Alibaba Cloud Log Service

ユーザーガイド

Document Version20190719

目次

1 準備

1.1 準備

Log Serviceは、複数のログ収集方法を提供します。 Log Serviceを使用して、ECS（Elastic Compute Service）ログ、ローカルサーバーログ、IoTデバイスログ、およびその他のクラウド製品ログを収集できます。

Log Serviceを使用する前に、まず次の準備を行う必要があります。

1. Log Serviceの有効化

   ご登録されたAlibaba CloudアカウントでLog Serviceプロダクトページにログインします。 Get it Freeをクリックします。 購入ページに自動的にリダイレクトされます。 I agree with Log Service Agreement of Serviceチェックボックスにチェックをいれ、「今すぐ有効にする」をクリックしてLog Serviceを有効にします。

2. アクセスキー（API / SDK用）を作成して有効にします。

   Logtailを使用してログを収集するには、AccessKeyが必要です。 Log Serviceを使用する前に、AccessKeyを作成する必要があります。

   Log Serviceコンソールで、右上のアバターにマウスを移動します。 ドロップダウンリストからアクセスキーを選択します 。 表示されたダイアログボックスでContinue to manage AccessKeyをクリックして、 Access Key Managementページに入ります。 AccessKeyを作成し、 作成されたAccessKeyが有効になっているかどうかを確認します。

3. プロジェクトの作成

   Log Serviceコンソールに初めてログインすると、プロジェクトの作成を求めるプロンプトが表示されます。 また、 右上にある プロジェクトの作成をクリックすることもできます。

   プロジェクト説明の変更と、プロジェクトの削除を実行することもできます。 詳細は、プロジェクトの作成を参照してください。

4. Logstoreの作成。

   プロジェクトを作成した後、システムがユーザーにLogstoreを作成するよう要求します。 また、プロジェクト名をクリックし、 右上の 作成をクリックすることもできます。 Logstoreを作成する際、これらのログの使用方法を指定する必要があります。

   また、Logstoreの変更や、削除を実行することもできます。 詳細は、Logstore の管理を参照してください。

5. シャードの管理（オプション）

   Logstoreを作成する際に、ログのボリュームと生成速度に基づいてシャードの数を指定できます。 また、Logstoreを変更する際に、シャードを分割または合併することで、シャードの数を変更することもできます。

   シャードの分割/合併の詳細は、シャードの分割を参照してください。

6. RAM権限付与の実行（オプション）

   クラウド製品のログを収集する必要がある場合、或いはLog ServiceデータをOSSまたは別の製品に保存および分析するために送信する必要がある場合は、Log Serviceまたは他のクラウド製品に関連する権限を付与する必要があります。

   サブアカウントを使用してLog Serviceで操作を実行するには、リソースアクセス管理（RAM）コンソールでサブアカウントにアクセス権限を付与する必要があります。

   権限付与ポリシーと手順の詳細については、概要を参照してください。

1.2 プロジェクトの作成

Log Service コンソールより、プロジェクトを作成および削除することができます。

プロジェクトの作成

注：

- 現時点では、Log Service コンソールでのみプロジェクトを作成できます。
- プロジェクト名は、Alibaba Cloud の全リージョンでグローバルに一意である必要があります。既に別のユーザーが使用しているプロジェクト名を入力すると、Project XXX already exists メッセージが表示されます。 別のプロジェクト名を入力し直します。
- プロジェクトを作成するには、収集するログのソースおよびその他の要件を基に、Alibaba Cloud リージョンを指定します。Alibaba Cloud の ECS (Elastic Compute Service) インスタンスのログを収集するには、ECS インスタンスと同じリージョンにプロジェクトを作成

することを推奨します。ログ収集が高速化します。ECS インスタンスのインターネット帯域幅を使用することなく、Alibaba Cloud のイントラネットを介してログが収集されます。

- プロジェクトの作成後は、プロジェクトのリージョンを変更できません。現時点では、Log Service プロジェクトを移行することはできないため、プロジェクトのリージョンは慎重にご選択ください。
- Alibaba Cloud の全リージョンで最大 50 のプロジェクトを作成できます。

手順

1. Log Service コンソールにログインします。
2. 右上隅のプロジェクトの作成をクリックします。
3. プロジェクト名を入力します。また、リージョンを選択して確認をクリックします。

構成項目	説明
プロジェクト名	長さは 3~63 字で、半角英数字 (小文字のみ)、ハイフン (-) を含むことができ、先頭および末尾には半角英数字 (小文字) にします。 注 : プロジェクトの作成後にプロジェクト名を変更することはできません。
説明	プロジェクトを作成後、その説明がプロジェクトリスト ページに表示され、 プロジェクトリストの右側にある変更をクリックして変更することができます。
リージョン	プロジェクトごとにリージョンを指定します。 プロジェクトを作成後、リージョンを変更することはできません、プロジェクトはリージョンをまたいだ移行はできません。

プロジェクトの削除

Log Service を無効化、プロジェクト内のログをすべて削除するといった場合には、プロジェクトを削除します。Log Service コンソールでプロジェクトを削除できます。

注 :
プロジェクトを削除すると、プロジェクトで管理していたすべてのログデータおよび構成情報が永久に削除され、復旧させることはできません。そのため、プロジェクトを削除する際は、データを損失しないようを避けるよう注意してください。

1. Log Service コンソールにログインします。

2. プロジェクト一覧ページで、削除するプロジェクトの右側の削除をクリックします。
3. 表示されるダイアログボックスの確認をクリックします 。

1.3 Logstore の管理

Logstore は、プロジェクトで作成されたリソースの集合です。Logstore 内のデータはすべて、同じデータソースからのデータです。Logstore 単位で、収集されたログデータを照会、分析、および送信します。Log Service コンソールでは、次の操作を行うことができます。

- Logstore の作成
- Logstore の設定変更
- Logstore の削除

Logstore の作成

注 :

- Logstore はプロジェクトに作成します。
- Log Service の各プロジェクトには最大 10 の Logstore を作成できます。
- Logstore の名前は、プロジェクト内で一意である必要があります。
- データ保管期間は、Logstore の作成後も変更できます。Logstore リストページで Logstore の右側の変更をクリックし、データ保管期間を変更して変更をクリックします。

1. プロジェクト一覧ページでプロジェクト名をクリックします。作成をクリックして Logstore を作成します。

   あるいは、プロジェクトを作成後に表示されるダイアログボックスの作成をクリックします。
2. 設定して確認をクリックします。

設定項目	説明
Logstore 名	プロジェクト内で一意である必要があります。名前は 3~63 文字で、半角英数字 (小文字のみ)、ハイフン (-)、アンダースコア (_) を含むことができ、先頭および末尾は半角英数字 (小文字) にする必要があります。 注 : Logstore の作成後に、Logstore 名を変更することはできません。

設定項目	説明
WebTracking	WebTracking 機能を有効にするかどうかを選択します。 この機能は、HTML、H5、iOS、または Android プラットフォームから Log Service へのログデータの収集をサポートします。 デフォルトでは無効となります。
データ保存期間	収集されたログを Logstore に保存する日数 (1~365 日間)。指定した期間を経過したログは破棄されます。
シャード数	Logstore のシャード数。各 Logstore には 1～10 のシャードを作成でき、各プロジェクトに最大 200 のシャードを作成できます。

Logstore の設定変更

Logstore の作成後、必要に応じて Logstore の設定を変更することができます。

1. Log Service コンソールにログインします。
2. プロジェクト一覧ページでプロジェクト名をクリックします。
3. Logstore リスト ページで、Logstore の右側の変更をクリックします。
4. Logstore 属性の変更ダイアログボックスが表示されます。Logstore の設定を変更し、ダイアログボックスを閉じます。

Logstore の削除

Logstore が不要になった場合には、Logstore を削除します。Logstore は、Log Service コンソールより削除することができます。

 注 :

- Logstore が削除されると、Logstore に保存されていたログデータは完全に削除され、復旧させることはできません。慎重に削除します。
- Logstore を削除する前に、紐づいている Logtail 構成をすべて削除します。

1. Log Service コンソールにログインします。
2. プロジェクト一覧ページでプロジェクト名をクリックします。
3. Logstore リストページで、削除する Logstore の右側にある削除をクリックします。
4. 表示されるダイアログボックスで確認をクリックします。

1.4 シャードの分割

Logstore の読み取り/書き込みログは、シャードに格納されます。各 Logstore はいくつかのシャードに分けられます。Logstore を作成する際に、シャード数を指定します。なお、シャードを分割または結合することで、シャード数は増減します。

既存のシャードに対して、次のことを行えます。

- シャードの分割
- シャードの結合
- シャードの削除

シャードの分割

シャードの読み書き速度は、書き込みは 5 MB/秒、読み込みは 10 MB/秒です。データトラフィックがシャードの容量を超える場合は、シャードを分割して、シャード数を増やすことをお勧めします。

手順

シャードを分割する際、readwrite (読み書き) ステータスのシャードの ShardId、および、MD5 を指定します。MD5 は、シャードの BeginKey より大きく、シャードの EndKey より小さくなければなりません。

分割すると、1 つのシャードは 2 つに分割され、シャード数は 1 つから 2 つになります。分割されたら、元のシャードのステータスは readwrite (読み書き) から readonly (読み取り) に切り替わります。データを読み込むことはできますが、新たにデータを書き込むことはできなくなります。新たに生成された 2 つのシャードは readwrite (読み書き) ステータスであり、元のシャードの後ろに追加されます。2 つのシャードの MD5 範囲は、元のシャードの範囲を含むものになります。

1. Log Service コンソールにログインします。
2. プロジェクト一覧ページで、プロジェクト名をクリックします。
3. Logstore リストページで、Logstore の右側の変更をクリックします。
4. 分割されるシャードの右側の分割をクリックします。

5. 確認をクリックし、ダイアログボックスを閉じます。

   分割すると、元のシャードは readonly (読み取り) ステータスに切り替わり、新たに生成された 2 つのシャードの MD5 範囲には、元のシャードの範囲が含まれます。

シャードの結合

シャードを結合すると、シャード数は減ります。指定されたシャードとその右隣りのシャードが結合されます。readwrite (読み書き) ステータスの新しいシャードが生成され、その MD5 範囲は、元の 2 つのシャードの全範囲が含まれます。元の 2 つのシャードは readonly (読み取り) ステータスになります。

注意事項

シャードを結合する際は、readwrite (読み書き) ステータスのシャードを指定します。指定したシャードが最後尾のシャードでないことを確認します。サーバーは、指定されたシャードの右隣りのシャードを自動検出して結合します。 結合後、指定されたシャードとその右隣りのシャードは readonly (読み取り) ステータスになります。データを読み取ることはできますが、書き込むことはできなくなります。readwrite (読み書き) ステータスの新しいシャードが生成され、その MD5 範囲には、元の 2 つのシャードの範囲が含まれます。

手順

1. Log Service コンソールにログインします。
2. プロジェクト一覧ページでプロジェクト名をクリックします。
3. Logstoreリストページで、Logstore の右側の変更をクリックします。
4. 結合するシャードの右側にある結合をクリックします。

   結合後、指定されたシャードとその右隣りのシャードが readonly (読み取り) ステータスに切り替わり、新しく生成されたシャードの MD5 範囲には、元の 2 つのシャードの範囲が含まれます。

シャードの削除

Logstore のライフサイクル (データ保管期間) は、永久または 1～365 日に指定できます。
シャードおよびシャードのログデータは、指定されたデータ保管期間を経過すると自動的に削除されます。読み取り専用シャードに対しては料金は発生しません。

また、Logstore を削除することでも Logstore 内のシャードはすべて削除されます。

2 データ収集

2.1 収集方法

LogHub は、さまざまなログ収集方法を提供するさまざまな RESTful API をサポートします。たとえば、1 つまたは複数のクライアント、Web サイト、プロトコル、SDK、および API を介したログ収集です。

データソース

Log Service は次のソースからログを収集できます。

タイプ	ソース	アクセス方式	Details
アプリケーション	プログラム出力	Logtail	-
	アクセスログ	Logtail	分析 - Nginx アクセスログ
	Link track	Jaeger Collector、Logtail	-
プログラミング言語	Java	SDK、Java Producer Library	-
	Log4J Appender	1.x、2.x	-
	LogBack Appender	LogBack	-
	C	Native	-
	Python	Python	-
	Python Logging	Python Logging Handler	-
	PHP	PHP	-
	C#	C#	-
	C++	C++ SDK	-
	Go	Go	-
	NodeJS	NodeJs	-
	JS	JS/Web Tracking	-
OS	Linux	Logtail	-
	Windows	Logtail	-

タイプ	ソース	アクセス方式	Details
	Mac/Unix	Native C	-
	Docker ファイル	Logtail ファイルコレクション	-
	Docker 出力	Logtail コンテナの標準出力	-
Mobile 端末	iOS/Android	iOS SDK、Android SDK	-
	ウェブサイト	JS/Web Tracking	-
	インテリジェント IoT	C プロデューサーライブラリ	-
クラウドプロダクト	ECS や OSS などのさまざまなプロダクト 詳細は、クラウドプロダクトのログ をご参照ください。	クラウドプロダクトコンソール	クラウドプロダクトログ
	MaxCompute データのインポート	Dataworks を使用した MaxCompute データのエクスポート	DataWorks より Log Service に MaxCompute データを収集
サードパーティソフトウェア	Logstash	Logstash	-

次の表に、Log Service でログを収集できるクラウドプロダクトを示します。

タイプ	クラウド製品名	有効化する方法	Details
エラスティックコンピューティング	Elastic Compute Service (ECS)	Logtail をインストールして有効化	Logtail の紹介
	Container Service / Container Service for Kubernetes	Container Service コンソールにて有効化	テキストログ と stdout
ストレージ	Object Storage Service (OSS)	OSS コンソールにて有効化	OSS アクセスログ
ネットワーク	Server Load Balancer (SLB)	SLB コンソールにて有効化	レイヤ 7 SLB のアクセスログ

タイプ	クラウド製品名	有効化する方法	Details
	Virtual Private Cloud (VPC)	VPC コンソールにて有効化	フローログ
	API Gateway	API Gateway コンソールにて有効化	API Gateway アクセスログ
セキュリティ	ActionTrail	ActionTrail コンソールにて有効化	ActionTrail の概要
	DDoS 防御	DDoS 防御コンソールにて有効化	DDoS 防御の概要
	Threat Detection Service	Threat Detection Service Enterprise Edition を購入し、Threat Detection Service コンソールでサービスを有効にします。	Log retrieval
	Anti-Bot Service	Anti-Bot Service コンソールにて有効化	Anti-Bot Service ログ
アプリケーション	Log Service (LOG)	Log Service コンソールにて有効化	Log Service の概要

ネットワークとアクセスポイントの選択

Log Service はサービスエンドポイントを各リージョンで提供しています。次の種類のネットワークアクセス方法がサポートされています：

- (おすすめ）イントラネット（クラシックネットワーク）およびプライベートネットワーク（VPC）：スムーズなサービスアクセスと高品質の帯域幅リンクがあるリージョンに適用可能。
- インターネット（クラシックネットワーク）：制限なくアクセス可能 アクセス速度は回線品質によって変わります。 アクセスセキュリティを維持するために HTTPS の使用をお勧めします。

よくある質問

- Q：どの種類のネットワークが物理接続に適用されますか？

  A：イントラネット/プライベートネットワーク

- Q：インターネットのデータ収集中にインターネットの IP アドレスを収集できますか？

  A：はい。 Logstore の管理 に記載されている手順に従って、インターネット IP アドレスの記録機能を有効にすることができます。
- Q：リージョン A にある ECS サーバーからログを収集し、リージョン B にある Log Service サーバー上のプロジェクトに送信する場合は、どの種類のネットワークを使用できますか。

  A：インターネット版の Logtail を ECS サーバーにインストールした後、インターネットを使用してログを転送します。 他のシナリオに関しては、ネットワークタイプの選択 の指示に従ってください。
- Q：アクセスが正常に確立されたかどうかはどうすれば判断できますか？

  A：次のコマンドを実行した後に情報が返されれば、アクセスは正常に確立されています。

  ```
  curl $myproject.cn-hangzhou.log.aliyuncs.com
  ```

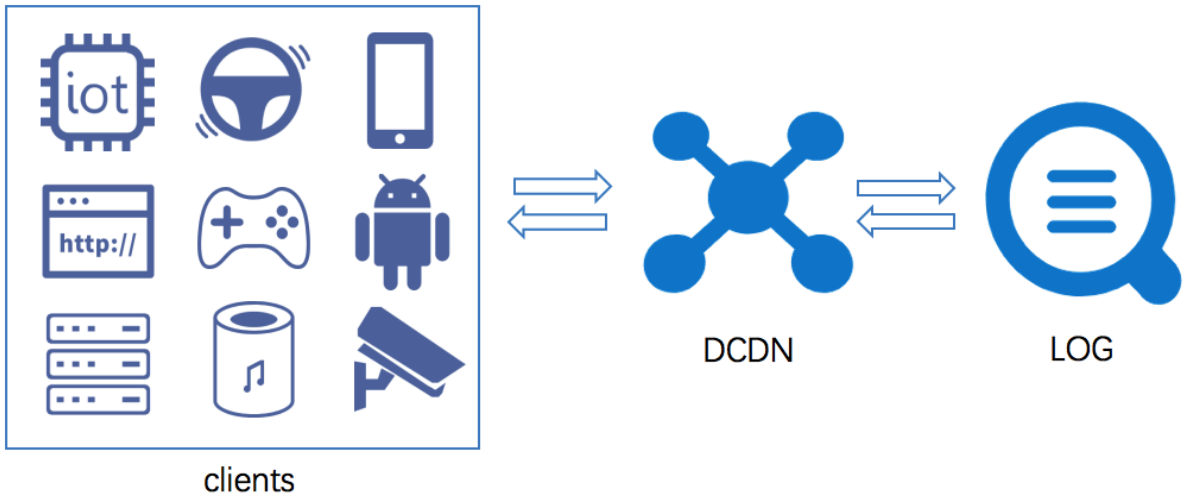
  `$myproject` はプロジェクト名を示し、 `cn-hangzhou.log.aliuncs.com` はアクセスポイントを示します。

2.2 収集の加速

2.2.1 概要

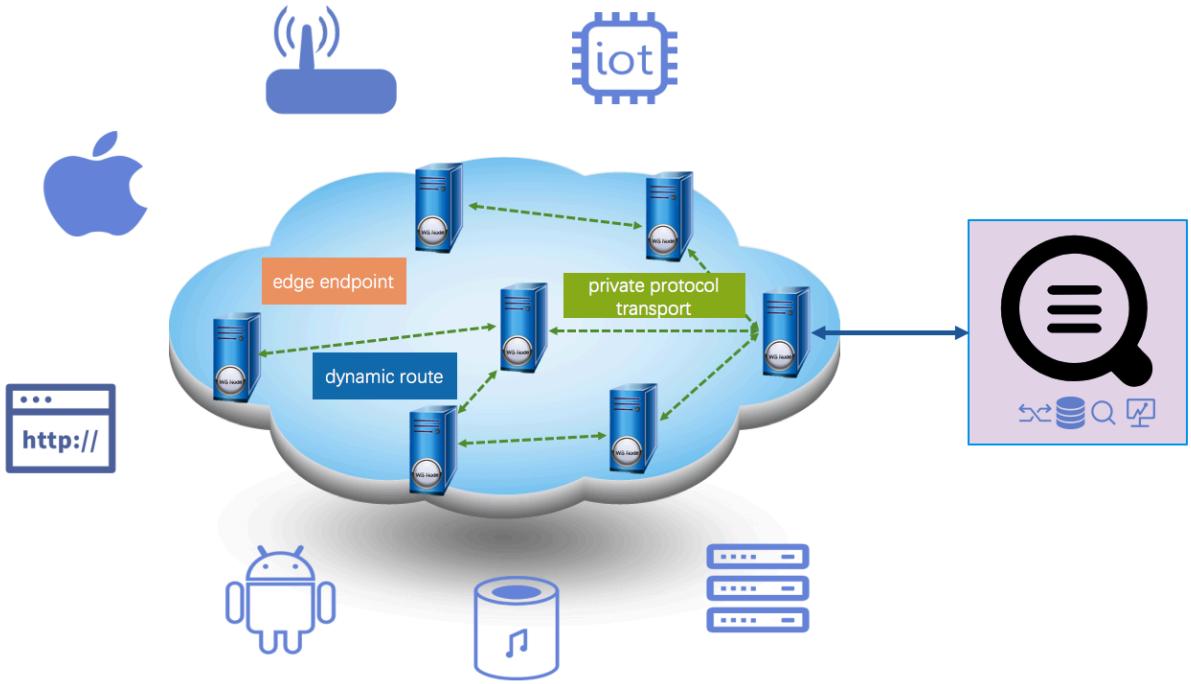
Log Serviceは、Virtual Private Cloud（VPC）およびパブリックネットワークに基づいて、Global Acceleration パブリックネットワークのネットワークタイプを追加します。 Global Acceleration パブリックネットワークは、通常のパブリックネットワークアクセスと比較して、遅延と安定性の面で大きな利点があり、データ収集、消費遅延、信頼性の高いシナリオに適しています。 Log Service 向けのGlobal Acceleration は、Alibaba Cloud Dynamic Routeが提供するCDNプロダクトのアクセラレーション環境に依存します。 この機能により、キャリア間のアクセス、不安定なネットワーク、トラフィックの急増、ネットワークの輻輳といった要因により発生する、遅い応答、パケットの消失、不安定なサービスなどの問題を解決して、サイト全体のパフォーマンスとユーザーエクスペリエンスを向上します。

Log Service 向けのGlobal Acceleration は、Alibaba Cloud Content Delivery Network（CDN）ハードウェアリソースに基づいており、携帯電話、IoT（Internet of Things）デバイス、スマートデバイスなどのさまざまなデータソースからのログ収集およびデータ転送の安定性を最適化します 、セルフビルドインターネットデータセンター（IDC）、およびその他のクラウドサーバーが含まれます。



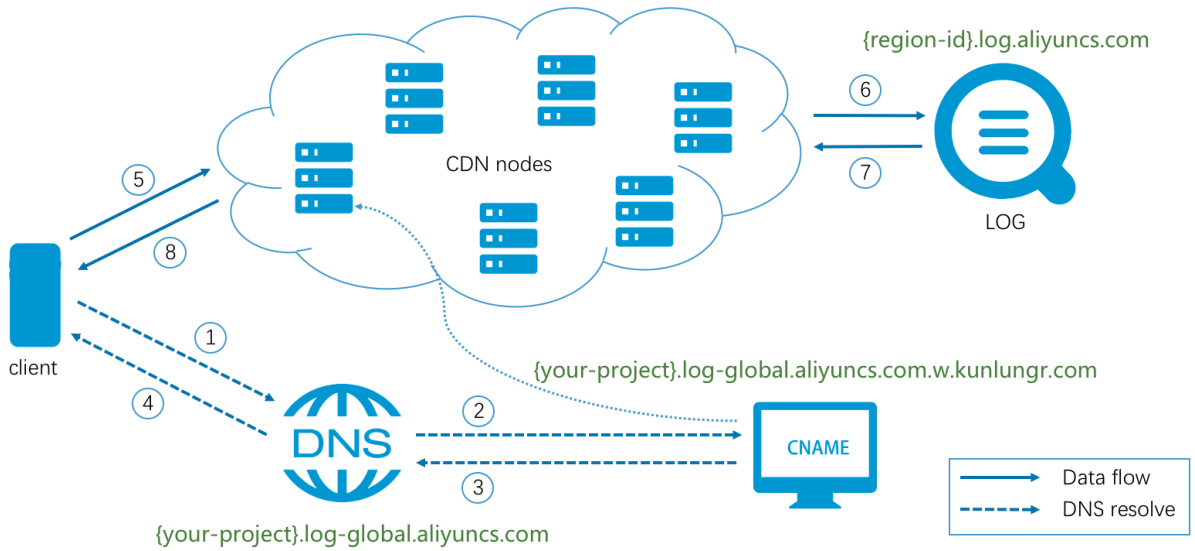
技術原理

Log Service 向けのGlobal Acceleration は、Alibaba Cloud CDNハードウェアリソースに基づいています。 お使いのグローバルアクセス端末（携帯電話、IOTデバイス、スマートデバイス、セルフビルドIDC、その他のクラウドサーバーなど）は、世界中のAlibaba Cloud CDNの最寄ノードにアクセスし、 CDN内部高速チャネルを通じてLog Serviceにルーティングします。この方法は、一般的なパブリックネットワーク転送と比較して、ネットワークの遅延とジッターを大幅に削減できます。



Log Service 向けのGlobal Acceleration リクエストの処理フローは上図の通りです。 全体的な流れは次のようになります：

1. クライアントは、ログアップロードまたはログダウンロードリクエストをログサービスアクセラレーションドメイン名*your-project*.log-global.aliyuncs.comに送信する前に、パブリックDNSにドメイン名解決リクエストを送信する必要があります。
2. パブリックDNSのドメイン名*your-project*.log-global.aliyuncs.comは、CNAMEアドレス*your-project*.log-global.aliyuncs.com.w.kunlungr.comを指します。 ドメイン名解決は、Alibaba Cloud CDNのCNAMEノードに転送されます。
3. Alibaba Cloud CDNスマートスケジューリングシステムに基づき、CNAMEノードは最適なCDNエッジノードのIPアドレスをパブリックDNSに返します。
4. パブリックDNSは、最終的にクライアントに解決されたIPアドレスを返します。
5. クライアントは、取得したIPアドレスに基づいてサーバにリクエストを送信します。
6. リクエストを受信すると、CDNエッジノードは、ダイナミックルート検索およびプライベートトランスポートプロトコルに基づいて、Log Serviceサーバに最も近いノードにリクエストをルーティングします。 リクエストは最終的にLog Serviceに転送されます。
7. Log Serviceのサーバは、CDNノードからリクエストを受信すると、そのリクエストの結果をCDNノードに返します。
8. CDNは、Log Serviceによって返された結果またはデータを透過的にクライアントに送信します。



課金方法

Log Service 向けGlobal Acceleration の課金項目は次のとおりです：

- Log Serviceへのアクセスによって発生する費用

  Log Serviceへのアクセスによって発生する費用は、一般的なパブリックネットワークの費用と同様です。 Log Serviceは、従量課金をサポートし、無料クォータも提供しています。 詳細は、Billing methodを参照してください。
- CDN用ダイナミックルートサービスの費用

  CDN用ダイナミックルートのクラウドプロダクト費用については、CDNのダイナミックルートの請求方法を参照してください。

シナリオ

- 広告

  広告閲覧回数とクリック回数に関するログデータは、広告の料金請求にとって非常に重要です。 広告キャリアは、携帯端末、H5ページ、PC端末などがあります。 一部の辺鄙な地方では、パブリックネットワークのデータ転送の安定性が低く、ログ損失のリスクが存在します。 Global Acceleration によって、より安定した信頼性の高いログアップロードチャネルを取得することができます。
- オンラインゲーム

  オンラインゲーム業界では、公式サイト、ログインサービス、セールスサービス、ゲームサービス、およびその他のサービスにおけるデータ収集のパフォーマンスと安定性に対する高い要件があります。 モバイルゲームデータ収集やグローバル化されたゲームのデータバック送信の場合、データ収集の適時性や安定性を保証するのは難しくなります。 この問題を解決するには、Log Service 向けの Global Acceleration を使用することを推奨します。
- 金融

  金融関係のアプリケーションでは、ネットワークの高可用性と高いセキュリティが求められます。 各トランザクションおよび各ユーザーアクションの監査ログは、安全かつ確実にサーバーに収集する必要があります。 現在、モバイルトランザクションが主流になっています。 たとえば、オンラインバンキング、クレジットカードモール、モバイル証券、およびその他のタイプのトランザクションは、Log Service 向け HTTPS Global Acceleration を使用して、安全で高速で安定したログ収集を実現します。
- IoT

  IoTデバイスやスマートデバイス（スマートスピーカーやスマートウォッチなど）は、データ分析のためにセンサーデータ、操作ログ、重要なシステムログ、およびその他のデータをサーバーに収集します。 これらのデバイスは、通常、世界中に分散されており、周囲のネッ

トワークは必ずしも信頼性の高いものではありません。 安定した信頼性の高いログ収集を実現するには、Log Service 向け Global Acceleration を使用することを推奨します。

アクセラレーションの効果

リージョン	遅延ms（一般的パブリックネットワーク）	遅延ms（アクセラレーション）	タイムアウト割合%（一般的パブリックネットワーク）	タイムアウト割合%（アクセラレーション）
杭州	152.881	128.501	0.0	0.0
ヨーロッパ	1750.738	614.227	0.5908	0.0
アメリカ	736.614	458.340	0.0010	0.0
シンガポール	567.287	277.897	0.0024	0.0
中東	2849.070	444.523	1.0168	0.0
オーストラリア	1491.864	538.403	0.014	0.0

テスト環境は次の通り：

- Log Serviceリージョン：中国（フフホト）
- アップロードするパケットの平均サイズ：10KB
- テスト時間範囲：1日（平均値）
- リクエストタイプ：HTTPS
- リクエストサーバー：Alibaba Cloud ECS（仕様：1C1GB）


注：
アクセラレーション効果は参考用です。

2.2.2 Global Acceleration の有効化

Global Accleration を有効化するには、次の手順に従ってください。

前提条件

- Log Serviceを有効にして、プロジェクトと Logstore が作成されていること。
- Dynamic Route for CDN が有効になっていること。
- HTTPS acceleration を有効にするには、 まず HTTP acceleration を有効にする必要があります。

設定

プロジェクトのHTTP Global Acceleration が有効になった後、必要に応じてLogtail、SDKなどの Global Acceleration を構成することもできます。

1. HTTP アクセラレーションの有効化。
2. Logtail、SDKなどの Global Acceleration を有効にする。
   - HTTPS

     HTTPSを使用してLog Serviceにアクセスする場合は、HTTPSアクセラレーションを確実に有効にしてください。 HTPSアクセラレーションを設定するには、HTTPS アクセラレーションの有効化を参照してください。
   - Logtailログ収集

     Logtailをインストールする際に、ページプロンプトでグローバルアクセラレーションネットワークタイプを選択します。 これにより、Logtailを使用してログを収集すると、Global Acceleration を取得できます。
   - SDK/Producer/Consumer

     エンドポイントをlog- global.aliyuncs.comに置き換えることで、SDK、Producer、ConsumerなどのLog Serviceにアクセスする他の方法をアクセラレーションすることができます。

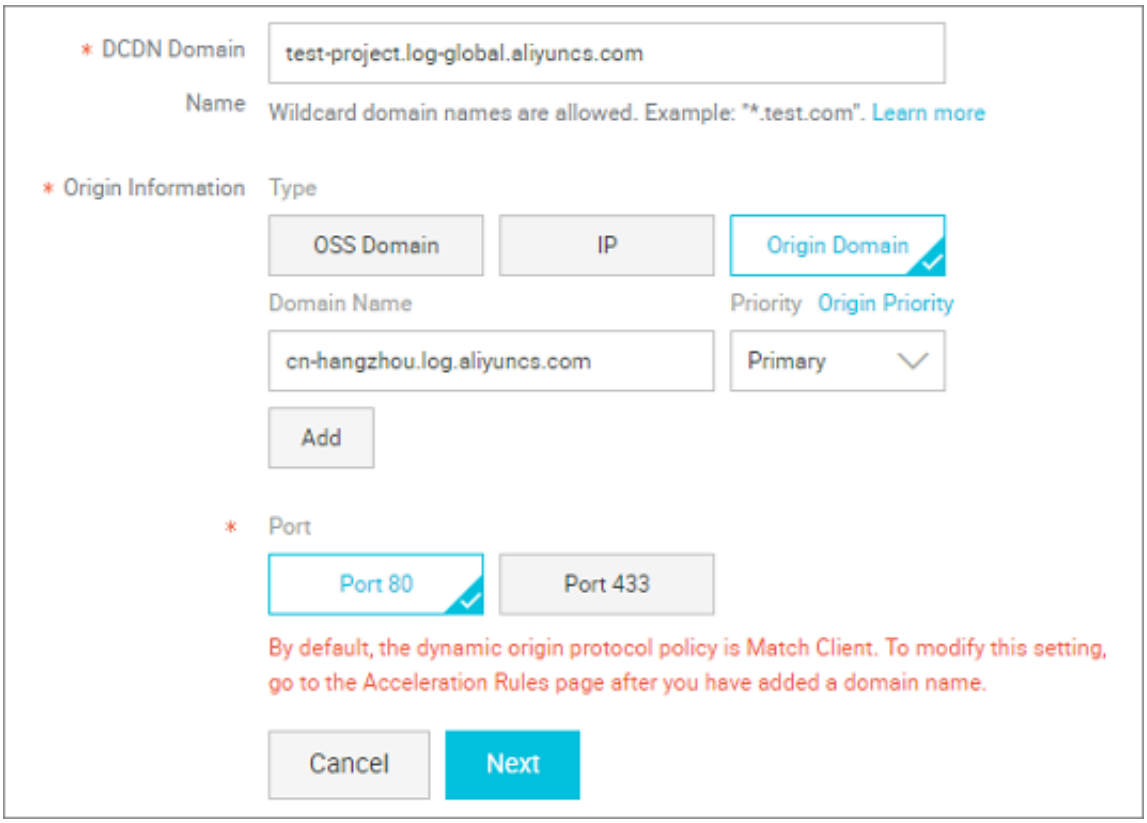
HTTPアクセラレーションの有効化

1. Dynamic Route for CDNコンソールにログインします。 左側のメニューでドメイン名をクイックしてドメイン名ページへ移動します。
2. 左上のドメイン名の追加ボタンをクイックしてドメイン名の追加ページへ移動します。
3. DCDN ドメイン名などの情報を入力して、次をクリックします。

構成項目	説明
ドメイン名のアクセラレーション	*project_name*.log-global.aliyuncs.com の *project_name* をご使用するプロジェクト名に置き換えます。
オリジン情報のタイプ	オリジンドメインを選択します。
ドメイン名	プロジェクトが属するリージョンのパブリックネットワークエンドポイントを入力します。 エンドポイントに関する情報は、サービスエンドポイントを参照してください。

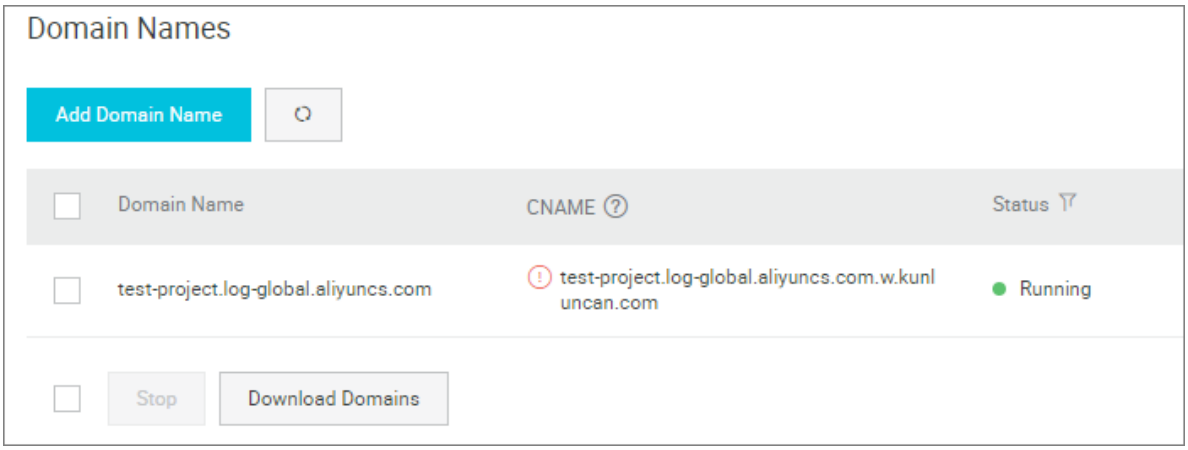
構成項目	説明
ポート	ポート80を選択します。 HTTPSアクセラレーションで要件がある場合、HTTPSアクセラレーションの有効化を参照してください。
アクセラレーションリージョン	この構成項目はデフォルトで表示されず、アクセラレーションリージョンは中国本土となります。 Global Acceleration が必要な場合は、ホワイトリストに申請するため、Dynamic Route for CDNへチケットを起票してサポートセンターへお問い合わせください。

ドメイン名の追加に関する詳細情報は、「ドメイン名の追加」に参照してください。

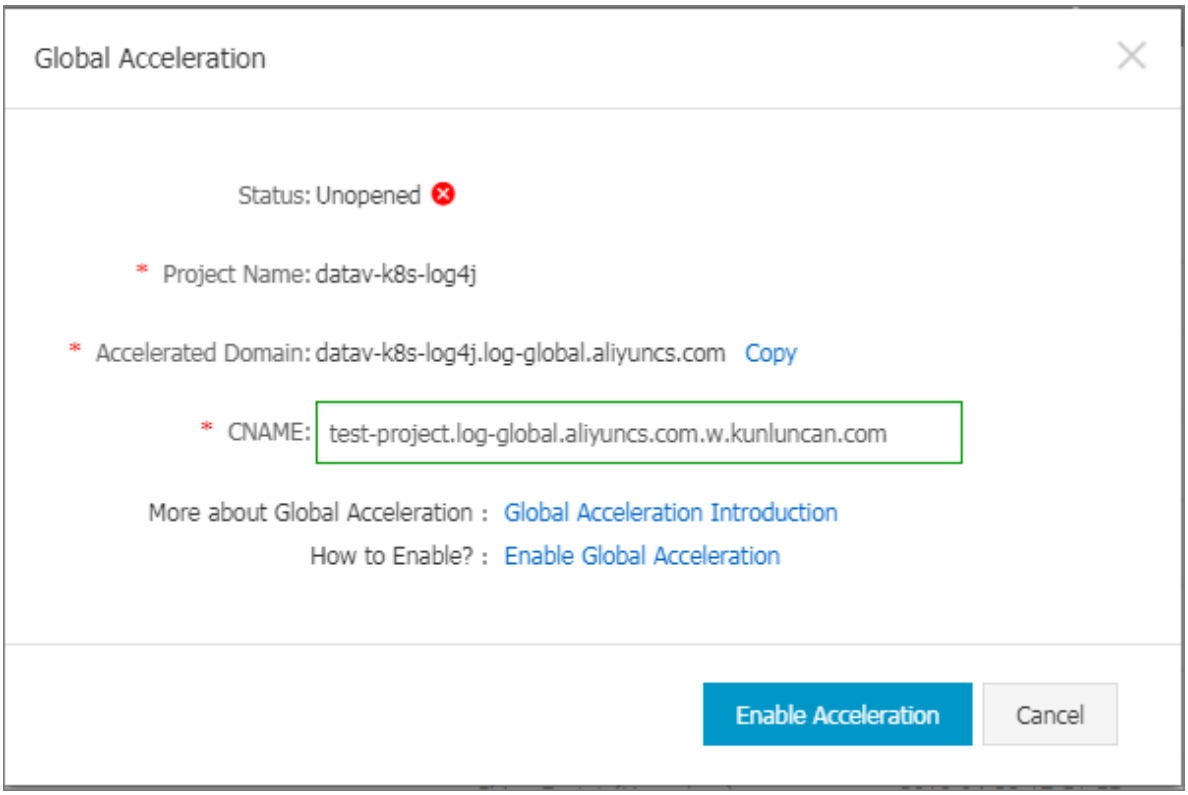


4. ページの指示に従ってドメイン名ページへ移動します。

   ドメイン名ページで、対応する各ドメイン名のCNAMEを表示できます。



5. Log Serviceコンソールにログインして、プロジェクトリスト内の特定プロジェクトの右側にあるGlobal Accelerationをクリックします。
6. ダイアログボックスで加速されたドメイン名に対応するCNAMEを入力します。 アクセラレーションを有効化をクリックします。



   上記の手順を実行すると、Log Service 向けGlobal Acceleration が有効になります。

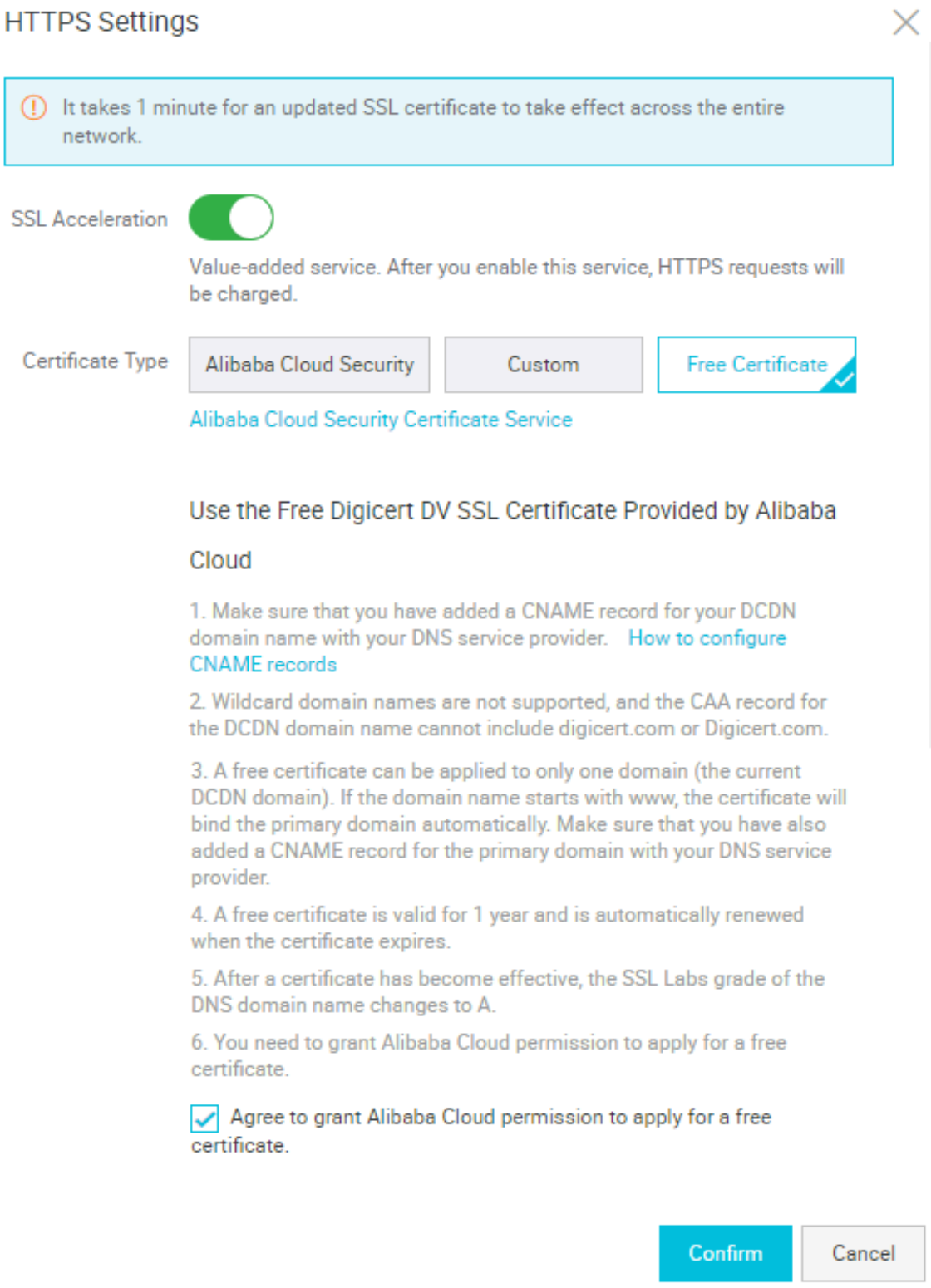
HTTPSアクセラレーションの有効化

HTTPアクセラレーションを有効にした後、HTTPSアクセスの要件がある場合、次の手順に従ってHTTPSアクセラレーションを有効にすることができます。

1. Dynamic Route for CDNコンソールにログインします。 左側のメニューで、ドメイン名をクリックしてドメイン名ページへ移動します。
2. 特定のドメイン名の右側にある設定をクリックします。
3. 左側のメニューで、HTTPS設定をクリックし、SSL証明書セッション内の変更をクイックしてHTTPS設定ページを開きます。

4. SSL アクセラレーションと証明書タイプを設定します。

   - SSL アクセラレーションを有効にします。
   - 証明書タイプで無料証明書を選択します。



設定が終わったら、Alibaba Cloudが無料証明書を申請することに同意する。にチェックを入れ、OKをクリックします。

アクセラレーション設定が有効になっているか確認する

よくある質問

- アクセラレーション設定が有効かどうかを確認する方法は？

  設定が終わったら、アクセラレートされたドメイン名にアクセスすることでアクセラレーションが有効になっているかどうかを確認できます。

  たとえば、Global Acceleration が *test - project* プロジェクトに対して有効になっている場合、`curl` を使用してアクセラレートされたドメイン名にリクエストを送信します。次のタイプのアウトプットが返されると、アクセラレーションが有効だと判断できます。

  ```
  $ curl   test - project . log - global . aliyuncs . com
  {" Error ":{" Code ":" OLSInvalid  Method "," Message ":" The
   script   name   is   invalid  : /"," RequestId ":" 5B55386A2C
   E41D1F4FBC  F7E7 "}}
  ```

  チェック方法に関する詳細情報は、アクセラレーション設定が有効かどうかを確認する方法を参照してください。

- アクセラレートされたドメイン名にアクセスする際に返される `project not exist` というエラーをどのように処理しますか？

  この問題は、通常、無効な送信元サイトアドレスによって発生します。 Dynamic Route for CDNコンソールにログインし、ソースサイトのアドレスを、プロジェクトが属するリージョンのパブリックネットワークアドレスに変更します。 アドレスリストに関する詳細情報は、サービスエンドポイントを参照してください。

  > 注：
  > ソースサイトアドレスを変更すると、数分間の同期化遅延が発生します。

2.2.3 Logtail 収集アクセラレーションの設定

Global Acceleration が有効になった後、Global Acceleration モードでインストールされた Logtail は、Gloabl Accelerationモードでログを自動的に収集します。 Global Acceleration が有効になる前にインストールされた Logtail の場合は、このドキュメントを参照し、アクセラレーションモードをに手動で切り替える必要があります。

前提条件

1. HTTPアクセラレーションの有効化.

2. (オプション) HTTPアクセラレーションの有効化.

   HTTPS を使用して Log Service にアクセスする場合は、HTTPS アクセラレーションが有効になっていることや、HTTPアクセラレーションの有効化の指示に従って HTTPS アクセラレーションが構成されていることを確認します。

3. アクセラレーションが正常に機能することを確認します

   グローバルアクセラレーションの有効化に記載されている手順に従います。

始める前に

Logtail 収集アクセラレーションを構成する前に、次の点に注意します：

- グローバルアクセラレーションを有効にしてから Logtail をインストールする場合は、Logtail の Linux へのインストールの指示に従い、インストールモードを グローバルアクセラレーション に設定する必要があります。 その後、Logtail はグローバルアクセラレーションモードの方法を使用してログを収集します。
- グローバルアクセラレーションが有効になる前に Logtail がインストールされている場合は、本ドキュメントを参照し、Logtail 収集モードをグローバルアクセラレーションに切り替える必要があります。

Logtail 収集モードをグローバルアクセラレーションへの切り替え

1. Logtail を停止します。
   - Linux の場合、管理者アカウントで `/ etc / init . d / ilogtaild  stop` を実行します。
   - Windows の場合：
     a. コントロールパネルで、システムとセキュリティ > 管理ツールを選択します。
     b. サービスプログラムを開き、LogtailWorker ファイルを検索します。
     c. ファイルを右クリックし、ショートカットメニューで停止をクリックします。
2. Logtail 起動構成ファイル *ilogtail_c onfig . json* を変更します。

   起動構成ファイル (ilogtail_config.json)の指示に従ってエンドポイントの `data_serve r_list` を `log - global . aliyuncs . com` に変更します。

3. Logtail を起動します。
   - Linuxの場合、管理者アカウントで `/ etc / init . d / ilogtaild  start` を実行します。
   - Windows の場合：
     a. コントロールパネルで、システムとセキュリティ > 管理ツールを選択します。
     b. サービスプログラムを開き、LogtailWorker ファイルを検索します。
     c. ファイルを右クリックし、ショートカットメニューで開始クリックします。

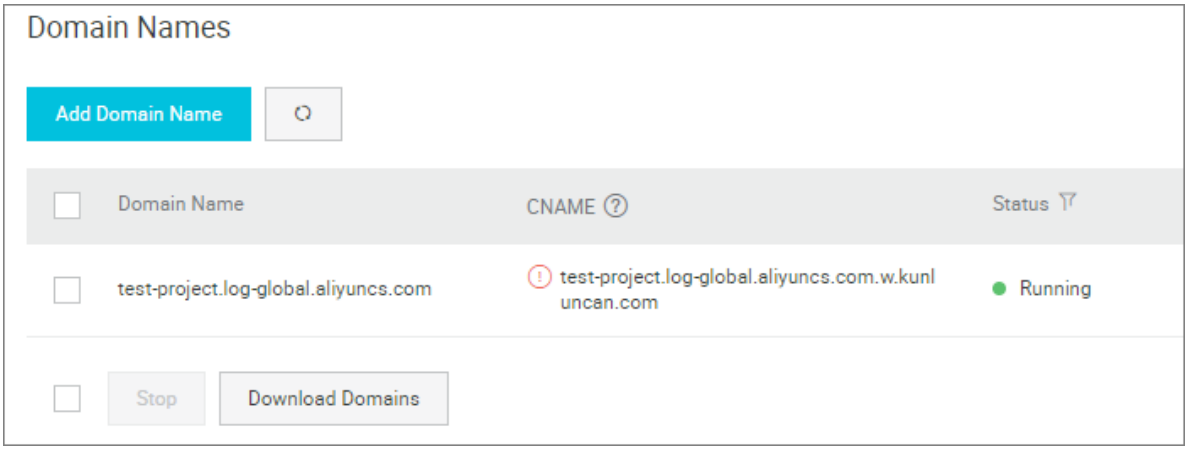
2.2.4 Global Accelerationを無効化する

Log ServiceのGlobal Acceleration を無効化するには、次の操作を実行します。

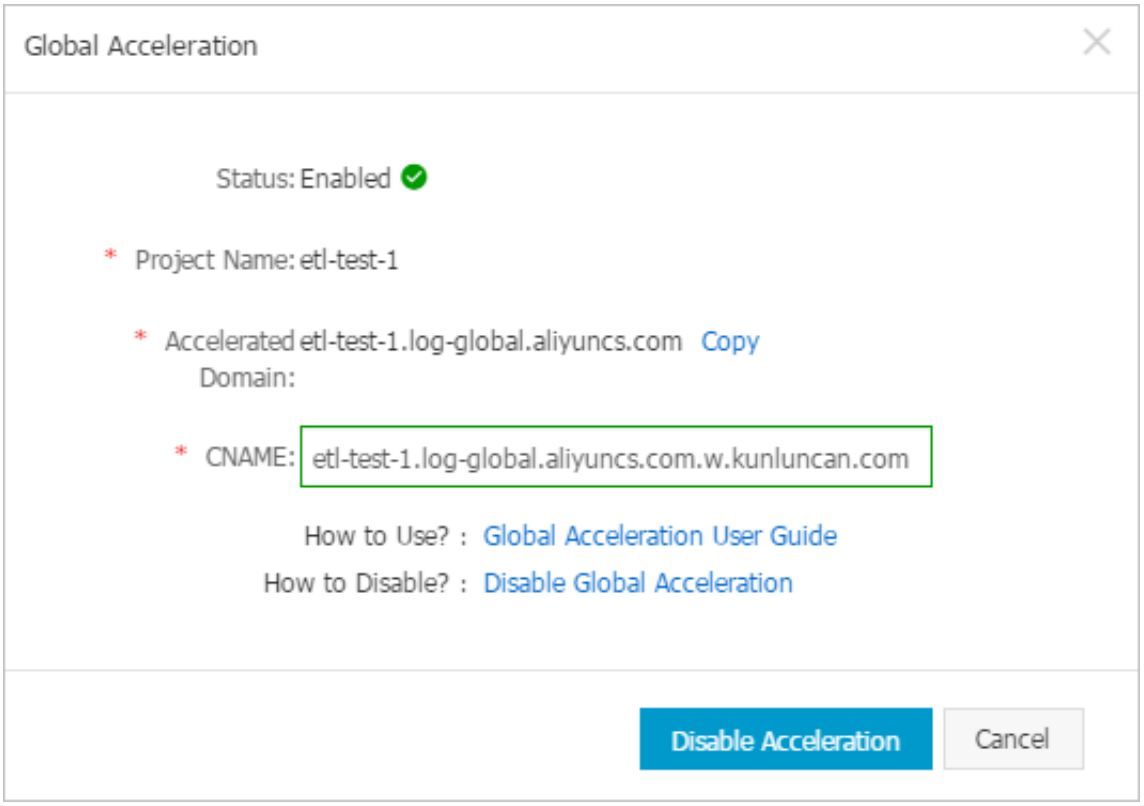
> 注：
> Global Acceleration を無効にすると、プロビジョニング中に設定したドメイン名が利用できなくなります。 Global Acceleration を無効にする前に、すべてのクライアントがドメイン名を使用してデータのアップロードやリクエストを行っていないことを確認してください。

Global Acceleration を無効化

1. DCDNコンソールにログインします。 左側のドメイン名をクリックしてドメイン名ページを開きます。
2. 無効化しようとするドメイン名に対応するCNAMEを確認します。



3. Log Serviceコンソールにログインします。 プロジェクトリストページで、対象プロジェクトの右側にあるGlobal Accelerationをクリックします。
4. CNAMEを入力して加速を無効化をクリックします。


Global Acceleration
Status: Enabled
* Project Name: etl-test-1
* Accelerated Domain: etl-test-1.log-global.aliyuncs.com Copy
* CNAME: etl-test-1.log-global.aliyuncs.com.w.kunluncan.com
How to Use? : Global Acceleration User Guide
How to Disable? : Disable Global Acceleration
Disable Acceleration
Cancel

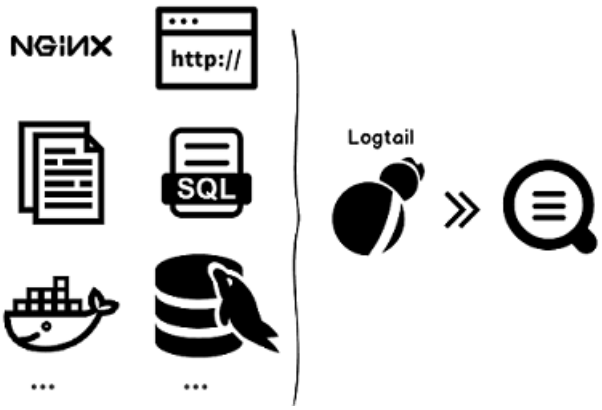
3 Logtailでの収集

3.1 概要

3.1.1 概要

Logtail アクセスサービスは、 Log Service によって提供されるログ収集エージェントです。 Logtail を使用して、Alibaba Cloud Elastic Compute Service （ECS）インスタンスなどのサーバーから、 Log Service コンソールでリアルタイムでログを収集することができます。

図 3-1 : 機能の利点



利点

- ログファイルに基づく非接続のログ収集。 アプリケーションのコードを変更する必要はなく、ログ収集はアプリケーションの動作ロジックに影響を与えません。
- テキストログの収集に加えて、binlog、http、およびcontainer stdoutなど、より多くの収集方法がサポートされています。
- コンテナーは十分にサポートされています。 このサービスは、標準コンテナー、集団クラスター、およびKubernetesクラスターでのデータ収集をサポートします。
- Logtail は、ログ収集プロセスで発生した例外を処理します。 ネットワークや Log Service などで異常が発生し、ユーザデータが一時的に予約帯域の書き込み制限を超えるといった問題が発生すると、Logtail はローカルで積極的にデータを再試行してキャッシュし、データセキュリティを保証します。
- ログサービスに基づく集中管理機能。 Logtail をインストールした後は、ログを収集するマシンや Log Service での一元的な収集方法などの設定を、サーバーにログインすることなく、また個別に設定することなく行うことができます。 Logtail のインストール方法について

は、Logtail の Windowsへのインストールと Logtail の Linux へのインストールを参照してください。

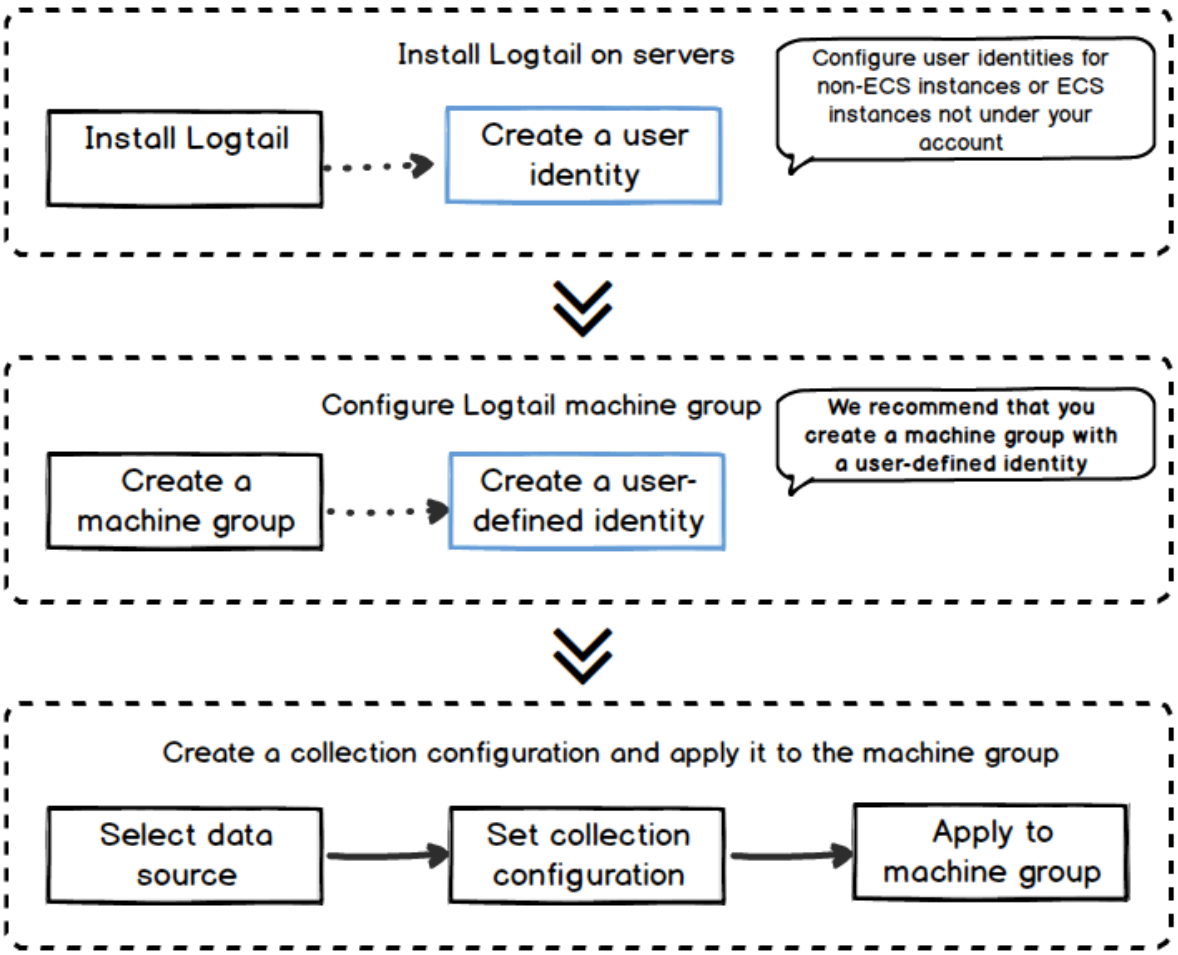
- 包括的な自己保護の仕組み。 マシン上で実行されている収集エージェントがサービスのパフォーマンスに大きな影響を与えないように、Logtail クライアントは CPU、メモリ、ネットワークリソースの使用を厳しく保護し、制限します。

処理能力と制限

制限を参照してください。

手順

図 3-2 : 設定プロセス



Logtail を使用してサーバからログを収集するには、次の手順に従います。

1. Logtail をインストールします。 ログを収集するサーバーに Logtail をインストールします。
   詳細：Logtail の Windowsへのインストール と Logtail の Linux へのインストール

2. マシングループにユーザー定義 ID を設定する。 Alibaba Cloud ECS インスタンスからログを収集しようとしている場合は、この手順をスキップしてください
3. Logtail マシングループの作成。 Log Service は、マシングループの形式で Logtail クライアントを使用してログを収集するすべてのサーバーを管理します。 Log Service は、IP またはユーザー定義の ID を使用してマシングループを定義することをサポートします。 マシングループに適用ページの指示どおりにマシングループを作成することもできます。
4. Logtail コレクション設定を作成し、それをマシングループに適用します。 データインポートウィザードで Logtail 設定を作成することにより、テキストファイルの収集やlogtailを用いたsyslogデータの収集などのデータを収集することができます。 次に、Logtail 設定をマシングループに適用することができます。

上記の手順を完了すると、ログを収集するサーバーの増分ログがアクティブに収集され、対応するログストアに送信されます。 履歴ログは収集されません。 コンソールまたは API/SDK を使用してこれらのログを照会することができます。 収集が正常かどうか、エラーが発生したかどうかなど、コンソールの Logtail ログ収集状況を照会することもできます。

Log Service コンソールの Logtail アクセスサービスの完全な手順については、Logtail を使用したテキストファイルの収集を参照してください。

コンテナー

- Alibaba クラウド Container Service ：Log Service の有効化を参照してください。
- Alibaba クラウドService Kubernetesクラスター：Kubernetes のログ収集
- 自己構築型Kubernetes：自分で構築した Kubernetes をインストールする
- その他自己構築Dockerクラスタ：標準の Docker ログの収集

主なコンセプト

- グループ：マシングループには、ログタイプを収集する 1 つ以上のマシンが含まれています。 Logtail 構成をマシングループに適用すると、 Log Service は同じ Logtail 構成に従ってマシングループ内のすべてのマシンからログを収集します。 Log Service コンソールでは、マシングループの作成/削除、マシングループへのマシンの追加 / 削除など、マシングループを管理することもできます。 マシングループは、Windows マシンと Linux マシンの両方を含むことはできませんが、異なるバージョンの Windows マシンまたは異なるバージョンの Linux マシンを含むことができます。
- Logtailクライアント： Logtail は、ログを収集し、ログが収集されるサーバー上で実行されるエージェントです。 Logtail のインストール方法については、Logtail の Windowsへのイ

ンストールとLogtail の Linux へのインストールを参照してください。 サーバに Logtail をインストールした後、Logtail 設定を作成し、それをマシングループに適用します。

- Linuxでは、Logtail は`/ usr / local / ilogtail`ディレクトリに インストールされ、ilogtail で始まる 2 つの独立したプロセス（収集プロセスとデーモンプロセス）を開始します。 プログラム実行ログは`/ usr / local / ilogtail / ilogtail . LOG`です。
- Windowsでは、Logtail は`C :\ Program Files \ Alibaba \ Logtail`ディレクトリ（32 ビットシステム用） または`C :\ Program Files ( x86 )\ Alibaba \ Logtail`ディレクトリ（64 ビット用 システム）にインストールされます。 Windows 管理ツール> サービスに移動すると、 LogtailWorker と LogtailDaemon の 2 つの Windows サービスを表示できます。 LogtailDaemon はデーモンとして動作します。 プログラム実行ログは、 インストールディレクトリの`logtail_ *. log`です。

・ Logtail 構成： Logtail 構成は、Logtail を使用してログを収集するポリシーの集合です。 データソースや収集モードなどの Logtail パラメーターを設定することで、マシングループ内のすべてのマシンのログ収集ポリシーをカスタマイズできます。 Logtail 設定は、マシンからログの種類を収集し、収集したログを解析し、指定された Log Service の Logstore にそれらを送信するために使用されます。 Logstore がこの Logtail 構成を使用して収集されたログを受信できるようにするには、コンソール内の各 Logstore に Logtail 構成を追加できます。

基本機能

Logtail アクセスサービスは、以下の機能を提供します。

・ リアルタイムログ収集： Logtail は動的にログファイルを監視し、増分ログをリアルタイムで読み込み、解析します。 一般に、ログが生成されてから Log Service ログが送信されるまでの間に、3 秒未満の遅延があります。

> 注：
> Logtail は履歴データの収集をサポートしていません。 ログが読み込まれてからログが生成されるまでの間隔が 5 分を超えるログは破棄されます。

・ 自動ログローテーション処理：多くのアプリケーションは、ログファイルをファイルサイズまたは日付に従ってローテーションします。 ローテーション処理中に、元のログファイルの名前が変更され、ログの書き込み用に新しい空のログファイルが作成されます。 例えば、監視された *app . LOG* は *app . LOG . 1* と *app . LOG . 2* を生成するためにローテーションします。 *app . LOG* のように、収集されたログが書き込まれるファイルを指定する

ことができます。 Logtail は自動的にログローテーションプロセスを検出し、このプロセス中にログデータが失われないことを保証します。

- 複数の収集入力ソース：テキストログ以外にも、Loglog は syslog、HTTP、MySQL、binlog などの入力ソースをサポートしています。 詳細については、「ログサービスユーザーガイド」の「データソース」を参照してください。
- オープンソース収集エージェントとの互換性：Logstashの入力ソースは、LogstashやBeatsなどのオープンソースソフトウェアによって収集されたデータにすることができます。 詳細については、「ログサービスユーザーガイド」の「データソース」を参照してください。
- 収集例外の自動処理： Log Service エラー、ネットワーク対策、制限を超えるクォータなどの例外によりデータ送信が失敗した場合、Logtail は特定のシナリオに基づいて積極的に再試行します。 再試行が失敗した場合、Logtail はローカルキャッシュにデータを書き込み、その後、自動的にデータを再送信します。
- 柔軟な収集ポリシー設定： Logtail 設定を使用して、サーバからログを収集する方法を柔軟に指定することができます。 具体的には、実際のシナリオに基づいて、ワイルドカードと完全一致またはファジー一致をサポートするログディレクトリとファイルを選択できます。 ログ収集の抽出方法と抽出されたフィールドの名前をカスタマイズすることができます。 Log Service、正規表現を使用してログを抽出できます。 Log Service のログデータモデルでは、各ログに正確なタイムスタンプが必要です。 したがって、Logtail はカスタムログの時刻形式を提供し、さまざまな形式のログデータから必要なタイムスタンプ情報を抽出することができます。
- 収集構成の自動同期：一般的に、 Log Service コンソールで構成を作成または更新すると、Logtail は自動的にその構成を受け入れ、3 分以内に構成を有効にします。 構成の更新時に収集されたデータは失われません。
- クライアントの自動アップグレード： Logtail をサーバーに手動でインストールすると、 Log Service は自動的に Operation＆Maintenance（O＆M） およびLogtail のアップグレードを実行します。 Logtail をアップグレードしてもログデータは失われません。
- ステータス監視： Logtail クライアントがリソースを消費し過ぎてサービスに影響を与えないようにするため、Logtail クライアントは CPU とメモリの消費量をリアルタイムで監視します。Logtail クライアントは、リソースの使用量が制限を超えた場合に自動的に再起動され、マシン上の他の操作に影響を与えないようにします。 Logtail クライアントは、ネットワークトラフィックを積極的に制限して、過剰な帯域幅消費を防ぎます。

- シグネチャによるデータ送信：送信プロセス中にデータが改ざんされるのを防ぐため、Logtail クライアントは Alibaba Cloud AccessKey を取得し、送信されるすべてのログデータパケットに署名を提供します。

> 注：
> Alibaba Cloud AccessKey のセキュリティを維持するため、Logtail は HTTPS トンネルを使用してAccessKeyを取得します。

3.1.2 Logtail の収集プロセス

Logtail クライアントがサーバーからログを収集する処理には、ファイルのモニタリング、ファイルの読み取り、ログの処理、ファイルのフィルタリング、ログの集約、およびログの転送の 6 つのステップがあります。

サーバーに Logtail クライアントをインストールして Logtail 構成を設定すると、Logtail は Log Service へのログの収集を開始します。 ログ収集は次の処理を行います。

1. ファイルのモニタリング
2. ファイルの読取り
3. ログの処理
4. ログのフィルタリング
5. ログの集約
6. ログの送信

> 注：
> - 詳細は、 Alibaba Cloud Community をご参照ください。
> - マシングループに Logtail 構成を設定すると、そのマシングループ内のサーバーのログに変更のなかったものは、過去のファイルとみなされます。過去のファイルは収集されません。過去のログを収集するには、過去ログのインポートをご参照ください。

ファイルのモニタリング

サーバーに Logtail クライアントをインストールし、データソースに合わせて Logtail 構成を設定すると、その Logtail 構成は Logtail にすぐに送信されます。Logtail 構成に基づいてファイルのモニタリングが開始します。

1. 具体的には、設定されたログパスと最大モニタリングディレクトリの階層に合わせて、ファイル命名規則に準拠した指定ログディレクトリとファイルが階層ごとにスキャンされます。

   ログ収集の効率と安定性を確保するために、Logtail は収集ディレクトリ (つまり、Linux の inotify ディレクトリまたは Windows の ReadDirectoryChangesW ディレクトリ) のイベントモニタリングを登録し、定期的にポーリングします。
2. モニタリング結果に、ファイル命名規則に準拠した指定されたディレクトリに変更のなかったログファイルがあった場合、そのファイルは収集されません。変更のあったログファイルがあれば、収集プロセスが起動し、そのファイルは読み取られます。

ファイルの読み取り

変更のあったファイルを読み取ります。

1. 初めて読み取るファイルの場合、ファイルサイズがチェックされます。
   - ファイルサイズが 1 MB より小さい場合、ファイルは先頭から読み取られます。
   - ファイルサイズが 1 MB より大きい場合、ファイルの最後の 1 MB が読み取られます。
2. 以前に読み込まれたことのあるファイルの場合、最後のチェックポイントからファイルが読み込まれます。
3. 一度に 512 KB のみが読み取り可能です。したがって、ログは 512 KB 以下にします。

> 注：
> サーバー時間を変更した場合は、手動で Logtail を再起動する必要があります。再起動しないと、ログ生成時間が正確なものではなくなり、誤ってログが削除される可能性があります。

ログの処理

ログは行に分割また解析され、時間フィールド設定が適正が確認されます。

1. 行に分割

   Logtail 構成で改行正規表現を指定すると、改行設定に従ってログは複数の行に分割されます。各行は 1 つのログとして処理されます。改行の正規表現を指定しない場合、データブロックが 1 つのログとして処理されます。
2. 解析

   Logtail 構成に設定された正規表現、区切り文字、JSON 配列に基づいてログコンテンツが解析されます。

   > 注：

> 正規表現を複雑にし過ぎると、CPU 使用率が異常に高くなる可能性があります。したがって、正規表現を効率のよいものにされることを推奨します。

3. 解析失敗処理

   Logtail 構成で解析失敗ログの破棄機能の有効/無効によって、解析失敗ログは次のように処理されます。

   - 有効化の場合、ログは破棄され、エラーが報告されます。
   - 無効化の場合、元のログと併せて raw_log のキーとログ内容の値をアップロードする必要があります。

4. 時間フィールド設定

   - 本フィールドを設定しない場合、現在の解析時間がログ生成時間に適用されます。
   - 本フィールドを設定した場合、ログ生成時間は、次のとおりとなります。
     - 現在時刻から 12 時間経過していない場合、解析済み時刻フィールドから時間が抽出されます。
     - 現在時刻から 12 時間以上経過した場合、ログは破棄されエラーが報告されます。

ログのフィルタリング

Logtail 構成のフィルター設定に従ってログがフィルタリングされます。

- フィルターを設定した場合、ログはフィルタリングされず、直接ログを統計します。
- フィルターを設定しない場合、各ログのすべてのフィールドがスキャンおよび検証されます。
  - フィルター設定に適合するログが収集されます。フィルター設定のすべてのフィールドがログに含まれ、すべてのフィールドが設定条件を満たしている。
  - フィルター設定に適合しないログは収集されません。

ログの集約

ログデータは Log Service に送信されます。その前にログは一旦キャッシュされます。ネットワークリクエストの数を減らすために、必要なログは集約、パッケージ化されてから Log Service に送信されます。

キャッシュされたログが次のいずれかに適合する場合、そのログはパッケージ化されてから送信されます。

- ログの集約に 3 秒を超える
- 集約するログが 4,096 を超える
- ログが 512 KB を超える

ログの送信

Log Service に集約ログが送信されます。起動パラメータの構成の手順に従って、起動パラメータの `max_bytes_ per_sec` (ログ送信速度) および `send_reque st_concurr ency` (同時送信できるログの最大数) を調整します。調整した値を超えて送信されることがなくなります。

ログの送信に失敗した場合、エラーメッセージのとおり、タスクは自動的に再試行または終了します。

エラーメッセージ	説明	処理方法
エラーコード：401	Logtail クライアントにはデータを収集する権限がありません。	Logtail はログパッケージを削除
エラーコード：404	Logtail Config で指定されたプロジェクトまたは Logstore が存在しません。	Logtail はログパッケージを削除
エラーコード：403	シャードクォータが上限を超えています。	3 秒後に再試行
エラーコード：500	サーバーでエラーが発生しました。	3 秒後に再試行
ネットワークタイムアウト	ネットワーク接続エラーが発生しました。	3 秒後に再試行

3.1.3 Logtail 構成とログファイル

Logtail は、いくつかの設定ファイルに基づいて実行され、特定の情報を記録したログファイルを生成します。本ドキュメントでは、一般的な生成ファイルの概要とパスについて説明します。

構成ファイル：

- 起動構成ファイル (ilogtail_config.json)
- AliUid 構成ファイル
- カスタム ID ファイル (user_defined_id)
- Logtail 構成ファイル (user_log_config.json)

記録ファイル：

- AppInfo 記録ファイル (app_info.json)
- Logtail 操作ログファイル (ilogtail.LOG)
- Logtail プラグインログファイル (logtail_plugin.LOG)
- コンテナパスマッピングファイル (docker_path_config.json)

起動構成ファイル (ilogtail_config.json)

Logtail の実行パラメータが記載されており、表示および設定できます。JSON 形式のファイルです。

Logtail をインストールした場合、本ファイルに対して次の操作ができます。

- Logtail 実行パラメータの変更

  CPU 使用率や常駐メモリー使用率のしきい値といった一般的な設定を変更します。
- インストールコマンドが正しいことの確認

  `config_ser ver_addres s` と `data_serve r_list` は、インストール時に指定したものになります。指定されたパラメータのリージョンが Log Service のリージョンと異なる場合、またはアドレスにアクセスできない場合は、インストール中に誤ったパラメータまたはコマンドが使用されたことを意味します。Logtail によるログ収集はできないため、再インストールする必要があります。

注 :

- ファイルは有効な JSON 配列である必要があります。有効な JSON 配列でない場合、Logtail は起動されません。
- ファイルに加えた変更は、Logtail を再起動しない限り反映されません。

下表は、デフォルトの構成項目一覧です。その他の構成項目については、「起動パラメータを設定」をご参照ください。

表 3-1 : 起動構成ファイルのデフォルト構成項目

構成項目	説明
config_server_address	サーバーから取得する Logtail 構成に設定されているアドレス (インストール時に指定) アクセス可能なアドレスであり、Log Service と同一リージョンのアドレスを指定する必要があります。
data_server_list	データサーバーのアドレス (インストール時に指定) アクセス可能なアドレスであり、パラメータで指定したリージョンは Log Service と同一リージョンに属している必要があります。
cluster	リージョン名

構成項目	説明
endpoint	サービスのエンドポイント
cpu_usage_limit	CPU 使用率のしきい値 (コア数による)
mem_usage_limit	常駐メモリの使用率しきい値
max_bytes_per_sec	送信可能な Raw データの最大サイズ。データ送信速度が 20 Mbit/s を超える場合は適用されない。
process_thread_count	Logtail がログファイルにデータを書き込むために使用するスレッド数
send_request_concurrency	Logtail が同時非同期に送信できるデータパケットの数。 デフォルトでは、Logtail はデータパケットを非同期で送信します。 書き込み TPS が非常に高い場合は、値を大きくします。

ファイルアドレス：

- Linux: */ usr / local / ilogtail / ilogtail_c onfig . json*
- コンテナ: ファイルは Logtail コンテナに格納され、ファイルアドレスは環境変数 `ALIYUN_LOG TAIL_CONFI G` に設定されます。アドレスの確認には、 `Docker inspect $ { logtail_co ntainer_na me } | grep ALIYUN_LOG TAIL_CONFI G` を実行します (例: */ Etc / ilogtail / CONF / CN - Hangzhou / FIG*)。
- Windows
  - x64: *C :\ Program Files (x86)\ Alibaba \ Logtail \ ilogtail_c onfig . json*
  - x32: *C :\ Program Files \ Alibaba \ Logtail \ ilogtail_c onfig . json*

ファイル例

```
$ cat / usr / local / ilogtail / ilogtail_c onfig . json
{
    " config_ser ver_addres s " : " http :// logtail . cn - hangzhou
 - intranet . log . aliyuncs . com ",
    " data_serve r_list " :
    [
        {
            " cluster " : " ap - southeast - 2 ",
            " endpoint " : " cn - hangzhou - intranet . log . aliyuncs
 . com "
        }
    ],
    " cpu_usage_ limit " : 0 . 4 ,
    " mem_usage_ limit " : 100 ,
    " max_bytes_ per_sec " : 2097152 ,
```

```
    " process_th  read_count " :  1 ,
    " send_reque  st_concurr  ency " :  4 ,
    " streamlog_  open " :  false
}
```

AliUid 構成ファイル

Alibaba Cloud アカウントの AliUid が含まれます。AliUid は、サーバーにアクセスしてログを収集する権限のある Alibaba Cloud アカウントであることを証明するものです。別の Alibaba Cloud アカウントの ECS インスタンス、または、オンプレミス IDC のログを収集する場合は、AliUid 構成ファイルを手動で作成します。詳細は、Alibaba Cloud ECS インスタンス以外または他のアカウントの ECS インスタンスからログを収集するをご参照ください。

 注:

- オプションファイルであり、別の Alibaba Cloud アカウントの ECS インスタンス、または、オンプレミス IDC からログを収集する場合にのみ使用されます。
- ファイルには現在の Alibaba Cloud アカウントの AliUid のみを含めることができます。現在の Alibaba Cloud アカウントであっても、RAM ユーザーアカウントの AliUid を含めることはできません。
- ファイル名に拡張子を含めることはできません。
- Logtail に複数の AliUid 構成ファイルを設定できますが、Logtail コンテナに設定できる AliUid 構成ファイルは 1 つのみです。

ファイルアドレス

- Linux: `/ etc / ilogtail / users /`
- コンテナ: Logtail コンテナの環境変数 `ALIYUN_LOG  TAIL_USER_  ID` に設定します。ファイルの確認には、`docker  inspect  ${ logtail_co  ntainer_na  me } | grep  ALIYUN_LOG  TAIL_USER_  ID` を実行します。
- Windows: `C :\ LogtailDat  a \ users \`

ファイル例

```
$ ls  / etc / ilogtail / users /
 1559122535  028493  1329232535  020452
```

カスタム ID ファイル(user_defined_id)

マシングループにカスタム ID を設定した場合に使用されます。 詳細は、「マシングループの作成とカスタム ID 割り当て」をご参照ください。

 注:

- カスタム ID でマシングループの構成にのみ使用されるファイルです。
- マシングループに複数のカスタム ID を設定する場合は、区切り文字で区切ります。

ファイルアドレス

- Linux: `/etc/ilogtail/user_defined_id`
- コンテナ: Logtail コンテナの環境変数 `ALIYUN_LOG_TAIL_USER_DEFINED_ID` にファイルが設定されています。ファイルの確認には、`docker inspect ${logtail_container_name} | grep ALIYUN_LOG_TAIL_USER_DEFINED_ID` を実行します。
- Windows: `C:\LogtailData\user_defined_id`

ファイル例

```
$ cat /etc/ilogtail/user_defined_id
aliyun-ecs-rs1e16355
```

Logtail 構成ファイル (user_log_config.json)

Logtail のサーバーより取得された Logtail 構成情報が記載されます。JSON 形式のファイルであり、Logtail 構成に変更があると、更新されます。サーバーに Logtail 構成が送信されているかどうかを確認する際に使用します。本ファイルがあり、内容が最新であれば、Logtail 構成情報は正常にサーバーに送信されています。

注:

- キーまたはデータベースのパスワードを変更する必要がない限り、手動でファイルを変更されないことをお勧めします。
- チケットを起票して、サポートセンターにお問い合わせの際は、本ファイルを添付してください。

ファイルアドレス

- Linux: `/usr/local/ilogtail/user_log_config.json`
- コンテナ: `/usr/local/ilogtail/user_log_config.json`
- Windows
  - x64: `C:\Program Files (x86)\Alibaba\Logtail\user_log_config.json`
  - x32: `C:\Program Files\Alibaba\Logtail\user_log_config.json`

ファイル例

```
$ cat  / usr / local / ilogtail / user_log_c  onfig . json
{
   " metrics " : {
      "## 1 . 0 ## k8s - log - c12ba2028 ***** 939f0b $ app - java " :
 {
         " aliuid " : " 16542189 ***** 50 ",
         " category " : " app - java ",
         " create_tim  e " :  1534739165 ,
         " defaultEnd  point " : " cn - hangzhou - intranet . log .
 aliyuncs . com ",
         " delay_alar  m_bytes " :  0 ,
         " enable " :  true ,
         " enable_tag " :  true ,
         " filter_key  s " : [],
         " filter_reg  s " : [],
         " group_topi  c " : "",
         " local_stor  age " :  true ,
         " log_type " : " plugin ",
         " log_tz " : "",
         " max_send_r  ate " : - 1 ,
         " merge_type " : " topic ",
         " plugin " : {
            " inputs " : [
               {
                  " detail " : {
                     " IncludeEnv " : {
                        " aliyun_log  s_app - java " : " stdout "
                     },
                     " IncludeLab  le " : {
                        " io . kubernetes . container . name " : "
 java - log - demo - 2 ",
                        " io . kubernetes . pod . namespace " : "
 default "
                     },
                     " Stderr " :  true ,
                     " Stdout ":  true ,
                  },
                  " type " : " service_do  cker_stdou  t "
               }
            ]
         },
         " priority " :  0 ,
         " project_na  me " : " k8s - log - c12ba2028c *****
 ac1286939f  0b ",
         " raw_log " :  false ,
         " region ":  " cn - hangzhou "、
         " send_rate_  expire " :  0 ,
         " sensitive_  keys " : [],
         " tz_adjust " :  false ,
         " version " :  1
      }
   }
```

```
}
```

AppInfo ログファイル (app_info.json)

Logtail の起動時間や、Logtail の IP アドレスおよびホスト名を取得した時間といったあらゆる時間情報が含まれます。マシングループを作成して IP アドレス割り当てるを実施する際に、本ファイルに記載されている IP アドレスが必要になります。

通常、Logtail は次の規則に従ってサーバの IP アドレスを取得します。

- サーバーファイル/ `etc` / `hosts` で IP アドレスにホスト名を割り当てている場合、Logtail は自動的に IP アドレスを取得します。
- ホストに IP アドレスが割り当てられていない場合、Logtail はホストのプライマリ NIC の IP アドレスを自動的に取得します。

 注：

- Logtail の内部情報のみが含まれています。手動でファイルに変更を加えても、Logtail の基本設定は変更されません。
- ホスト名の変更といった、サーバーのネットワーク設定を変更した場合は、Logtail を再起動して新しい IP アドレスを取得します。

表 3-2 : フィールド説明

フィールド	フィールドの説明
UUID	サーバーのシリアル番号
hostname	ホスト名
instance_id	ランダムに生成された Logtail 固有の識別子
ip	Logtail の取得した IP アドレス。フィールドが空の場合、Logtail は IP アドレスを取得できず、正常に動作していないことになります。この場合は、サーバーに IP アドレスを設定して Logtail を再起動します。  注： マシングループの識別に IP アドレスを使用している場合、本フィールドはマシングループに設定されている IP アドレスになっているはずです。サーバーに誤った IP アドレスを設定していた場合には、マシングループの IP アドレスを修正し、1 分後に再度ご確認ください。

フィールド	フィールドの説明
logtail_version	Logtail クライアントのバージョン
os	OS のバージョン
update_time	Logtail の最終起動時間

ファイルアドレス

- **Linux:** */ usr / local / ilogtail / app_info . json*
- **コンテナ:** */ usr / local / ilogtail / app_info . json*
- **Windows**
  - **x64:** *C :\ Program Files (x86)\ Alibaba \ Logtail \ app_info . json*
  - **x32:** *C :\ Program Files \ Alibaba \ Logtail \ app_info . json*

ファイル例

```
$ cat  / usr / local / ilogtail / app_info . json
{
   " UUID " : "",
   " hostname " : " logtail - ds - slpn8 ",
   " instance_i  d " : " E5F93BC6 - B024 - 11E8 - 8831 - 0A58AC1403
 9E_172 . 20 . 3 . 158_153605  3315 ",
   " ip " : " 172 . 20 . 3 . 158 ",
   " logtail_ve  rsion " : " 0 . 16 . 13 ",
   " os " : " Linux ;  3 . 10 . 0 - 693 . 2 . 2 . el7 . x86_64 ; # 1
 SMP  Tue  Sep  12  22 : 26 : 13  UTC  2017 ;  x86_64 ",
   " update_tim  e " : " 2018 - 09 - 04  09 : 28 : 36 "
}
```

Logtail 操作ログファイル(ilogtail.LOG)

Logtail クライアントに関する実行中の情報が含まれています。ログは `INFO` 、 `WARN` 、 `ERROR` の順に重要度が高くなります。 `INFO` タイプのログは無視して構いません。

注:

- 収集のエラー診断を実施し、エラーおよび Logtail 操作ログをもとにトラブルシューティングします。
- Logtail 収集の例外に関して、チケットを起票して、サポートセンターにお問い合わせの際は、本ファイルを添付してください。

ファイルアドレス

- **Linux:** */ usr / local / ilogtail / ilogtail . LOG*

- コンテナ: */ usr / local / ilogtail / ilogtail . LOG*
- Windows
  - x64: *C :\ Program Files (x86)\ Alibaba \ Logtail \ logtail_ *. log*
  - x32: *C :\ Program Files \ Alibaba \ Logtail \ logtail_ *. log*

ファイル例

```
$ tail  / usr / local / ilogtail / ilogtail . LOG
[ 2018 - 09 - 13   01 : 13 : 59 . 024679 ]    [ INFO ]    [ 3155 ]
   [ build / release64 / sls / ilogtail / elogtail . cpp : 123 ]
 change   working   dir :/ usr / local / ilogtail /
[ 2018 - 09 - 13   01 : 13 : 59 . 025443 ]    [ INFO ]    [ 3155 ]
  [ build / release64 / sls / ilogtail / AppConfig . cpp : 175 ]
 load   logtail   config   file ,  path :/ etc / ilogtail / conf / ap
 - southeast - 2 / ilogtail_c  onfig . json
[ 2018 - 09 - 13   01 : 13 : 59 . 025460 ]    [ INFO ]    [ 3155 ]
  [ build / release64 / sls / ilogtail / AppConfig . cpp : 176 ]
 load   logtail   config   file ,  detail :{
   " config_ser  ver_addres  s " : " http :// logtail . ap - southeast
 - 2 - intranet . log . aliyuncs . com ",
   " data_serve  r_list " : [
      {
         " cluster " : " ap - southeast - 2 ",
         " endpoint " : " ap - southeast - 2 - intranet . log .
 aliyuncs . com "
      }
]
```

Logtail プラグインログファイル (logtail_plugin.LOG)

コンテナの標準出力、binlog、http プラグイン、その他プラグインに関する実行中の情報が含まれています。 ログは、`INFO` 、`WARN` 、`ERROR` の順に重要度が高くなります。`INFO` タイプのログは無視していただいて構いません。

CANAL_RUNTIME_ALARM といったプラグインエラーがあり、収集例外を診断を実施する際は、Logtail プラグインログのエラーをもとにトラブルシューティングします。

注 :

プラグイン例外に関して、チケットを起票して、サポートセンターにお問い合わせの際は、本ファイルを添付してください。

ファイルアドレス

- Linux: */ usr / local / ilogtail / logtail_pl ugin . LOG*
- コンテナ: */ usr / local / ilogtail / logtail_pl ugin . LOG*
- Windows: プラグインのログには対応していません。

ファイル例

```
$ tail  / usr / local / ilogtail / logtail_pl  ugin . LOG
 2018 - 09 - 13   02 : 55 : 30  [ INF ] [ docker_cen  ter . go : 525 ]
 [ func1 ]  docker   fetch   all : start
 2018 - 09 - 13   02 : 55 : 30  [ INF ] [ docker_cen  ter . go : 529 ]
 [ func1 ]  docker   fetch   all : stop
 2018 - 09 - 13   03 : 00 : 30  [ INF ] [ docker_cen  ter . go : 525 ]
 [ func1 ]  docker   fetch   all : start
 2018 - 09 - 13   03 : 00 : 30  [ INF ] [ docker_cen  ter . go : 529 ]
 [ func1 ]  docker   fetch   all : stop
 2018 - 09 - 13   03 : 03 : 26  [ INF ] [ log_file_r  eader . go :
 221 ] [ ReadOpen ] [## 1 . 0 ## sls - zc - test - hz - pub $ docker -
 stdout - config , k8s - stdout ]     open   file   for   read ,  file
 :/ logtail_ho  st / var / lib / docker / containers / 7f46afec6a
 14de39b59e  e9cdfbfa8a  70c2fa26f1  148b2e2f31  bd3410f5b2  d624
 / 7f46afec6a  14de39b59e  e9cdfbfa8a  70c2fa26f1  148b2e2f31
 bd3410f5b2  d624 - json . log      offset : 40379573      status :
 794354 - 64769 - 40379963
 2018 - 09 - 13   03 : 03 : 26  [ INF ] [ log_file_r  eader . go :
 221 ] [ ReadOpen ] [## 1 . 0 ## k8s - log - c12ba2028c  fb444238cd
 9ac1286939  f0b $ docker - stdout - config , k8s - stdout ]
 open   file   for   read ,  file :/ logtail_ho  st / var / lib /
 docker / containers / 7f46afec6a  14de39b59e  e9cdfbfa8a  70c2fa26f1
 148b2e2f31  bd3410f5b2  d624 / 7f46afec6a  14de39b59e  e9cdfbfa8a
 70c2fa26f1  148b2e2f31  bd3410f5b2  d624 - json . log      offset :
 40379573      status : 794354 - 64769 - 40379963
 2018 - 09 - 13   03 : 04 : 26  [ INF ] [ log_file_r  eader . go :
 308 ] [ CloseFile ] [## 1 . 0 ## sls - zc - test - hz - pub $ docker
 - stdout - config , k8s - stdout ]     close   file ,  reason : no
   read   timeout      file :/ logtail_ho  st / var / lib / docker
 / containers / 7f46afec6a  14de39b59e  e9cdfbfa8a  70c2fa26f1
 148b2e2f31  bd3410f5b2  d624 / 7f46afec6a  14de39b59e  e9cdfbfa8a
 70c2fa26f1  148b2e2f31  bd3410f5b2  d624 - json . log      offset :
 40379963      status : 794354 - 64769 - 40379963
 2018 - 09 - 13   03 : 04 : 27  [ INF ] [ log_file_r  eader . go :
 308 ] [ CloseFile ] [## 1 . 0 ## k8s - log - c12ba2028c  fb444238cd
 9ac1286939  f0b $ docker - stdout - config , k8s - stdout ]     close
   file ,  reason : no   read   timeout      file :/ logtail_ho  st /
 var / lib / docker / containers / 7f46afec6a  14de39b59e  e9cdfbfa8a
 70c2fa26f1  148b2e2f31  bd3410f5b2  d624 / 7f46afec6a  14de39b59e
 e9cdfbfa8a  70c2fa26f1  148b2e2f31  bd3410f5b2  d624 - json . log
   offset : 40379963      status : 794354 - 64769 - 40379963
 2018 - 09 - 13   03 : 05 : 30  [ INF ] [ docker_cen  ter . go : 525 ]
 [ func1 ]  docker   fetch   all : start
 2018 - 09 - 13   03 : 05 : 30  [ INF ] [ docker_cen  ter . go : 529 ]
 [ func1 ]  docker   fetch   all : stop
```

コンテナパスマッピングファイル (docker_path_config.json)

コンテナファイルが収集されるときにのみ自動生成されるファイルです。コンテナファイルのパスと実際のファイルパスとのマッピングが記録されます。JSON 形式のファイルです。

収集のエラー診断を実施する際、DOCKER_FILE_MAPPING_ALARM エラー報告がある場合、Logtail は Docker ファイルマッピングに追加することができません。エラーのトラブルシューティングに本ファイルを使用します。

注:

- 情報のみが含まれているファイルです。ファイルに変更を加えても何の効果もありません。ファイルが削除された場合には自動的に再生成されます。 サービスには影響ありません。
- コンテナログの収集に例外が発生に関し、チケットを起票して、サポートセンターにお問い合わせの際は、本ファイルを添付してください。

ファイルアドレス

/ usr / local / ilogtail / ilogtail_c onfig . json

ファイル例

```
$ cat  / usr / local / ilogtail / docker_pat  h_config . json
{
   " detail " : [
      {
         " config_nam  e " : "## 1 . 0 ## k8s - log - c12ba2028c
 fb444238cd  9ac1286939  f0b $ nginx ",
         " container_  id " : " df19c06e85  4a0725ea7f  ca7e0378b0
 450f7bd312  2f94fe3e75  4d8483fd33  0d10 ",
         " params " : "{\ n    \" ID \" : \" df19c06e85  4a0725ea7f
 ca7e0378b0  450f7bd312  2f94fe3e75  4d8483fd33  0d10 \",\ n
 \" Path \" : \"/ logtail_ho  st / var / lib / docker / overlay2
 / 947db34669  5a1f65e63e  582ecfd10a  e1f57019a1  b99260b6c8
 3d00fcd189  2874 / diff / var / log \",\ n    \" Tags \" : [\ n
      \" nginx - type \",\ n       \" access - log \",\ n       \"
 _image_nam  e_ \",\ n       \" registry . cn - hangzhou . aliyuncs
 . com / log - service / docker - log - test : latest \",\ n       \"
 _container  _name_ \",\ n       \" nginx - log - demo \",\ n       \"
 _pod_name_ \",\ n       \" nginx - log - demo - h2lzc \",\ n       \"
 _namespace  _ \",\ n       \" default \",\ n       \" _pod_uid_ \",\
 n       \" 87e56ac3 - b65b - 11e8 - b172 - 00163f0086  85 \",\ n
   \" _container  _ip_ \",\ n       \" 172 . 20 . 4 . 224 \",\ n
 \" purpose \",\ n       \" test \"\ n    ]\ n }\ n "
      }
   ],
   " version " : " 0 . 1 . 0 "
}
```

3.2 ネットワークタイプの選択

収集されたログデータは、Alibaba Cloud イントラネット、インターネット、またはGlobal Accelerationを介して Log Service に送信されます。

ネットワークタイプ

- インターネット: インターネットを介したログデータの送信は、ネットワーク帯域幅の制限を受ける可能性があります。 また、ジッタ、遅延、パケット損失といったネットワークの問題により、データ転送の速度と安定性が影響を受ける可能性もあります。

- Alibaba Cloud イントラネット: Alibaba Cloud イントラネットの共有帯域幅はギガビットレベルであるため、インターネットよりも安定しており、ログデータの送信は高速です。 イントラネットには、Virtual Private Cloud (VPC) 環境とクラシックネットワーク環境があります。
- Global Acceleration: Alibaba Cloud の Content Delivery Network (CDN) のエッジノードを使用したネットワークサービスであり、ログ収集が高速化されます。インターネットと比較して、Global Acceleration はより低遅延に高い安定性を確保できます。

ネットワークタイプの選択

- イントラネット

  サーバータイプ、また、サーバーと Log Service プロジェクトが同じリージョンにあるかどうかで、ログデータが Alibaba Cloud イントラネットを介して送信されるかどうかが決まります。Alibaba Cloud イントラネットを介してログデータを送信できるのは次の場合のみです。

  - アカウントの ECS インスタンスおよび Log Service プロジェクトのリージョンが同じである
  - 別アカウントの ECS インスタンスおよび Log Service プロジェクトのリージョンが同じである

  したがって、Log Service プロジェクトは、ECS インスタンスと同じリージョンに作成してログ収集されることを推奨します。ECS インスタンスのログデータは、インターネットの帯域幅を使用せずに、Alibaba Cloud イントラネットを介して Log Service に書き込まれます。

   注:
  サーバーに Logtail クライアントをインストールする際、Log Service プロジェクトと同じリージョンを選択します。リージョンが異なる場合、ログデータは収集されません。

- Global Acceleration

  海外のオンプレミスサーバー、または、海外の他クラウドベンダーのサーバーからインターネット経由でデータを送信すると、遅延が大きくなり、送信が不安定になるといった問題が発生します。そういった場合に、Global Accelerationを採用します。 Global Accelerationは、Alibaba Cloud CDN のエッジノードを使用するため、ログ収集が高速化されます。インターネット経由でのデータ転送と比較して、Global Acceleration は最小限の転送遅延でより安定したネットワークとなります。

・ インターネット

次の場合には、インターネットを選択することをお勧めします。

- サーバーは ECS インスタンスであるが、Log Service プロジェクトと同じリージョンではない
- オンプレミスサーバーや他ベンダーのサーバーである

サーバータイプ	プロジェクトと同じリージョン	AliUid を設定する必要がある	ネットワークタイプ
使用中のアカウントの ECS インスタンス	はい	いいえ	Alibaba Cloud イントラネット
	いいえ	いいえ	インターネット、または Global Acceleration
その他アカウントの ECS インスタンス	はい	はい	Alibaba Cloud イントラネット
	いいえ	はい	インターネット、または Global Acceleration
他クラウドベンダーまたはオンプレミスサーバー	-	はい	インターネット、または Global Acceleration

 注：

Log Service は、別のアカウントや他のサーバーの ECS インスタンスの所有者情報を取得できません。 そのため、Logtail クライアントをインストール後に、各サーバに AliUid を設定します。 AliUid の設定されていないサーバーとは正常なハートビートが取れず、ログを収集できません。 詳細は、「非 Alibaba Cloud ECS インスタンスまたはAlibaba Cloud 別アカウントの ECS インスタンスのログ収集」をご参照ください。

ネットワークタイプの選択例

次の例では、いくつかの一般的なシナリオで適切なネットワークを選択する方法について説明します。

 注：

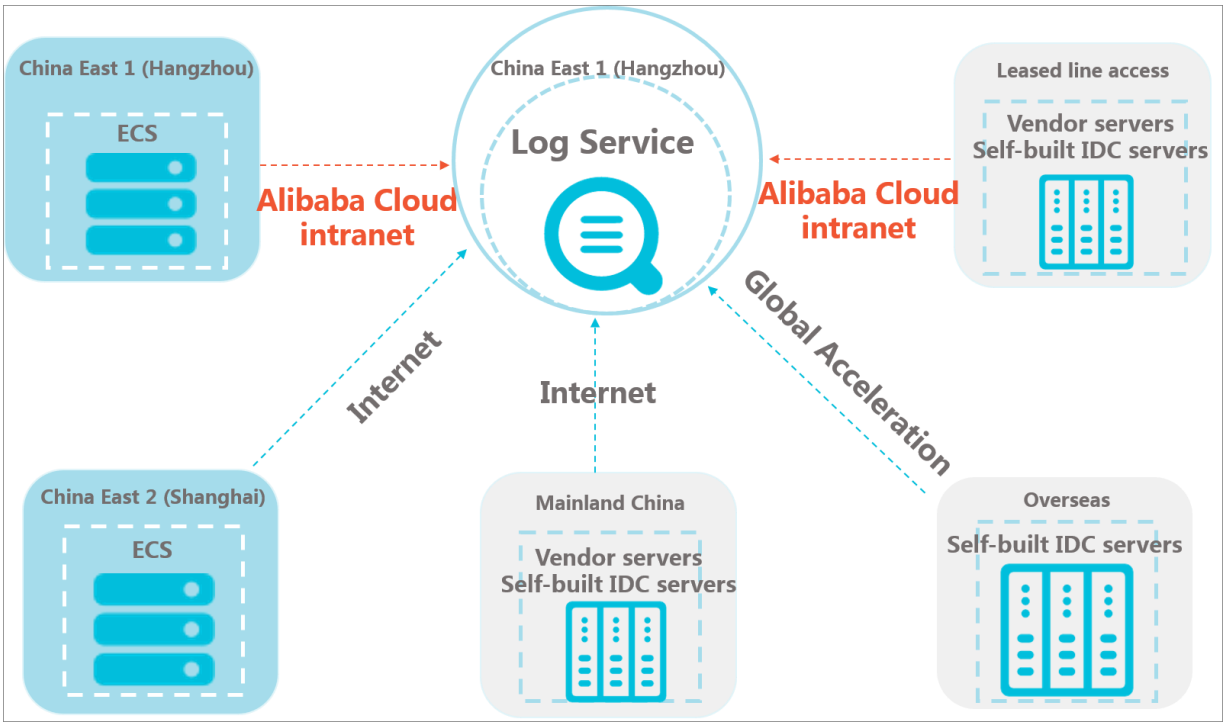
Global Acceleration シナリオでは、データ収集の速度と信頼性が極めて重要になります。Log Service プロジェクトを香港リージョンに作成しても、接続先のオンプレミスサーバーは世界中

に散らばっています。そのため、このシナリオで Logtail クライアントをインストールする際には、香港リージョンの Global Accelleration ネットワークタイプを選択することをお勧めします。Global Acceleration の場合、インターネットよりも高い安定性とパフォーマンスでログデータを送信されます。

シナリオ	Log Service プロジェクトのリージョン	サーバータイプ	ECS インスタンスのリージョン	Logtail クライアントインストール時に選択したリージョン	ネットワークタイプ	AliUid を設定する必要がある
ECS とプロジェクトが同じリージョンにある	中国 (杭州)	現在のアカウントの ECS	中国 (杭州)	中国 (杭州)	イントラネット	いいえ
ECS とプロジェクトのリージョンが異なる	中国 (上海)	現在のアカウントの ECS	中国 (北京)	中国 (北京)	インターネット	いいえ
その他アカウント	中国 (上海)	その他アカウントの ECS インスタンス	中国 (北京)	中国 (北京)	インターネット	はい
オンプレミスサーバー	中国 (深セン)	オンプレミス	-	中国 (深セン)	インターネット	はい

シナリオ	Log Service プロジェクトのリージョン	サーバータイプ	ECS インスタンスのリージョン	Logtail クライアントインストール時に選択したリージョン	ネットワークタイプ	AliUid を設定する必要がある
Global Acceleration	中国 (香港)	オンプレミス	-	中国 (香港)	Global Acceleration	はい

図 3-3 : ネットワークタイプの選択例



クラシックネットワークを VPC に切り替えてから構成を更新

Logtail クライアントをインストール後、ECS インスタンスがクラシックネットワークから VPC に切り替えられた場合は、ネットワーク構成を更新する必要があります。 構成を更新するには、次の手順に従います。

1. Logtail クライアントを管理者として再起動します。

   - Linux

     ```
     sudo  / etc / init . d / ilogtaild   stop
     ```

```
sudo / etc / init . d / ilogtaild start
```

- Windows

  コントロールパネルの管理ツールをダブルクリックします。サービスをクリックし、*LogtailWor ker* を右クリックして再起動を選択します。

2. マシングループ構成を更新します。

   - カスタム ID

     マシングループの識別にカスタム ID を定義している場合は、マシングループ構成を変更することなく VPC ネットワークをそのまま使用できます。

   - IP アドレス

     マシングループの識別に ECS インスタンスの IP アドレスを使用している場合は、元の IP アドレスを再起動した Logtail クライアントより新たに取得された IP アドレスに置き換える必要があります (*app_info . json* ファイルの IP アドレスフィールド)。

     app_info . json のファイルパス

     - Linux: */ usr / local / ilogtail / app_info . json*
     - Windows x64: *C :\ Program Files (x86)\ Alibaba \ Logtail \ app_info . json*
     - Windows x32: *C :\ Program Files \ Alibaba \ Logtail \ app_info . json*

3.3 インストール

3.3.1 Logtail の Linux へのインストール

対応システム

Logtail は、次のリリースの Linux x86-64 (64 ビット) サーバーをサポートしています。

- Aliyun Linux
- Ubuntu
- Debian
- CentOS
- OpenSUSE

Logtail のインストール

Logtail を上書きモードでインストールします。 以前の Logtail がインストールされている場合、インストーラはその Logtail をアンインストールし、`/ usr / local / ilogtail` ディレクトリを削除してから、Logtail をインストールします。 デフォルトでは、インストール後に Logtail が起動し、スタートアップに登録されます。

マシンのネットワーク環境と Log Service のリージョンに基づいて、インストーラをダウンロードします。 インストールごとに異なるパラメータを選択します。

このドキュメントのインストール方法に従って、Logtail をインストールします。 インストールが失敗した場合は、チケットを起票し、サポートセンターへお問い合わせください。

インストール方法

Logtail をインストールするには、インストールスクリプトをダウンロードして実行します。 リージョンとネットワークタイプに基づいて、インストールパラメーターを選択する必要があります。

インストールパラメーター

 注：
Docker または Kubernetes に Logtail をインストールする場合、`${ yourregion _name }` に、次の表のパラメーターを設定します。 対応するインストールステートメントを直接コピーします。

それぞれのリージョンとネットワークタイプに対応するインストールパラメーターは、次のとおりです (対応するインストールステートメントを直接コピーすることを推奨します)。

リージョン	クラシックネットワークと VPC	インターネット (自己構築型 IDC)
中国 (青島)	cn-qingdao	cn-qingdao-internet
中国 (北京)	cn-beijing	cn-beijing-internet
中国 (杭州)	cn-hangzhou	cn-hangzhou-internet
中国 (上海)	cn-shanghai	cn-shanghai-internet
中国 (張家口)	cn-zhangjiakou	cn-zhangjiakou-internet
中国 (フフホト)	cn-huhehaote	cn-huhehaote-internet
中国 (深セン)	cn-shenzhen	cn-shenzhen-internet
中国 (成都)	cn-chengdu	cn-chengdu-internet

リージョン	クラシックネットワークと VPC	インターネット (自己構築型 IDC)
中国 (香港)	cn-hongkong	cn-hongkong-internet
米国 (シリコンバレー)	us-west-1	us-west-1-internet
米国(バージニア)	us-east-1	us-east-1-internet
シンガポール	Ap-southeast-1	ap-southeast-1-internet
オーストラリア (シドニー)	ap-southeast-2	ap-southeast-2-internet
マレーシア(クアラルンプール)	ap-southeast-3	ap-southeast-3-internet
インドネシア (ジャカルタ)	ap-southeast-5	ap-southeast-5-internet
インド (ムンバイ)	ap-south-1	ap-south-1-internet
日本 (東京)	ap-northeast-1	ap-northeast-1-internet
ドイツ (フランクフルト)	eu-central-1	eu-central-1-internet
UAD (ドバイ)	me-east-1	me-east-1-internet
イギリス (ロンドン)	eu-west-1	eu-west-1-internet
中国 (杭州) (financial cloud)	cn-hangzhou-finance	None
中国 (上海) (financial cloud)	cn-shanghai-finance	None
中国 (深セン) (financial cloud)	cn-shenzhen-finance	None

ECS (クラシックネットワーク、VPC)

ECS 上のデータは、Alibaba Cloud イントラネットを介して Log Service に書き込まれます。インターネット帯域幅は使用しません。

リージョンパラメーターの自動選択：

ECS が配置されているリージョンまたは ECS の ID を特定できない場合、Logtail インストーラの auto パラメーターを使用して、Logtail をインストールすることができます。 このパラメーターを指定すると、Logtail インストーラはサーバーを介してメタデータを取得し、リージョンを自動的に決定します。

手順は次のとおりです。

1. パブリックネットワーク経由で Logtail インストーラを入手します。 このステップでは、パブリックネットワークにアクセスし、約 10 KB のパブリックネットワークトラフィックを使用します。

```
$ wget http :// logtail - release - cn - hangzhou . oss - cn -
 hangzhou . aliyuncs . com / linux64 / logtail . sh - O logtail .
 sh ; chmod 755 logtail . sh
```

2. インストールパラメーター auto を使用します。 このステップでは、対応するリージョンのインストールプログラムを自動的にダウンロードします。パブリックネットワークトラフィックは使用しません。

```
$ ./ logtail . sh install auto
```

region パラメータを手動で選択する

auto パラメータを指定したインストールが失敗した場合は、手動でインストールすることができます。 次のコマンドを実行して、直接インストールしてください。

注：

- 次のコマンドの `${ your_regio n_name }` を、ECS が配置されているリージョン (`cn - beijing` や `cn - hangzhou` など) に置き換えます。
- 内部ネットワークを介してインストーラを取得します。パブリックネットワークトラフィックを使用しません。

```
wget http://logtail-
release-${your_region_name}.oss-${your_region_name}-
internal.aliyuncs.com/linux64/logtail.sh -O logtail.sh; chmod 755
 logtail.sh; ./logtail.sh install ${your_region_name}
```

ECS が配置されているリージョンに応じて、次のいずれかのコマンドを実行し、インストールを実行することもできます。

- 中国 (北京)

```
wget http :// logtail - release - cn - beijing . oss - cn -
beijing - internal . aliyuncs . com / linux64 / logtail . sh - O
  logtail . sh ; chmod 755 logtail . sh ; ./ logtail . sh
install cn - beijing
```

- 中国 (青島)

```
wget http :// logtail - release - cn - qingdao . oss - cn -
qingdao - internal . aliyuncs . com / linux64 / logtail . sh - O
```

```
  logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - qingdao
```

- **中国 (杭州)**

```
wget   http :// logtail - release - cn - hangzhou . oss - cn -
hangzhou - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - hangzhou
```

- **中国 (上海)**

```
wget   http :// logtail - release - cn - shanghai . oss - cn -
shanghai - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - shanghai
```

- **中国 (深セン)**

```
wget   http :// logtail - release - cn - shenzhen . oss - cn -
shenzhen - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - shenzhen
```

- **中国 (張家口)**

```
wget   http :// logtail - release - cn - zhangjiako  u . oss - cn -
zhangjiako  u - internal . aliyuncs . com / linux64 / logtail . sh
- O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
  install   cn - zhangjiako  u
```

- **中国 (フフホト)**

```
wget   http :// logtail - release - cn - huhehaote . oss - cn -
huhehaote - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - huhehaote
```

- **中国 (成都)**

```
wget   http :// logtail - release - cn - chengdu . oss - cn -
chengdu - internal . aliyuncs . com / linux64 / logtail . sh  - O
  logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - chengdu
```

- **中国 (香港)**

```
wget   http :// logtail - release - cn - hongkong . oss - cn -
hongkong - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - hongkong
```

- **米国 (シリコンバレー)**

```
wget   http :// logtail - release - us - west - 1 . oss - us -
west - 1 - internal . aliyuncs . com / linux64 / logtail . sh  -
```

```
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   us - west - 1
```

- 米国 (バージニア)

```
wget   http :// logtail - release - us - east - 1 . oss - us -
east - 1 - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   us - east - 1
```

- シンガポール

```
wget   http :// logtail - release - ap - southeast - 1 . oss - ap -
southeast - 1 - internal . aliyuncs . com / linux64 / logtail . sh
- O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
  install   ap - southeast - 1
```

- オーストラリア (シドニー)

```
wget   http :// logtail - release - ap - southeast - 2 . oss - ap -
southeast - 2 - internal . aliyuncs . com / linux64 / logtail . sh
- O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
  install   ap - southeast - 2
```

- マレーシア (クアラルンプール)

```
wget   http :// logtail - release - ap - southeast - 3 . oss - ap -
southeast - 3 - internal . aliyuncs . com / linux64 / logtail . sh
- O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
  install   ap - southeast - 3
```

- インドネシア (ジャカルタ)

```
wget   http :// logtail - release - ap - southeast - 5 . oss - ap -
southeast - 5 - internal . aliyuncs . com / linux64 / logtail . sh
- O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
  install   ap - southeast - 5
```

- 日本 (東京)

```
wget   http :// logtail - release - ap - northeast - 1 . oss - ap -
northeast - 1 - internal . aliyuncs . com / linux64 / logtail . sh
- O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
  install   ap - northeast - 1
```

- インド (ムンバイ)

```
wget   http :// logtail - release - ap - south - 1 . oss - ap -
south - 1 - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   ap - south - 1
```

- ドイツ (フランクフルト)

```
wget   http :// logtail - release - eu - central - 1 . oss - eu -
central - 1 - internal . aliyuncs . com / linux64 / logtail . sh  -
```

```
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   eu - central - 1
```

- UAE (ドバイ)

```
wget   http :// logtail - release - me - east - 1 . oss - me -
east - 1 - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   me - east - 1
```

- イギリス (ロンドン)

```
wget   http :// logtail - release - eu - west - 1 . oss - eu -
west - 1 - internal . aliyuncs . com / linux64 / logtail . sh  -
O   logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   eu - west - 1
```

インターネット (自己構築型 IDC、または他のクラウドホスト)

データは、インターネット帯域幅を専有するインターネットを介して Log Service に書き込まれ、Alibaba Cloud 以外の仮想マシンや他の IDC に適用されます。

> 注:
> Log Service は Alibaba Cloud 以外のマシンの所有者情報を取得できません。 したがって、Logtail をインストールした後、手動でユーザー ID を設定する必要があります。 詳細：Alibaba Cloud ECS インスタンス以外または他のアカウントの ECS インスタンスからログを収集する。 ユーザー ID を設定しないと、 Logtail は異常と判断し、ログを収集することができません。

次のコマンドの `${ your_regio  n_name }` を、Log Service プロジェクトが配置されているリージョン (`cn - beijing` や `cn - hangzhou` など) に置き換えます。

```
wget http://logtail-
release-${your_region_name}.oss-${your_region_name}.aliyuncs.com/
linux64/logtail.sh -O logtail.sh; chmod 755 logtail.sh; ./logtail.sh
 install ${your_region_name}-internet
```

Log Service プロジェクトが配置されているリージョンに応じて、次のいずれかのコマンドを実行し、インストールを実行することもできます。

- 中国 (北京)

```
wget   http :// logtail - release - cn - beijing . oss - cn -
beijing . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn -
beijing - internet
```

- 中国 (青島)

```
wget   http :// logtail - release - cn - qingdao . oss - cn -
qingdao . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
```

```
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn -
qingdao - internet
```

- 中国 (杭州)

```
wget   http :// logtail - release - cn - hangzhou . oss - cn -
hangzhou . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn -
hangzhou - internet
```

- 中国 (上海 (中国)

```
wget   http :// logtail - release - cn - shanghai . oss - cn -
shanghai . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn -
shanghai - internet
```

- 中国 (深セン)

```
wget   http :// logtail - release - cn - shenzhen . oss - cn -
shenzhen . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn -
shenzhen - internet
```

- 中国 (張家口)

```
wget   http :// logtail - release - cn - zhangjiako  u . oss -
cn - zhangjiako  u . aliyuncs . com / linux64 / logtail . sh  - O
  logtail . sh ;  chmod   755   logtail . sh ; ./ logtail . sh
install   cn - zhangjiako  u - internet
```

- 中国 (フフホト)

```
wget   http :// logtail - release - cn - huhehaote . oss - cn -
huhehaote . aliyuncs . com / linux64 / logtail . sh  - O   logtail
. sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn
- huhehaote - internet
```

- 中国 (成都)

```
wget   http :// logtail - release - cn - chengdu . oss - cn -
chengdu . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn -
chengdu - internet
```

- 中国 (香港)

```
wget   http :// logtail - release - cn - hongkong . oss - cn -
hongkong . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   install   cn -
hongkong - internet
```

- 米国 (シリコンバレー)

```
wget   http :// logtail - release - us - west - 1 . oss - us - west
- 1 . aliyuncs . com / linux64 / logtail . sh  - O   logtail . sh
```

```
; chmod 755 logtail . sh ; ./ logtail . sh install us -
west - 1 - internet
```

- 米国 (バージニア)

```
wget http :// logtail - release - us - east - 1 . oss - us - east
- 1 . aliyuncs . com / linux64 / logtail . sh - O logtail . sh
; chmod 755 logtail . sh ; ./ logtail . sh install us -
east - 1 - internet
```

- シンガポール

```
wget http :// logtail - release . oss - cn - hangzhou . aliyuncs
. com / linux64 / logtail . sh - O logtail . sh ; chmod 755
logtail . sh ; sh logtail . sh install ap - southeast - 1 -
internet
```

- オーストラリア (シドニー)

```
wget http :// logtail - release - ap - southeast - 2 . oss -
ap - southeast - 2 . aliyuncs . com / linux64 / logtail . sh - O
logtail . sh ; chmod 755 logtail . sh ; ./ logtail . sh
install ap - southeast - 2 - internet
```

- マレーシア (クアラルンプール)

```
wget http :// logtail - release - ap - southeast - 3 . oss -
ap - southeast - 3 . aliyuncs . com / linux64 / logtail . sh - O
logtail . sh ; chmod 755 logtail . sh ; ./ logtail . sh
install ap - southeast - 3 - internet
```

- インドネシア (ジャカルタ)

```
wget http :// logtail - release - ap - southeast - 5 . oss -
ap - southeast - 5 . aliyuncs . com / linux64 / logtail . sh - O
logtail . sh ; chmod 755 logtail . sh ; ./ logtail . sh
install ap - southeast - 5 - internet
```

- 日本 (東京)

```
wget http :// logtail - release - ap - northeast - 1 . oss -
ap - northeast - 1 . aliyuncs . com / linux64 / logtail . sh - O
logtail . sh ; chmod 755 logtail . sh ; ./ logtail . sh
install ap - northeast - 1 - internet
```

- ドイツ (フランクフルト)

```
wget http :// logtail - release - eu - central - 1 . oss - eu
- central - 1 . aliyuncs . com / linux64 / logtail . sh - O
logtail . sh ; chmod 755 logtail . sh ; ./ logtail . sh
install eu - central - 1 - internet
```

- UAD (ドバイ)

```
wget http :// logtail - release - me - east - 1 . oss - me - east
- 1 . aliyuncs . com / linux64 / logtail . sh - O logtail . sh
```

```
; chmod 755 logtail . sh ; ./ logtail . sh install me -
east - 1 - internet
```

- インド (ムンバイ)

```
wget http :// logtail - release - ap - south - 1 . oss - ap -
south - 1 . aliyuncs . com / linux64 / logtail . sh - O logtail
. sh ; chmod 755 logtail . sh ; ./ logtail . sh install ap
- south - 1 - internet
```

- イギリス (ロンドン)

```
wget http :// logtail - release - eu - west - 1 . oss - eu - west
- 1 . aliyuncs . com / linux64 / logtail . sh - O logtail . sh
; chmod 755 logtail . sh ; ./ logtail . sh install eu -
west - 1 - internet
```

Logtail バージョンの表示

Logtail のバージョン情報は、`/ usr / local / ilogtail / app_info . json` ファイルの `logtail_ve rsion` フィールドに記録されています。例：

```
$ cat / usr / local / ilogtail / app_info . json
{
   " UUID " : " 0DF18E97 - 0F2D - 486F - B77F -*********",
   " hostname " : " david *******",
   " instance_i d " : " F4FAFADA - F1D7 - 11E7 - 846C - 00163E3034
 9E_ ********* _151512954 8 ",
   " ip " : "**********",
   " logtail_ve rsion " : " 0 . 16 . 0 ",
   " os " : " Linux ; 2 . 6 . 32 - 220 . 23 . 2 . ali1113 . el5 .
 x86_64 ; # 1 SMP Thu Jul 4 20 : 09 : 15 CST 2013 ;
 x86_64 ",
   " update_tim e " : " 2018 - 01 - 05 13 : 19 : 08 "
}
```

Logtail の更新

Logtail を更新する手順は、Logtail をインストールする手順と同じです。Logtail を更新すると、Logtail が自動的にアンインストールされ、次に Logtail の最新バージョンがインストールされます。

手動による Logtail の起動と停止

- Logtail の起動

  管理者として次のコマンドを実行します。

```
/ etc / init . d / ilogtaild   start
```

- Logtail の停止

  管理者として次のコマンドを実行します。

```
/ etc / init . d / ilogtaild   stop .
```

Logtail のアンインストール

インストーラlogtail.shをダウンロードします。 詳細については、Logtail のインストールをご参照ください。 シェルモードで次のコマンドを管理者として実行します。

```
wget   http :// logtail - release - cn - hangzhou . oss - cn -
hangzhou . aliyuncs . com / linux64 / logtail . sh  - O   logtail .
sh ;  chmod   755   logtail . sh ; ./ logtail . sh   uninstall
```

3.3.2 Logtail の Windowsへのインストール

対応システム

Logtailは、Windows Server 2003 (32/64 ビット) 及びそれ以降のバージョンをサポートしています。を参照ごください：

- Windows 7 (クライアント) 32bit
- Windows 7 (クライアント) 64bit
- Windows Server 2003 32bit
- Windows Server 2003 64bit
- Windows Server 2008 32bit
- Windows Server 2008 64bit
- Windows Server 2012 64bit

Logtailのインストール

1. インストールパッケージをダウンロードします。

   インストールパッケージは、こちらからダウンロードできます。
2. 現行ディレクトリにlogtail.zipを解凍します。

3. マシンのネットワーク環境とLog Serviceのリージョンに基づいて、Logtailをインストールします。

   Windows PowerShellかcmd.exeを起動して、`logtail_in staller`ディレクトリに移動します。 マシンのネットワーク環境とリージョンに基づいて、対応するコマンドを実行します。

   インストールコマンド：

リージョン	クラシックネットワークとVPC	インターネット(自己構築型IDC)	Global Acceleration
中国 (青島)	.\logtail_installer.exe install cn-qingdao	.\logtail_installer.exe install cn-qingdao-internet	.\logtail_installer.exe install cn-qingdao-acceleration
中国 (北京)	.\logtail_installer.exe install cn-beijing	.\logtail_installer.exe install cn-beijing-internet	.\logtail_installer.exe install cn-beijing-acceleration
中国 (張家口)	.\logtail_installer.exe install cn-zhangjiakou	.\logtail_installer.exe install cn-zhangjiakou-internet	.\logtail_installer.exe install cn-zhangjiakou-acceleration
中国 (フフホト)	.\logtail_installer.exe install cn-huhehaote	.\logtail_installer.exe install cn-huhehaoteinternet	.\logtail_installer.exe install cn-huhehaote-acceleration
中国 (杭州)	.\logtail_installer.exe install cn-hangzhou	.\logtail_installer.exe install cn-hangzhou-internet	.\logtail_installer.exe install cn-hangzhou-acceleration
中国 (上海)	.\logtail_installer.exe install cn-shanghai	.\logtail_installer.exe install cn-shanghai-internet	.\logtail_installer.exe install cn-shanghai-acceleration
中国 (深圳)	.\logtail_installer.exe install cn-shenzhen	.\logtail_installer.exe install cn-shenzhen-internet	.\logtail_installer.exe install cn-shenzhen-acceleration

リージョン	クラシックネットワークとVPC	インターネット(自己構築型IDC)	Global Acceleration
中国 (成都)	.\logtail_installer.exe install cn-chengdu	.\logtail_installer.exe install cn-chengdu-internet	.\logtail_installer.exe install cn-chengdu-acceleration
中国 (香港)	.\logtail_installer.exe install cn-hongkong	.\logtail_installer.exe install cn-hongkong-internet	.\logtail_installer.exe install cn-hongkong-acceleration
米国（シリコンバレー）	.\logtail_installer.exe install us-west-1	.\logtail_installer.exe install us-west-1-internet	.\logtail_installer.exe install us-west-1-acceleration
米国 (バージニア)	.\logtail_installer.exe install us-east-1	.\logtail_installer.exe install us-east-1-internet	.\logtail_installer.exe install us-east-1-acceleration
シンガポール	.\logtail_installer.exe install ap-southeast-1	.\logtail_installer.exe install ap-southeast-1-internet	.\logtail_installer.exe install ap-southeast-1-acceleration
オーストラリア (シドニー)	.\logtail_installer.exe install ap-southeast-2	.\logtail_installer.exe install ap-southeast-2-internet	.\logtail_installer.exe install ap-southeast-2-acceleration
マレーシア (クアラルンプール)	.\logtail_installer.exe install ap-southeast-3	.\logtail_installer.exe install ap-southeast-3-internet	.\logtail_installer.exe install ap-southeast-3-acceleration
インドネシア (ジャカルタ)	.\logtail_installer.exe install ap-southeast-5	.\logtail_installer.exe install ap-southeast-5-internet	.\logtail_installer.exe install ap-southeast-5-acceleration
インド (ムンバイ)	.\logtail_installer.exe install ap-south-1	.\logtail_installer.exe install ap-south-1-internet	.\logtail_installer.exe install ap-south-1-acceleration
日本 (日本)	.\logtail_installer.exe install ap-northeast-1	.\logtail_installer.exe install ap-northeast-1-internet	.\logtail_installer.exe install ap-northeast-1-acceleration

リージョン	クラシックネットワークとVPC	インターネット(自己構築型IDC)	Global Acceleration
ドイツ (フランクフルト)	.\logtail_installer.exe install eu-central-1	.\logtail_installer.exe install eu-central-1-internet	.\logtail_installer.exe install eu-central-1-acceleration
UAE (ドバイ)	.\logtail_installer.exe install me-east-1	.\logtail_installer.exe install me-east-1-internet	.\logtail_installer.exe install me-east-1-acceleration
イギリス（ロンドン）	.\logtail_installer.exe install eu-west-1	.\logtail_installer.exe install eu-west-1	.\logtail_installer.exe install eu-west-1-acceleration

 注：

Log ServiceはAlibaba Cloud以外のマシンの所有者情報を取得できません。 そのため、自己構築型IDCや他のクラウドホストでLogtailを使用する場合、Logtailをインストール後に手動でユーザー ID を設定する必要があります。 ユーザーIDを設定しないと、Logtailは異常と判断し、ログを収集することができません。 詳細は、Alibaba Cloud ECS インスタンス以外または他のアカウントの ECS インスタンスからログを収集するを参照してください。

Logtailのアンインストール

Windows PowerShell か cmd.exe を起動して `logtail_in staller` ディレクトリに移動し、次のコマンドを実行します。

```
.\ logtail_in  staller . exe   uninstall
```

3.3.3 Logtail 起動設定パラメーター

このドキュメントでは、Logtail 起動設定パラメーターについて説明します。 特別な要件がある場合は、このドキュメントに従って起動パラメーターを設定できます。

シナリオ

次のシナリオでは、Logtail 起動設定パラメーターを設定する必要があります。

- 各ファイルのメタデータ情報（ファイルの署名、収集場所、ファイル名など）は、メモリに保持する必要があります。
- そのため、多数のログファイルを収集する場合は、メモリ使用量が大きくなることがあります。

- ログデータの量が多く、 Log Service 送信されるトラフィックが多いため、CPU 使用率が高くなります。
- Syslog/TCP データストリームが収集されます。

起動の設定

- ファイルパス

```
/ usr / local / ilogtail / ilogtail_c  onfig . json
```

- ファイル形式

  JSON

- ファイルサンプル（部分的な設定項目のみを表示）

```
{
    ...
    " cpu_usage_  limit " :  0 . 4 ,
    " mem_usage_  limit " :  100 ,
    " max_bytes_  per_sec " :  2097152 ,
    " process_th  read_count " :  1 ,
    " send_reque  st_concurr  ency " :  4 ,
    " buffer_fil  e_num " :  25 ,
    " buffer_fil  e_size " :  20971520 ,
    " buffer_fil  e_path " : "",
    ...
}
```

一般的な設定パラメーター

パラメーター名	パラメーター説明	値
cpu_usage_limit	CPU 使用率のしきい値。コアごとに計算されます。 たいていの場合、シングルコア処理能力はシンプルモードでは約24 MB /秒で、では約12 MB /秒です。	Double型。 最小値は0.1で、最大値は現在のマシンのCPUコア数です。 デフォルト値は2です。 たとえば、値 0.4 は、Logtail の CPU 使用率がシングルコア CPU の 40% に制限されていることを示します。 しきい値を超えた場合、Logtail は自動的に再起動します。

パラメーター名	パラメーター説明	値
mem_usage_limit	常駐メモリの使用しきい値。 1000 を超えるファイルを収集するには、しきい値を適切に増やします。	Int型。 MB で測定されます。 最小値は128で、最大値は現在のマシンの有効メモリ値です。 デフォルト値は2048です。 たとえば、値 100 は、Logtail のメモリ使用量が 100 MB に制限されていることを示します。 しきい値を超えた場合、Logtail は自動的に再起動します。
max_bytes_per_sec	Logtail が送信した生データのトラフィック制限は、20MB /秒を超えるストリームには制限されません。	Int型。 バイト/秒で測定されます。 範囲は1024 - 52428800で、デフォルト値は20971520です。 たとえば、値 2097152 は、Logtail のデータ転送速度が 2 MB/秒に制限されていることを示します。
process_thread_count	Logtail がログファイルのデータを書き込んだスレッドの数。 通常、シンプルモードでは 24 MB/秒、フルモードでは 12 MB/秒の書き込み速度をサポートします。 デフォルトでは、この値を調整する必要はありませんが、必要に応じてしきい値を増やすことができます。	Int型。 単位：個。 範囲：1~64。デフォルト値は1。

パラメーター名	パラメーター説明	値
send_request_concurrency	非同期並行処理の数。デフォルトでは、Logtail はデータパケットを非同期で送信します。 書き込み TPS が大きい場合は、より大きな非同期並行性値を設定できます。 Can be supported with a single concurrency of 0.5 Mb/s ~ It is based on the network delay to calculate the throughput of 1 Mb/s network. 注 : 単一の並行処理が全体を通して 0.5-1 MB/秒のネットワークをサポートしているという条件に基づいて同時実行数量を計算できます。 実際の並行処理量は、ネットワーク遅延によって異なります。	Int型。 単位：個。 範囲：1~1000、デフォルト値は20です。
buffer_file_num	ネットワーク例外が発生するか、または書き込み制限を超えた場合、Logtail は、リアルタイムで解析されたログをキャッシュとして、インストールディレクトリにあるローカルファイルに書き込みます、復帰後にログをLog Serviceに再送します。 このパラメーターは、キャッシュファイルの最大数を制限します。	Int型。 単位：個。 範囲：1~100、デフォルト値は25です。

パラメーター名	パラメーター説明	値
buffer_file_size	キャッシュファイルの最大Byteを設定できます。 buffer_file_num * buffer_file_size がキャッシュファイルが使用できる最大のディスク容量です。	Int型。 単位：個。 範囲：1048576 - 104857600、デフォルト値は20971520 Bytes (20 MB)です。
buffer_file_path	キャッシュファイルを保存するディレクトリ。 このパラメータ値を変更した後、旧キャッシュディレクトリの *logtail_buffer_file_** の形式でファイルを手動で新しいキャッシュディレクトリに移動し、ログを送信後 Logtail がキャッシュファイルを読み込んで削除できるようにする必要があります。	デフォルト値は null で、キャッシュファイルが Logtail インストールディレクトリ(*/usr/local/ilogtail*)に保存されていることを示します。
bind_interface	ローカルマシンにバインドされているネットワークカードの名前（eth1 など）（Linux バージョンのみがサポートされています）。	このパラメータはデフォルトで空となります。 利用可能なネットワークカードが自動的にバインドされます。 このパラメータが設定されている場合、Logtail はログをアップロードするためにこのネットワークカードを強制的に使用します。

パラメーター名	パラメーター説明	値
check_point_filename	チェックポイントファイルに保存されているフルパス。Logtail のチェックポイントの保存位置をカスタマイズするために使用されます。 Dockerユーザーがこのファイルストレージアドレスを変更し、チェックポイントファイルが存在するディレクトリをホストにマウントすることを推奨します。 そうしないと、チェックポイント情報がないためにコンテナが解放されたときに重複したコレクションが発生します。 例えば、Docker のcheck_point_filenameを/data/logtail/check_point.datとして設定し、-v /data/docker1/logtail:/data/logtailをDockerスタートアップコマンドに追加して /data/docker1/logtail ディレクトリを Docker の/data/logtailディレクトリにコピーします。	デフォルト値は/ *tmp* / *logtail_check_point* です。

注：

・ 上記の表には、注意が必要な一般的な起動パラメーターのみが記載されています。 *ilogtail_config.json* のパラメーターが表にない場合、デフォルト値が適用されます。

- 必要に応じて設定パラメーターの値を追加または変更します。 未使用の設定パラメーター（syslog データストリームの収集に関連するパラメーターなど）は `ilogtail_c onfig . json` に追加する必要はありません。

設定の変更

1. 必要に応じて `ilogtail_c onfig . json` を設定してください。

   変更された設定が有効な JSON 形式であることを確認します。
2. 変更された設定を適用するには、Logtail を再起動します。

```
/ etc / init . d / ilogtaild   stop
/ etc / init . d / ilogtaild   start
/ etc / init . d / ilogtaild   status
```

3.4 マシングループ

3.4.1 概要

Log Service では、マシングループで、ログ収集クライアントである Logtail を実装したサーバーをまとめて管理します。

マシングループは、複数サーバーの集まった仮想のグループです。マシングループにサーバーを追加し、そのマシングループに Logtail 構成を適用することで、複数のサーバーの各 Logtail クライアントを同じ構成にすることができます。

マシングループを次のいずれかの方法で定義します。

- IP アドレス: マシングループにすべてのサーバーの IP アドレスを追加します。グループの各サーバーは、それぞれに固有の IP アドレスで特定することができます。
- カスタム ID: マシングループに ID を設定し、その ID をグループの各マシンに割り当てます。

注 :

- 他のクラウドベンダーのサーバーやお客様のローカル IDC、または、別のアカウントの ECS インスタンスをマシングループに追加する前に、サーバーまたはインスタンスに AliUid を設定する必要があります。 詳細は、「非 Alibaba Cloud ECS インスタンスまたは別アカウントの Alibaba Cloud ECS インスタンス」をご参照ください。
- Windows サーバーと Linux サーバーを同じマシングループに追加することはできません。

IP アドレスベースのマシングループ

マシングループに複数のサーバーを追加するには、マシングループに各サーバーの IP アドレスを追加します。マシングループ内の全サーバーに Logtail クライアントを一括して設定できるようになります。

- ECS サーバーに hostname がバインドされていなく、かつネットワークタイプが変更されていない場合、その ECS サーバーのプライベート IP アドレスをマシングループに設定します。
- それ以外の場合は、Logtail クライアントの自動取得するサーバーの IP アドレスをマシングループに指定します。 各サーバーの IP アドレスは、サーバー上の `app_info . json` サーバーファイルの IP アドレスフィールドに登録されています。

注：

`app_info . json` ファイルには、Logtail クライアントに関する情報が記録されています。 その情報には、Logtail クライアントの自動取得したサーバーの IP アドレスが含まれます。 このファイルの IP アドレスフィールドを手動で変更しても、Logtail クライアントの取得する IP アドレスは変更されません。

Logtail クライアントがサーバーの IP アドレスを自動取得する方法は、次のとおりです。

- サーバーの `/ etc / hosts` ファイルに IP アドレスと hostname がバインドされている場合は、その IP アドレスを Logtail クライアントは自動取得します。
- サーバーの IP アドレスと hostname がバインドされていない場合、Logtail はサーバーのネットワークインタフェース（NI）のプライマリ IP アドレスを自動取得します。

注：

データ収集に Alibaba Cloud のイントラネットが使用されるかどうかは、プライベート IP アドレスベースのマシングループかどうかは関係がありません。 Alibaba Cloud ECS インスタンスを使用しており、その ECS インスタンスに Logtail をインストールする際に Alibaba Cloud イントラネット (クラシックネットワークと VPC)を選択した場合にのみ、そのサーバーのログデータは Alibaba Cloud イントラネットを介して収集されます。

詳細は、「マシングループの作成と IP アドレス割り当て」をご参照ください。

カスタム ID ベースのマシングループ

IP アドレス以外に、カスタム ID でマシングループを定義する方法もあります。

以下の場合は、マシングループにカスタム ID を定義されることをお勧めします。

- VPC といったカスタムネットワークの場合、1 つの IP アドレスを複数のサーバーが使用していることがあります。そういった場合、Log Service ではサーバー上の Logtail クライアントを管理することはできません。 この問題は、カスタム ID 定義によるマシングループで解決できます。
- 1 つのカスタム ID で、マシングループ内の各サーバーを自動スケーリングすることができます。 新しいサーバーに同じカスタム ID を設定する場合、Log Service は新しいサーバーを自動的に識別してそれをマシングループにそのサーバーを追加します。

通常、システムには複数のモジュールがあります。各モジュールは水平方向にスケーリングできます。つまり、各モジュールに複数のサーバーを追加できます。モジュールごとにマシングループを作成することで、モジュールごとにログを収集できます。そのためには、各モジュールにカスタム ID を作成し、各モジュールの各サーバーにマシングループ ID を設定する必要があります。たとえば、一般的な Web サイトには、HTTP リクエスト処理モジュール、キャッシュモジュール、ロジック処理モジュール、およびストアモジュールがあります。この場合には、HTTP リクエスト処理モジュールのカスタム ID は http_module、キャッシュモジュールは cache_module、ロジック処理モジュールは logic_module、ストアモジュールは store_module に設定することができます。

詳細は、マシングループ作成とカスタム ID 設定をご参照ください。

3.4.2 Logtail マシングループの作成

Log Service は、マシングループの形式で Logtail クライアントを使用してログを収集する必要のあるすべての ECS（ Elastic Compute Service ）インスタンスを管理します。

Logtail 設定を作成した後、マシングループページでマシングループを作成するか、 データインポートウィザードのマシングループに適用ページでマシングループの作成をクリックしてください。

以下を使用してマシングループを定義できます。

- マシングループ名を定義し、マシングループのイントラネット IP アドレスを追加します。

  マシングループに複数の ECS インスタンスを追加して、Alibaba Cloud ECS インスタンスのイントラネット IP アドレスを追加して Logtail 設定を統一することができます。 非 ECS インスタンス用のマシングループの作成方法については、Alibaba Cloud ECS インスタンス以外または他のアカウントの ECS インスタンスからログを収集するを参照してください。

- ユーザー定義の ID：マシングループの ID を定義し、対応するマシン上で ID を設定して関連付けを行います。

  システムは複数のモジュールで構成されています。 各モジュールの各部分は水平方向にスケーラブルであり、複数のマシンを含むことができます。 ログを個別に収集するには、モジュールごとにマシングループを作成します。 したがって、モジュールごとにユーザー定義の ID を作成し、各モジュールのサーバー上で ID を構成する必要があります。 たとえば、一般的に、ウェブサイトは、フロントエンド HTTP のリクエスト処理モジュール、キャッシュモジュール、ロジック処理モジュール、およびストレージモジュールで構成されいます。 ユーザー定義の ID は、http_module、cache_module、logic_module、および store_module として定義できます。

1. [Log Service コンソール]にログインします。 プロジェクト一覧ページで プロジェクト名をクリックします。 ログストアリストページを開きます。
2. ログストアリストページで、左側のナビゲーションウィンドウで LogHub -ログ収集 > Logtail マシングループをクリックします。マシングループページが表示されます。 右上のマシングループの作成をクリックします。

   また、マシングループの作成 （データインポートウィザードのマシングループに適用ページにあります）をクリックすることもできます。
3. グループ名を入力します。

   名前は 3～128 文字で、小文字、数字、ハイフン（ - ）、およびアンダースコア（_）を含むことができ、小文字または小文字の先頭または末尾に入力する必要があります。
4. マシングループの識別子を選択します。

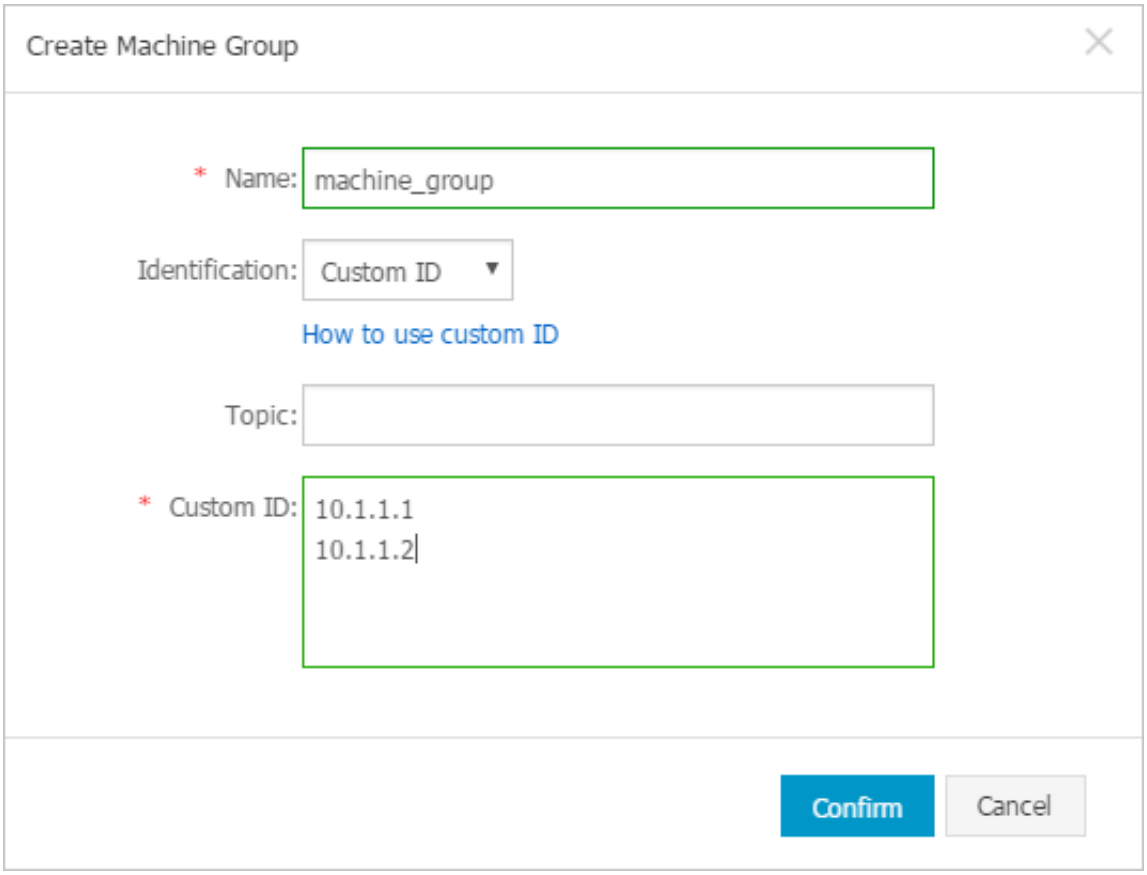
   - IP

     このオプションを選択して、 IP フィールドに ECS インスタンスのイントラネット IP アドレスを入力します。

     > 注 :
     >
     > - 入力された ECS インスタンスが現在のログイン Alibaba Cloud アカウントに属していることを確認します。
     > - 入力された ECS インスタンスと現在の Log Service プロジェクトが同じ Alibaba Cloud リージョンにあることを確認します。
     > - ECS インスタンスのイントラネット IP アドレスを使用していることを確認します。 複数の IP アドレスを区切るには、改行を使用します。

- Windows ECS インスタンスと Linux ECS インスタンスを同じマシングループに追加することはできません。
- 現在、Log Service は、Cloudtail を使用して Logtail クライアントをリモートにインストールする機能を無効にしています。 Logtail をインストールするには、Logtail の Linux へのインストールを参照してください。

図 3-4 : IPアドレス



・ ユーザー定義の固有設定

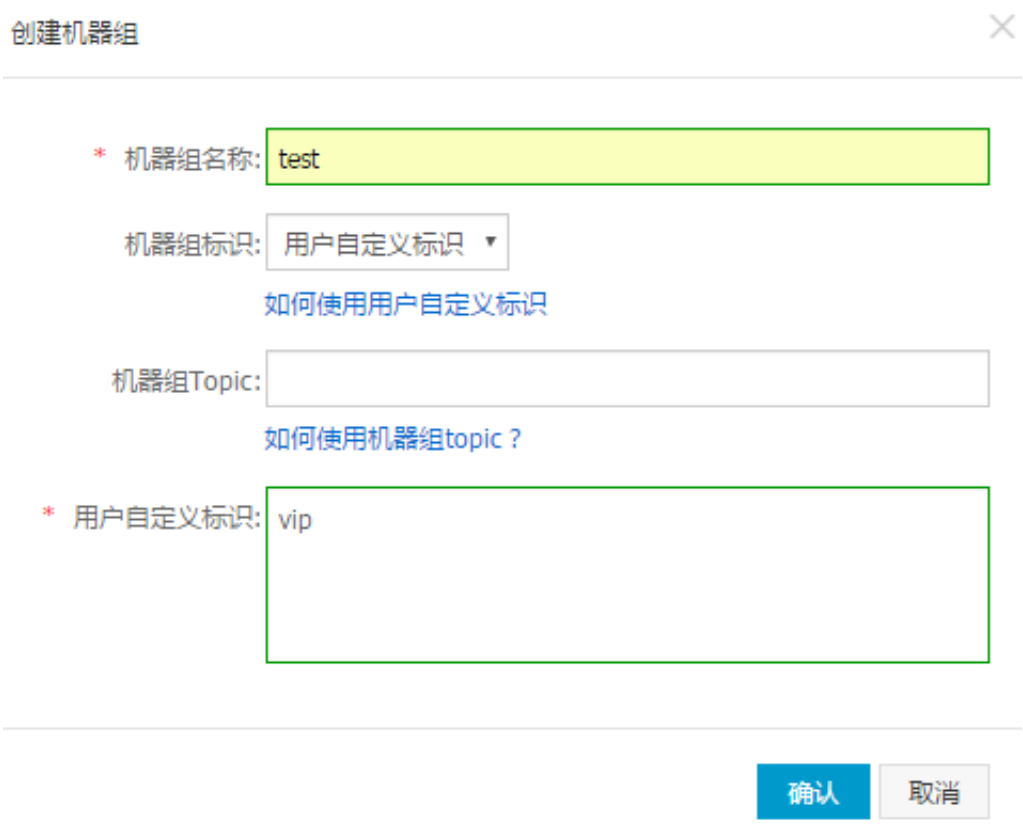
このオプションを選択した状態で、ユーザー定義の `ID` フィールドに カスタムID を入力します。

この手順を実行する前に、ログを収集するサーバー上にユーザー定義の ID を作成したことを確認してください。ユーザー定義の ID の使用方法については、マシングループにユーザー定義 ID を設定するを参照してください。

たとえば、フロントエンドモジュール用のサーバーを追加するために、モジュールのマシンを展開するには、Logtail をインストールし、追加するサーバー上でユーザー定義 ID が `http_modul e` のファイルを作成して、異なるマシングループの構成を自動的に同期

させます。 操作が成功したら、マシンステータスをクリックして、追加されたマシンを表示します。

図 3-5 : カスタムID



5. マシングループトピックを入力します。
6. 確認をクリックします。

   作成したマシングループは、マシングループページで表示できます。

図 3-6 : マシングループリスト

マシングループの作成後、マシングループのリストの表示、マシングループの変更、ステータスの表示、構成の管理、マシングループの削除ができます。

3.4.3 マシングループにユーザー定義 ID を設定する

IP アドレスの他に、ラベルのユーザー定義 ID を使用して、マシングループを動的に定義することができます。

ユーザー定義のIDは、以下のシナリオで有利です。

- 仮想プライベートクラウド（VPC）などのカスタムネットワーク環境では、異なるマシンのIP アドレスが互いに競合する可能性があり、Log Service が Logtail の管理に失敗することがあります。 ユーザー定義の ID は、このような状況を回避するのに役立ちます。
- 複数のマシンは同じラベルを使用してマシングループの自動スケーリングを実装します。 新しく追加されたマシンに対して同じユーザ定義 ID だけを設定する必要があります。 Log Service はそれを自動的に識別し、マシングループに追加することができます。

手順

ユーザ定義の ID を使用してマシングループを動的に定義するには、次の手順を実行します。

1. ユーザ定義の ID を有効にする
   - Linux Logtail

     `/ etc / ilogtail / user_defin  ed_id` ファイルを使ってユーザ定義の ID を設定します。

     たとえば、次のようにマシンのユーザー定義の ID を設定します。

     ```
     # cat  / etc / ilogtail / user_defin  ed_id
     ```

   - Windows Logtail

     `C :\ LogtailDat  a \ user_defin  ed_id` ファイルを使用してユーザ定義の ID を設定します。

     たとえば、次のようにマシンのユーザー定義の ID を設定します。

     ```
     C :\ LogtailDat  a > more   user_defin  ed_id
     aliyun - ecs - rs1e16355
     ```

     `aliyun - ecs - rs1e16355` をマシングループに追加します。 設定は 1 分後に有効になります。

     > 注：
     > ディレクトリ*/ etc / ilogtail /*または *C :\ LogtailDat a* 、ファイル*/ etc / ilogtail / user_defin ed_id* または *C :\ LogtailDat a \ user_defin ed_id* が存在しない場合は、手動で作成してください。

2. マシングループを作成する

   a. マシングループページで、右上のマシングループの作成をクリックします。

   b. マシングループの設定を完了します。

      - グループ名：マシングループの名前を入力します。
      - マシングループ ID ： ユーザー定義 IDを選択します。
      - ユーザー定義 ID ：手順 1 で設定したユーザー定義 ID を入力します。

   c. 確認をクリックして、マシングループを作成します。 マシンを拡張するには、追加するサーバーでステップ 1 を完了します。

3. ステータスを表示する

   マシングループページで、マシングループの右側にあるマシンステータスをクリックして、同じユーザー定義の ID とハートビートステータスを使用するマシンのリストを表示します。

その他の操作

ユーザー定義の ID を無効にする

IP アドレスをマシングループの識別情報として使用するには、 user_defined_id ファイルを削除します。 設定は 1 分後に有効になります。

```
rm  - f  / etc / ilogtail / user_defin  ed_id
```

- Linux OS

```
 rm  - f  / etc / ilogtail / user_defin  ed_id
```

- Windows Logtail

```
 Del   c : \  logtaildat  a  \  user_defin  ed_id
```

有効時間

user_defined_id ファイルを追加、削除、または変更すると、最新の設定がデフォルトで有効になります。

設定をすぐに有効にするには、次のコマンドを実行して Logtail を再起動します。

```
/ etc / init . d / ilogtaild   stop
/ etc / init . d / ilogtaild   start
```

- Linux OS

```
 / etc / init . d / ilogtaild   stop
```

```
/ etc / init . d / ilogtaild   start
```

- Windows Logtail

  Windows コントロールパネル > 管理ツール > サービスに移動し、サービスリストの LogtailWorker サービスを右クリックし、再起動を選択して設定を有効にします。

例

一般に、システムは複数のモジュールで構成されています。 各モジュールは、複数のマシンを含めることができ、例えば、一般的なウェブサイトは、フロントエンド HTTP リクエスト処理モジュール、キャッシュモジュール、ロジック処理モジュール、およびストレージモジュールで構成されています。 各パートは水平方向に拡張できます。そのため、マシンを追加するときにログをリアルタイムで収集する必要があります。

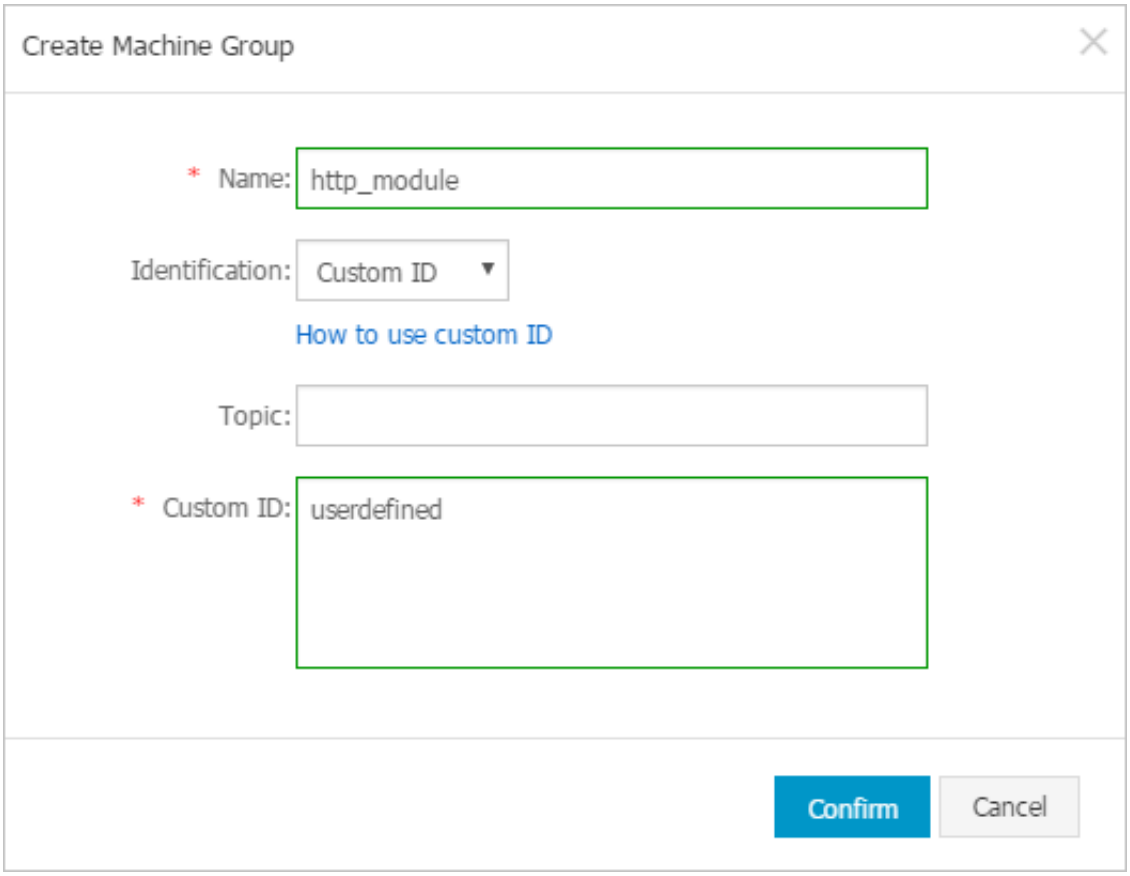
1. ユーザー定義の ID を作成します。

   Logtail クライアントをインストールした後、サーバーのユーザー定義 ID を有効にします。前述の例のモジュールの場合、ユーザー定義の ID は http_module、cache_module、logic_module、および store_module として定義できます。

2. マシングループを作成します。

   マシングループの作成時に、 User-defined Identityフィールドに、マシングループが対応するユーザー定義の ID を入力します。 http_module マシングループの次の設定を参照してください。 http_module マシングループは下図の通りです：

   図 3-7 : マシングループを作成します。



3. マシングループの右側にあるマシンステータスをクリックして、同じユーザー定義の ID とそのハートビートステータスを使用するマシンのリストを表示します。
4. フロントエンドモジュールにマシン 10.1.1.3 が追加されている場合は、新しく追加されたマシンでステップ 1 を完了します。 操作が正常に終了すると、マシングループステータスダイアログボックスで追加したマシンを表示できます。

3.4.4 Alibaba Cloud ECS インスタンス以外または他のアカウントの ECS インスタンスからログを収集する

Logtail を使用して、Alibaba Cloud ECS 以外またはご自身で作成されたものではない ECS インスタンスからログを収集するには、サーバーにログサービスに Logtail をインストールしてユーザー ID（アカウント ID）を設定し、ご自身のアカウントからそのサーバーにアクセスできることを検証してください。 そうしなければ、ハートビートステータスが異常と設定さ

れ、Logtail は Log Service にデータを収集することができません。 以下の手順に従って、ユーザー ID（アカウント ID）を設定します。

1. Logtail のインストール

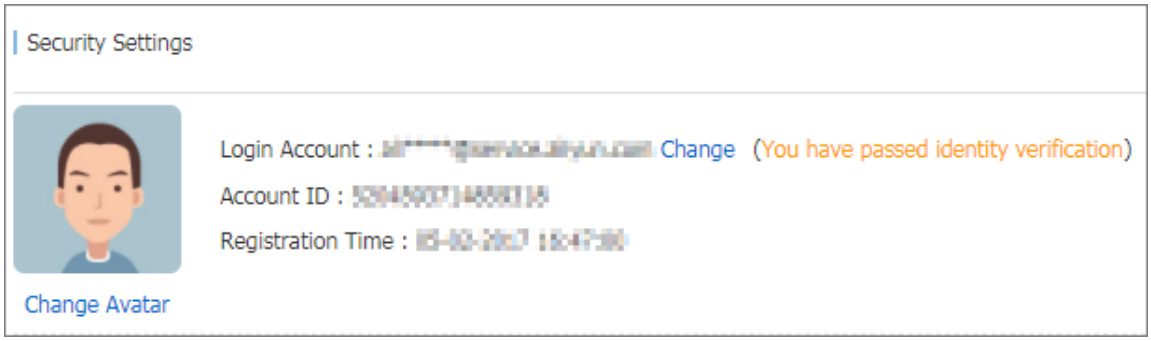
   ログを収集したいサーバーに Logtail をインストールするには、Linux の場合は、Logtail の Linux へのインストールを、Windowsの場合は、Logtail の Windowsへのインストールを参照してください。

2. ユーザ ID の設定

   a) Alibaba Cloud アカウント ID の表示

   Alibaba Cloud Account Management ページにログインし、Log Service プロジェクトのアカウント ID を表示します。

   図 3-8 : ユーザ ID の表示



   b) サーバーにアカウント ID ファイルを設定

   - Linux

     / etc / ilogtail / users ディレクトリにアカウント ID の名前の付いたファイルを作成します。ディレクトリが存在しない場合は、手動で作成できます。 以下のように、複数のアカウント ID を 1 台のマシン上に作成することもできます。

```
touch  / etc / ilogtail / users / 1559122535  028493
```

```
touch  / etc / ilogtail / users / 1329232535  020452
```

Log Service プロジェクトにデータを収集するために Logtail を必要としない場合は、ユーザー ID を削除してください。

```
rm  / etc / ilogtail / users / 1559122535  028493
```

- Windows OS

  ユーザーIDを設定するには、 *C :\ LogtailDat a \ users* ディレクトリにアカウントIDの名前の付いたファイルを作成します。 ユーザー IDを削除するには、このファイルを直接削除してください。

  C :\ LogtailDat a \ users \ 1559122535 028493 。

  注：

  - マシン上でアカウント ID を設定した後、 クラウドアカウントには、Logtail を使用してマシン上のログを収集する権限が与えられます。 適宜、マシンから不要なアカウント ID ファイルをクリアします。
  - ユーザー ID の追加または削除は 1 分以内に有効になります。

3.4.5 Logtail 設定の作成

Logtail クライアントは、Log Service コンソールの Elastic Compute Service（ECS）インスタンスからログを簡単に収集する方法を提供します。 Logtail クライアントをインストールしたら、Logtail クライアントのログ収集設定を作成する必要があります。 Logtail をインストールする方法については、Logtail の Linux へのインストールとLogtail の Windowsへのインストールを参照してください。 Logstore List ページで Logstore の Logtail 設定を作成および変更することができます。

Logtail 設定を作成する

Log Service コンソールでLogtail 設定を作成する方法については、テキストファイルの収集およびlogtailを用いたsyslogデータの収集 を参照してください。

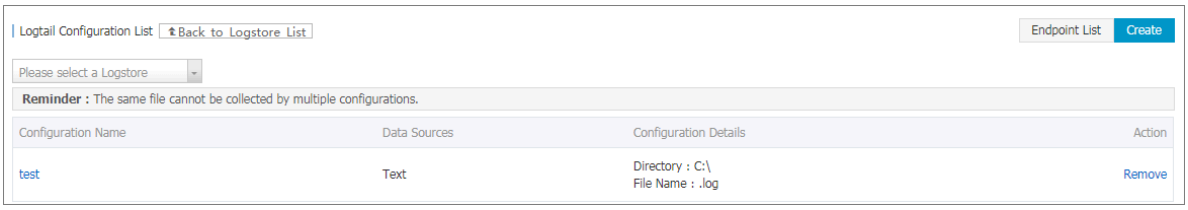
Logtail 設定リストを表示する

1. Log Serviceコンソールにログインします。
2. プロダクトページでプロジェクト名をクリックしてLogstore リストページへ移動します。

3. Logstore リストページで、Logstore の右側にある管理をクリックします。 Logtail 設定ページが 表示されます。

   このLogstore のすべての設定が、設定名、データソース、設定の詳細など、このページに表示されます。 データソースがテキストの場合、 ファイルパスとファイル名が構成の詳細に表示されます。

   図 3-9 : Logtail 設定リスト



> 注 :
> ファイルは、1つの設定だけで収集できます。

Logtail 設定の変更

1. Log Service コンソールにログインします。
2. プロダクトページでプロジェクト名をクリックします。
3. Logstore リストページで、Logstore の右側にある管理をクリックします。 Logtail 設定ページが表示されます。
4. 変更するLogtail の名前をクリックします。

   ログ収集モードを変更し、この変更された設定を適用するマシングループを指定できます。 設定変更プロセスは、設定作成プロセスと同じです。

Logtail 設定の削除

1. Log Serviceコンソールにログインします。
2. プロダクトページでプロジェクト名をクリックします。
3. Logstore リストページで、Logstore の右側にある管理をクリックします。 Logtail 設定ページが表示されます。
4. 削除しようとするLogtail 設定の右側にある削除をクリックします。

   設定が正常に削除されると、この構成を適用したマシングループからアンバインドされ、Logtailは削除された構成のログファイルの収集を停止します。

> 注 :
> Logstoreを削除する前に、LogstoreのすべてのLogtail 設定を削除する必要があります。

3.4.6 マシングループの管理

Log Service は、マシングループの形式で Logtail クライアントを使用してログを収集する必要のあるすべての Elastic Compute Service（ECS）インスタンスを管理します。 マシングループ ページに移動するには、プロジェクトリスト ページのプロジェクト名をクリックし、左側のナビゲーションで LogHub - ログ収集 > Logtail マシングループをクリックします。 マシングループの作成、変更、削除、マシングループのリストとステータスの表示、構成の管理、マシングループ ID の使用を Log Service のコンソールで行うことができます。

マシングループを作成する

以下を使用してマシングループを定義できます。

- IP：マシングループ名を定義し、マシングループのイントラネットIPアドレスを追加します。
- ユーザー定義の ID：マシングループの ID を定義し、対応するマシン上で ID を設定して関連付けを行います。

マシングループの作成方法については、Logtail マシングループの作成を参照してください。

マシングループリストを表示する

1. Log Service コンソールにログオンする。
2. Logstoreページで、左側のナビゲーションウィンドウで LogHub - ログ収集> Logtail マシングループをクリックします。 マシングループページが表示されます。

   プロジェクト内のすべてのマシングループを表示します。

   図 3-10 : マシングループリストを表示する

マシングループを変更する

マシングループを作成したら、必要に応じてマシングループ内の ECS インスタンスを調整できます。

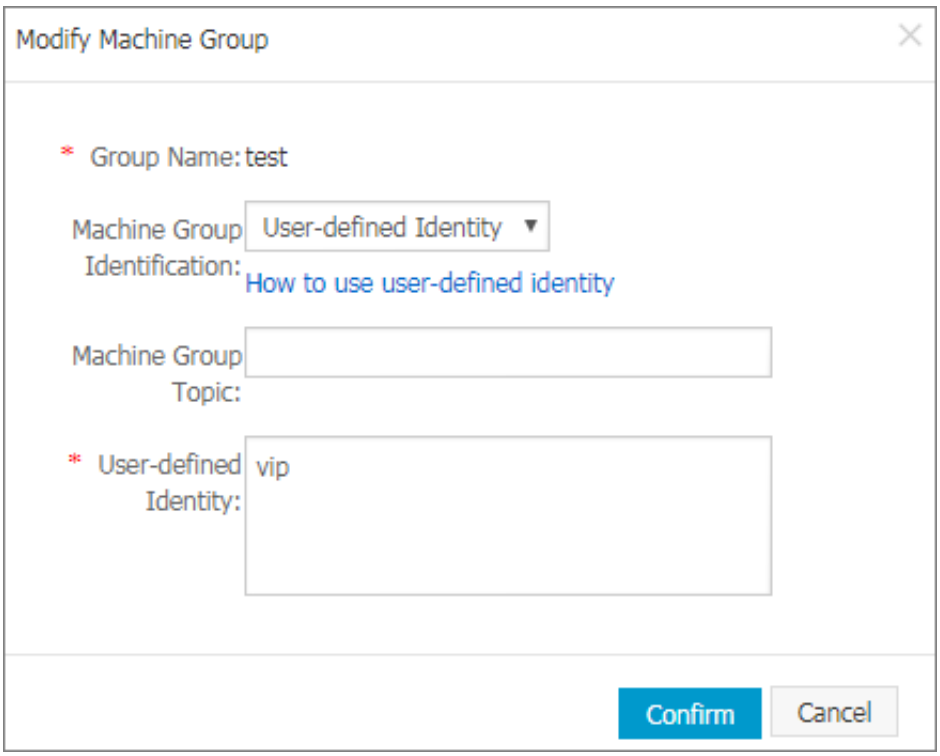
> 注 :
> マシングループ名は、マシングループの作成後に変更することはできません。

1. ログサービスコンソールにログインする。

2. Logstore ページで、左側のナビゲーションウィンドウで LogHub - ログ収集> Logtail マシングループをクリックします。 マシングループページが表示されます。

   Project内のすべてのマシングループが表示されます。
3. マシングループの右側にある変更をクリックします。
4. 設定を変更し、確認をクリックします。

図 3-11 : マシングループの変更



ステータス

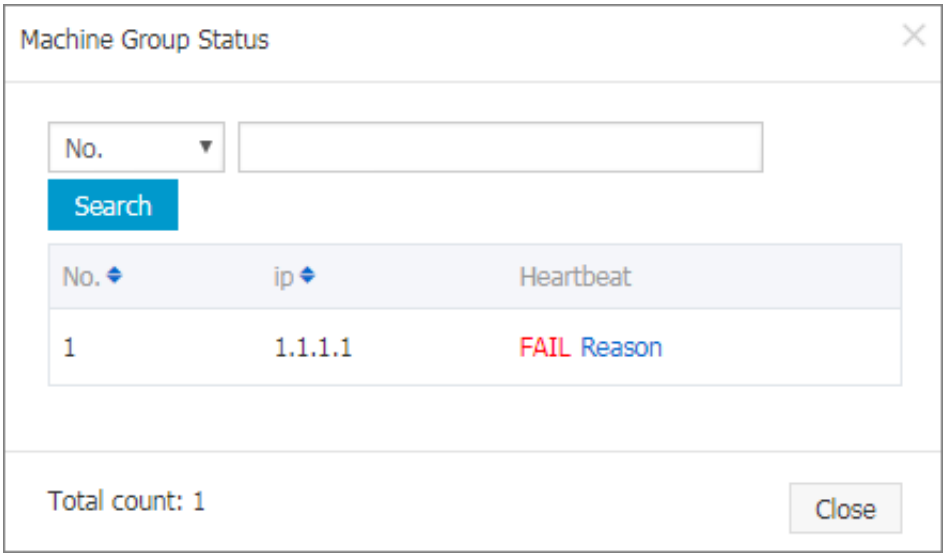
Logtail クライアントがマシングループ内のすべての ECS インスタンスに正常にインストールされたことを確認するには、Logtail クライアントのハートビートステータスを表示します。

1. Log Service コンソールにログインする。
2. プロジェクトページでプロジェクト名をクリックします。 Logstoreページで、左側のナビゲーションウィンドウで LogHub - ログ収集> Logtail マシングループをクリックします。 マシングループページが表示されます。
3. マシングループの右側にあるマシンステータスをクリックします。

   Logtailクライアントがすべての ECSインスタンスに正常にインストールされている場合、ECSインスタンスのハートビートステータスは `OK` です。 ハートビートのステータスが

FAIL の場合、ページで指示された理由を見つけることを推奨します。 問題が自分で解決できない場合は、チケットを起票し、サポートセンターへお問い合わせください。

図 3-12 : マシングループのステータスの表示



注 :

- ハートビートステータス OK は、 Logtail クライアントがログサービスに正しく接続していることを示します。 マシンをマシングループに追加した後、ハートビートのステータス OK を表示する前に、数分の遅延が存在する可能性があります。
- ECS インスタンスのハートビート状態が常に FAIL の場合、Logtail の Linux へのインストールとLogtail の Windowsへのインストールを参照してください。

設定の管理

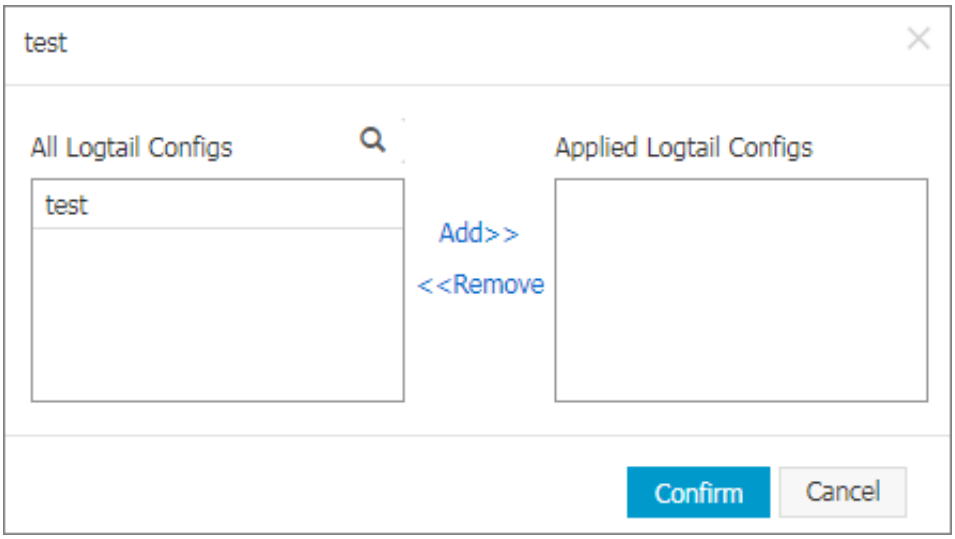
ログサービスは、マシングループを使用してログを収集する必要があるすべてのサーバーを管理します。 1つの重要な管理項目は、Logtail クライアントの収集設定です。 詳細については、テキストファイルの収集およびlogtailを用いたsyslogデータの収集を参照してください。 マシングループとの間でLogtail 設定を適用または削除して、収集されるログ、ログの解析方法、各 ECS インスタンスの Logtail によってログが送信される Logstore を決定することができます。

1. Log Service コンソールにログする。
2. Logstoreページで、左側のナビゲーションウィンドウでLogHub - ログ収集> Logtail マシングループをクリックします。 マシングループページが表示されます。
3. マシングループの右側にある設定をクリックします。

4. Logtail 設定を選択し、追加または削除をクリックして、マシングループとの間で設定を追加または削除します。

   Logtail 設定が追加されると、 マシングループ内の各 ECS インスタンスの Logtail クライアントに発行されます。 Logtail 設定が削除されると、 Logtail クライアントから削除されます。

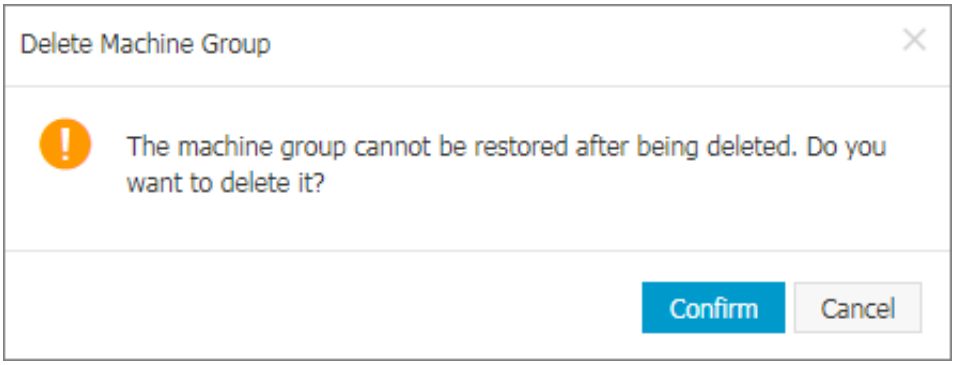
図 3-13 : マシングループ設定の管理



マシングループを削除する

1. Log Service コンソールにログインする。
2. プロジェクトページでプロジェクト名をクリックします。 Logstoreページで、左側のナビゲーションウィンドウでLogHub - ログ収集> Logtail マシングループをクリックします。 マシングループページが表示されます。
3. マシングループの右側にある削除をクリックします。
4. 表示されたダイアログボックスで確認をクリックします。

図 3-14 : マシングループの削除



3.5 テキストログ

3.5.1 テキストファイルの収集

Logtail クライアントは、Log Service ユーザーがコンソールで ECSインスタンスまたはローカルサーバーからログを収集する際に補助します。

前提条件

- ログを収集する前に Logtail をインストールする必要があります。 Logtail は、Windows および Linux オペレーティングシステムをサポートしています。インストール方法については、Logtail の Linux へのインストールとLogtail の Windowsへのインストールを参照してください。
- ECSインスタンスまたはローカルサーバーからログを収集するには、ポート80と443が開いていることを確認してください。

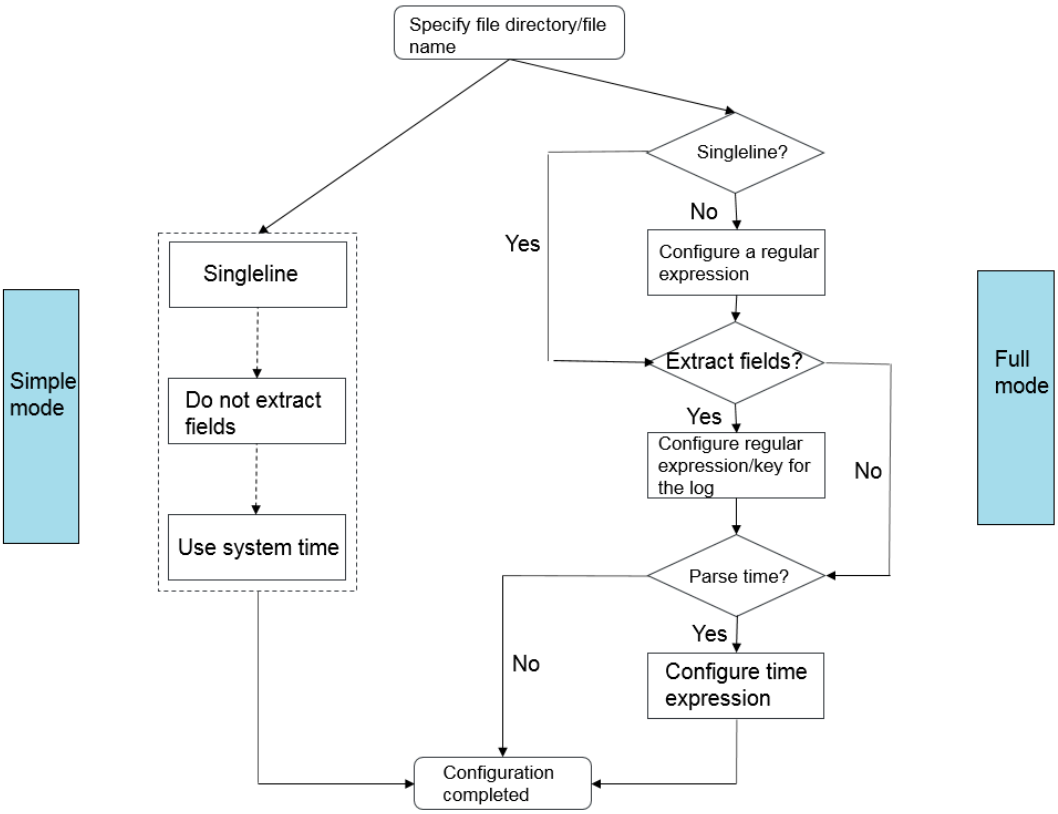
制限

- 単一のファイルは、1 つのみの設定で収集できます。 複数の設定を使用してファイルを収集する必要がある場合は、ソフトリンクが推奨されます。 例えば、`/ home / log / nginx / log` の下にあるファイルは、2 つの設定を使用して収集する必要があります。一方は元のパスを設定し、もう一方はフォルダ用に作成されたソフトリンク `ln - s / home / log / nginx / log / home / log / nginx / link_log` を設定します。
- オペレーティングシステムのバージョンの詳細は概要を参照してください。
- クラシックネットワークや仮想プライベートクラウド（VPC）、または Log Service プロジェクトの ECS インスタンスは、同一リージョンでなければなりません。 ソースデータがインターネットを通して送信される場合（IDC の使用法と同様）、リージョンサービスのプロジェクトが存在するリージョンをリージョンの説明に基づいて選択できます。

ログ収集の設定プロセス

ログサービスコンソールでは、シンプルモード、デリミタモード、JSON モード、 フルモードなどのモードでテキストログを収集するように Logtail を設定できます。 シンプルモードとフルモードを例に挙げます。 構成の手順は次の通りです。

図 3-15 : 手順



手順

1. プロジェクト名をクリックしてLogstore リストを開きます。
2. 対象の Logstore の右側にあるデータ・インポート・ウィザードをクリックします。
3. データソースを選択します。

   他のソースの下のテキストを選択することで、データソースの設定に入ります。
4. 設定名を指定します。

   設定名の長さは 3～63 文字で、小文字、数字、ハイフン（-）、アンダースコア（_）を含めることができます。小文字の英字で開始し、終了する必要があります。

> **注 :**
> 設定名は、後から変更することはできません。

5. ログディレクトリとファイル名を指定します。

   ディレクトリ構造は、完全パスまたはワイルドカードを含むパスである必要があります。

   > 注：
   > *と?のみがディレクトリ内でワイルドカードとして 使用できます。

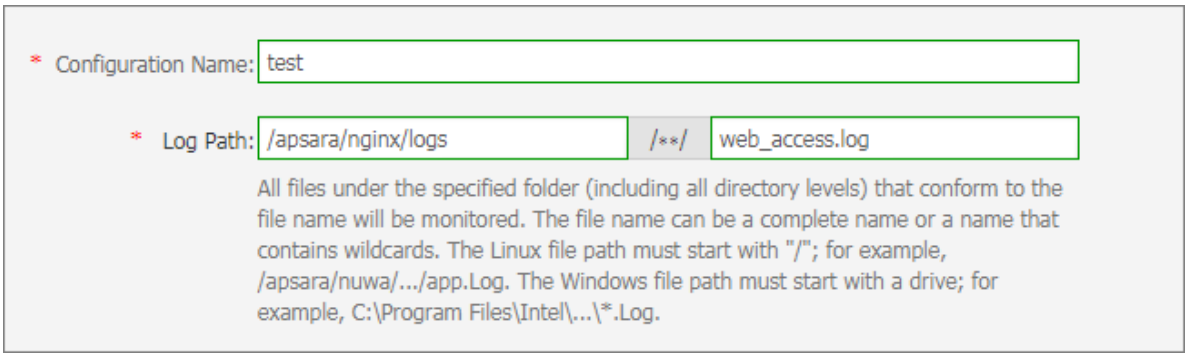
   ログファイル名は、完全なファイル名またはワイルドカードを含む名前でなければなりません。 ファイル名の規則については、 Wildcard matchingを参照してください。

   ログファイルの検索モードはマルチレベルのディレクトリマッチングモードです。つまり、指定したフォルダ（このフォルダのすべてのサブディレクトリを含む）で、ファイル名検索モードに適合するすべてのファイルがモニターされます。こちらに2つの例があります：

   - `/ apsara / nuwa / … /*. Log` は接尾辞が`. Log` で、`/ apsara / nuwa` ディレクトリ（再帰的なサブディレクトリを含む）に存在するファイルを意味します。
   - `/ var / logs / app_ * … /*. Log *`は`. Log` を含むファイル名を持ち、 `app_ *` 検索モード（再帰的なサブディレクトリを含む）に準拠するすべてのディレクトリに存在するファイルを意味します。`/ var / logs` ディレクトリの下にあります。

   > 注：
   > 1つのファイルは、1つの設定でしか収集できません。

   図 3-16 : ディレクトリとファイル名の指定



6. 収集モードを設定します。

   Logtail はシンプルモード、デリミタモード、JSON モード、正規表現モード、およびその他のログ収集メソッドをサポートしています。 詳細は収集方法を参照してください。 この例では、シンプルモードと正規表現モードを使用して収集モードの設定を紹介しています。

   ・ シンプルモード

     シンプルモード、つまりシングルラインモードでは、 デフォルトで 1 行のデータがログとして扱われます。すなわち、2行のログはログファイルの改行で区切られます。 システムはログフィールドを抽出しません（つまり、デフォルトでの正規表現は（.*））。ログ生成時刻として現在のサーバのシステム時刻が使用されます。 より詳細な設定を行うには、設定

をフルモードに変更して設定を調整します。 Logtail 設定の変更方法については、Logtail 設定の作成を参照してください。

シンプルモードで、ファイルディレクトリとファイル名を指定するだけで、 Logtail は行ごとにログを収集します。Logtail はログコンテンツからフィールドを抽出しません。さらに、ログ時間は、ログがキャプチャされた時間に設定されます。

- モード

ログコンテンツ（クロスラインログやフィールド抽出など）のパーソナライズされたフィールド抽出設定を設定するには、フルモードを選択します 。 これらのパラメーターの具体的な意味と設定方法の詳細については、概要を参照してください。

a. Enter ログサンプルを入力します。

ログサンプルを提供する目的は、ログサービスコンソールがログ内の正規表現モードを自動的に抽出するのを容易にすることを目的としています。 実際の環境からログを使用してください。

b. Disable シングルラインを無効にします。

シングルラインモードがデフォルトのオプションです。これは、ログが1行ずつ区切られていることを意味します。 クロスラインログ（Java プログラムログなど）を収集する必要がある場合は、 シングルラインを無効にし、 正規表現を設定する必要があります。

c. 正規表現を設定します。

このオプションは、自動生成と手動入力という 2 つの機能を提供します。 ログサンプルを入力した後、 自動生成をクリックすると、 正規表現が自動的に生成されます。 失敗した場合は、手動モードに切り替えて正規表現を入力して検証することができます。

d. フィールドの抽出を設定します。

ログコンテンツでフィールドを1つずつ分析して処理する必要がある場合は、フィールドの抽出機能を使用して、 指定されたフィールドをキーと値のペアに変換してからサーバーに送信します。 したがって、ログコンテンツ（具体的には正規表現）を解析する方法を指定する必要があります。

ログサービスコンソールでは、構文解析のための正規表現を2通りの方法で指定できます。 最初のオプションは、簡単な対話を介して正規表現を自動的に生成することです。抽出するフィールドを示すために、ログサンプルの”ドラッグ選択”メソッドを使用し

て、” 正規表現の作成 “をクリックすると、ログサービスコンソールが自動的に正規表現を生成します。

このようにして、自らが書くことなく正規表現を生成することができます。 さらに、手動で正規表現を入力することもできます。 [手動で正規表現入力]をクリックすると、 手動入力モードに切り替えることができます。 式が入力されたら、右側の[検証]をクリックして、 式がサンプルログを解析して抽出できるかどうかを確認します。

ログ解析の正規表現が自動的に生成されるか手動で入力されるかにかかわらず、抽出された各フィールドに名前を付ける必要があります （つまり、フィールドのキーを設定する必要があります）。

図 3-17 :

e. [システム時間を使用] を設定します。

[システム時間を使用] はデフォルトで設定されています。 無効になっている場合は、フィールド抽出中の時間フィールドとして使用する特定のフィールド（値）を指定し、このフィールドの名前を `time` とする必要があります。 `time` フィールドを選択した後、 自動生成 （時間フォーマット内にあります） をクリックして、このフィールドを

解析するメソッドを生成できます。 ログの時刻形式の詳細については、テキストログ－日付形式を参照してください。

7. 状況に応じて高度なオプションを設定して、次へをクリックします。

ローカルキャッシュ、オリジナルログのアップロード、トピック生成モード、ログファイルエンコーディング、最大モニタディレクトリの深さ、タイムアウト、およびフィルター設定を要件に応じて設定します。それ以外の場合は、デフォルトのままにしておきます。

構成項目	詳細
ローカルキャッシュ	ローカルキャッシュを有効にするかどうかを選択します。 この機能を有効にすると、ログサービスが利用できないときにログがマシンのローカルディレクトリにキャッシュされ、サービスの復旧後にログサービスに引き続き送信されます。 デフォルトでは、最大で1GBのログをキャッシュできます。
オリジナルログのアップロード	オリジナルログをアップロードするかどうかを選択します。有効にすると、デフォルトで新しいフィールドが追加され、オリジナルログがアップロードされます。
トピック生成モード	・ Null - トピックを生成しない：デフォルトオプションであり、トピックをヌル文字列として設定し、トピックを入力せずにログを照会できることを示します。 ・ マシングループのトピック属性：異なるフロントエンドサーバーで生成されたログデータを明確に区別するために使用されます。 ・ ファイルパスレギュラー：このオプションを選択すると、正規表現を使用してパスからコンテンツをトピックとして抽出するには、カスタム正規表現を入力する必要があります。 ユーザーとインスタンスによって生成されたログデータを区別するために使用されます。 ユーザーとインスタンスによって生成されたログデータを区別するために使用されます。
カスタム正規表現	トピック生成モードとしてファイルパスレギュラーを選択すると、カスタム正規表現を入力する必要があります。
ログファイルエンコーディング	・ utf8: UTF-8エンコーディングを使用。 ・ gbk: GBKエンコーディングを使用
最大モニターディレクトリの深さ	ログソースからログを収集するときに監視するディレクトリの最大深度を指定します。要するに、最大でモニターできるログレベルを指定します。 範囲は0~1000で、0を指定した場合は、現行ディレクトリレベルのみをモニターすることになります。

構成項目	詳細
タイムアウト	指定された時間内に更新がない場合、ログファイルはタイムアウトします。 タイムアウトの次の設定を構成できます。 ・ タイムアウトにならない：すべてのログファイルを永続的にモニターし、ログファイルがタイムアウトしないように指定します。 ・ 30分のタイムアウト：ログファイルが30分以内に更新がないければタイムアウトになり、モニターされなくなります。
フィルター構成	フィルター条件に完全に準拠したログのみ、収集できます。 たとえば： ・ 条件に一致するログを収集します：Key: level Regex:WARNING|ERRORは、レベルがWARNINGまたはERRORのログのみを収集することを示します。 ・ 条件に適合しないログをフィルターします： - `Key : level Regex :^(?!. *( INFO | DEBUG ))`、INFOまたはDEBUGレベルのログを収集しないことを示します。 - `Key : url Regex :. *^(?!.*( healthchec k )). *`、urlにヘルスチェックを含むログをフィルターすることを示します。 keyがurlでvalueが`/ inner / healthchec k / jiankong . html` のログなどは収集されません。 同様の事例についてはregex-exclude-wordとregex-exclude-patternを参照してください。

8. 設定が完了したら、次へをクリックします。

マシングループを作成していなければ、1つを作成する必要があります。 マシングループの作成方法は、「マシングループの作成」 Logtail マシングループの作成を参照してください。

注：

・ Logtail 設定が有効になるまでには最大3分かかります。
・ IISアクセスログを収集するには、Logstash を使用した IIS ログの収集を参照してください・

- Logtail 設定を作成後、Logtail 設定リストを表示したり、Logtail 設定を変更したり、Logtail 設定を削除することができます。 詳細は、Logtail 設定の作成を参照してください。

図 3-18 : マシングループに設定の適用

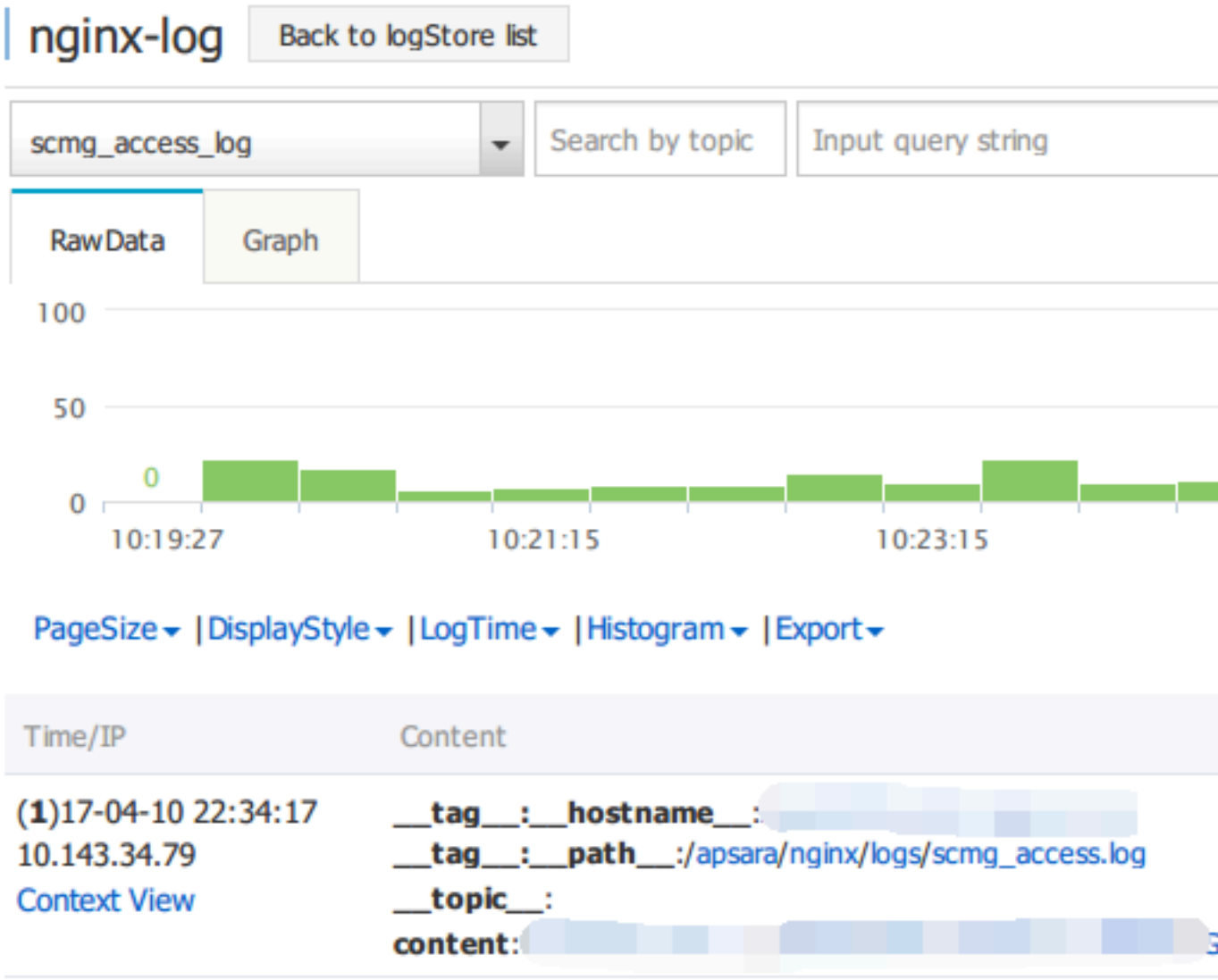
ログサービスは、設定が完了するとログを収集し始めます。

その他の操作

上記の設定が完了すると、ページの指示どおりに検索、分析、ビジュアライゼーション、シッパーとETLを設定することができます。

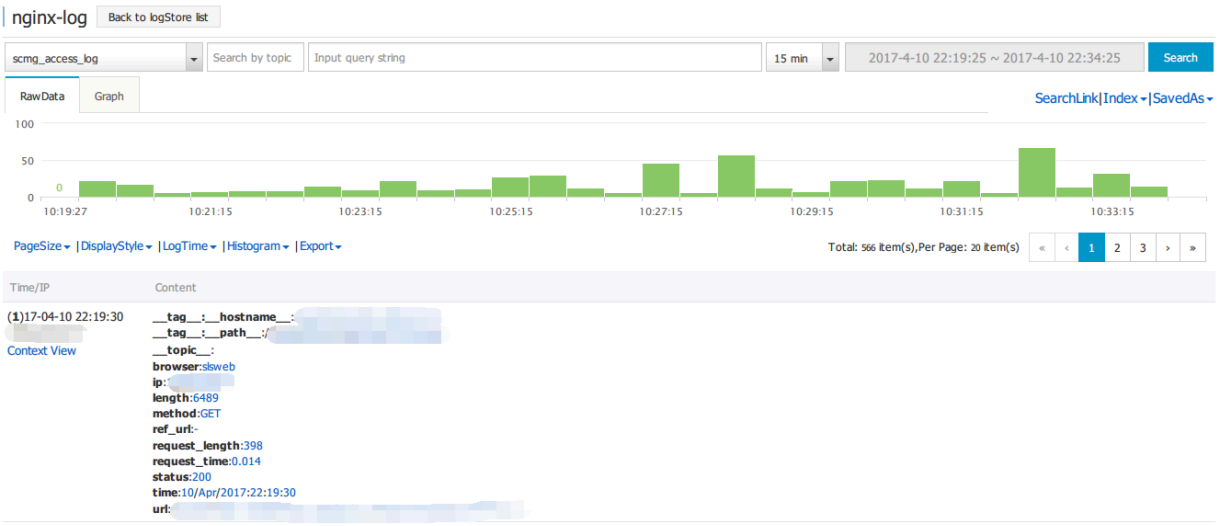
シンプルモードで Log Service に収集されたログは次のとおりです。 すべてのログの内容は、contentという名前のキーの下に表示されます。

図 3-19 : プレビュー



フルモードでログサービスに収集されたログは、次のとおりです。 設定されたキー値に従って、各ログの内容がログサービスに収集されます。

図 3-20 : プレビュー



設定項目をログに記録

Logtail を設定するときは、設定項目を完了する必要があります。 一般的に使用される構成項目の説明と制限は次のとおりです。

構成項目	説明
ログパス	ログ監視ディレクトリとログファイル名がマシン上のファイルと一致することを確認してください。 ディレクトリはファジー一致をサポートしておらず、絶対パスに設定する必要がありますが、ログファイル名はファジー一致をサポートしています。 ワイルドカードを含むパスは、複数のレベルのディレクトリと一致する可能性があります、つまり、指定されたフォルダ（すべてのレベルのディレクトリを含む）の下の、ファイル名に適合するすべてのファイルを監視することができます。
ログファイル名	収集されたログ・ファイルの名前を示します。名前は大文字と小文字を区別し、`*. log` のようにワイルドカードを含むことがあります。Linuxのファイル名ワイルドカードには、*、?、[…]があります。
ローカルストレージ	短期間のネットワーク中断のために送信できないログをローカルキャッシュに一時的に保存するかどうかを指定します。

First-line log header	正規表現による複数行ログの開始行を示します。 複数行のログ収集（たとえば、スタック情報を持つアプリケーションログ）では、個々のログを区切るために行を使用することはできません。 この場合、複数行ログの開始行を指定する必要があります。 この行が検出されると、最後のログが終了し、新しいログが開始されたことを示します。 そのため、開始ヘッダーに一致するルール、つまり正規表現を指定する必要があります。
ログ解析式	ログ情報を抽出し、Log Serviceでサポートされているログ形式に変換する方法を示します。 必要なログフィールドを抽出し、各フィールドに名前を付けるには、正規表現を指定する必要があります。
ログ時刻形式	ログデータのタイムスタンプ文字列の時刻形式を解析する方法を定義します。 詳細はテキストログー 日付形式を参照してください。

ログの書き込み方法

ログを収集するために Logtail を使用することに加えて、Log Service はログを書くのに役立つAPIとSDKも提供します。

- API を使用してログを書き込む

  ログサービスには、ログを書き込むのに役立つ RESTful API が用意されています。 PostLogStoreLogs インターフェイスを PutLogStoreLogs 使用してデータを書き込むことができます。 完全な API リファレンスについては、Overviewを参照してください。

- SDK を使用してログを書き込む

  ログサービスは、API に加えて、複数の言語（Java、.NET、PHP、およびPython）で SDK を提供しているため、簡単にログを書き込むことができます。 SDK リファレンスについては、SDKリファレンス を参照してください 概要。

3.5.2 テキストログの設定と解析

ログラインの分離方法を指定する

一般に、完全なアクセスログ（例えば、 Nginx アクセスログ）は 1 行を占めます。2 つのログは改行で区切られています。 たとえば、次の 2 つのシングルラインアクセスログを参照してください。

```
10 . 1 . 1 . 1  - - [ 13 / Mar / 2016 : 10 : 00 : 10  + 0800 ] " GET
 /  HTTP / 1 . 1 "  0 . 011   180   404   570  "-" " Mozilla / 4 . 0
( compatible ;  MSIE   6 . 0 ;  Windows   NT   5 . 1 ;  360se )"
10 . 1 . 1 . 1  - - [ 13 / Mar / 2016 : 10 : 00 : 11  + 0800 ] " GET
 /  HTTP / 1 . 1 "  0 . 011   180   404   570  "-" " Mozilla / 4 . 0
( compatible ;  MSIE   6 . 0 ;  Windows   NT   5 . 1 ;  360se )"
```

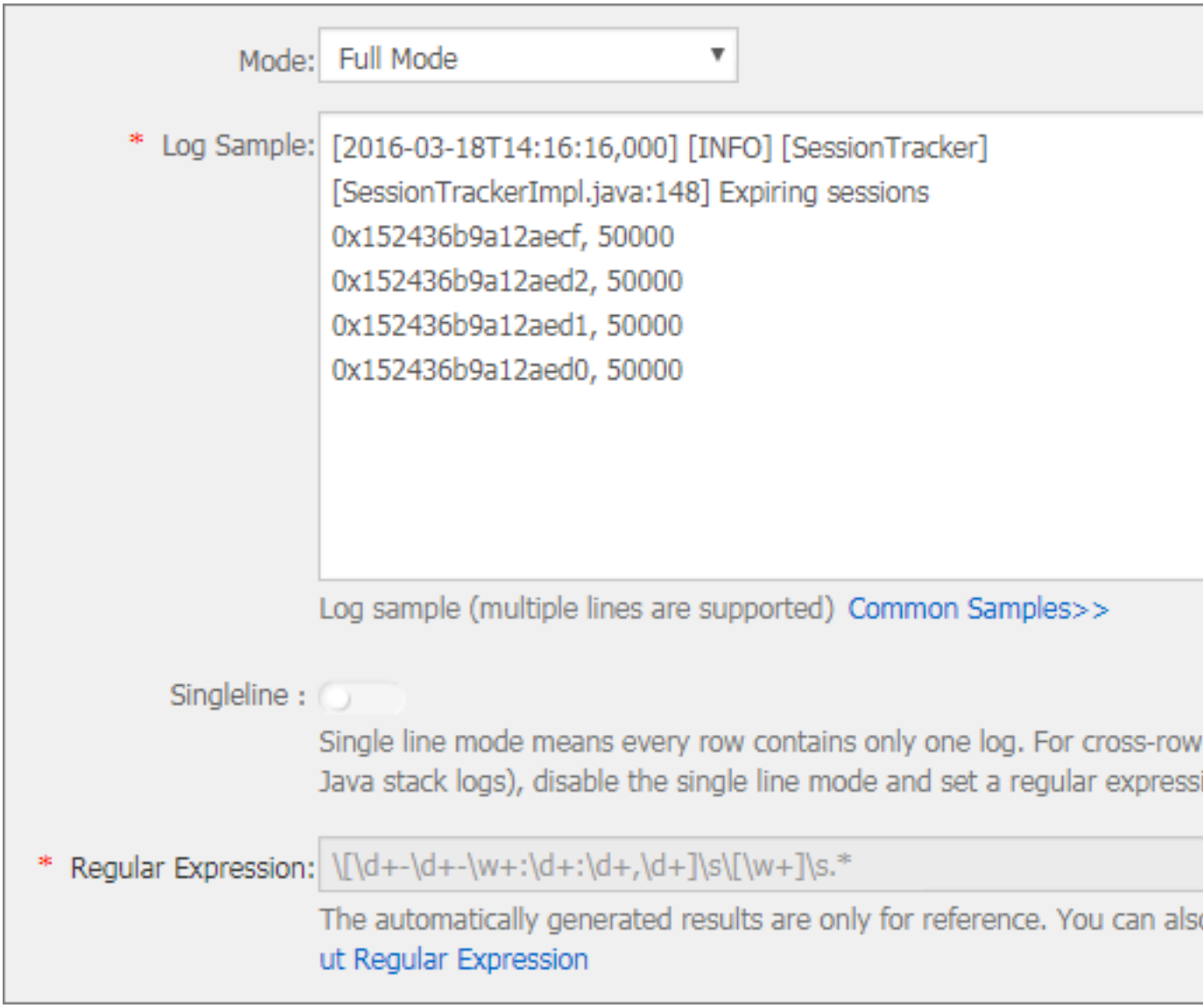
Java アプリケーションの場合、プログラムログは通常複数の行にまたがります。 特性ログヘッダーは、 2 つのログを分離するために使用されます。 たとえば、次の Java プログラムログを参照してください。

```
[ 2016 - 03 - 18T14 : 16 : 16 , 000 ] [ INFO ] [ SessionTra  cker ] [
 SessionTra  ckerImpl . java : 148 ]  Expiring   sessions
 0x152436b9  a12aecf ,  50000
 0x152436b9  a12aed2 ,  50000
 0x152436b9  a12aed1 ,  50000
```

```
0x152436b9  a12aed0 ,  50000
```

前述の Java ログには、日時形式の開始フィールドがあります。 正規表現は \[\d+-\d+-\w+:\d+:\d+,\d+]\s.* です。 以下のように、コンソールで設定を完了してください。

図 3-21 : フルモード解析の正規表現



ログフィールドを抽出する

Log Service data modelsによれば、ログには 1 つ以上のキーと値のペアが含まれています。 分析のために指定したフィールドを抽出するには、正規表現を設定する必要があります。 ログコンテンツを処理する必要がない場合、ログはキーと値のペアと見なすことができます。

上記のアクセスログの場合：

- フィールドが抽出されるとき

  Regular expression: `(\ S +)\ s -\ s -\ s \[(\ S +)\ s [^]]+]\ s "(\ w +).*`, Extracted contents: `10 . 1 . 1 . 1` , `13 / Mar / 2016 : 10 : 00` and `GET` .

- フィールドが抽出されない場合

  Regular expression: `(. *)`, Extracted contents: `10 . 1 . 1 . 1 - - [ 13 / Mar / 2016 : 10 : 00 : 10 + 0800 ] " GET / HTTP / 1 . 1 " 0 . 011 180 404 570 "-" " Mozilla / 4 . 0 ( compatible ; MSIE 6 . 0 ; Windows NT 5 . 1 ; 360se )""`

ログ時間を指定する

Log Service data models によると、ログには UNIX のタイムスタンプ形式の時間フィールドが必要です。 現在、ログ時間は、 Logtail がログ内容を収集する時のシステム時刻に設定することができます。

上記のアクセスログの場合：

- ログコンテンツの時間フィールドを抽出する `Time : 13 / Mar / 2016 : 10 : 00 : 10` Time expression: `% d /% b /% Y :% H :% M :% S`
- ログが収集されるシステム時刻 Time: Timestamp when the log is collected.

3.5.3 テキストログー 日付形式

Log Serviceの概要に記載されているように、Log Service の各ログはログが発生した際のタイムスタンプを有しています。 Logtail は、ログファイルからログを収集する際に、各ログのタイムスタンプ文字列を抽出し、タイムスタンプに解析する必要があります。 したがって、Logtail のログのタイムスタンプ形式を指定する必要があります。

Linux では、Logtail は strftime function が提供するすべての日時形式をサポートしています。 Logtail は、strftime 関数で定義されたログ形式で表現できるタイムスタンプ文字列を解析して使用することができます。

実際には、ログのタイムスタンプ文字列には複数の形式があります。 設定を簡単にするために、Logtail は以下の共通のログ日時形式をサポートしています。

フォーマット	意味	例（説明）
%a	1 週間のうちの 1 日の省略形。	例：金

フォーマット	意味	例（説明）
%A	1 週間に 1 日の名前。	例：金曜日
%b	月の省略形。	例：Jan
%B	月の名前。	例：1 月
%d	10 進形式の月の日[01,31]。	例：07,31
%h	月の省略形。 `% b` と同じ。	例：Jan
%H	時間は 24 時間形式です。	例：22
%I	12 時間形式の時間。	例：11
%m	10 進形式の月。	例：08
%M	10 進形式の分[00,59]。	例：59
%n	改行	改行
%p	ローカル時間AM または PM 。	例：AM/PM
%r	`% I :% M :% S % p` に相当する 12 時間形式の時刻。	例：11:59:59 AM
%R	時間と分で表される時間。これは `% H :% M` に相当します。	例：23:59
% S	2 番目は 10 進形式[00,59]です。	例：59
%t	タブ文字	タブ文字
%y	10 進形式ではない年[00,99]。	例：04,98
%Y	10 進形式の年。	例：2004 年、1998 年
%C	10 進形式の世紀[00-99]。	例：16
%e	10 進形式の月の日[1,31]。一桁の前にスペースがあります。	例：7,31
%j	10 進数形式の年の日[001,366]。	例：365
%u	10 進形式の曜日[1,7]。 1 は月曜日を表します。	例：2
%U	週の週番号（週の最初の日の日曜日）[00,53]。	例：23

フォーマット	意味	例（説明）
%V	週の週番号（週の最初の曜日）[01,53]。 1 月初めの週が 4 日以上であれば、今週は今年の最初の週です。 それ以外の場合は、翌週がその年の最初の週とみなされます。	例：24
%w	10進数形式の曜日[0,6]です。 0 は日曜日を表します。	例：5
%W	週の週番号（週の最初の日の月曜日）[00,53]。	例：23
%c	標準的な日時表現。	長い日付や短い日付などの詳細情報を指定するには、前述のサポートされている形式を使用してより正確な式を使用することを推奨します。
%x	標準の日付表現。	長い日付や短い日付などの詳細情報を指定するには、前述のサポートされている形式を使用してより正確な式を使用することを推奨します。
%X	標準時間表記。	長い日付や短い日付などの詳細情報を指定するには、前述のサポートされている形式を使用してより正確な式を使用することを推奨します。
%s	UNIX タイムスタンプ。	例: 1476187251

3.5.4 履歴ログのインポート

Logtail はデフォルトでインクリメンタルログのみを収集します。 履歴ログをインポートする場合は、Logtail の履歴ログインポート機能を使用します。

- Logtail のバージョンは `0 . 16 . 6` 以上であること。
- 対象の履歴ログは、収集設定に属していなければならず、Logtail によって収集さていないこと。
- 履歴ログの最終変更時刻は、Logtail 設定時刻よりも前であること。
- ローカル設定の生成とインポートの最大間隔は1分であること。
- ローカル設定をロードする特別なアクションのため、Logtail がお使いのサーバーに `LOAD_LOCAL _EVENT_ALA RM` を送信してこの動作を通知すること。

Logtail は、リスニングをオンにして検出されたイベントに基づいてログを収集します。 Logtail はローカル設定をロードし、ログ収集をトリガーすることもできます。 Logtail は、ローカル設定をロードして履歴ログを収集します。

1. 収集設定情報の作成

   収集を設定し、マシングループに適用します。 対象のログが収集設定に属していることを確認します。 収集の設定についての詳細は、テキストファイルの収集を参照してください。

2. 設定の一意の ID を取得します。

   次の例のように、ローカル `/ usr / local / ilogtail / user_log_c onfig . json` から設定の一意のIDを取得します：

```
grep  "##" / usr / local / ilogtail / user_log_c  onfig . json  |
awk  '{ print  $ 1 }'
                    "## 1 . 0 ## log - config - test $ multi "
                    "## 1 . 0 ## log - config - test $ ecs - test "
                    "## 1 . 0 ## log - config - test $ metric_sys
tem_test "
                    "## 1 . 0 ## log - config - test $ redis -
status "
```

3. ローカルイベントを追加します。

   ローカルイベントを 次のフォーマットを使用して JSON ファイル `/ usr / local / ilogtail / local_even t . json` に保存します。

```
[
  {
    " config " : "${ your_confi  g_unique_i  d }",
    " dir " : "${ your_log_d  ir }",
    " name " : "${ your_log_f  ile_name }"
   },
  {
   ...
   }
   ...
]
```

   - 構成項目

構成項目	説明	例
Config	ステップ2で取得した構成の一意のID。	##1.0#
dir	ログが存在するフォルダー。 注： フォルダーが/で終わることはできません。	/ dat

構成項目	説明	例
name	ログの名前。	acce

注：

Logite が無効な JSON ファイルを読み込まないようにするには、ローカルイベントの構成情報を一時ファイルに保存し、一時ファイルを編集した後に/ usr / local / ilogtail / local_even t . json にその内容をコピーすることをおすすめします。

- 設定の例

```
$ cat / usr / local / ilogtail / local_even t . json
[
  {
    " config " : "## 1 . 0 ## log - config - test $ ecs - test ",
    " dir " : "/ data / log /",
    " name " : " access . log . 2017 - 08 - 08 "
  },
  {
    " config " : "## 1 . 0 ## log - config - test $ ecs - test ",
    " dir " : "/ tmp ",
    " name " : " access . log . 2017 - 08 - 09 "
  }
]
```

- Logtail が設定をロードしたかどうかを確認するには？

  ローカルファイル local_even t . json を保存すると、Logtailはこのローカル構成ファイルを1分以内にメモリにロードし、 local_even t . json の内容をクリアします。

  Logtailがローカルイベントを読み込んでいるかどうかは、次の方法で確認できます：

  - local_even t . json 内のコンテンツが消去されているかどうかを確認します。 消去された場合、Logtail はローカル設定情報を読み込みます。
  - / usr / local / ilogtail / ilogtail . LOG ファイルに次の情報が含まれているかどうかを確認します。 process local event キーワード。 local_even t . json の内容が消去されたが、これらのキーワードが見つからない場合、ローカル設定ファイルが無効でフィルタされている可能性があります。
  - #unique_75で LOAD_LOCAL _EVENT_ALA RM アラームがあるかどうかをクエリします。

- Logtail は設定情報をロードしましたが、それでもデータを収集できません。 この問題をどう処理しますか？

  この問題は、以下の理由により発生する可能性があります：

  - 設定情報が無効です。
  - ローカル設定の `config` 項目は存在しません。
  - 対象のログは、収集設定内で指定されたパスにありません。
  - 対象のログは既に収集されています。

- 収集済みのデータをどのように収集できますか？

  収集済みのデータを収集するには、次の手順を実行します：

  1. `/ etc / init . d / ilogtaild  stop` コマンドを実行してLogtail を停止します。
  2. `/ tmp / logtail_ch  eck_point` ファイルでログのパスを検索します。
  3. このログのチェックポイント（JSONオブジェクト）を削除し、変更を保存します。
  4. 手順3に従って、ローカルイベントを追加します。
  5. `/ etc / init . d / ilogtaild  start` コマンドを実行してLogtail を起動します。

3.5.5 ログトピック

注：
syslog データのトピックは設定できません。

トピック生成モード

Logtail を使用してログを収集したり、API/SDK を使用してデータをアップロードするときに、トピックを設定できます。 現在、次のトピック生成モードがコンソールでサポートされています。Null - トピックを生成しない、マシングループトピック属性、およびファイルパスレギュラー。

- Null - トピックを生成しない

  コンソールにテキストログを収集するために Logtail を設定するとき、デフォルトのログトピック生成モードは、 Null - トピックを生成しないです。つまり、トピックがヌル文字列であるときにトピックを入力せずにログをクエリできます。

- マシングループのトピック属性

  マシングループのトピック属性モードは、異なるサーバーで生成されたログデータを明確に区別するために使用されます。 異なるサーバーのログデータが同じファイルパスとファイル名に格納されている場合、異なるマシングループにマシンを分けて、異なるサーバーのログデータをトピックごとに区別することができます。 これを行うには、マシングループを作成するときに異なるマシングループに異なるトピック属性を設定し、トピック生成モードをマシングループトピック属性に設定します。以前に作成した Logtail 設定をそれらのマシングループに適用して設定を完了します。

  このモードを選択すると、データを報告するときに、現在のマシンが属しているマシングループのトピック属性がトピック名として Log Service にアップロードされます。 ログ索引分析機能を使用してログを照会するときは、トピックを指定する必要があります。つまり、ターゲット・マシン・グループのトピック属性を照会条件として指定する必要があります。

- ファイルパスレギュラー

  このモードは、ユーザーとインスタンスによって生成されたログデータを区別するために使用されます。サービスログがユーザーまたはインスタンスに基づいて異なるディレクトリに格納されているが、サブディレクトリとログファイル名が同じ場合、Log Service はログファイルを収集するときにログを生成するユーザーまたはインスタンスを明確に区別できません。 この場合、トピック生成モードをファイルパスレギュラーに設定し、ファイルパスの正規表現を入力してトピックをインスタンス名として設定することができます。

  このモードを選択すると、データを報告するときに Logtail がトピック名としてインスタンス名を Log Service にアップロードします。 ディレクトリ構造と設定に従って異なるトピックが生成されます。 ログ索引分析機能を使用してログを照会するときは、トピック名をインスタンス名として指定する必要があります。

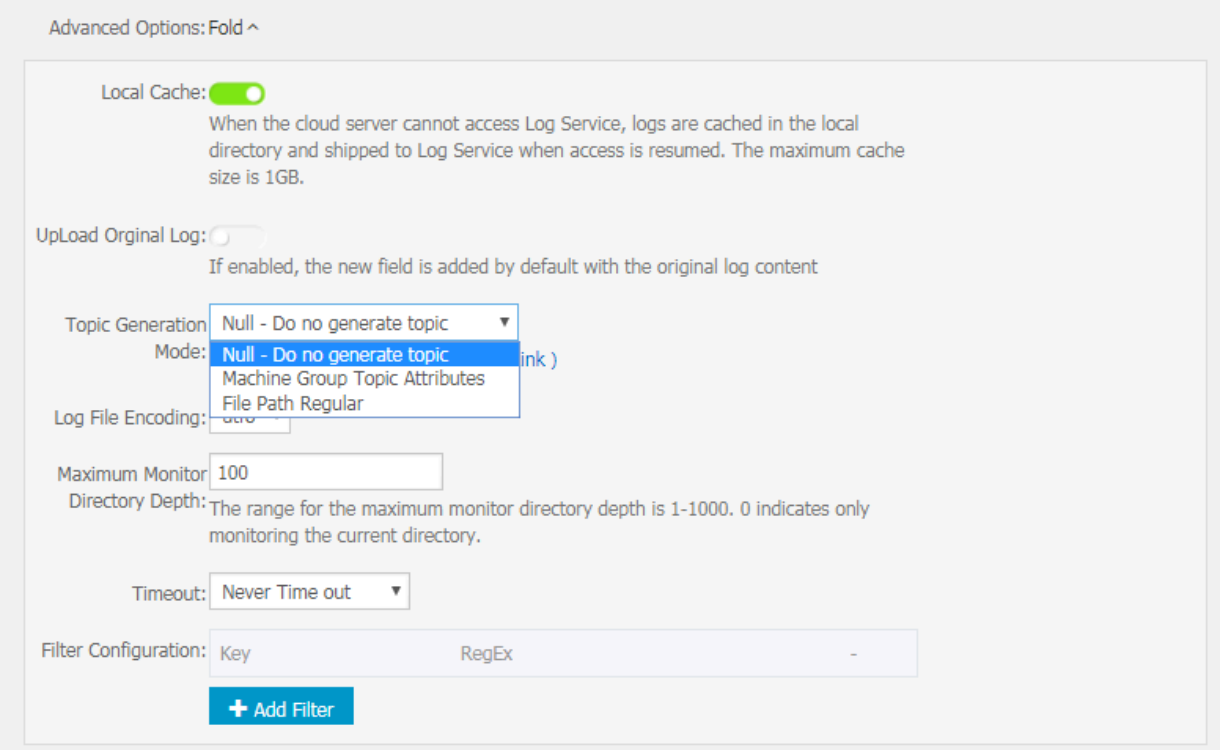
ログトピックの設定

1. テキストファイルの収集に従って、コンソールに Logtail を設定します。

   トピック生成モードをマシングループトピック属性に設定するには、マシングループの作成/変更時にマシングループトピックを設定します。

2. データインポートウィザードの詳細オプションを展開し、トピック生成モードドロップダウンリストからトピック生成モードを選択します。

図 3-22 : ログトピックの設定



ログトピックの変更

ログトピックの生成モードを変更するには、データインポートウィザードでトピック生成モードオプションを直接変更します。

注 :
変更された設定は、変更が有効になった後に収集されたデータにのみ適用されます。

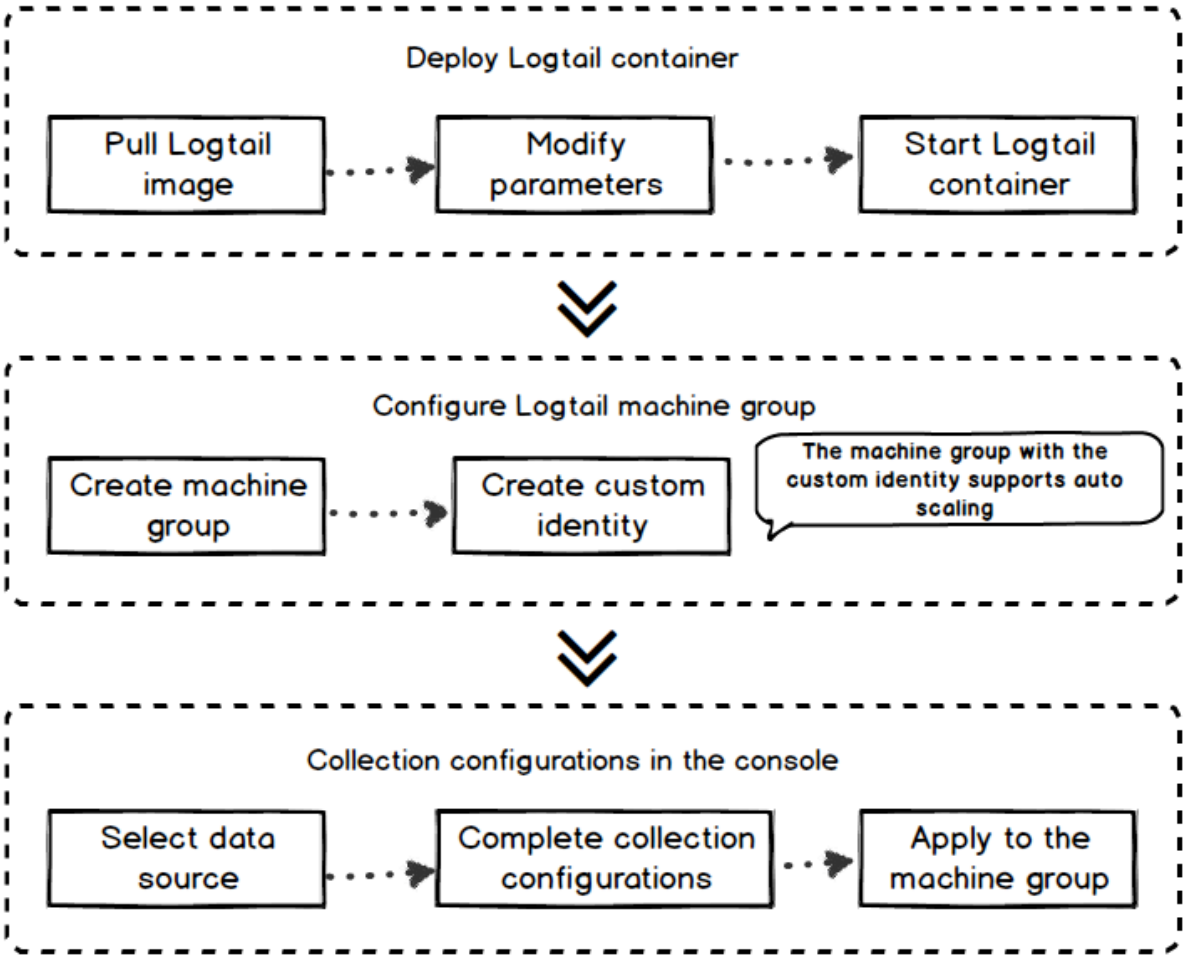
3.6 コンテナログの収集

3.6.1 標準の Docker ログの収集

Logtail は、標準の Docker ログを収集し、これらのログをコンテナ関連のメタデータ情報と一緒に Log Service にアップロードすることをサポートしています。

設定プロセス

図 3-23 : 設定プロセス



1. Logtail コンテナをデプロイします。
2. Logtail コンテナを構成します。

   ログサービスコンソールで、ユーザー定義の ID を持つマシングループを作成します。 その後、コンテナークラスターを拡張または縮小するために追加の操作および保守（O&M）は必要ありません。
3. コンソールにコレクション設定を作成します。

   ログサービスコンソールでコレクション構成を作成します。 すべてのコレクション構成はサーバー側用です。 ローカル設定は必要ありません。

ステップ 1 Logtail コンテナをデプロイする

1. Logtail 画像を引き出します。

```
docker pull registry.cn-hangzhou.aliyuncs.com/log-service/logtail
```

2. Logtail コンテナを開始します。

スタートアップテンプレートの${your_region_name}、${your_aliyun_user_id}、${your_machine_group_name}の 3 つのパラメータを置き換えてください。

```
docker run -d -v /:/logtail_host:ro -v /var/run/docker.sock:/var/run/docker.sock --env ALIYUN_LOG_TAIL_CONFIG=/etc/ilogtail/conf/${your_region_name}/ilogtail_config.json --env ALIYUN_LOG_TAIL_USER_ID=${your_aliyun_user_id} --env ALIYUN_LOG_TAIL_USER_DEFINED_ID=${your_machine_group_user_defined_id} registry.cn-hangzhou.aliyuncs.com/log-service/logtail
```

> 注：
> 構成パラメータの前に次の構成を実行してください。そうしないと、他のコンテナを削除する際に次のエラーが発生する可能性があります。container_text_file_busy。
>
> - CentOS バージョン 7.4 以降では fs.may_detach_mounts = 1が設定されています。詳細は、Bug 1468249、Bug 1441737およびissue 34538 をご参照ください。
> - Logtail に privileged 権限を与え、起動パラメータに--- privileged を追加します。 詳細は、docker run command をご参照ください。

パラメータ	説明
${your_region_name}	リージョン名。 作成したログサービスプロジェクトが存在するリージョンに置き換えます。 リージョン名については、Logtail の Linux へのインストール で使われているリージョン名をご参照ください。 注： リージョン名をリストから直接コピーすることをお勧めします。

パラメータ	説明
`$ { your_aliyu n_user_id }`	ユーザ ID。メインの Alibaba Cloud アカウントの ID で置き換えてください。 String 型の Alibaba Cloud メインアカウントの ID に置き換えてください。 ID の確認方法については、Alibaba Cloud ECS インスタンス以外または他のアカウントの ECS インスタンスからログを収集する の 2.1 をご参照ください。
`${ your_machi ne_group_u ser_define d_id }`	クラスタマシングループのユーザー定義の ID。 ユーザー定義の ID がまだ有効になっていない場合は、マシングループにユーザー定義 ID を設定するの該当する手順に従って userdefined-id を有効にします。

```
docker run -d -v /:/ logtail_ho st : ro -v / var / run /
docker . sock :/ var / run / docker . sock
-- env ALIYUN_LOG TAIL_CONFI G =/ etc / ilogtail / conf /
cn_hangzho u / ilogtail_c onfig . json -- env
ALIYUN_LOG TAIL_USER_ ID = 1654218 ******-- env ALIYUN_LOG
TAIL_USER_ DEFINED_ID = log - docker - demo registry . cn -
hangzhou . aliyuncs . com / log - service / logtail
```

注：

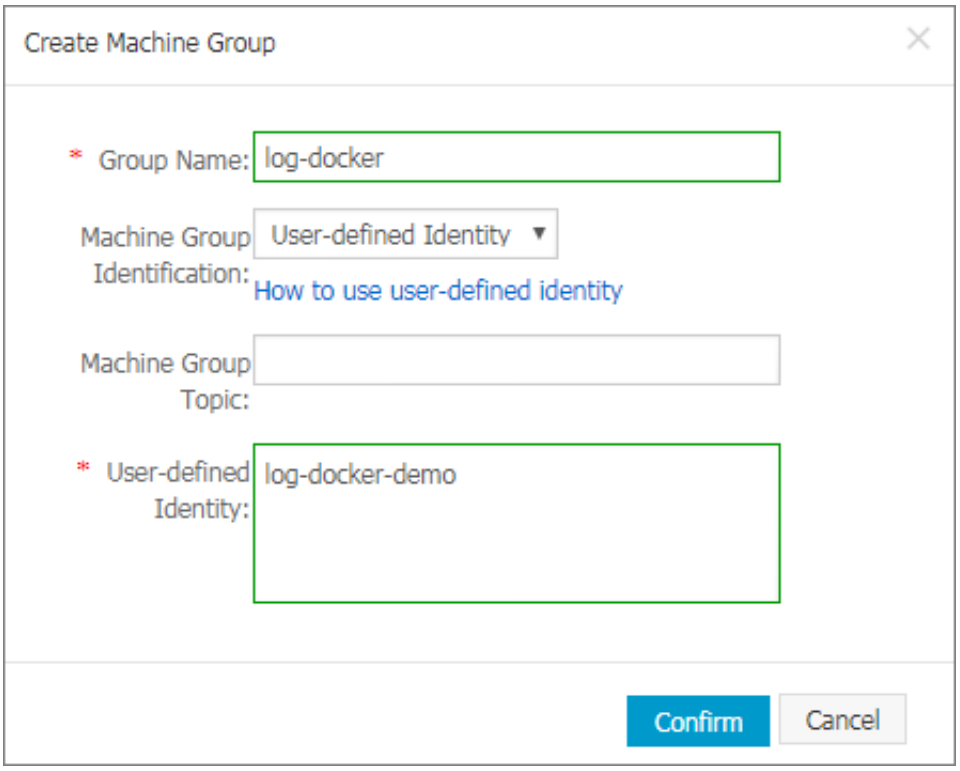
次の条件が満たされている場合は、Logtail コンテナの起動パラメータ設定をカスタマイズできます。

1. Logtail コンテナを起動するときは、`ALIYUN_LOG TAIL_USER_ DEFINED_ID`、`ALIYUN_LOG TAIL_USER_ ID`、および `ALIYUN_LOG TAIL_CONFI G` という 3 つの環境変数があります。
2. Docker のドメインソケットは `/ var / run / docker . sock` にマウントされています。
3. 他のコンテナまたはホストファイルからログを収集する場合、ルートディレクトリは Logtail コンテナの / logtail_host ディレクトリにマウントされます。
4. Logtail ログ `/ usr / local / ilogtail / ilogtail . LOG` にはパラメータ無効： `uuid = none` というエラーログがある場合、ホストマシンで *product_uu id* ファイルを作成し、有効な UUID（例： `169E98C9 - ABC0 - 4A92 - B1D2 - AA6239C0D2 61`）を入力して、Logtail コンテナの */ sys / class / dmi / id / product_uu id* パスにマウントします。

ステップ 2. マシングループを構成する

1. Log Service を有効にして、プロジェクトとログストアを作成します。 詳細については、準備 をご参照ください。
2. Log Service コンソール内の[マシングループ]ページでLogtail マシングループの作成 をクリックします。
3. ユーザー定義の ID を 「マシングループの識別」ドロップダウンリストから選択します。 前の手順で設定した `ALIYUN_LOG_TAIL_USER_DEFINED_ID` を[ユーザー定義の ID]フィールドに入力します。

   図 3-24 : マシングループの構成



[確認]をクリックして、マシングループを作成します。 1 分後、マシングループページの右側にあるマシンステータスをクリックして、デプロイされた Logtail コンテナのハートビートステータスを表示します。 詳細については、マシングループの管理のステータスの表示をご参照ください。

ステップ 3. 収集の構成情報を作成する

必要に応じてコンソールに Logtail 収集構成情報を作成します。 収集構成情報の作成方法については、以下をご参照ください。

- Container text log (recommended)
- Container standard output (recommended)

- ホストテキストファイル

  デフォルトでは、ホストのルートディレクトリは Logtail 容器の `/ logtail_ho st` ディレクトリにマウントされています。 構成パスの先頭に/ logtail_hostを付ける必要があります。 たとえば、ホストの `/ home / logs / app_log /` ディレクトリからデータを収集するには、構成ページのログパスを `/ logtail_ho st / home / logs / app_log /` に設定します。
- logtailを用いたsyslogデータの収集

その他の操作

- Logtail コンテナの動作状態を確認する

  `docker exec $ { logtail_co ntainer_id } / etc / init . d / ilogtaild status` コマンドを実行すると、Logtail の実行状態を確認できます。
- バージョン番号、IP、および Logtail の起動時間を確認する

  `docker exec $ { logtail_co ntainer_id } cat / usr / local / ilogtail / app_info . json` コマンドを実行して、Logtail に関する情報を確認できます。
- ログの実行ログを確認する

  Logtail 実行ログは、`/ usr / local / ilogtail /` ディレクトリに格納されます。 ファイル名は、 `ilogtail . LOG` です。 rotation ファイルは圧縮され、 `ilogtail . LOG . x . gz` として保存されます。

  例えば：

```
[ root @ iZbp17enxc 2us3624wex h2Z ilogtail ]# docker exec
 a287de895e 40 tail - n 5 / usr / local / ilogtail / ilogtail
 . LOG
[ 2018 - 02 - 06 08 : 13 : 35 . 721864 ] [ INFO ] [ 8 ] [ build /
 release64 / sls / ilogtail / LogtailPlu gin . cpp : 104 ] logtail
   plugin Resume : start
[ 2018 - 02 - 06 08 : 13 : 35 . 722135 ] [ INFO ] [ 8 ] [ build /
 release64 / sls / ilogtail / LogtailPlu gin . cpp : 106 ] logtail
   plugin Resume : success
[ 2018 - 02 - 06 08 : 13 : 35 . 722149 ] [ INFO ] [ 8 ] [ build /
 release64 / sls / ilogtail / EventDispa tcher . cpp : 369 ] start
   add existed check point events , size : 0
[ 2018 - 02 - 06 08 : 13 : 35 . 722155 ] [ INFO ] [ 8 ] [ build /
 release64 / sls / ilogtail / EventDispa tcher . cpp : 511 ] add
   existed check point events , size : 0 cache size : 0
 event size : 0 success count : 0
```

```
[ 2018 - 02 - 06   08 : 13 : 39 . 725417 ] [ INFO ] [ 8 ] [ build /
 release64 / sls / ilogtail / ConfigMana  ger . cpp : 3776 ]  check
   container   path   update   flag : 0   size : 1
```

コンテナの stdout は参照になりません。次の stdout の出力は無視してください。

```
 start   umount   useless   mount   points , / shm $|/ merged $|/
 mqueue $
 umount : / logtail_ho  st / var / lib / docker / overlay2 /
 3fd0043af1  74cb0273c3  c7869500fb  e2bdb95d13  b1e110172e
 f57fe840c8  2155 / merged :  must   be   superuser   to   unmount
 umount : / logtail_ho  st / var / lib / docker / overlay2 /
 d5b10aa193  99992755de  1f85d25009  528daa749c  1bf8c16edf
 f44beab6e6  9718 / merged :  must   be   superuser   to   unmount
 umount : / logtail_ho  st / var / lib / docker / overlay2 /
 5c3125dadd  acedec29df  72ad0c52fa  c800cd56c6  e880dc4e8a
 640b1e16c2  2dbe / merged :  must   be   superuser   to   unmount
......
 xargs :  umount :  exited   with   status   255 ;  aborting
 umount   done
 start   logtail
 ilogtail   is   running
 logtail   status :
 ilogtail   is   running
```

- Logtail の再起動

  Logtail を再起動するには、次の例をご参照ください。

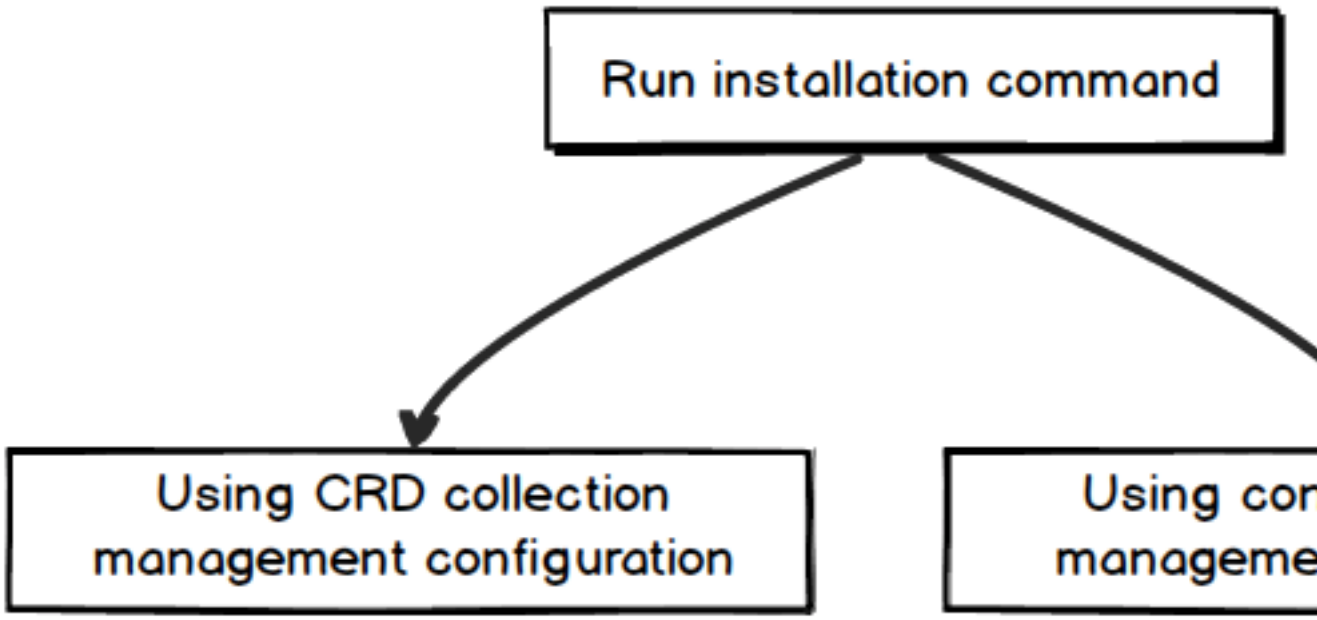
```
[ root @ iZbp17enxc  2us3624wex  h2Z   ilogtail ]#  docker   exec
 a287de895e  40  / etc / init . d / ilogtaild   stop
 kill   process   Name :  ilogtail   pid :  7
 kill   process   Name :  ilogtail   pid :  8
 stop   success
[ root @ iZbp17enxc  2us3624wex  h2Z   ilogtail ]#  docker   exec
 a287de895e  40  / etc / init . d / ilogtaild   start
 ilogtail   is   running
```

3.6.2 Kubernetes のログ収集

Log Service は、 Logtail を使用して Kubernetes クラスターログを収集し、カスタムリソース定義 (CRD) を使用して、収集設定を管理します。 ここでは、Logtailのインストール方法、及びLogtailを使用してKubernetesクラスターログを収集する方法について説明します。

設定プロセス

図 3-25 : 設定プロセス



1. インストールコマンドを実行して、alibaba-log-controller Helm パッケージをインストールします。
2. 収集設定を管理するために、必要に応じて CRD またはコンソールを選択します。

手順 1. インストール

Alibaba Cloud Container Service に Kubernetes をインストールする

インストール手順

1. Alibaba Cloud Container Service Kubernetes のマスターノードにログインします。 ログインの詳細については、「SSH による Kubernetes クラスターへのアクセス」をご参照ください。
2. `${ your_k8s_c luster_id }` を Kubernetes クラスタ ID に置き換え、次のコマンドを実行します。

```
wget http :// logtail - release . oss - cn - hangzhou . aliyuncs
. com / linux64 / alicloud - log - k8s - install . sh - O
alicloud - log - k8s - install . sh ; chmod 744 ./ alicloud -
log - k8s - install . sh ; sh ./ alicloud - log - k8s - install .
sh ${ your_k8s_c luster_id }
```

インストール後、 Log ServiceはKubernetesクラスタの同じリージョンにLog Serviceプロジェクトを自動的に作成します。 作成されるプロジェクトの名前は `k8s - log -${`

`your_k8s_c luster_id }` です。プロジェクトに、マシングループ `k8s - group - ${ your_k8s_c luster_id }`が自動的に作成されます。

> 注：
> Under `k8s - log -${ your_k8s_c luster_id }` プロジェクトに、 `config - operation - log` というという名前の Logstore が自動的に作成されます。 Logstore を削除しないでください。

インストールの例

実行が成功すると、次の情報が出力されます。

```
[ root @ iZbp ****** biaZ ~]# wget http :// logtail - release .
 oss - cn - hangzhou . aliyuncs . com / linux64 / alicloud - log - k8s
 - install . sh - O alicloud - log - k8s - install . sh ; chmod
 744 ./ alicloud - log - k8s - install . sh ; sh ./ alicloud - log
 - k8s - install . sh c12ba20 ********** 86939f0b
....
....
....
 alibaba - cloud - log / Chart . yaml
 alibaba - cloud - log / templates /
 alibaba - cloud - log / templates / _helpers . tpl
 alibaba - cloud - log / templates / alicloud - log - crd . yaml
 alibaba - cloud - log / templates / logtail - daemonset . yaml
 alibaba - cloud - log / templates / NOTES . txt
 alibaba - cloud - log / values . yaml
 NAME : alibaba - log - controller
 LAST DEPLOYED : Wed May 16 18 : 43 : 06 2018
 NAMESPACE : default
 STATUS : DEPLOYED
 RESOURCES :
==> v1beta1 / ClusterRol eBinding
 NAME AGE
 alibaba - log - controller 0s
==> v1beta1 / DaemonSet
 NAME DESIRED CURRENT READY UP - TO - DATE AVAILABLE NODE
   SELECTOR AGE
 logtail 2 2 0 2 0 0s
==> v1beta1 / Deployment
 NAME DESIRED CURRENT UP - TO - DATE AVAILABLE AGE
 alibaba - log - controller 1 1 1 0 0s
==> v1 / Pod ( related )
 NAME READY STATUS RESTARTS AGE
 logtail - ff6rf 0 / 1 ContainerC reating 0 0s
 logtail - q5s87 0 / 1 ContainerC reating 0 0s
 alibaba - log - controller - 7cf6d7dbb5 - qvn6w 0 / 1 ContainerC
 reating 0 0s
==> v1 / ServiceAcc ount
 NAME SECRETS AGE
 alibaba - log - controller 1 0s
==> v1beta1 / CustomReso urceDefini tion
名前年齢
 aliyunlogc onfigs . log . alibabaclo ud . com 0s
==> v1beta1 / ClusterRol e
 alibaba - log - controller 0s
```

```
[ SUCCESS ] install helm package : alibaba - log - controller
 success .
```

`helm Status alibaba - log - controller` を実行して、Pod の現在のステータスを確認できます。 すべてのステータスが成功すると、インストールは成功です。

インストールが正常に完了したら、Log Service コンソールにログインします。 自動的に作成された Log Service プロジェクトが、コンソールに表示されます。(プロジェクトがたくさんある場合は、キーワード `k8s - log` で検索してください。)

自分で構築した Kubernetes をインストールする

制限事項

1. Kubernetes クラスタは、バージョン 1.8 以降でなければなりません。
2. Helm 2.6.4 以降がインストールされていなければなりません。

インストール手順

1. Log Service コンソールで、プロジェクトを作成します。 プロジェクト名は、`k8s - log - custom -`で始める必要があります。
2. 次のコマンドで、パラメーターをご自分のパラメーターに置き換え、コマンドを実行します。

```
wget http :// logtail - release . oss - cn - hangzhou . aliyuncs
. com / linux64 / alicloud - log - k8s - custom - install . sh -
O alicloud - log - k8s - custom - install . sh ; chmod 744
./ alicloud - log - k8s - custom - install . sh ; sh ./ alicloud
- log - k8s - custom - install . sh { your - project - suffix }
{ region - id } { aliuid } { access - key - id } { access - key -
secret }
```

パラメーターと説明は次のとおりです。

名前	説明
{your-project-suffix}	2 番目の手順で作成したプロジェクト名の `k8s - log - custom -`の後の部分。たとえば、作成したプロジェクトが `k8s - log - custom - xxxx` の場合 、 `xxxx` を入力します。
{regionId}	プロジェクトが配置されているリージョンの ID。 Service endpointを表示できます。たとえば、`中国（杭州）` のリージョン ID は、`cn - hangzhou` です。

名前	説明
{aliuid}	ユーザー ID は、Alibaba Cloud マスターアカウントのユーザー IDで置き換えてください。 マスターアカウントのユーザーIDはString型で、ID の表示方法については、ユーザー ID 設定のセクション 2.1 をご参照ください。
{access-key-id}	アカウントアクセスキー ID。 サブアカウントアクセスキーを使用し、 権限を与えることを推奨します。概要
{access-key-secret}	アカウントのアクセスキーシークレット。 サブアカウントの AccessKey を使用し、AliyunLogFullAccess 権限を与えることを推奨します。 詳細については、概要を参照してください。

インストール後、 Log Service はマシングループをプロジェクトに自動的に作成します。 マシングループ名は、 `k8s - group -${ your_k8s_c luster_id }`です。

注：

- Logstore `config - operation - log` は、k8s-log-${your_k8s_cluster_id} プロジェクトに自動的に作成されます。 この Logstore は削除しないでください。
- 自分で構築した kubernetes のインストールの後、Logtail は `privileged` 権限を付与します。それは他のPOD を削除中に `container text file busy` にならないようにするためです。 詳細については、bug 1468249、bug 1441737 、およびissue 34538をご参照ください。

インストールの例

実行に成功すると、次のように出力されます。

```
[ root @ iZbp1dsxxx xxqfbiaZ ~]# wget http :// logtail - release
 . oss - cn - hangzhou . aliyuncs . com / linux64 / alicloud - log
 - k8s - custom - install . sh - O alicloud - log - k8s - custom
 - install . sh ; chmod 744 ./ alicloud - log - k8s - custom -
 install . sh ; sh ./ alicloud - log - k8s - custom - install .
 sh xxxx cn - hangzhou 165xxxxxxx x050 LTAxxxxxxx xxxx
 AIxxxxxxxx xxxxxxxxxx xxxxxxxxxx xe
....
....
....
 NAME : alibaba - log - controller
 LAST DEPLOYED : Fri May 18 16 : 52 : 38 2018
 NAMESPACE : default
 STATUS : DEPLOYED
```

```
 RESOURCES :
==>  v1beta1 / ClusterRol  eBinding
 NAME   AGE
 alibaba - log - controller   0s
==>  v1beta1 / DaemonSet
 NAME   DESIRED   CURRENT   READY   UP - TO - DATE   AVAILABLE   NODE
   SELECTOR   AGE
 logtail - ds   2   2   0   2   0    0s
==>  v1beta1 / Deployment
 NAME   DESIRED   CURRENT   UP - TO - DATE   AVAILABLE   AGE
 alibaba - log - controller   1   1   1   0   0s
==>  v1 / Pod ( related )
 NAME   READY   STATUS   RESTARTS   AGE
 logtail - ds - 7xf2d   0 / 1   ContainerC  reating   0   0s
 logtail - ds - 9j4bx   0 / 1   ContainerC  reating   0   0s
 alibaba - log - controller - 796f8496b6 - 6jxb2   0 / 1   ContainerC
 reating   0   0s
==>  v1 / ServiceAcc  ount
 NAME   SECRETS   AGE
 alibaba - log - controller   1   0s
==>  v1beta1 / CustomReso  urceDefini  tion
 NAME   AGE
 aliyunlogc  onfigs . log . alibabaclo  ud . com   0s
==>  v1beta1 / ClusterRol  e
 alibaba - log - controller   0s
[ INFO ]  your   k8s   is   using   project  :  k8s - log - custom
 - xxx ,  region  :  cn - hangzhou ,  aliuid  :  1654218 *********,
 accessKeyI  d  :  LTAxxxxxxx  xxxx
[ SUCCESS ]  install   helm   package  :  alibaba - log - controller
 success .
```

`helm status alibaba - log - controller` を使用してPodの現行状態をチェックできます。 すべてのステータスが成功すると、インストールは完了です。

インストール後、Log Service コンソールにログインします。 自動的に作成された Log Service プロジェクトを表示できます。 プロジェクトがたくさんある場合は、キーワード `k8s - log` で検索してください。

手順 2. 設定

ログ収集は、デフォルトでコンソール設定モードをサポートしています。 また、Kubernetes マイクロサービス開発の CRD 設定モードも提供されています。 kubectl を使用して設定を管理できます。 2 つの設定の比較は次のとおりです。

-	CDRモード	コンソールモード
操作上の複雑さ	低	中
機能	コンソールモードを除く、高度な設定をサポート。	中
複雑さ	中	低
ネットワーク接続	Kubernetes クラスタに接続	インターネットに接続

-	CDRモード	コンソールモード
デプロイメントコンポーネントとの連携	サポート	未サポート
認証方法	Kubernetes 認証	クラウドアカウント認証

Kubernetes のデプロイメント、および公開プロセスと連携されているため、収集設定管理に CRD メソッドを使用することを推奨します。

コンソール上でコレクションの設定を管理する

必要に応じてコンソールにLogtail収集設定を作成します。 設定手順については、以下をご参照ください。

- コンテナテキストログ（推奨）
- コンテナスタンダードアウトプット（推奨）
- ホストテキストファイル

  デフォルトでは、ホストのルートディレクトリはLogtail容器の`/ logtail_ho st`ディレクトリにマウントされています。 パスを設定する際、このプレフィックスを追加する必要があります。 たとえば、ホストの`/ home / logs / app_log /`ディレクトリからデータを収集するには、構成ページのログパスを`/ logtail_ho st / home / logs / app_log /`に設定します。
- logtailを用いたsyslogデータの収集

CRD 管理による取得設定

kubernetes マイクロサービス開発モデルでは、ログサービスも CRD を設定する方法を提供し、kubectl を直接使用して設定を管理できます。また、kubernetes デプロイメントと連携し、公開プロセスをより完全に行うことができます。

詳細については、CRD での Kubernetes ログ収集の設定を参照してください。

その他の操作

Glasonset デプロイメントの移行手順

以前に使用した WebSphere の set メソッドを使用して Log Service をデプロイした場合、設定管理に CRD を使用することはできません。 次の方法で新しいバージョンに移行できます。

注：

アップグレード中に、いくつかのログが複製されます。 CRD 管理設定は、CRD を使用して作成された設定に対してのみ使用できます。 履歴設定は、 CRD モードを使用して作成されていないため、CRD 管理モードはサポートしていません。

1. 新しいバージョンの形式でインストールすると、インストールコマンドには、以前のkubernetes クラスタで使用されていた Log Service プロジェクト名のパラメーターが最後に追加されます。

   たとえば、プロジェクト名が `k8s - log - demo` で、クラスタ ID が `c12ba2028c xxxxxxxxxx 6939f0b` の場合、インストールコマンドは次のようになります。

   ```
   wget
            http :// logtail - release . oss - cn - hangzhou .
   aliyuncs . com / linux64 / alicloud - log - k8s - install . sh  -
   O
            alicloud - log - k8s - install . sh ;  chmod   744  ./
   alicloud - log - k8s - install . sh ;  sh
           ./ alicloud - log - k8s - install . sh   c12ba2028c
   xxxxxxxxxx  6939f0b   k8s - log - demo .
   ```

2. インストールが成功したら、Log Service コンソールで、履歴収集設定を新しいマシングループ `k8s - group -${ your_k8s_c luster_id }`に適用します。
3. 1 分後、履歴収集設定はマシングループの履歴にバインドされます。
4. ログ収集が正常であれば、以前にインストールした Logtail DaemonSet を削除できます。

同じ Log Service プロジェクトで複数のクラスタを使用する

複数のクラスタを使用して、同じ Log Service プロジェクトにログを収集することができます。他のクラスタ Log Service コンポーネントをインストールする際、インストールパラメーターの`${ your_k8s_c luster_id }`を、最初にインストールしたクラスタ ID に置き換える必要があります。

たとえば、 ID が abc001、abc002、および abc003 の 3 つのクラスタがあるとします。 3 つのクラスタのインストールパラメーター`${ your_k8s_c luster_id }`は、すべて `abc001` にする必要があります。

 注：

リージョン間の Kubernetes マルチクラスタ共有では、この方法はサポートしていません。

Logtail コンテナログ

Logtail ログは、Logtail コンテナの/ usr / local / ilogtail /ディレクトリに格納されており、ファイル名は、 ilogtail . LOG および ilogtail . plugin です。コンテナの stdout に参照の意味はありません。そのため、次の stdout の出力を無視できます。

```
start umount useless mount points , / shm $|/ merged $|/
mqueue $
umount : / logtail_ho st / var / lib / docker / overlay2 /
3fd0043af1 74cb0273c3 c7869500fb e2bdb95d13 b1e110172e
f57fe840c8 2155 / merged : must be superuser to unmount
umount : / logtail_ho st / var / lib / docker / overlay2 /
d5b10aa193 99992755de 1f85d25009 528daa749c 1bf8c16edf
f44beab6e6 9718 / merged : must be superuser to unmount
umount : / logtail_ho st / var / lib / docker / overlay2 /
5c3125dadd acedec29df 72ad0c52fa c800cd56c6 e880dc4e8a
640b1e16c2 2dbe / merged : must be superuser to unmount
......
xargs : umount : exited with status 255 ; aborting
umount done
start logtail
ilogtail is running
logtail status :
ilogtail is running
```

Kubernetes クラスタ内のログ関連コンポーネントのステータスを表示する

```
helm status alibaba - log - controller
```

alibaba-log-controller の起動に失敗した場合は？

次のようにインストールを実行してください。

1. インストールコマンドは、kubernetes クラスタのマスターノードで実行されます
2. インストールコマンドパラメーターは、クラスタ ID に入力されます。

これらの問題によりインストールが失敗した場合は、 helm del — purge alibaba - log - controller r を使用して、インストールパッケージを削除し、再度インストールを実行してください。

インストールの失敗が続く場合は、チケットを起票し、サポートセンターへお問い合わせください。

Kubernetes クラスタの Logtail DaemonSet のステータスを確認する

kubectl get ds - n kube - system コマンドを実行し、Logtail の実行状態を確認できます。

注：

Logtail のデフォルトの namespace は、 kube - system です。

Logtail のリソース制限を調整する方法

デフォルトでは、Logtail は最大 40% の CPU と 200M の RAM しか占有できません。 処理速度を上げる必要がある場合は、次の 2 つのセクションでパラメーターを調整する必要があります。

- YAMLテンプレートの `resources` の `limits` と `requests` 。
- Logtail 起動設定ファイルのパスは、YAML テンプレートの `ALIYUN_LOG_TAIL_CONFIG` 環境変数です。 変更方法については、「Logtail 起動設定パラメーター」 をご参照ください。

Logtail DaemonSet の強制更新

`logtail-daemonset.yaml` ファイルを修正後、次のコマンドを実行して、Logtail DaemonSet を強制的に更新します。

```
kubectl --namespace=kube-system delete ds logtail
kubectl apply -f ./logtail-daemonset.yaml
```

注:

強制更新中にデータの重複が発生することがあります。

Logtail DaemonSet の設定情報を確認する

```
'kubectl describe ds logtail -n kube-system'
```

Logtail のバージョン番号、IP、開始時刻などを確認する

例は次のとおりです。

```
[root@iZbp1dsu6v77zfb40qfbiaZ ~]# kubectl get po -n
kube-system -l k8s-app=logtail
NAME READY STATUS RESTARTS AGE
logtail-gb92k 1/1 Running 0 2h
logtail-wm7lw 1/1 Running 0 4d
[root@iZbp1dsu6v77zfb40qfbiaZ ~]# kubectl exec logtail-
gb92k -n kube-system cat /usr/local/ilogtail/app_info
.json
{
"UUID": "",
"hostname": "logtail-gb92k",
"instance_id": "**********************************",
"ip": "*.*.*.*",
"logtail_version": "0.16.2",
"os": "Linux; 3.10.0-693.2.2.el7.x86_64 ; # 1
SMP Tue Sep 12 22:26:13 UTC 2017; x86_64",
"update_time": "2018-02-05 06:09:01"
```

```
}
```

Logtail の実行ログを表示する

Logtail 実行ログは、/ usr / local / ilogtail /ディレクトリに格納されます。ファイル名は、 ilogtail . LOG です。ファイルは圧縮され、 ilogtail . LOG . x . gz として保存されます。

例は次のとおりです。

```
[ root @ iZbp1dsu6v  77zfb40qfb  iaZ  ~]#  kubectl   exec   logtail
 - gb92k  - n   kube - system   tail  / usr / local / ilogtail /
 ilogtail . LOG
[ 2018 - 02 - 05   06 : 09 : 02 . 168693 ] [ INFO ] [ 9 ] [ build /
 release64 / sls / ilogtail / LogtailPlu  gin . cpp : 104 ]  logtail
 plugin   Resume : start
[ 2018 - 02 - 05   06 : 09 : 02 . 168807 ] [ INFO ] [ 9 ] [ build /
 release64 / sls / ilogtail / LogtailPlu  gin . cpp : 106 ]  logtail
 plugin   Resume : success
[ 2018 - 02 - 05   06 : 09 : 02 . 168822 ] [ INFO ] [ 9 ] [ build /
 release64 / sls / ilogtail / EventDispa  tcher . cpp : 369 ]  start
 add   existed   check   point   events ,  size : 0
[ 2018 - 02 - 05   06 : 09 : 02 . 168827 ] [ INFO ] [ 9 ] [ build /
 release64 / sls / ilogtail / EventDispa  tcher . cpp : 511 ]  add
   existed   check   point   events ,  size : 0   cache   size : 0
 event   size : 0   success   count : 0
```

pod の Logtail を再起動する

例は次のとおりです。

```
[ root @ iZbp1dsu6v  77zfb40qfb  iaZ  ~]#  kubectl   exec   logtail -
 gb92k  - n   kube - system  / etc / init . d / ilogtaild   stop
 kill   process   Name :  ilogtail   pid :  7
 kill   process   Name :  ilogtail   pid :  9
 stop   success
[ root @ iZbp1dsu6v  77zfb40qfb  iaZ  ~]#  kubectl   exec   logtail -
 gb92k  - n   kube - system  / etc / init . d / ilogtaild   start
 ilogtail   is   running
```

3.6.3 コンテナテキストログ

Logtail は、コンテナで生成されたテキストログを収集し、収集されたログをコンテナメタデータと一緒に Log Service にアップロードします。

機能の特徴

基本的なログファイル収集と比較して、Docker ファイル収集は、次の機能もサポートしています。

- コンテナ内のログパスの設定。このパスからホストへのマッピングを気にする必要はありません。

- Label を使用した収集するコンテナの指定。
- Label を使用した特定のコンテナの除外。
- 収集するコンテナを指定する environment。
- 特定のコンテナを除外する environment。
- 複数行ログ (Java スタックログなど) 。
- コンテナデータの自動タグ付け。
- Kubernetes コンテナの自動タグ付け。

制限事項

- 収集を停止するポリシー。 コンテナが停止すると、Logtail は (1-3 秒の遅延で) コンテナの `die` イベントをリッスンした後、コンテナからのログの収集を停止します。 この間に収集遅延が発生すると、停止前にログの一部を失う可能性があります。
- Logtail の実行方法。 Logtail はコンテナとして実行し、Logtail のデプロイメント方法に従う必要があります。
- Label。 Label は、Docker inspect のラベル情報であり、Kubernetes 設定のラベルではありません。
- Environment。 Environment は、コンテナの起動時に設定された環境情報です。

手順

1. Logtail コンテナをデプロイして設定します。
2. Log Service で収集設定を行います。

1. Logtail のデプロイメントと設定

- Kubernetes

  Kubernetes ログ収集の詳細については、Kubernetes のログ収集をご参照ください 。
- その他のコンテナ管理メソッド

  Swarm や Mesos などの他のコンテナ管理メソッドの詳細については、標準の Docker ログの収集をご参照ください。

2. Log Service の収集設定

1. Logstore リストページで、データ・インポート・ウィザードアイコンをクリックして設定プロセスを入力します。
2. データソースを選択します。

   サードパーティ製のソフトウェアでDocker ファイルを選択し、[次へ] をクリックします。

3. データソースを設定します。

設定項目	必須かどうか	説明
Docker ファイル	はい	収集する該当ファイルが Docker ファイルであるかどうかを確認します。
ホワイトリストにラベルを付ける	オプション	LabelKey は必須です。 LabelValue が空でない場合、LabelKey = LabelValue がラベルに含まれているコンテナのみが収集されます。 LabelValue が空の場合、LabelKey がラベルに含まれているすべてのコンテナが収集されます。  注 : a. キーと値のペアには互いに OR 関係があります。つまり、ラベルにキーと値のペアのいずれかが含まれていればコンテナは収集されます。 b. ここでは、ラベルは Docker inspect のラベル情報です。
ブラックリストに Label を付ける	オプション	LabelKey は必須です。 LabelValue が空でない場合、LabelKey = LabelValue がラベルに含まれているコンテナのみが除外されます。 LabelValue が空の場合、LabelKey がラベルに含まれているすべてのコンテナが除外されます。  注 : a. キーと値のペアには互いに OR 関係があります。つまり、 ラベルにキーと値のペアのいずれかが含まれている場合、コンテナは除外されます。 b. ここでは、ラベルは Docker inspect のラベル情報です。

設定項目	必須かどうか	説明
環境変数ホワイトリスト	オプション	EnvKey は必須です。 EnvValue が空でない場合、EnvKey = EnvValue が環境変数に含まれているコンテナのみが収集されます。 EnvValue が空の場合、EnvKey が環境変数に含まれているすべてのコンテナが収集されます。 注： ・ キーと値のペアは相互に OR 関係を持ちます。つまり、環境変数にキーと値のペアのいずれかが含まれている場合、コンテナは収集されます。 ・ ここでは、環境変数とは、コンテナの起動時に設定される環境情報です。
環境変数ブラックリスト	オプション	EnvKeyis は必須です。 EnvValue が空でない場合、EnvKey = EnvValue が環境変数に含まれているコンテナのみが除外されます。 EnvValue が空の場合、EnvKey が環境変数に含まれているすべてのコンテナが除外されます。 注： a. キーと値のペアは相互に OR 関係を持ちます。環境変数にキーと値のペアのいずれかが含まれている場合、コンテナが収集されます。 b. ここでは、環境変数とは、コンテナの起動時に設定される環境情報です。
その他の設定	-	その他の収集設定とパラメーターの説明については、「テキストファイルの収集」をご参照ください。

4. マシングループに適用します。

   [マシングループに適用] ページで、収集する Logtail マシングループを選択し、マシングループに適用をクリックして、選択したマシングループに設定を適用します。 マシングループを作成していない場合は、マシングループの作成をクリックしてマシングループを作成します。

5. コンテナのテキストログにアクセスするプロセスを完了します。

   [検索]、[分析]、[可視化]、および [Shipper と ETL] 機能を設定するために、ページの指示どおりに設定を完了します。

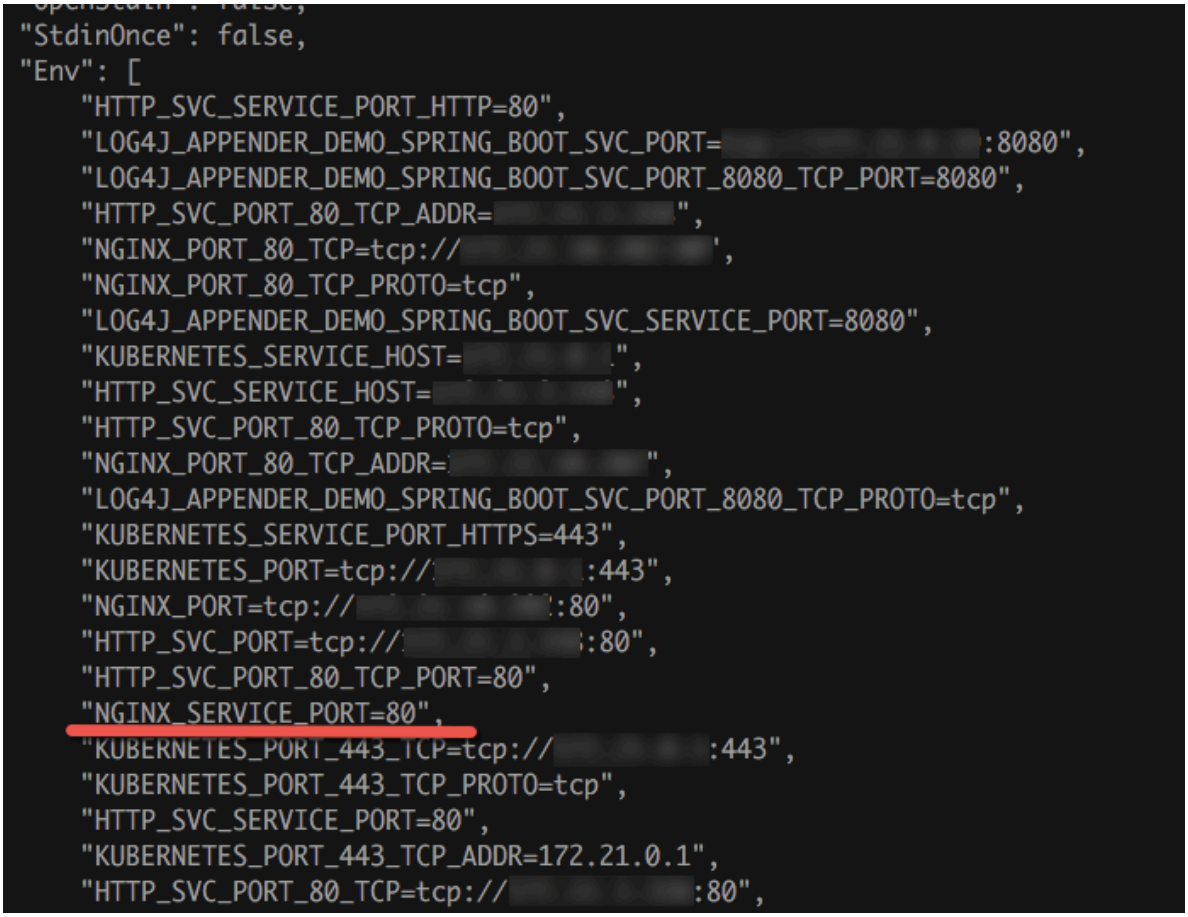
設定例

- Environment 設定

  Environment が `NGINXPORT_ 80_TCP_POR T = 80` で、 `POD_NAMESP ACE = kube - system` ではないコンテナのログを収集します。 ログファイルパスは、`/ var / log / nginx / access . log` で、ログはシンプルモードで解析されます。

  > 注 :
  > Environment は、コンテナの起動時に設定された環境情報です。

  図 3-26 : Environment 設定の例



この例のデータソースの設定は、次のとおりです。 その他の収集設定とパラメーターの説明については、「テキストファイルの収集」をご参照ください。

- Label 設定

  Label が `io . kubernetes . container . name = nginx` で、type=pre ではないコンテナのログを収集します。 ログファイルのパスは`/ var / log / nginx / access . log` で、ログはシンプルモードで解析されます。

  > 注：
  > Label は、Docker inspect のラベル情報であり、Kubernetes 設定のラベルではありません。

  図 3-27 : Label モードの例

  この例のデータソースの設定は、次のとおりです。 その他の収集設定とパラメーターの説明については、「テキストファイルの収集」をご参照ください。

デフォルトフィールド

標準 Docker

各ログによって次のフィールドが、デフォルトでアップロードされます。

フィールド	説明
`_image_nam e_`	イメージ名。
`_container _name_`	コンテナ名。

Kubernetes

クラスターが Kubernetes クラスターの場合、各ログによって次のフィールドがデフォルトでアップロードされます。

フィールド	説明
`_image_nam e_`	イメージ名。
`_container _name_`	コンテナ名。
`_pod_name_`	pod 名。
`_namespace _`	pod の名前空間。
`_pod_uid_`	pod の一意の識別子。

3.6.4 コンテナ標準出力

Logtail では、コンテナの標準出力ストリームを入力ソースとして使用し、標準出力ストリームをコンテナメタデータと一緒に Log Service にアップロードすることができます。

機能の特徴

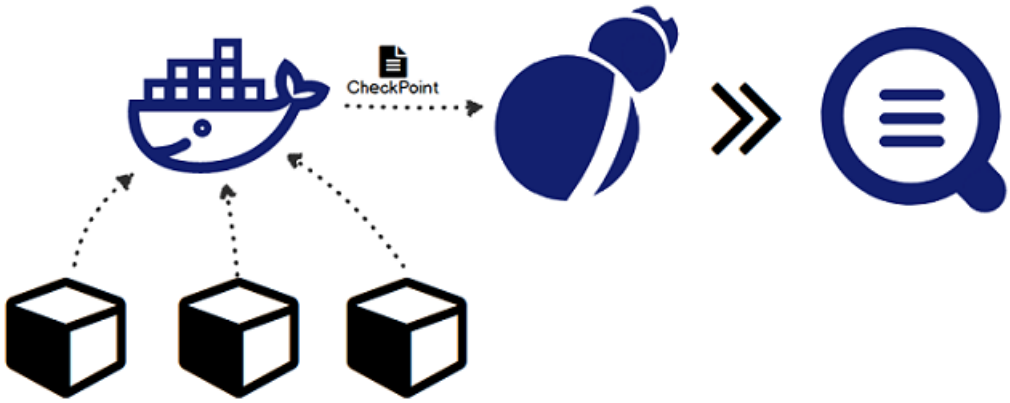
- stdout と stderr の収集をサポートします。
- ラベルを使用して収集するコンテナを指定することをサポートします。
- ラベルを使用して特定のコンテナを除外することをサポートします。
- environments を使用して収集するコンテナを指定することをサポートします。
- environments を使用して収集するコンテナを除外することをサポートします。
- 複数行ログ（Java スタックログなど）をサポートします。
- コンテナデータの自動タグ付けをサポートします。
- Kubernetes コンテナの自動タグ付けをサポートします。

実装の原則

次の図に示すように、Logtail は Docker の Domain Socket と通信し、 Docker 上で実行されているすべてのコンテナを照会し、ラベル情報に従って収集するコンテナを特定します。Logtail は、Docker log コマンドを使用して、指定されたコンテナログを取得します。

Logtail は、コンテナの標準出力を収集するときに定期的に収集したポイント情報をチェックポイントファイルに保存します。 Logtail が停止後に再起動されると、ログは最後に保存されたポイントから収集されます。

図 3-28 : 実装の原則



制限

- 現在、この機能は Linux のみをサポートしており、Logtail 0.16.0 以降のバージョンに依存しています。 バージョンの確認とアップグレードについては、Logtail の Linux へのインストールをご参照ください。
- デフォルトでは、Logtail は`/ var / run / docker . sock`を使用して Docker Engine にアクセスします。 ドメインソケットが存在し、アクセス権があることを確認します。
- 複数行ログの制限。 複数の行で構成されるログが出力遅延のために分割されないようにするため、最後に収集された複数行のログはデフォルトで短時間キャッシュされます。 デフォルトのキャッシュ時間は 3 秒ですが、`BeginLineT imeoutMs`パラメータを使用して変更できます。 ただし、この値は 1000 未満にすることはできません。 そうしないと、エラーが発生する可能性があります。
- 収集を停止するための戦略。 コンテナが停止すると、Logtail はコンテナの`die`イベントをリスニング後、コンテナからの標準出力の収集を停止します。 この間に収集遅延が発生すると、停止前に出力の一部を失う可能性があります。
- Context limit。 デフォルトでは、コレクション構成は同じコンテキストにあります。 コンテナの各タイプごとに異なるコンテキストを設定するには、各タイプの収集設定を作成します。
- データ処理。収集されたデータのデフォルトのフィールドは`content`で、これは共通の処理設定をサポートします。

- Label。 Label は、Docker 検査のラベル情報であり、Kubernetes 構成のラベルではありません。
- Environment。Environment は、コンテナの起動時に構成された環境情報です。

構成プロセス

1. Logtail コンテナをデプロイして構成します。
2. ログサービスで収集の構成を行います。

1.Logtail コンテナをデプロイして構成する

- Kubernetes
- その他コンテナ管理方法

2.ログサービスで収集の構成を行う

1. Logstore List ページで、Data Import Wizard アイコンをクリックし、設定プロセスに入ります。
2. データソースを選択します。

   サードパーティ製ソフトウェアの Docker Stdout を選択してから、Next をクリックします。
3. データソースを設定します。

   データソースの設定ページで、コレクションの設定を完了します。 次の例をご参照ください。

```
{
 " inputs ": [
     {
         " type ": " service_do  cker_stdou  t ",
         " detail ": {
             " Stdout ":  true ,
             " Stderr ":  true ,
             " IncludeLab  el ": {
                 " io . kubernetes . container . name ": " nginx "
             },
             " ExcludeLab  el ": {
                 " io . kubernetes . container . name ": " nginx -
 ingress - controller "
             },
             " IncludeEnv ": {
                 " NGINX_SERV  ICE_PORT ": " 80 "
             },
             " ExcludeEnv ": {
                 " POD_NAMESP  ACE ": " kube - system "
             }
         }
     }
 ]
```

```
}
```

4. マシングループに適用します。

   マシングループに適用ページで、収集する Logtail マシングループの作成 を選択し、マシングループに適用をクリックして、選択したマシングループに設定を適用します。マシングループを作成していない場合は、マシングループの作成をクリックしてマシングループを作成します。

構成項目の説明

入力ソースタイプは `service_do cker_stdou t` です。

設定項目	型	必須	説明
IncludeLabel	マッピングタイプ。キーと値の両方がStringです。	はい	デフォルトでは空です。 空の場合、すべてのコンテナが収集されます。キーが空ではなく値が空の場合、ラベルにこのキーが含まれているすべてのコンテナが収集されます。 注： 1. キーと値のペアには OR 関係があります。つまり、ラベルにキーと値のペアが含まれていればコンテナが収集されます。 2. ここで、ラベルは Docker inspect のラベル情報です。
ExcludeLabel	マッピングタイプ。キーと値の両方が String です。	いいえ	デフォルトでは空です。 空の場合、コンテナは除外されません。 キーが空ではなく値が空の場合、ラベルにこのキーが含まれるすべてのコンテナが除外されます。 注： 1. キーと値のペアには互いに OR 関係があります。 つまり、ラベルにキーと値のペアが含まれている場合、コンテナは除外されます。 2. ここで、ラベルは Docker inspect のラベル情報です。

設定項目	型	必須	説明
IncludeEnv	マッピングタイプ。キーと値の両方が String です。	いいえ	デフォルトでは空です 空の場合、すべてのコンテナが収集されます。キーが空ではなく値が空の場合、このキーを含む環境を持つすべてのコンテナが収集されます。 注： 1. キーと値のペアは相互に OR 関係を持ちます。つまり、環境にキー/値のペアが含まれている場合、コンテナが収集されます。 2. 環境は、コンテナの起動時に構成された環境情報です。
ExcludeEnv	マッピングタイプ。キーと値の両方が String です。	いいえ	デフォルトでは空です 空の場合、コンテナは除外されません。 キーが空ではなく値が空の場合、このキーを含む環境を持つすべてのコンテナが除外されます。 注： 1. キーと値のペアには OR 関係があります。つまり、環境にキーと値のペアが含まれている場合、コンテナは除外されます。 2. 環境は、コンテナの起動時に構成された環境情報です。
Stdout	ブール	いいえ	デフォルトでは True です。 false の場合、stdout データは収集されません。
Stderr	ブール	いいえ	デフォルトでは True です。 false の場合、stderr データは収集されません。
BeginLineRegex	String	いいえ	デフォルトでは空です。 空でない場合は、行の先頭に一致する正規表現です。 この正規表現が行と一致する場合、その行は新しいログとして扱われます。 それ以外の場合、データの行は前のログに接続されます。

設定項目	型	必須	説明
BeginLineTimeoutMs	int	いいえ	行の先頭と一致するためのタイムアウト（ミリ秒単位）。デフォルト値は 3000 です。 新しいログが 3 秒以内に表示されない場合は、最後のログが出力されます。
BeginLineCheckLength	int	いいえ	正規表現との一致に使用される行の先頭の長さ（バイト単位）。 デフォルト値は 10 \ 1024 です。 正規表現が最初の N バイト以内の行と一致する場合は、このパラメータを設定して一致効率を上げます。
MaxLogSize	int	いいえ	ログの最大長（バイト単位）。 デフォルト値は 512 \ 1024 です。 ログの長さが設定された値を超えると、一致した行の先頭を検索することなく、ログが直接アップロードされます。

デフォルトフィールド

ノーマルドッカー

次のフィールドは、デフォルトで各ログによってアップロードされます。

フィールド名	説明
`_time_`	データ時間。 たとえば、`2018 - 02 - 02T02 : 18 : 41 . 979147844Z` となります。
`_source_`	入力ソースタイプ（stdout または stderr）。
`_image_nam e_`	イメージ名。
`_container _name_`	コンテナ名。

Kubernetes

クラスタが Kubernetes クラスタの場合、デフォルトで各ログによって次のフィールドがアップロードされます。

フィールド名	説明
time	データ時間。 たとえば、 2018 - 02 - 02T02 : 18 : 41 . 979147844Z となります。
source	入力ソース・タイプ（stdout または stderr のいずれか）。
_image_nam e_	イメージ名。
_container _name_	コンテナ名。
_pod_name_	ポッド名。
_namespace _	ポッドが存在する名前空間。
_pod_uid_	ポッドの一意の識別子。

設定例

一般的な設定

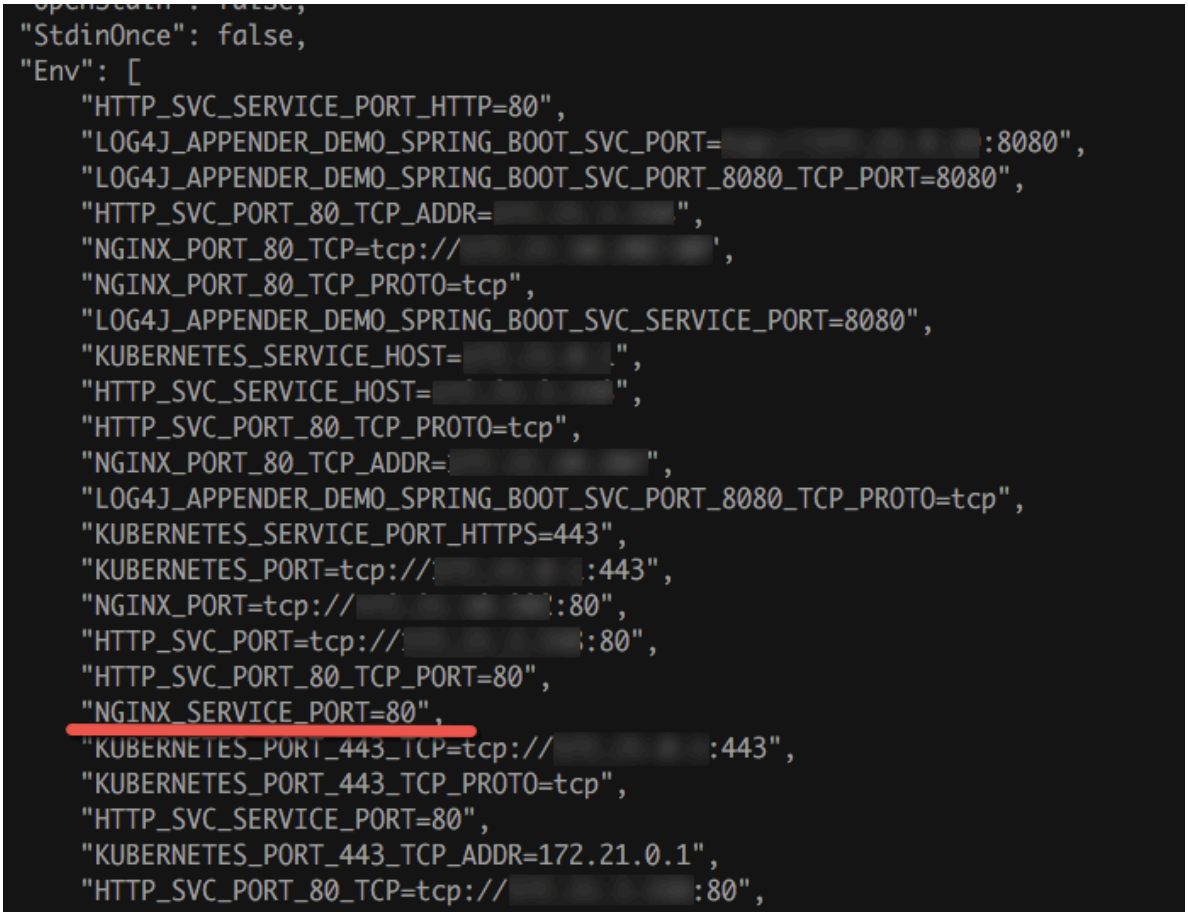
- Environment の構成

  Environment が NGINX_PORT _80_TCP_PO RT = 80 で、 POD_NAMESP ACE = kube - system ではないコンテナの stdout ログと stderr ログを収集する：

   注：

Environment は、コンテナの起動時に構成された環境情報です。

図 3-29 : Environment 構成の例



収集の構成

```
{
    " inputs ": [
        {
            " type ": " service_do  cker_stdou  t ",
            " detail ": {
                " Stdout ":  true ,
                " Stderr ":  true ,
                " IncludeEnv ": {
                    " NGINX_PORT  _80_TCP_PO  RT ": " 80 "
                },
                " ExcludeEnv ": {
                    " POD_NAMESP  ACE ": " kube - system "
                }
            }
        }
    ]
```

```
}
```

- Label 構成

  Label が `io . kubernetes . container . name = nginx` で `type = pre` ではないコンテナの stdout ログと stderr ログを収集します。

> 注：
> こちらの label は Docker です Kubernetes 構成のラベルではありません。

図 3-30 : Label 構成の例


```
{
    " inputs ": [
        {
            " type ": " service_do  cker_stdou  t ",
            " detail ": {
                " Stdout ":  true ,
                " Stderr ":  true ,
                " IncludeLab  el ": {
                    " io . kubernetes . container . name ": " nginx
 "
                },
                " ExcludeLab  el ": {
                    " type ": " pre "
                }
            }
        }
    ]
}
```

複数行ログの収集設定

マルチラインログ収集は、Java 例外スタック出力の収集にとって特に重要です。 ここでは、Java 標準出力ログの標準収集設定を紹介します。

・ ログサンプル：

```
2018 - 02 - 03   14 : 18 : 41 . 968   INFO  [ spring - cloud -
monitor ] [ nio - 8080 - exec - 4 ]  c . g . s . web . controller .
DemoContro  ller  :  service   start
2018 - 02 - 03   14 : 18 : 41 . 969   ERROR  [ spring - cloud -
monitor ] [ nio - 8080 - exec - 4 ]  c . g . s . web . controller .
DemoContro  ller  :  java . lang . NullPointe  rException
at   org . apache . catalina . core . Applicatio  nFilterCha  in .
internalDo  Filter ( Applicatio  nFilterCha  in . java : 193 )
at   org . apache . catalina . core . Applicatio  nFilterCha  in .
doFilter ( Applicatio  nFilterCha  in . java : 166 )
at   org . apache . catalina . core . StandardWr  apperValve .
invoke ( StandardWr  apperValve . java : 199 )
at   org . apache . catalina . core . StandardCo  ntextValve .
invoke ( StandardCo  ntextValve . java : 96 )
...
2018 - 02 - 03   14 : 18 : 41 . 968   INFO  [ spring - cloud -
monitor ] [ nio - 8080 - exec - 4 ]  c . g . s . web . controller .
DemoContro  ller  :  service   start   done
```

・ コレクションの構成：

ラベルが `app = monitor` で、行の先頭が日付型のコンテナの入力ログを収集する（マッチング効率を上げるために、行の最初の 10 バイトだけが正規表現との一致をチェックするために使用される）。

```
{
" inputs ": [
  {
    " detail ": {
      " BeginLineC  heckLength ":  10 ,
      " BeginLineR  egex ": "\\ d +-\\ d +-\\ d +. *",
      " IncludeLab  el ": {
        " app ": " monitor "
      }
    },
    " type ": " service_do  cker_stdou  t "
  }
]
}
```

収集したデータを処理する

Logtail は収集された Docker の標準出力に対して common data processing methods をサポートしています。 前のセクションの複数行のログ形式に基づいて、正規表現を使用して時刻、モジュール、スレッド、クラス、および情報のログを解析することをお勧めします。

・ 収集の構成：

ラベルが `app = monitor` で、行の先頭が日付型のコンテナの入力ログを収集する（マッチング効率を上げるために、行の最初の 10 バイトだけが正規表現との一致をチェックするために使用される）

```
{
```

```
" inputs ": [
  {
    " detail ": {
      " BeginLineC  heckLength ":  10 ,
      " BeginLineR  egex ": "\\ d +-\\ d +-\\ d +. *",
      " IncludeLab  el ": {
        " app ": " monitor "
      }
    },
    " type ": " service_do  cker_stdou  t "
  }
],
" Processors  ":[
    {
        " type ": " processor_  regex ",
        " detail ": {
            " SourceKey ": " content ",
            " Regex ": "(\\ d +-\\ d +-\\ d + \\ d +:\\ d +:\\ d +\
\.\\ d +)\\ s +(\\ w +)\\ s +\\[([^]]+)]\\ s +\\[([^]]+)]\\ s +:\\ s
 +(. *)",
            " Keys ": [
                " time ",
                " module ",
                " thread ",
                " class ",
                " info "
            ],
            " NoKeyError ":  true ,
            " NoMatchErr  or ":  true ,
            " KeepSource ":  false
        }
    }
]
}
```

- **サンプル出力：**

  ログ `2018 - 02 - 03  14 : 18 : 41 . 968  INFO  [ spring - cloud - monitor ] [ nio - 8080 - exec - 4 ]  c . g . s . web . controller . DemoContro  ller  :  service  start  done` **を処理した後の出力は、次のようになります。**

```
__tag__ : __hostname  __ : logtail - dfgef
_container  _name_ : monitor
_image_nam  e_ : registry . cn - hangzhou . aliyuncs . xxxxxxxxxx
xxxxx
_namespace  _ : default
_pod_name_ : monitor - 6f54bd5d74 - rtzc7
_pod_uid_ : 7f012b72 - 04c7 - 11e8 - 84aa - 00163f00c3  69
_source_ : stdout
_time_ : 2018 - 02 - 02T14 : 18 : 41 . 979147844Z
Time :  2018 - 02 - 02  02 : 18 : 41 .  968
level : INFO
module : spring - cloud - monitor
Thread :  fig
Class :  c . g . s .  web .  Controller .  demcontrol  ler
```

```
message : service   start   done
```

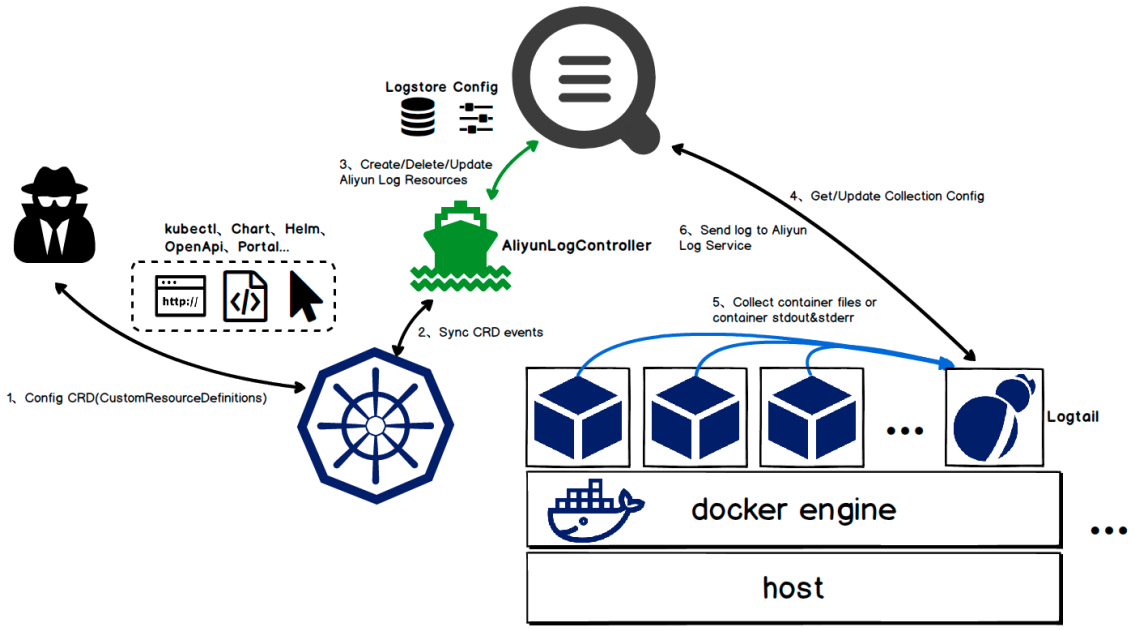
3.6.5 CRD での Kubernetes ログ収集の設定

ログ収集は、デフォルトでコンソールに設定されます。 Log Service では、Kubernetes マイクロサービス開発用のログ収集の CRD 設定も提供します。 これにより、kubectl を使用して設定を管理できます。

この方法は、Kubernetes のデプロイメント、および公開プロセスと連携されているため、収集設定管理に CRD メソッドを使用することを推奨します。

実装の原則

図 3-31 : 実装の原則



インストールコマンドを実行して、 `alibaba - log - controller  Helm` パッケージをインストールします。 Helm パッケージは、主に次の操作を実行します。

1. aliyunlogconfigs CRD (カスタムリソース定義) を作成します。
2. alibaba-log-controller をデプロイします。
3. Logtail DaemonSet をデプロイします。

設定の内部ワークフローは次のとおりです。

1. `kubectl` または他のツールを使用して、aliyunlogconfigs CRD 設定を適用します。
2. alibaba-log-controller は、設定のアップデートを検出します。

3. alibaba-log-controller は、CRD の内容とサーバーのステータスに基づき、Logstore の作成、設定の作成、およびマシングループのアプリケーション設定のリクエストを自動的に送信します。
4. DaemonSet モードで実行される Logtail は、サーバー設定のリクエストを定期的に送信し、新しい設定または更新された設定を取得し、高速ロードを実行します。
5. Logtail は、設定情報に基づき、各コンテナ (pod) から標準出力またはファイルを収集します。
6. Logtail は、処理され集められたデータを Log Service に送信します。

設定方法

注：

DaemonSet モードでデプロイされた Logtail を使用した場合、CRD モードでは設定を管理できません。 詳細は、このドキュメントの DaemonSet デプロイメントモードの移行プロセスをご参照ください。

設定を作成するには、AliyunLogConfig の CRD を定義する必要があります。設定を削除するには、対応する CRD リソースを削除する必要があります。 CRD は次のように設定されています。

```
apiVersion: log.alibabacloud.com/v1alpha1 ## デフォルト値。変更する必要はありません。
kind: AliyunLogConfig ## デフォルト値。変更する必要はありません。
metadata:
  name: simple-stdout-example ## リソース名。ラスター内で一意である必要があります。
spec:
  logstore: k8s-stdout ## Logstore 名。存在しなければ自動的に作成されます。
  shardCount: 2 ## [オプション] Logstore のパーティション数。 デフォルト値は 2 です。 値の範囲は 1 ～ 10 です。
  lifeCycle: 90 ## [オプション] Logstore の保存期間。 デフォルト値は 90 です。 値の範囲は 1 ～ 7300 です。 値 7300 は、永久的な保存を示しています。
  logtailConfig: ## 詳細な設定
    inputType: plugin ## 収集の入力タイプ。 通常、値はファイルかプラグインです。
    configName: simple-stdout-example ## 収集設定名。 値はリソース名 ( metadata.name ) と同じでなければなりません。
    inputDetail: ## 詳細な設定情報。例をご参照ください。
      ...
```

設定が完了して適用されると、alibaba-log-controller が自動的に作成されます。

設定の表示

Kubernetes CRD またはコンソールで設定を表示できます。

コンソールでの設定の表示方法については、「Logtail 設定の作成」をご参照ください。

注：

CRD 方式を使用して設定を管理すると、CRD で設定を更新する際、コンソールで行った設定変更が上書きされます。

- **すべての設定を表示するには、** `kubectl get aliyunlogc onfigs` **を実行します。**

```
[ root @ iZbp1dsbia Z ~]# kubectl get aliyunlogc onfigs
 NAME AGE
 regex - file - example 10s
 regex - stdout - example 4h
 simple - file - example 5s
```

- **詳細な設定とステータスを表示するには、** `kubectl get aliyunlogc onfigs ${ configname } - o yaml` **を実行します。**

  設定の `status` **フィールドには、設定の実行結果が表示されます。 設定が正常に適用された場合、** `statusCode` **の値は** `status` **フィールドで 200 になります。** `statusCode` **の値が 200 でない場合、設定の適用に失敗しています。**

```
[ root @ iZbp1dsbia Z ~]# kubectl get aliyunlogc onfigs
 simple - file - example - o yaml
 apiVersion : log . alibabaclo ud . com / v1alpha1
 kind : AliyunLogC onfig
 metadata :
 annotation s :
   kubectl . kubernetes . io / last - applied - configurat ion : |

 clusterNam e : ""
 creationTi mestamp : 2018 - 05 - 17T08 : 44 : 46Z
 generation : 0
 name : simple - file - example
 namespace : default
 resourceVe rsion : " 21790443 "
 selfLink : / apis / log . alibabaclo ud . com / v1alpha1 /
 namespaces / default / aliyunlogc onfigs / simple - file -
 example
 uid : 8d3a09c4 - 59ae - 11e8 - 851d - 00163f0086 85
 spec :
 lifeCycle : null
 logstore : k8s - file
 logtailCon fig :
   configName : simple - file - example
   inputDetai l :
     dockerFile : true
     dockerIncl udeEnv :
       ALIYUN_LOG TAIL_USER_ DEFINED_ID : ""
     filePatter n : simple . LOG
     logPath : / usr / local / ilogtail
     logType : common_reg _log
   inputType : file
 machineGro ups : null
 project : ""
 shardCount : null
 status :
```

```
status : OK
statusCode : 200
```

設定例

コンテナ標準出力

コンテナ標準出力では、`inputType` を `plugin` に設定します。`inputDetai l` の `plugin` フィールドの詳細情報を入力します。設定フィールドの詳細については、「コンテナ標準出力」をご参照ください。

- シンプル収集モード

  環境変数の設定 `COLLECT_ST DOUT_FLAG = false` があるコンテナを除く、すべてのコンテナの標準出力 (stdout と stdeer) を収集します。

```
apiVersion : log . alibabaclo ud . com / v1alpha1
kind : AliyunLogC onfig
metadata :
 # your config name , must be unique in your k8s
cluster
  name : simple - stdout - example
spec :
 # logstore name to upload log
  logstore : k8s - stdout
 # logtail config detail
  logtailCon fig :
   # docker stdout ' s input type is ' plugin '
    inputType : plugin
   # logtail config name , should be same with [
metadata . name ]
    configName : simple - stdout - example
    inputDetai l :
      plugin :
        inputs :
         -
           # input type
            type : service_do cker_stdou t
            detail :
             # collect stdout and stderr
              Stdout : true
              Stderr : true
             # collect all container ' s stdout except
containers with " COLLECT_ST DOUT_FLAG : false " in docker
env config
              ExcludeEnv :
```

```
                    COLLECT_ST  DOUT_FLAG : " false "
```

- **カスタム収集モード**

  Grafana のアクセスログを収集し、アクセスログを構造化データに解析します。

  Grafana コンテナには環境変数 `GF_INSTALL  _PLUGINS = grafana - piechart -…. ` **の設定があります。 Logtail がこのコンテナのみから標準出力を収集できるように** `IncludeEnv` **を** `GF_INSTALL  _PLUGINS : ‘’` **に設定できます。**

  図 3-32 : カスタム収集モード

  Grafana のアクセスログの形式は次のとおりです。

```
t = 2018 - 03 - 09T07 : 14 : 03 + 0000   lvl = info   msg ="
Request   Completed "  logger = context   userId = 0   orgId = 0
uname =  method = GET   path =/  status = 302   remote_add  r = 172
. 16 . 64 . 154   time_ms = 0   size = 29   referer =
```

  正規表現を使用してアクセスログを解析します。 詳細な設定は次のとおりです。

```
apiVersion :  log . alibabaclo  ud . com / v1alpha1
kind :  AliyunLogC  onfig
metadata :
 #  your   config   name ,  must   be   unique   in   your   k8s
cluster
  name :  regex - stdout - example
spec :
 #  logstore   name   to   upload   log
  logstore :  k8s - stdout - regex
 #  logtail   config   detail
  logtailCon  fig :
   #  docker   stdouts   input   type   is   plugin
    inputType :  plugin
   #  logtail   config   name ,  should   be   same   with  [
metadata . name ]
    configName :  regex - stdout - example
    inputDetai  l :
      plugin :
        inputs :
         -
           #  input   type
            type :  service_do  cker_stdou  t
            detail :
             #  stdout  出力のみを収集し、 stdeer  出力は収集しません。
```

```
              Stdout :  true
              Stderr :  false
            # コンテナの環境変数設定でキーが “ GF_INSTALL  _PLUGINS
” である  stdout  出力のみを収集します。
              IncludeEnv :
                GF_INSTALL  _PLUGINS : ''
        processors :
         -
           # 正規表現を使用します。
            type :  processor_  regex
            detail :
             #  docker  によって収集されたデータには、デフォルトで、キー
“コンテンツ” があります。
              SourceKey :  content
              # 抽出の正規表現
              Regex : ' t =(\ d +-\ d +-\ w +:\ d +:\ d +\+\ d +)
lvl =(\ w +)  msg ="([^"]+)"  logger =(\ w +)  userId =(\ w +)
orgId =(\ w +)  uname =(\ S *)  method =(\ w +)  path =(\ S +)
status =(\ d +)  remote_add  r =(\ S +)  time_ms =(\ d +)  size =(\
d +)  referer =(\ S *). *'
              # 抽出されたキー
              Keys : [' time ', ' level ', ' message ', ' logger ',
' userId ', ' orgId ', ' uname ', ' method ', ' path ', ' status ',
' remote_add  r ', ' time_ms ', ' size ', ' referer ']
              # 元のフィールドを保持します。
              KeepSource :  true
              NoKeyError :  true
              NoMatchErr  or :  true
```

設定が適用されると、 Log Service によって収集されるデータは次のようになります。

図 3-33 : 収集されるログデータ

コンテナファイル

- シンプルファイル

  環境変数設定にキー `ALIYUN_LOG  TAIL_USER_  DEFINED_ID` が含まれるコンテナからログファイルを収集します。 ログファイルのパスは`/ data / logs / app_1` で、ファイル名は `simple . LOG` です。

```
apiVersion :  log . alibabaclo  ud . com / v1alpha1
kind :  AliyunLogC  onfig
metadata :
 #  your  config  name ,  must  be  unique  in  your  k8s
cluster
  name :  simple - file - example
```

```
spec :
 #  logstore   name   to   upload   log
  logstore :  k8s - file
 #  logtail   config   detail
  logtailCon  fig :
   #  log   file ' s   input   type   is  ' file '
    inputType :  file
   #  logtail   config   name ,  must   same   with  [ metadata .
name ]
    configName :  simple - file - example
    inputDetai  l :
     # 正規表現型のログについては、 logType  を “ common_reg  _log ”に設
定します。
      logType :  common_reg  _log
     # ログファイルフォルダ
      logPath : / data / logs / app_1
     # ワイルドカードをサポートするファイル名。たとえば、 log . log
      filePatter  n :  simple . LOG
     # コンテナからファイルを収集します。 dockerFile  フラグは  true  に設
定されています。
      dockerFile :  true
     #  Only   collect   container   with  " ALIYUN_LOG
TAIL_USER_  DEFINED_ID "  in   docker   env   config
      dockerIncl  udeEnv :
        ALIYUN_LOG  TAIL_USER_  DEFINED_ID : ""
```

- 正規表現ファイルを完成する

  Java プログラムログの例を次に示します。

```
[ 2018 - 05 - 11T20 : 10 : 16 , 000 ] [ INFO ] [ SessionTra  cker ]
 [ SessionTra  ckerImpl . java : 148 ]  Expiring   sessions
java . sql . SQLExcepti  on :  Incorrect   string   value : '\ xF0
\ x9F \ x8E \ x8F ",...'  for   column  ' data '  at   row   1
at   org . springfram  ework . jdbc . support . AbstractFa
llbackSQLE  xceptionTr  anslator . translate ( AbstractFa
llbackSQLE  xceptionTr  anslator . java : 84 )
at   org . springfram  ework . jdbc . support . AbstractFa
llbackSQLE  xception
```

  ログにはエラースタック情報が含まれているため、ログエントリは複数の行に分割されることがあります。 したがって、行の先頭に正規表現を設定する必要があります。 各フィールドを抽出するには、正規表現を使用します。 設定の詳細は次のとおりです。

```
apiVersion :  log . alibabaclo  ud . com / v1alpha1
kind :  AliyunLogC  onfig
metadata :
 #  your   config   name ,  must   be   unique   in   your   k8s
cluster
  name :  regex - file - example
spec :
 #  logstore   name   to   upload   log
  logstore :  k8s - file
  logtailCon  fig :
   #  log   file ' s   input   type   is  ' file '
    inputType :  file
   #  logtail   config   name ,  should   be   same   with  [
metadata . name ]
    configName :  regex - file - example
    inputDetai  l :
```

```
      # 正規表現型のログについては、 logType を“ common_reg _log ”に設
定します。
       logType : common_reg _log
      # ログファイルフォルダ
       logPath : / app / logs
      # ワイルドカードをサポートするファイル名。たとえば、 log . log
       filePatter n : error . LOG
      # 最初の行の正規表現
       logBeginRe gex : '\[\ d +-\ d +-\ w +:\ d +:\ d +,\ d +]\ s
 \[\ w +]\ s . *'
      # 正規表現を解析します。
       regex : '\[([^]]+)]\ s \[(\ w +)]\ s \[(\ w +)]\ s \[([^:]+):
(\ d +)]\ s (. *)'
      # 抽出されたキーのリスト
       key : [" time ", " level ", " method ", " file ", " line ",
 " message "]
      # 正規表現のログ。 ログの time はデフォルトで時間解析のために抽出されま
す。 時間が必要ない場合は、フィールドを無視します。
       timeFormat : '% Y -% m -% dT % H :% M :% S '
      # コンテナからファイルを収集します。 dockerFile フラグは true に設
定されています。
       dockerFile : true
      # Only collect container with " ALIYUN_LOG
 TAIL_USER_ DEFINED_ID " in docker env config
       dockerIncl udeEnv :
         ALIYUN_LOG TAIL_USER_ DEFINED_ID : ""
```

設定が適用されると、 Log Service によって収集されるデータは次のようになります。

図 3-34 : 収集されるログデータ

- 区切り文字パターンファイル

  Logtail は、区切り文字モードでのログ解析をサポートしています。例は次のとおりです。

```
apiVersion : log . alibabaclo ud . com / v1alpha1
kind : AliyunLogC onfig
metadata :
 # your config name , must be unique in your k8s
cluster
  name : delimiter - file - example
spec :
 # logstore name to upload log
  logstore : k8s - file
  logtailCon fig :
   # log file ' s input type is ' file '
    inputType : file
    configName : delimiter - file - example
```

```
    #  logtail   config   name ,  should   be   same   with  [
metadata . name ]
    inputDetai  l :
     # 区切り文字タイプのログの場合は、 logType  を  delimiter_  log  に
設定します。
      logType :  delimiter_  log
     # ログファイルフォルダ
      logPath : / usr / local / ilogtail
     # ワイルドカードをサポートするファイル名。たとえば、  log . log
      filePatter  n :  delimiter_  log . LOG
     # 複数文字の区切り文字を使用します。
      separator : '|&|'
     # 抽出されたキーのリスト
      key  : [' time ', ' level ', ' method ', ' file ', ' line ',
' message ']
     # 解析時間のキー。 時間の解析が不要な場合はフィールドを無視します。
      timeKey : ' time '
     # 時間解析メソッド。 時間の解析が不要な場合はフィールドを無視します。
      timeFormat : '% Y -% m -% dT % H :% M :% S '
     # コンテナからファイルを収集します。 dockerFile  フラグは  true  に設
定されています。
      dockerFile :  true
     #  Only   collect   container   with  " ALIYUN_LOG
TAIL_USER_  DEFINED_ID "  in   docker   env   config
      dockerIncl  udeEnv :
        ALIYUN_LOG  TAIL_USER_  DEFINED_ID : ''
```

- JSON モードファイル

  ファイル内の各データ行が JSON オブジェクトの場合、JSON メソッドを解析に使用できます。例は次のとおりです。

```
apiVersion :  log . alibabaclo  ud . com / v1alpha1
kind :  AliyunLogC  onfig
metadata :
 #  your   config   name ,  must   be   unique   in   you   k8s
cluster
  name :  json - file - example
spec :
 #  logstore   name   to   upload   log
  logstore :  k8s - file
  logtailCon  fig :
   #  log   file ' s   input   type   is  ' file '
    inputType :  file
   #  logtail   config   name ,  should   be   same   with  [
metadata . name ]
    configName :  json - file - example
    inputDetai  l :
     # 区切り文字タイプのログの場合は、 logType  を  json_log  に設定しま
す。
      logType :  json_log
     # ログファイルフォルダ
      logPath : / usr / local / ilogtail
     # ワイルドカードをサポートするファイル名。たとえば、 log_ *. log
      filePatter  n :  json_log . LOG
     # 解析時間のキー。 時間の解析が不要な場合はフィールドを無視します。
      timeKey : ' time '
     # 時間解析メソッド。 時間の解析が不要な場合はフィールドを無視します。
      timeFormat : '% Y -% m -% dT % H :% M :% S '
     # コンテナからファイルを収集します。 dockerFile  フラグは  true  に設
定されている
      dockerFile :  true
```

```
      #  Only   collect   container   with  " ALIYUN_LOG
TAIL_USER_  DEFINED_ID "  in   docker   env   config
      dockerIncl  udeEnv :
        ALIYUN_LOG  TAIL_USER_  DEFINED_ID : ""
```

3.6.6 Kubernetes-Sidecar ログ収集モード

Logtail は Kubernetes から Sidecar モードでログを収集し、ログ収集を必要とするサービス容器ごとに Sidecar 容器を作成できるため、マルチ容器の分離が容易になり、収集パフォーマンスが向上します。

現在、Kubernetes クラスタにインストールされているデフォルトのログコンポーネントは DaemonSet です。これは、O&M の操作を簡素化し、リソースの占有が少なく、容器の標準出力と容器ファイルの収集をサポートし、柔軟に構成できます。

ただし、DaemonSet モードでは、Logtail はノード上のすべての容器からログを収集する必要があります。 これはパフォーマンスのボトルネックにつながり、サービスログ間の完全な分離はできません。 この問題を解決するために、Logtail は Sidecar を提供します。これにより、Logtail はログ収集を必要とする各サービス容器に対して Sidecar 容器を作成できます。 このモードでは、マルチ容器間の分離が大幅に強化され、収集パフォーマンスが向上します。 大規模な Kubernetes クラスタ、および複数のサービスを提供する PaaS プラットフォームとして機能するクラスタには、Sidecar モードの使用をお勧めします。

機能

- Sidecar モードは、Kubernetes、オンプレミス ECS Kubernetes、および IDC のオンプレミス Kubernetes の容器サービスに適用できます。
- Sidecar モードでは、Logtail は Pod名、Pod IP アドレス、Pod ネームスペース、および Pod が属するノードの名前と IP アドレスを含む、Pod メタデータを収集できます。
- Sidecar モードでは、Logtail は CustomResourceDefinition（CRD）を介してプロジェクト、Logstores、インデックス、Logtail 構成、マシングループなどの Log Service リソースを自動的に作成できます。
- Sidecar モードは動的スケーリングもサポートします。 レプリカの数はいつでも調整でき、変更はすぐに有効になります。

コンセプト

Sidecar モードでは、ログ収集で Logtail がログディレクトリをサービス容器と共有する必要があります。 簡単に言うと、サービス容器はログをログディレクトリに書き込み、Logtail はログディレクトリ内のログファイルの変更をモニタリングしてログを収集します。 詳細については、以下の説明をご参照ください。

1. Sidecar ログ収集モードの紹介

2. Sidecar モードの例

前提条件

1. Log Service を有効化しました。

   まだ Log Service を有効にしていない場合は 、まず有効化します。

2. CRD ベースの設定用にKubernetes のログ収集をインストールしました。

制限事項

1. Logtail はログディレクトリをサービス容器と共有する必要があります。
2. Sidecar モードは容器の標準出力の収集をサポートしません。

Sidecar 構成

Sidecar 構成には以下が含まれます：

1. 基本操作パラメータの設定
2. マウントパスの設定

例は次となります：

```
apiVersion :  batch / v1
kind :  Job
metadata :
  name :  nginx - log - sidecar - demo
  namespace :  default
spec :
  template :
    metadata :
      name :  nginx - log - sidecar - demo
    spec :
      restartPol  icy :  Never
      containers :
     -  name :  nginx - log - demo
        image :  registry . cn - hangzhou . aliyuncs . com / log -
service / docker - log - test : latest
        command : ["/ bin / mock_log "]
        args : ["-- log - type = nginx ", "-- stdout = false ", "--
stderr = true ", "-- path =/ var / log / nginx / access . log ", "--
total - count = 1000000000 ", "-- logs - per - sec = 100 "]
        volumeMoun  ts :
       -  name :  nginx - log
          mountPath : / var / log / nginx
     #####  logtail   sidecar   container
     -  name :  logtail
       #  more   info :  https :// cr . console . aliyun . com /
repository / cn - hangzhou / log - service / logtail / detail
       #  this   images   is   released   for   every   region
        image :  registry . cn - hangzhou . aliyuncs . com / log -
service / logtail : latest
        livenessPr  obe :
          exec :
            command :
           - / etc / init . d / ilogtaild
           -  status
```

```
          initialDel  aySeconds :  30
          periodSeco  nds :  30
        resources :
          limits :
            memory :  512Mi
          requests :
            cpu :  10m
            memory :  30Mi
        env :
         #####  base   config
         #  user   id
         -  name : " ALIYUN_LOG  TAIL_USER_  ID "
            value : "${ your_aliyu  n_user_id }"
         #  user   defined   id
         -  name : " ALIYUN_LOG  TAIL_USER_  DEFINED_ID "
            value : "${ your_machi  ne_group_u  ser_define  d_id }"
         #  config   file   path   in   logtail ' s   container
         -  name : " ALIYUN_LOG  TAIL_CONFI  G "
            value : "/ etc / ilogtail / conf /${ your_regio  n_config
}/ ilogtail_c  onfig . json "
         #####  env   tags   config
         -  name : " ALIYUN_LOG  _ENV_TAGS "
            value : " _pod_name_ | _pod_ip_ | _namespace  _ |
_node_name  _ | _node_ip_ "
         -  name : " _pod_name_ "
            valueFrom :
              fieldRef :
                fieldPath :  metadata . name
         -  name : " _pod_ip_ "
            valueFrom :
              fieldRef :
                fieldPath :  status . podIP
         -  name : " _namespace  _ "
            valueFrom :
              fieldRef :
                fieldPath :  metadata . namespace
         -  name : " _node_name  _ "
            valueFrom :
              fieldRef :
                fieldPath :  spec . nodeName
         -  name : " _node_ip_ "
            valueFrom :
              fieldRef :
                fieldPath :  status . hostIP
        volumeMoun  ts :
       -  name :  nginx - log
          mountPath : / var / log / nginx
     #####  share   this   volume
      volumes :
     -  name :  nginx - log
        emptyDir : {}
```

構成 1：基本動作パラメータを設定します。

主なパラメータとその設定は次のとおりです。

```
#####  base   config
         #  user   id
         -  name : " ALIYUN_LOG  TAIL_USER_  ID "
            value : "${ your_aliyu  n_user_id }"
         #  user   defined   id
         -  name : " ALIYUN_LOG  TAIL_USER_  DEFINED_ID "
```

```
            value : "${ your_machi  ne_group_u  ser_define  d_id }"
          #  config   file   path   in   logtail ' s   container
          -  name : " ALIYUN_LOG  TAIL_CONFI  G "
             value : "/ etc / ilogtail / conf /${ your_regio  n_config
 }/ ilogtail_c  onfig . json "
```

パラメーター	説明
`$ { your_regio n_config }`	このパラメータは、プロジェクトのリージョンとネットワークの種類によって決まります。 ネットワークの種類に応じて適切な値に設定します。 有効な値： ・ インターネット： `region - internet` 。 たとえば、中国（杭州）リージョンの値は `cn - hangzhou - internet` です。 ・ Alibaba Cloud イントラネット： `region` 。 たとえば、中国（杭州）リージョンの値は `cn - hangzhou` です。 このパラメータでの region は#unique_42/unique_42_Connect_42_table_eyz_pmv_vdbです。 プロジェクトが属するリージョンに設定します。
`${ your_aliyu n_user_id }`	このパラメータは、ユーザー ID を指定します。これは、文字列形式の Alibaba Cloud アカウント ID に置き換える必要があります。 IDのクエリの方法については、 ユーザー ID の設定のセクション 2.1 をご参照ください。 注： このパラメータ値は、 Alibaba Cloud アカウント ID である必要があります。 RAM ユーザー ID の場合は無効になります。
`${ your_machi ne_group_u ser_define d_id }`	このパラメータは、クラスタ内のマシングループのカスタム ID を指定します。 ID は、Log Service がデプロイされているリージョン内で一意である必要があります。 詳細は、マシングループにユーザー定義 ID を設定するを参照ください。

構成 2：マウントパスを設定します。

1. Logtail とサービス容器は同じディレクトリにマウントする必要があります。
2. `emptyDir` のマウント方法をお勧めします。

マウントパスの例は、前述の構成例に示されています。

ログ収集設定

ログ収集は、CRD または Log Service コンソールを介して設定できます。 CRD ベースの設定は、プロジェクト、ログストア、インデックス、マシングループ、および Logtail 構成の自動作成をサポートし、Kubernetes と簡単に統合できます。 そのため、CRD ベースの設定をお勧め

します。 Kubernetes のログ収集を初めて使用、またはデバッグするユーザーにとっても、コンソールベースの設定が簡単です。

CRD ベースの設定

詳細は、CRD での Kubernetes ログ収集の設定をご参照ください。 DaemonSet 収集モードと比較して、CRD ベースの設定には以下の制限があります。

1. ログ収集が必要なプロジェクトの名前を指定しなければなりません。 そうしないと、ログが収集され、ログコンポーネントがデフォルトでインストールされているプロジェクトに送信されます。
2. 設定を有効にするには、マシングループを指定する必要があります。 そうしないと、設定は DaemonSet が属するマシングループにデフォルトで適用されます。
3. Sidecar モードはファイル収集のみをサポートします、ファイル収集している間に `dockerFile` を false に設定する必要があります。

詳細は、次の例をご参照ください。

コンソールベースの設定

1. マシングループの構成。

   Log Service コンソールで、Pod IP アドレスの変更に動的に適応するために、ID がカスタム ID に設定された Logtail マシングループを作成します。 そうするには、次の手順を実行します：

   a. Log Service を有効にして、プロジェクトとログストアを作成します。 詳細は、準備をご参照ください。
   b. マシングループリストページで、 マシングループの作成をクリックします。
   c. ID をカスタム ID `ALIYUN_LOG_TAIL_USER_DEFINED_ID` に設定します。

2. 収集モードの設定。

   対象のファイルの収集詳細を設定します。 現在、シンプルモード、Nginx アクセスモード、区切り文字モード、JSON モード、通常モードなど、さまざまなモードがサポートされています。 詳細は、テキストファイルの収集次をご参照ください。

   次の図にこの例の設定を示します。

   注：

Docker ファイルを無効にする必要があります。


* 配置名称:
nginx-log-sidecar
* 日志路径:
/var/log/nginx
/**/
access
指定文件夹下所有符合文件名称的文件都会被监控到(包含所有层次的目
符模式匹配。Linux文件路径只支持/开头，例：/apsara/nuwa/.../app.Lo
如：C:\Program Files\Intel\...*.Log
是否为Docker文件:
如果是Docker容器内部文件，可以直接配置内部路径与容器Tag，Logta
进行过滤采集指定容器的日志，具体说明参考文档链接
模式:
分隔符模式
如何设置Delimiter类型配置
日志样例:
2018-09-26T03:16:53.033307075Z 10.200.98.220 - - "POST /PutData
Category=YunOsAccountOpLog&AccessKeyId=Uxxxx45A&Date=Fri%
A53%3A30%20GMT&Topic=raw&Signature=pD12XYLmGxKQ%2Bm
18204 200 37 "-" "aliyun-sdk-java" 1
请贴入需要解析的日志样例(支持多条) 常见样例>>
* 分隔符:
空格
引用符:
双引号
双引号（"）作为Quote时，内部包含分隔符的字段需要被一对Quote包
包含空格、制表符等字符，请修改格式。

例

シナリオ：

1. **Kubernetes クラスタは IDC のオンプレミスクラスタであり、Log Service がデプロイされているリージョンは中国（杭州）です。 ログはインターネットから収集されます。**
2. **次の例では、マウントオブジェクトは `nginx - log` で、マウントタイプは `emptyDir` です。 これらはそれぞれ nginx-log-demo および logtail 容器内の `/ var / log / nginx` ディレクトリにマウントされています。**
3. **アクセスログは `/ var / log / nginx / access . log` で、保存先ログストアは `nginx - access` です。**
4. **エラーログは `/ var / log / nginx / error . log` で、保存先ログストアは `nginx - error` です。**

・ **Sidecar 設定：**

```
apiVersion :  batch / v1
kind :  Job
metadata :
  name :  nginx - log - sidecar - demo
  namespace :  default
spec :
  template :
    metadata :
      name :  nginx - log - sidecar - demo
    spec :
      restartPol  icy :  Never
      containers :
     -  name :  nginx - log - demo
        image :  registry . cn - hangzhou . aliyuncs . com / log -
service / docker - log - test : latest
        command : ["/ bin / mock_log "]
        args : ["-- log - type = nginx ", "-- stdout = false ", "--
stderr = true ", "-- path =/ var / log / nginx / access . log ",
"-- total - count = 1000000000 ", "-- logs - per - sec = 100 "]
        volumeMoun  ts :
       -  name :  nginx - log
          mountPath : / var / log / nginx
     #####  logtail   sidecar   container
     -  name :  logtail
       #  more   info :  https :// cr . console . aliyun . com /
repository / cn - hangzhou / log - service / logtail / detail
       #  this   images   is   released   for   every   region
        image :  registry . cn - hangzhou . aliyuncs . com / log -
service / logtail : latest
        livenessPr  obe :
          exec :
            command :
           - / etc / init . d / ilogtaild
           -  status
          initialDel  aySeconds :  30
          periodSeco  nds :  30
        env :
         #####  base   config
```

```
          #  user   id
          -  name : " ALIYUN_LOG  TAIL_USER_  ID "
             value : " xxxxxxxxxx "
          #  user   defined   id
          -  name : " ALIYUN_LOG  TAIL_USER_  DEFINED_ID "
             value : " nginx - log - sidecar "
          #  config   file   path   in   logtail ' s   container
          -  name : " ALIYUN_LOG  TAIL_CONFI  G "
             value : "/ etc / ilogtail / conf / cn - hangzhou -
 internet / ilogtail_c  onfig . json "
          #####  env   tags   config
          -  name : " ALIYUN_LOG  _ENV_TAGS "
             value : " _pod_name_ | _pod_ip_ | _namespace  _ |
 _node_name  _ | _node_ip_ "
          -  name : " _pod_name_ "
             valueFrom :
               fieldRef :
                 fieldPath :  metadata . name
          -  name : " _pod_ip_ "
             valueFrom :
               fieldRef :
                 fieldPath :  status . podIP
          -  name : " _namespace  _ "
             valueFrom :
               fieldRef :
                 fieldPath :  metadata . namespace
          -  name : " _node_name  _ "
             valueFrom :
               fieldRef :
                 fieldPath :  spec . nodeName
          -  name : " _node_ip_ "
             valueFrom :
               fieldRef :
                 fieldPath :  status . hostIP
        volumeMoun  ts :
        -  name :  nginx - log
           mountPath : / var / log / nginx
      #####  share   this   volume
       volumes :
      -  name :  nginx - log
         emptyDir : {}
```

- **CRD 設定：**

```
#  config   for   access   log
 apiVersion :  log . alibabaclo  ud . com / v1alpha1
 kind :  AliyunLogC  onfig
 metadata :
  #  your   config   name ,  must   be   unique   in   you   k8s
 cluster
   name :  nginx - log - access - example
 spec :
  #  project   name   to   upload   log
   project :  k8s - nginx - sidecar - demo
  #  logstore   name   to   upload   log
   logstore :  nginx - access
  #  machine   group   list   to   apply   config ,  should   be
 same   with   your   sidecar ' [ ALIYUN_LOG  TAIL_USER_  DEFINED_ID
 ]
   machineGro  ups :
  -  nginx - log - sidecar
  #  logtail   config   detail
   logtailCon  fig :
```

```
    #  log   file ' s   input   type   is  ' file '
     inputType :  file
    #  logtail   config   name ,  should   be   same   with  [
 metadata . name ]
     configName :  nginx - log - access - example
     inputDetai  l :
      #  Simple   logs   with   logType   set   to   common_reg
 _log
       logType :  common_reg  _log
      #  Log   folder
       logPath : / var / log / nginx
      #  File   name   with   wildcards   supported ,  for   example
 ,  log_ *. log
       filePatter  n :  access . log
      #  Sidecar   mode   with   dockerFile   set   to   false
       dockerFile :  false
      #  Line   start   regular   expression ,  which   is   set
 to  .*  is   the   log   contains   only   a   line
       logBeginRe  gex : '. *'
      #  Regular   expression   for   parsing
       regex : '(\ S +)\ s (\ S +)\ s \ S +\ s \ S +\ s "(\ S +)\ s
 (\ S +)\ s +([^"]+)"\ s +(\ S +)\ s (\ S +)\ s (\ d +)\ s (\ d +)\
 s (\ S +)\ s "([^"]+)"\ s . *'
      #  List   of   the   extracted   keys
       key  : [" time ", " ip ", " method ", " url ", " protocol ",
 " latency ", " payload ", " status ", " response - size ", ser -
 agent "]
#  config   for   error   log
```

```
#  config   for   error   log
 apiVersion :  log . alibabaclo  ud . com / v1alpha1
 kind :  AliyunLogC  onfig
 metadata :
  #  your   config   name ,  must   be   unique   in   you   k8s
 cluster
   name :  nginx - log - error - example
 spec :
  #  project   name   to   upload   log
   project :  k8s - nginx - sidecar - demo
  #  logstore   name   to   upload   log
   logstore :  nginx - error
  #  machine   group   list   to   apply   config ,  should   be
 same   with   your   sidecar ' [ ALIYUN_LOG  TAIL_USER_  DEFINED_ID
 ]
   machineGro  ups :
  -  nginx - log - sidecar
  #  logtail   config   detail
   logtailCon  fig :
    #  log   file ' s   input   type   is  ' file '
     inputType :  file
    #  logtail   config   name ,  should   be   same   with  [
 metadata . name ]
     configName :  nginx - log - error - example
     inputDetai  l :
      #  Simple   logs   with   logType   set   to   common_reg
 _log
       logType :  common_reg  _log
      #  Log   folder
       logPath : / var / log / nginx
      #  File   name   with   wildcards   supported ,  for   example
 ,  log_ *. log
       filePatter  n :  error . log
      #  Sidecar   mode   with   dockerFile   set   to   false
```

```
        dockerFile :  false
```

- ログ収集エラーの表示

  上記の設定が Kubernetes クラスタに適用されると、Logtail 容器は対応するプロジェクト、ログストア、マシングループ、および Logtail 構成を自動的に作成し、収集されたログを Log Service に自動的に送信します。 Log Service コンソールにログインして詳細を表示できます。

3.7 制限

表 3-3 : ファイル収集の制限

項目	能力と制限
ファイルエンコーディング	UTF-8 と GBK でエンコードされたログファイルがサポートされています。 他の形式でエンコードされたログファイルは、不器用さやデータ損失などの未定義の動作につながります。 処理パフォーマンスを向上させるには、UTF-8 エンコーディングを使用することをお勧めします。
ログファイルサイズ	無制限。
ログファイルのローテーション	`. log *`と`. log` ファイルの両方がサポートされています。
ログ解析の輻輳時のログ収集動作	ログ解析で輻輳が発生すると、Logtail FD のオープン状態が維持されます。 輻輳中にログファイルのローテーションが複数回発生すると、Logtail は回転ログの解析シーケンスを保持しようとします。 解析されていない 20 以上のログが回転した場合、Logtail は後続のログファイルを処理しません。 ソフトリンクのサポート詳細は、こちらを参照してください。
シングルログサイズ	監視対象ディレクトリはソフトリンクにすることができます。

項目	能力と制限
シングルログサイズ	各ログのサイズは 512 KB を超えることはできません。 複数行のログが行頭の正規表現で分割されている場合、各ログの最大サイズは 512 KBです。ログサイズが 512 KBを超えると、ログは収集のために複数の部分に分割されます。 たとえば、一つのログは 1025 KBです。最初の 512 KBは最初に処理され、その後の 512 KBは 2 回目に処理され、最後の 1 KBは 3 回目に処理されます。
正規表現タイプ	Perl と互換性のある正規表現を使用します。
同じファイルの複数の収集設定	サポートされていません。 ログファイルをログストアに収集し、複数のサブスクリプションを構成することをお勧めします。この機能が必要な場合は、ログファイルのソフトリンクを設定してこの制限をバイパスします。
ファイルオープン動作	Logtail はファイルをオープン状態で収集します。 5 分間ファイルを変更しないと、Logtail はファイルを閉じます。
最初のログ収集動作	Logtail は、増分ログファイルのみを収集します。 最初にファイルに変更があり、ファイルサイズが 1 MB を超える場合、Logtail は最後の 1 MB からログを収集します。 それ以外の場合、Logtail は最初からログを収集します。設定が発行された後にログファイルが変更されない場合、Logtail はこのファイルを収集しません。
非標準テキストログ	ログに’\0’を含む行。 ログは最初の ‘0’に切り捨てられます。

表 3-4 : チェックポイント管理

項目	能力と制限
チェックポイントタイムアウト期間	ファイルが 30 日以上変更されていない場合、チェックポイントは削除されます。
チェックポイントストレージポリシー	定期的に 15 分ごとに保存し、プログラムが終了すると自動的に保存されます。

項目	能力と制限
チェックポイント保存パス	デフォルトの保存パスは/ tmp / logtail_ch eckpoint です。Logtail 起動設定パラメーターに従ってパラメータを変更できます。

表 3-5 : 構成の制限

項目	能力と制限
構成情報の更新	更新された構成は約 30 秒後に有効になります。
動的構成ロード	サポートされる。 構成の更新は他のコレクションには影響しません。
構成の数	理論的には無制限。 サーバーの収集構成の数は 100 以下にすることをお勧めします。
マルチテナント分離	収集構成情報間の分離。

表 3-6 : リソースとパフォーマンスの制限

項目	能力と制限
ログ処理のスループット	raw ログトラフィックのデフォルトの制限は 2 MB/s です。（データはエンコードされ、圧縮された後にアップロードされますが、一般に圧縮率は 5～10 倍です）。ログトラフィックが制限を超えると、ログが失われる可能性があります。 パラメータを調整するには、 Logtail 起動設定パラメーター configurations parameters を参照してください。
最大パフォーマンス	単一のコア条件での最大処理能力：単純なログファイルの場合は 100 MB/秒、正規表現を使用するログファイルの場合はデフォルトで 20 MB/秒（正規表現の複雑さに応じて）、a の場合は 40 MB/秒区切り文字ログファイル、JSON ログファイルの場合は 30 MB/秒です。 複数のログ処理スレッドを開始すると、パフォーマンスが 1.5～3 倍向上します。
監視対象ディレクトリの数	Logtail は監視対象ディレクトリの深さを積極的に制限し、リソースを節約します。上限に達すると、Logtail はさらに多くのディレクトリとログファイルの監視を停止します。 Logtail は、最大 3,000 のディレクトリ（サブディレクトリを含む）を監視します。
既定のリソース制限	既定では、Logtail は CPU 使用率の最大 40% と 256 MB のメモリ使用量を占めます。 ログが高速に生成された場合は、Logtail 起動設定パラメーターを使用してパラメータを調整できます。
リソース制限を超える処理ポリシー	3 分で Logtail が占めるリソースが上限を超えた場合、Logtail は強制的に再起動され、データの損失や重複が発生する可能性があります。

表 3-7 : エラー処理の制限

項目	能力と制限
ネットワークエラー処理	ネットワーク接続が異常な場合、Logtail は積極的に再試行し、自動的に再試行間隔を調整します。
リソースクォータの処理が最大割り当て量を超えた場合	データ転送速度が Logstore の最大割り当て量を超えた場合、Logtail はログ収集をブロックし、自動的に再試行します。
タイムアウトの最大リトライ時間	データ送信が連続して 6 時間以上失敗した場合、Logtail はデータを破棄します。
ステータス自己チェック	プログラムの異常終了やリソース制限を超えるなど、例外が発生した場合、Logtail は自動的に再起動します。

表 3-8 : その他の制限

項目	能力と制限
ログ収集遅延	通常、ログがディスクにフラッシュされた後、Logtail によるログ収集の遅延は（輻輳を除いて）1 秒を超えません。
ログアップロードポリシー	Logtail は、ログをアップロードする前に自動的に同じファイルにログを集約します。 ログのアップロードは、2,000 を超えるログが生成されたり、ログファイルが 2 MB を超えたり、ログ収集が 3 秒を超えたりするという条件でトリガされます。

4 クラウドプロダクトのログ収集

4.1 クラウドプロダクトログ

Log Service は、ECS、OSS、SLB などのさまざまなクラウド製品からログを収集できます。 ログには、操作情報、操作状況、サービス状況などの製品情報が記録されています。

次の表に、Log Service でログを収集できるクラウドプロダクトを示します。

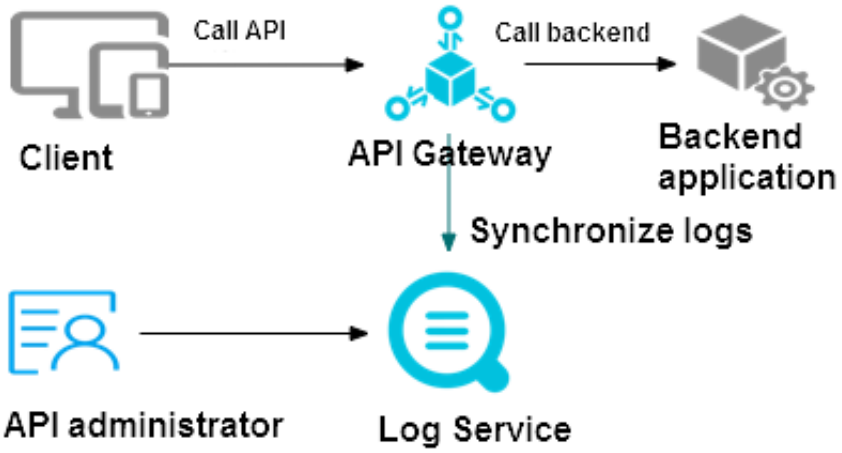
タイプ	クラウド製品名	有効化する方法	Details
エラスティックコンピューティング	Elastic Compute Service (ECS)	Logtail をインストールして有効化	Logtail の紹介
	Container Service / Container Service for Kubernetes	Container Service コンソールにて有効化	テキストログ と stdout
ストレージ	Object Storage Service (OSS)	OSS コンソールにて有効化	OSS アクセスログ
ネットワーク	Server Load Balancer (SLB)	SLB コンソールにて有効化	レイヤ 7 SLB のアクセスログ
	Virtual Private Cloud (VPC)	VPC コンソールにて有効化	フローログ
	API Gateway	API Gateway コンソールにて有効化	API Gateway アクセスログ
セキュリティ	ActionTrail	ActionTrail コンソールにて有効化	ActionTrail の概要
	DDoS 防御	DDoS 防御コンソールにて有効化	DDoS 防御の概要
	Threat Detection Service	Threat Detection Service Enterprise Edition を購入し、Threat Detection Service コンソールでサービスを有効にします。	Log retrieval
	Anti-Bot Service	Anti-Bot Service コンソールにて有効化	Anti-Bot Service ログ

タイプ	クラウド製品名	有効化する方法	Details
アプリケーション	Log Service (LOG)	Log Service コンソールにて有効化	Log Service の概要

4.2 API Gateway アクセスログ

Alibaba Cloud API Gateway は、マイクロサービスの集約、フロントエンドとバックエンドの分離、およびシステム統合を容易にする API ホスティングサービスを提供します。 アクセスログは、 Web サービスによって生成されるログです。 各 API リクエストは、呼び出し元 IP、要求された URL、応答待ち時間、返されたステータスコード、各要求と応答のバイト数、およびその他の情報を含むアクセスレコードに対応します。 前述の情報を使用すると、Web サービスの動作状況を理解できます。

図 4-1 : APIゲートウェイ



Log Service を使用すると、 Data Import Wizard を使用して API Gateway アクセスログを収集できます。

特徴

1. オンラインログクエリ：ログ内の任意のキーワードを使用して、迅速で正確なファジークエリを実行できます。 この機能を使用すると、問題の特定やクエリのカウントに使用できます。
2. 詳細な通話記録： API 通話記録の詳細を検索することができます。
3. カスタマイズされた分析チャート：あなたのビジネス要件を満たす統計的な要件に応じて、任意のログ項目を統計チャートにカスタマイズすることができます。
4. 解析レポート： API Gateway は、要求量、成功率、失敗率、待ち時間、API を呼び出すアプリケーションの数、障害統計、TOP グループ化、 TOP API、および TOP 遅延を含むいくつかのグローバル統計図を事前定義します。

フィールドの説明

ログフィールド	説明
apiGroupUid	API グループ ID。
apiGroupName	API グループ名。
apiUid	API ID。
apiName	API 名。
apiStageUid	API ステージ ID。
apiStageName	API ステージ名。
httpMethod	呼び出される HTTP メソッド。
path	要求されたパス。
domain	呼び出されるドメイン名。
statusCode	HTTP ステータスコード
ErrorMessage	エラーメッセージ。
appId	呼び出し元のアプリケーション ID。
appName	呼び出し元のアプリケーション名。
clientIp	呼び出し側のクライアント IP。
Exception	バックエンドによって返された特定のエラーメッセージ。
providerAliUid	API プロバイダのアカウント ID。
region	リージョン、例えば cn-hangzhou。
requestHandleTime	リクエスト時間（GMT）。
RequestId	リクエスト ID。グローバルに一意です。
requestSize	リクエストのサイズ（バイト単位）。
Responsesize	返されるデータのサイズ（バイト単位）。
Servicelatency	バックエンドの待ち時間（ミリ秒単位）。

手順

1. プロジェクトと Logstore を作成します。

   プロジェクトと Logstore の作成方法については、準備をご参照ください。

   Logstore がすでに存在する場合は、この手順をスキップします。

2. データアクセスウィザードを開始します。

   Logstore を作成したら、Logstore リストページで Data Import Wizard アイコンをクリックしてください。

3. データタイプを選択します。

   クラウド製品ログの API Gateway をクリックし、 次へをクリックしてデータソース設定の手順に進みます。

4. データソースを設定します。

   データソース設定ステップで、次の設定を完了しているかどうかを確認します。

   a. API Gateway サービスを有効にする。

      API Gateway は完全な API ホスティングサービスを提供し、能力、サービス、およびデータを API の形式でパートナーに公開するのに役立ちます。

      API Gateway サービスを有効にしていない場合は、関連ページの指示に従ってサービスを有効にします。

   b. 完全な Resource Access Management （RAM） の承認ログ情報を Logstore に収集できるように、配布ルールを設定する前に RAM を使用して Log Service を認可します。

      配布ルールを設定する前に RAM を使用して Log Service を認可します。

      承認を迅速に行うには、右上隅の承認をクリックします。

   c. 配信ルールを確立する。

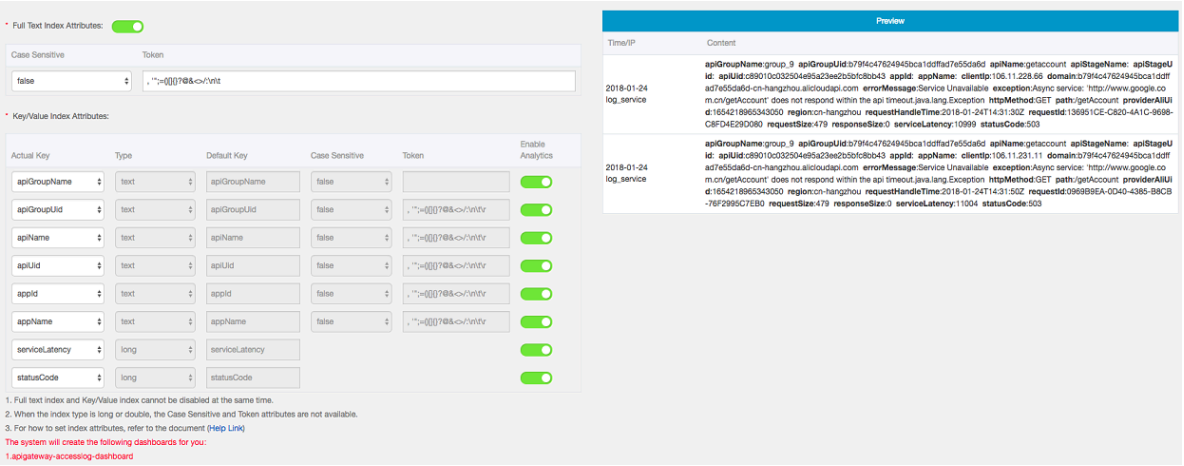
      この手順を初めて実行する場合、システムは自動的に API Gateway ログをインポートし、配布ルールを確立します。 以前に API Gateway log collection を設定していた場合、Log distribution rules already exists メッセージが表示されます。 既存の配布ルールを削除することもできます。

   次へをクリックして、検索と分析と視覚化ページに入ります。

5. 検索と分析と可視化を設定します。

   次の図に示すように、索引を構成します。 ダッシュボードでこの設定を使用するので、この設定を変更するときは注意する必要があります。

   図 4-2 : 索引の構成



   次へをクリックして設定を完了します。 ログシッパーは、必要に応じて個別に設定することができます。

ウィザードの初期化が完了しました。 設定した Logstore api-gateway-access-log を選択して、ログを照会および分析したり、ダッシュボードにアクセスしてレポートを表示することができます。

4.3 レイヤー 7 Server Load Balancer のアクセスログ

Alibaba Cloud の Server Load Balancer を使用することにより、複数の Elastic Compute Service (ECS) インスタンスにトラフィックを分散させることができます。Server Load Balancer には、レイヤー 4 Server Load Balancer (TCP)、および、レイヤー 7 Server Load Balancer (HTTP/HTTPS) があります。 Server Load Balancer を使用すると、1 つの ECS インスタンスに例外があった際、サービスへの影響が抑えられ、システム可用性は向上します。 Auto Scaling を併用し、 ECS のトラフィック量に応じてバックエンドサーバーを自動拡張/縮小させることもできます。

Server Load Balancer へのアクセス要求はすべて、アクセスログに記録されます。 アクセスログには、リクエスト時間、クライアントの IP アドレス、遅延、リクエストパス、およびサーバーレスポンスといった、Server Load Balancer に送信されるリクエストの詳細がすべて記録されます。 Server Load Balancer はインターネットアクセスポイントであるため、大量のアクセス要求が処理されます。 そのアクセスログより、クライアントユーザーの行動パターンや地理

的分布を分析することができます。また、問題のトラブルシューティングに役立てることもできます。

Server Load Balancer のアクセスログの収集には Log Service をご利用ください。 継続してレイヤー 7 (HTTP/HTTPS) のアクセスログをモニタリング、調査、診断、通知受信していくことにより、Server Load Balancer インスタンス全体を把握することができます。

 注 :
Log Service は、レイヤー 7 Server Load Balancer にのみ対応していますが、 全リージョンで利用できます。 詳細は、「#unique_123」をご参照ください。

利点

- シンプル: 開発者および管理者は、ログの処理に時間と手間をかける必要がなくなり、サービス開発および技術研究に専念できるようになります。
- 大容量処理: アクセスログのデータ量は、Server Load Balancer インスタンスのリクエスト PV に比例します。 一般的にデータ量が多いため、 アクセスログの処理に、コストパフォーマンスを考慮することは必須です。 Log Service は 1 秒で 1 億のログを分析することができるため、オープンソースのソリューションと比較してコスト面において確実に利点があります。
- リアルタイム性: DevOps、モニタリング、警告には、ログデータにリアルタイム性が求められます。 しかし、従来のデータ保存および分析ツールにはそのリアルタイム性がありません。 たとえば、Hive データの ETL は非常に時間がかかりますが、その大半はデータの統合処理です。 強力なコンピューティング機能を搭載した Log Service は、アクセスログを数秒で処理および分析します。
- 柔軟性: Server Load Balancer インスタンスごとにアクセスログの取得を有効/無効にすることができます。Server Load Balancer インスタンスごとにアクセスログの取得を有効/無効にすることができます。 また、保存期間 (1 ~ 365 日) も設定できます。なお、Logstore のサイズはサービスの成長に合わせて自動的に拡張されます。

Log Service にレイヤー 7 Server Load Balancer のアクセスログを収集するための設定

前提条件

1. Server Load Balancer および Log Service が有効になっていること。 また、作成した #unique_124、Log Service プロジェクト、および Logstore は同一リージョンであること。

    注 :

レイヤー 7 Server Load Balancer のアクセスログのみを収集することができます。 すべてのリージョンでご利用いただけます。

2. RAM ユーザーが SLB アクセスログへのアクセス権を有していること。 詳細は、「#unique_125」をご参照ください。

手順

1. Log Service コンソールにログインします。
2. プロジェクトおよび Logstore を作成後、ページの指示に従ってデータインポートウィザードを起動します。(または、Logstore リストページのデータインポートウィザードアイコンをクリックします。)
3. データソースを選択します。

   Cloud Services の Server Load Balancer 、次へを順にクリックします。
4. RAM の権限付与

   ページの指示に従って権限付与、 権限付与ポリシーの確認を順にクリックし、Server Load Balancer に Log Service へのアクセス権を付与します。
5. 送信ルールを設定します。 送信設定をクリックし、Server Load Balancer コンソールに移動します。

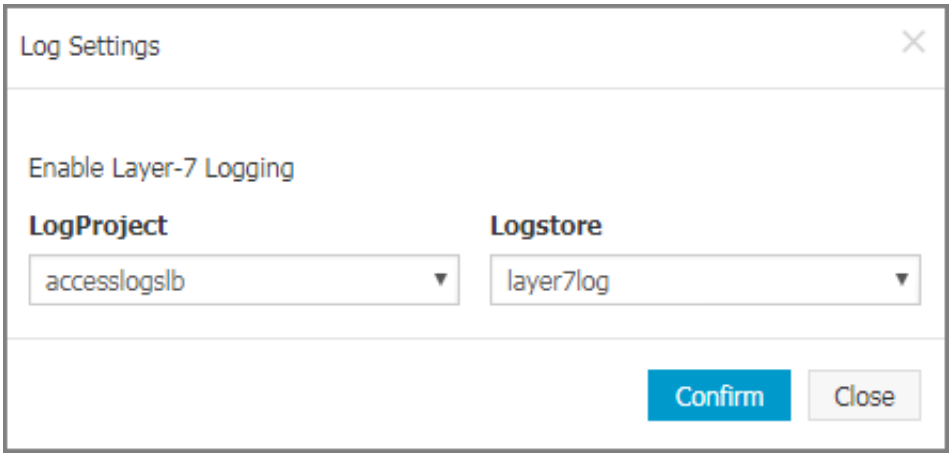
   a. 左側のナビゲーションメニューより、ログ > アクセスログを順にクリックします。

   b. Server Load Balancer インスタンスの右側の設定をクリックします。

   注 :

Log Service プロジェクトおよび SLB インスタンスが同じリージョンであること。

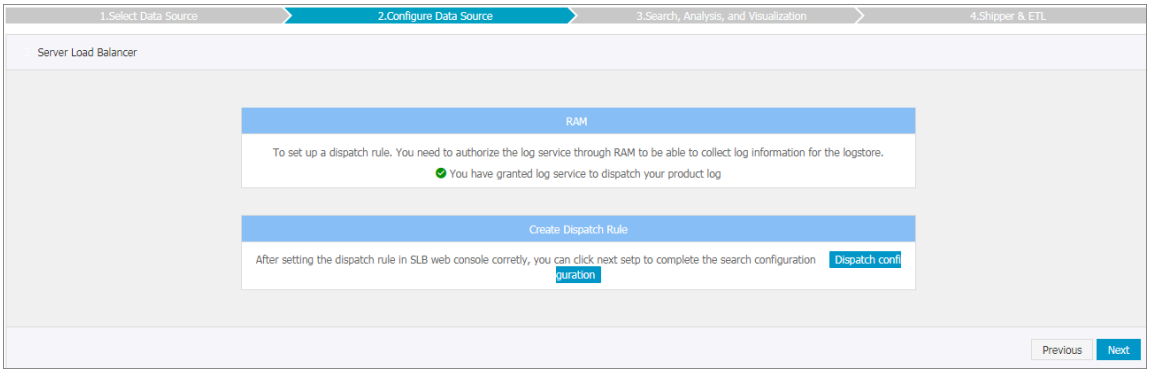
図 4-3 : ログの設定



c. Log Service のプロジェクトおよび Logstore を選択し、 確認をクリックします。

d. 設定し終わったら、ダイアログボックスを閉じます。 データインポートウィザードに戻り、次へをクリックします。

図 4-4 : データソースの設定



6. 照会/分析/可視化

Log Service には Server Load Balancer の照会フィールドがあらかじめ定義されています。 各フィールドの詳細は、下記「フィールドの説明」をご参照ください。 次へをクリックします。

注 :

Logstore 名で始まる、{LOGSTORE}-slb_layer7_access_center ダッシュボード および {LOGSTORE}-slb_layer7_operation_center ダッシュボードが自動生成されます。 設定が完了すると、ダッシュボードページに表示されます。

7. 確認をクリックしてデータインポートウィザードを終了します。

その他の操作

- リアルタイムにログを照会

  ログ内のキーワードを使用して、迅速かつ正確なクエリまたはあいまいクエリを実行することができます。 問題の特定や、統計クエリに使用します。

- 分析レポートのテンプレート

  Server Load Balancer には、アクセス数の多いクライアント、リクエストステータスコードの分布、アクセス数の多い URI、リクエストのトラフィック量の変化、およびサーバーの応答時間の統計といった、包括的な統計グラフが事前に定義されています。

- 分析グラフの作成

  任意のログ項目に対してアドホッククエリを実行し、その実行結果をグラフとして保存し、サービスの運用管理に役立てることができます。

- ログモニタリングアラームの設定

  Server Load Balancer のアクセスログに対してカスタム分析を実行し、その実行結果をクイック照会として保存することができます。 また、クイック照会をアラーム通知として設定すると、 リアルタイムログの処理結果が、設定されているしきい値を超えた場合、システムよりアラーム通知が送信されます。

フィールド説明

フィールド	説明
body_bytes_sent	クライアントに送信された HTTP 本文のサイズ (単位: byte)
client_ip	リクエストを送信したクライアントの IP アドレス
host	リクエストパラメータの host 値 (リクエストパラメータに host 値がない場合は、host ヘッダーの host 値。 host ヘッダーにも host 値がない場合は、リクエストのバックエンドサーバーの IP アドレスが host 値)
http_host	リクエストの host ヘッダーのコンテンツ
http_referer	プロキシの受信したリクエストの HTTP リファラーヘッダーのコンテンツ
http_user_agent	プロキシの受信したリクエストの HTTP user-agent ヘッダーのコンテンツ
http_x_forwarded_for	プロキシの受信したリクエストの x-forwarded-for のコンテンツ

フィールド	説明
http_x_real_ip	送信元 IP アドレス
read_request_time	プロキシのリクエスト読み込み時間 (単位: ミリ秒)
request_length	startline、HTTP ヘッダー、および HTTP 本文を含めたリクエストメッセージの長さ
request_method	リクエストメソッド
Request_time	プロキシが最初のリクエストを受信してからレスポンスを返すまでの時間 (単位: 秒)
request_uri	プロキシの受信したリクエストの URI
scheme	リクエストスキーマ（http または https）
server_protocol	プロキシの受信した HTTP プロトコルのバージョン (例: HTTP/1.0 または HTTP/1.1)
slb_vport	Server Load Balancer のリスニングポート
slbid	Server Load Balancer インスタンスの ID
ssl_cipher	暗号スイート (例: ECDHE-RSA-AES128-GCM-SHA256/)
ssl_protocol	SSL/TSL 接続のプロトコル (例: TLS v1.2)
status	メッセージに対するプロキシのレスポンスステータス
tcpinfo_rtt	クライアントの tcp RTT (単位: マイクロ秒)
time	ログが書き込まれた時間
Upstream_addr	バックエンドサーバーの IP アドレスおよびポート
upstream_response_time	Server Load Balancer のバックエンドサーバーへの接続確立から、データを送信し、接続が切断されるまでの時間 (単位: 秒)
upstream_statu	プロキシの受信したバックエンドサーバーのレスポンスステータスコード
vip_addr	VIP アドレス
write_response_time	プロキシのレスポンス書き込み時間 (ミリ秒)

4.4 DDoS ログ収集

4.4.1 概要

Alibaba Cloud Anti-DDoS Pro は、インターネットサーバー（Alibaba Cloud 以外のホストを含む）向けの有料サービスです。 大トラフィック DDoS 攻撃後にサービスが利用できなくなるリスクを回避するために、有料サービスを適用できます。 Anti-DDoS Pro を構成し、攻撃トラフィックをハイプロテクション IP へ誘導することで、送信元の安定さ、及び信頼性を確保します。

背景情報

インターネットのセキュリティは常に課題に直面しています。 DDoS 攻撃を始めとしてのネットワークの脅威は、ネットワークセキュリティに深刻な影響を及ぼします。

DDoS 攻撃は、大規模化、モバイル化、及びグローバル化の方向に向かっています。 最近の調査報告によると、DDoS 攻撃の頻度は増加傾向です。 ハッカーの攻撃は隠密性が高く、セキュリティ対策が不十分なクラウドサービスプロバイダをコントロールして、IDC、そして大量なカメラに攻撃を仕掛けることができます。 その攻撃は既に闇産業チェーンに形成し、より組織的になってきています。 同時に、攻撃モードは極性化し、スロー攻撃、混合攻撃、特に CC 攻撃の割合が増加するため、防御の検出がより困難になります。 1Tbps を超える攻撃のピーク値がもはや一般的であり、100GB の攻撃の回数は倍で成長していく。 一方、アプリケーション層の攻撃も大幅に増加しています。

Kaspersky 2018Q1 DDoS リスクレポートによると、中国は依然として DDoS 攻撃の主な標的である。 攻撃を受けている主な産業は、インターネット、ゲーム、ソフトウェア、そして金融会社です。 DDoS 攻撃の 80% 以上が HTTP と CC 攻撃を組み合わせており、高レベルの隠密性を持っています。 そのため、ログを使用してアクセスと攻撃の動作を分析し、保護戦略を適用することが特に重要である。

Log Service は、Alibaba Cloud Anti-DDoS Pro の Web サイトアクセスログ、CC 攻撃ログのリアルタイム収集をサポートし、収集されたログデータのリアルタイムクエリと分析をサポートします。 クエリの結果はダッシュボードの形式で表示されます。

利点

- 簡単な構成：リアルタイムで保護されたログを取得するように簡単に構成できます。
- リアルタイム分析：ログサービスと連動することで、リアルタイムのログ分析とすぐに使用可能なレポートセンターを提供し、CC 攻撃のステータスとカスタマーアクセスの詳細に関する情報を提供します。

- リアルタイムアラーム：特定のインジケータに基づいたカスタムモニタリングとアラームをリアルタイムでサポートし、重要なビジネス例外にタイムリーに対応します。
- エコシステム：ストリームコンピューティング、クラウドストレージ、さらにデータ価値を探求するための視覚化ソリューションなど、他のエコシステムのドッキングをサポートします。
- 無料利用クォータ：無料のデータインポートクォータ、3 日間の無料ログ保存、クエリ、およびリアルタイム分析を提供します。 コンプライアンス管理、トレース、およびファイリングのために保管期間を自由に拡張できます。 無制限の保管時間をサポートし、保管コストは1ヶ月あたり0.35 USD / GBとなります。

制限と説明

- 専用ログストアは追加データの書込みをサポートしません。

  専用ログストアは Anti-DDoS Pro Web サイトのログの保存のみに使用されるため、他のデータの書き込みはサポートされていません。 クエリ、統計、アラーム、ストリーミングの消費など、他の機能に対する制限はありません。
- 従量課金。DDoS ログ収集保護が有効になっていない場合、料金は発生しません。

  DDOS ログ収集機能は、Log Service の請求項目に従って課金されます。 DDoS ログ収集保護が有効になっていない場合、料金は発生しません。 Log Service は従量課金をサポートし、無料利用クォータも提供しています。 詳細は、課金方法をご参照ください。

シナリオ

- Webサイトアクセス例外のトラブルシューティング

  Log Service は DDoS ログを収集するように構成されると、収集したログをリアルタイムでクエリおよび分析できます。 SQL ステートメントを使用して DDoS アクセスログを分析することで、Web サイトのアクセス例外をすばやく確認して分析し、読み書きの遅延やキャリアの分布などの情報を表示できます。

  たとえば、次のステートメントを使用して DDoS アクセスログを表示します。

```
__topic__ : ddos_acces s_log
```

- CC 攻撃元の追跡

  CC 攻撃の分布と発生源は DDoS アクセスログに記録されます。 DDoS アクセスログに対してリアルタイムのクエリと分析を実行することで、ソースの追跡、CC 攻撃の追跡、および応答戦略の参照を提供できます。

  たとえば、DDoS アクセスログに記録されている CC 攻撃の国別分布を次の文で分析します。

```
__topic__ : ddos_acces s_log and cc_blocks > 0 | SELECT
  ip_to_coun try ( if ( real_clien t_ip ='-', remote_add r ,
```

```
real_clien t_ip )) as country , count ( 1 ) as "攻撃回数"
group by country
```

- たとえば、次の文で PV アクセスを表示します。

```
__topic__ : ddos_acces s_log | select count ( 1 ) as PV
```

- ウェブサイト運営分析

  DDoS アクセスログはリアルタイムでウェブサイトのアクセスデータを記録します。 収集したアクセスログデータの SQL クエリ分析を実行して、Web サイトの人気度、アクセスの発信元とチャネル、クライアントの分布などのリアルタイムのアクセス状況を取得し、Web サイトの運営分析をサポートできます。

  たとえば、複数のネットワーククラウドからの訪問者トラフィック分布を表示します。

```
__topic__ : ddos_acces s_log | select ip_to_prov ider ( if (
real_clien t_ip ='-', remote_add r , real_clien t_ip )) as
provider , round ( sum ( request_le ngth )/ 1024 . 0 / 1024 . 0
, 3 ) as mb_in group by provider having ip_to_prov
ider ( if ( real_clien t_ip ='-', remote_add r , real_clien
t_ip )) <> '' order by mb_in desc limit 10
```

4.4.2 収集手順

Anti-DDoS Pro コンソールで、Web サイトの DDos ログ収集機能を有効にできます。

1. Anti-DDoS Pro ログ収集機能を有効にし、Anti-DDoS Pro インスタンスを購入してから、オンライン構成を行います。
2. Anti-DDoS Pro ログ収集機能を有効にし、Anti-DDoS Pro インスタンスを購入します。
3. Log Service を有効化します。

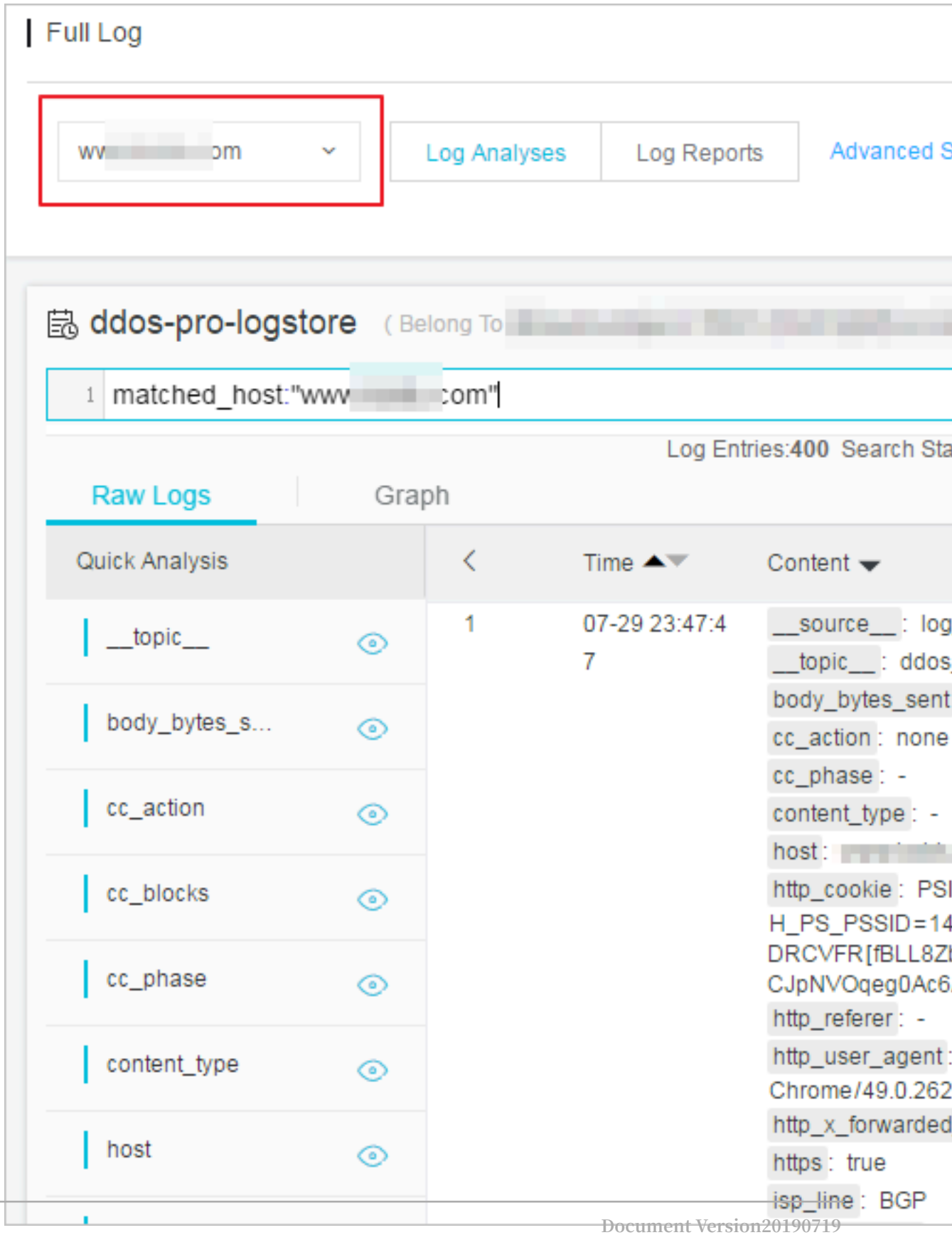
Log Service は、Alibaba Cloud Anti-DDoS Pro Web サイトのアクセスログ、CC 攻撃ログのリアルタイム収集をサポートし、収集されたログデータのリアルタイムクエリと分析をサポートします。 クエリの結果はダッシュボードの形式で表示され、ログはアクセスと攻撃の様子をリアルタイムで分析し、セキュリティ部門が保護の方針を策定するのに役立ちます。

1. Anti-DDoS Pro コンソールにログインし、左側のナビゲーションペインで Log > Full Log を選択します。 Full Log ページを開きます。
2. 初めて DDoS ログ収集を設定する場合は、ページの指示に従います。

   DDoS は、権限付与されると DDoS ログをログストアに配信する権限を持ちます。

3. DDoS ログ収集機能を有効にする Web サイトを選択し、ステータス がオンになっていることを確認します。

図 4-5 : 機能の有効化



これで、現行の Web サイトで DDoS ログ収集を有効にしました。 Log Service はご使用するアカウントの下に自動的にログストアを作成します。 DDoS は、この機能が有効になっている Web サイトのすべてのログをこのログストアにインポートします。 ログストアのデフォルト構成について、デフォルト構成をご参照ください。

表 4-1 : デフォルト構成

デフォルト構成項目	構成内容
Project	デフォルトでは、 `ddos - pro - logstore` プロジェクトが作成されています。
Logstore	デフォルトでは、ログストアが作成されています。 ログストア名は、購入した DDoS のドメインによって決まります。 ・ 中国本土の DDoS インスタンス：`ddos-pro-project-Alibaba Cloud Account ID-cn-hangzhou` ・ その他の DDoS インスタンス： `ddos-pro-project-Alibaba Cloud Account ID-ap-southeast-1` DDoS ログ収集機能により生成されたすべてのログは、このログストアに保存されます。
リージョン	・ DDoS リージョンが中国本土にある場合、デフォルトのプロジェクトは China East 1 に保存されます。 ・ DDoS リージョンが中国本土外にある場合、デフォルトのプロジェクトは Asia Pacific SE 1 に保存されます。
Shard	デフォルトでは、2 つのシャードが作成され、自動分割シャード 機能がオンになっています。
ログの保存期間	デフォルトの保管期間は、無料クォータ内で 3 日間です。 3 日後にログは自動的に削除されます。 保管期間を長くするには、構成をカスタマイズします。 詳細については、課金方法セクション内のWeb サイトログの保存期間を変更する方法を参照してください。

デフォルト構成項目	構成内容
ダッシュボード	デフォルトでは、2 つのダッシュボードが作成されます。 ・ `ddos - pro - logstore_ ddos_operation_center`: オペレーションセンター ・ `ddos - pro - logstore_ ddos_access_center`:アクセスセンター ダッシュボードの詳細は、ログレポート をご参照ください。

現行の Full Log ページで、収集したログをリアルタイムでクエリおよび分析できます。 ログフィールドの説明については、次の図を参照してください。 さらに、Log Service は DDoS オペレーションセンターとアクセスセンターの 2 つのダッシュボードを作成しています。
ダッシュボードの構成情報をカスタマイズできます。

フィールド	説明	例
__topic__	ログのトピックは ddos_access_log に限定されています。	-
body_bytes_sent	Body のサイズを送信するようリクエストします。 単位：バイト	2
content_type	コンテンツのタイプ。	application/x-www-form-urlencoded
host	ソース Web サイト。	api.zhihu.com
http_cookie	リクエスト cookie。	k1=v1;k2=v2
http_referer	リクエストリファラー ヌルの場合は、`-` と表示されます。	http://xyz.com
http_user_agent	ユーザーエージェントのリクエスト。	Dalvik/2.1.0 (Linux; U; Android 7.0; EDI-AL10 Build/HUAWEIEDISON-AL10)
http_x_forwarded_for	プロキシにリダイレクトされたアップストリームユーザー IP。	-

フィールド	説明	例
https	HTTPS リクエストであるかどうかを判断します。そのうち： ・ true: HTTPS リクエスト。 ・ false: HTTP リクエスト。	true
matched_host	構成がマッチングしたソース Web サイトは、汎ドメイン名である可能性があります。マッチングしていない場合は、 - と表示されます。	*.zhihu.com
real_client_ip	カスタマーの実 IP にアクセスします。 利用できない場合は、 - と表示されます。	1.2.3.4
isp_line	BGP、テレコミュニケーション、Unicom などの回線情報。	テレコミュニケーション
remote_addr	リクエストの発信元クライアント IP 。	1.2.3.4
remote_port	リクエストの発信元クライアントポート。	23713
request_length	リクエストの長さ。 単位：バイト。	123
request_method	HTTP リクエストの方式。	GET
request_time_msec	リクエスト時刻。 単位：マイクロ秒。	44
request_uri	リクエストパス。	/answers/377971214/banner
server_name	マッチングした host 名。マッチングしていない場合、 default と表示されます。	api.abc.com
status	HTTP ステータスコード。	200
time	時刻。	2018-05-02T16:03:59+08:00

フィールド	説明	例
cc_action	none、challenge、pass、close、captcha、wait、login、n などの CC 保護ポリシー。	close
cc_blocks	CC 保護がブロックされているかどうかを示します。 ・ 1: ブロックされています。 ・ その他コード: Passed.	1
cc_phase	seccookie、server_ip_blacklist、static_whitelist、 server_header_blacklist、server_cookie_blacklist、server_args_blacklist、qps_overmax などの CC 保護ポリシー。	server_ip_blacklist
ua_browser	ブラウザー。	ie9
ua_browser_family	ブラウザーシリーズ。	internet explorer
ua_browser_type	ブラウザータイプ。	web_browser
ua_browser_version	ブラウザーバージョン。	9.0
ua_device_type	クライアントデバイスタイプ。	computer
ua_os	クライアント OS。	windows_7
ua_os_family	クライアント OS シリーズ。	windows
upstream_addr	送信元アドレスリストを返します。形式： `IP : Port`。 複数のアドレスはカンマで区切ります。	1.2.3.4:443
upstream_ip	実際の返信元アドレス IP。	1.2.3.4
upstream_response_time	ソースの応答時間。 単位：秒。	0.044
upstream_status	ソースリクエストの HTTP ステータスを返します。	200
user_id	Alibaba Cloud ユーザー ID。	12345678

- 収集されたログデータでログ分析、クエリ分析の順にクリックします。
- ログレポートをクリックして組み込みダッシュボードを表示します。
- 詳細管理をクリックして Log Service コンソールに移動し、統計情報のクエリと収集、ストリームの消費、収集したログデータのアラームの設定を行います。

4.4.3 ログ分析

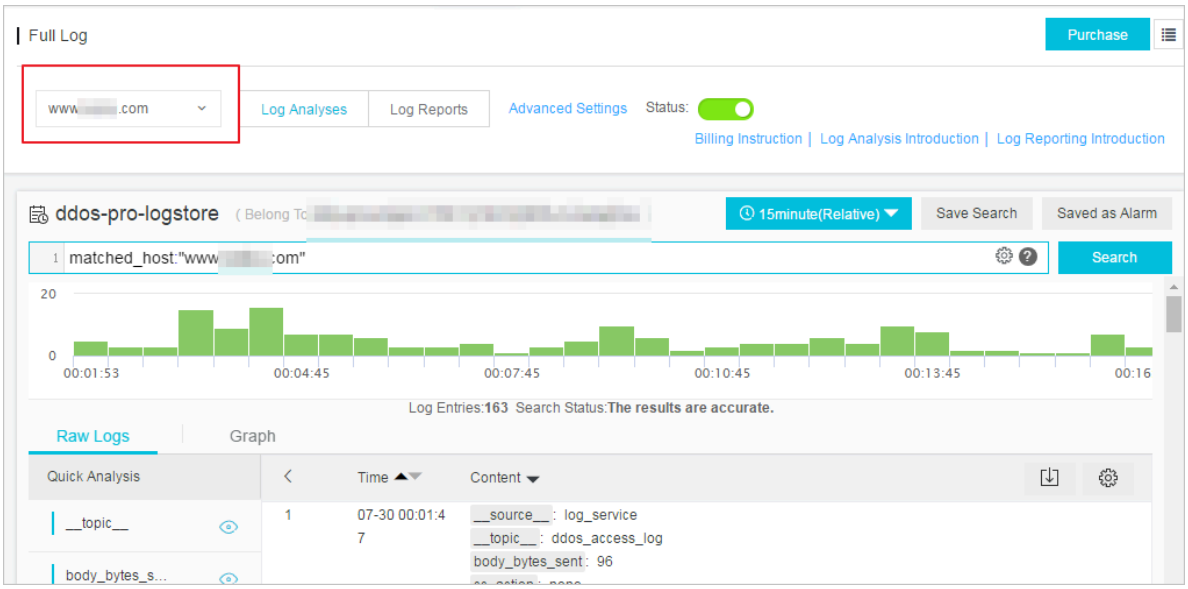
Anti-DDoS Pro は、Log Service のフルログページをログ分析とログレポートに埋め込んでいます。 特定の Web サイトに対して DDoS ログ保護機能を有効にすると、現行ページでリアルタイムに収集されたログデータを照会および分析し、ダッシュボードを表示または編集し、モニタリングアラームを設定できます。

手順

1. Anti-DDoS Pro コンソールにログインし、左側のナビゲーションペインで Log > Full logを選択します。
2. DDoS ログ収集保護を有効にする Web サイトを選択して、ステータスがオンになっていることを確認します。
3. ログ分析をクリックします。

   現行のページには Log Service のクエリ分析ページが埋め込まれており、システムは自動的に `matched_ho st : www . aliyun . com` などのクエリステートメントを入力して、選択した Web サイトに基づいてログデータを表示します。

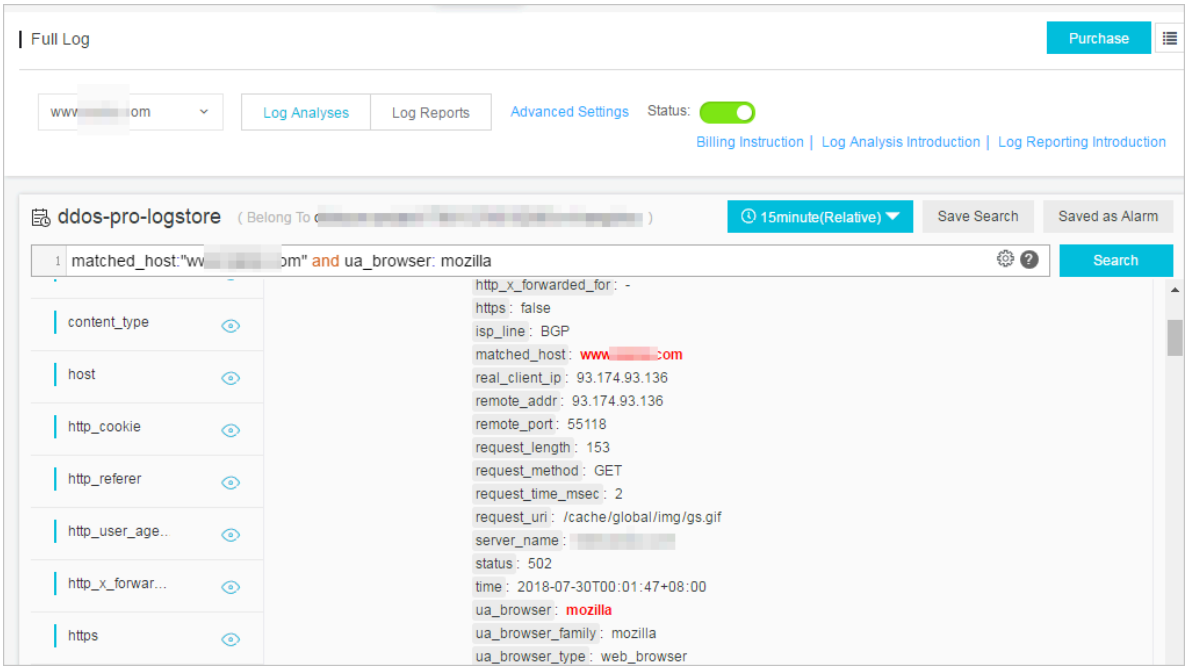
図 4-6 : ログ分析



4. クエリ分析ステートメントを入力し、ログ時間範囲を選択してクエリをクリックします。

> 注：
> DDoS ログのデフォルトの保存期間は3日です。 3 日後、ログデータは削除されます。 デフォルトでは、過去 3 日間のログデータのみをクエリできます。 ログ保存時間を変更するには、ログ保存時間の変更をご参照ください。

図 4-7 : ログクエリ



[クエリと分析]ページでは、以下の操作も実行できます。

- カスタムクエリと分析

  Log Service は、多様な複雑なシナリオでログクエリをサポートするために、さまざまなクエリおよび分析構文を提供します。 詳細は、カスタムクエリと分析をご参照ください。

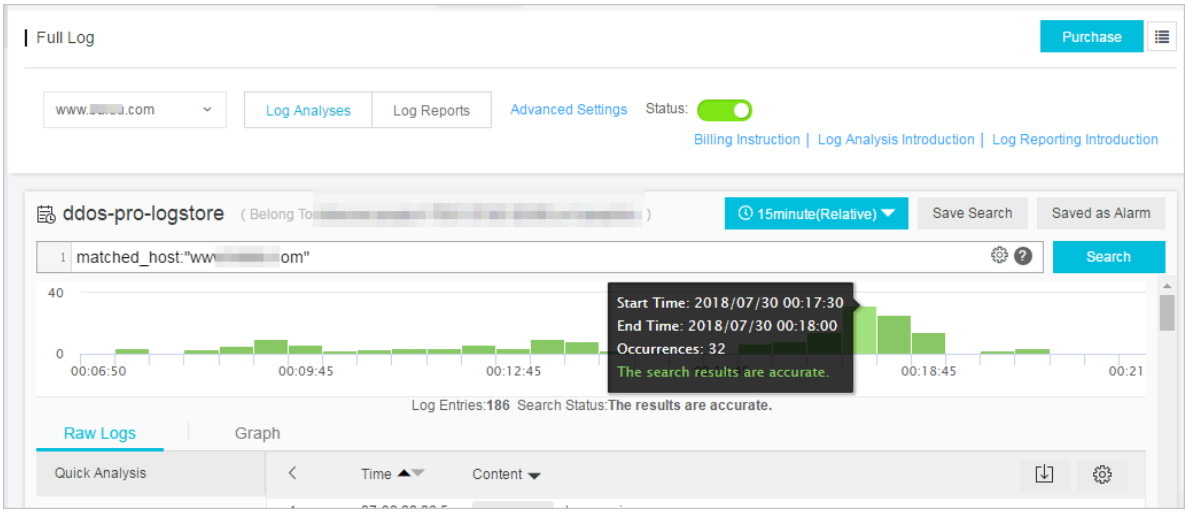
- ログ時間分布を表示する

  検索ボックスの下には、クエリ時刻とクエリステートメントに一致するログの時間分布が表示されます。 時間分布は、横軸と縦軸のヒストグラム形式で表示されます。 クエリされたログの総数が表示されます。

> 注：

ヒストグラムをスライドさせて、より絞り込んだ範囲のタイムゾーンを選択すると、タイムピッカーが選択した時間範囲を自動的に更新して結果を更新します。

図 4-8 : ログ時間分布を表示する



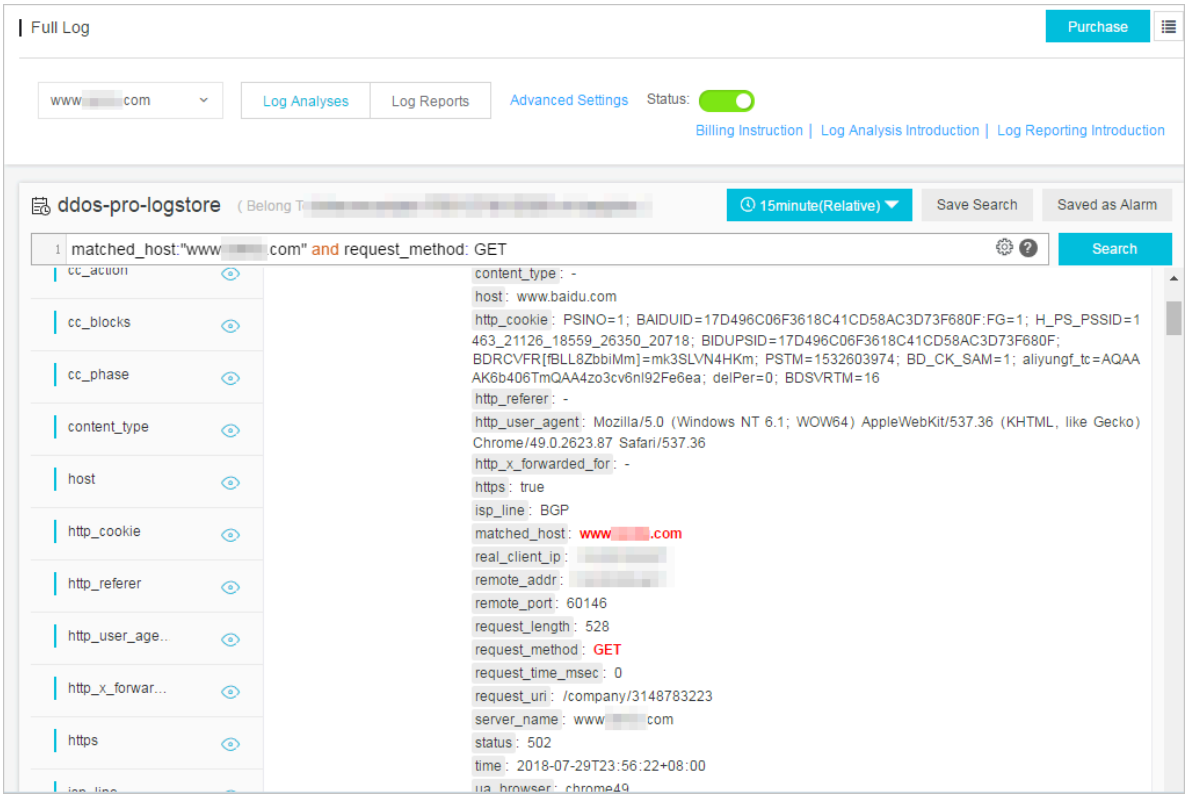
- Raw ログを表示する

  Raw ログでは、各ログの詳細がページ区切りで表示されます。時間、内容、及びそのうちの各フィールドも含まれます。 列の並べ替え、現行のクエリ結果のダウンロード、歯車アイコンをクリックして特定のフィールドを選択して表示することなどができます。

  ページ内の対応するフィールドの値または一部をクリックすると、検索ボックスに適切な検索条件が自動的に入力されます。 たとえば、 `request_me thod : GET` で値 `GET` をクリックすると、次のステートメントが自動的に検索ボックスに追加されます。

```
Raw search statement and request_me thod : GET
```

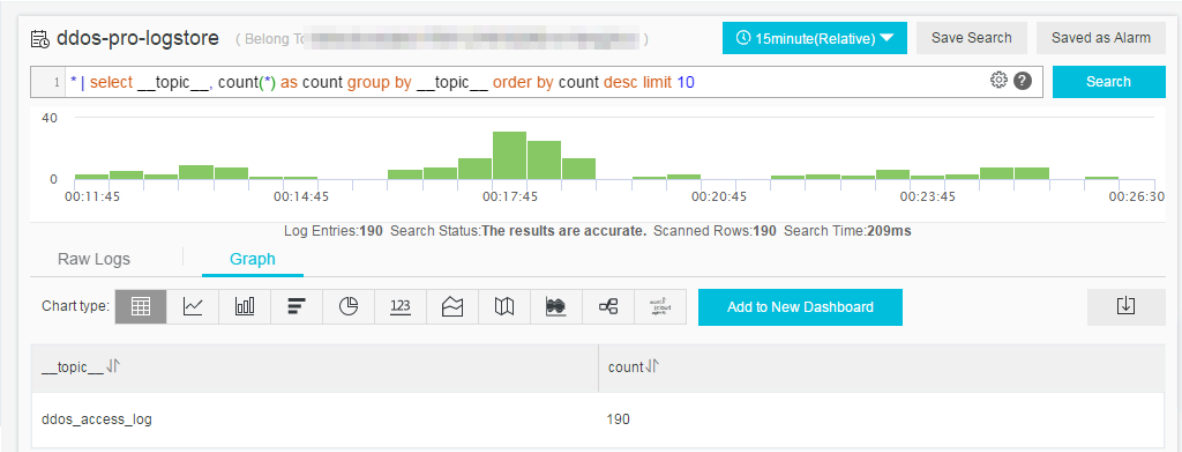
図 4-9 : Raw ログ



- 分析グラフの表示

  Log Service は分析結果のグラフィック表示をサポートしています。統計グラフページでさまざまなグラフタイプを選択できます。 詳細は、分析グラフを参照してください。

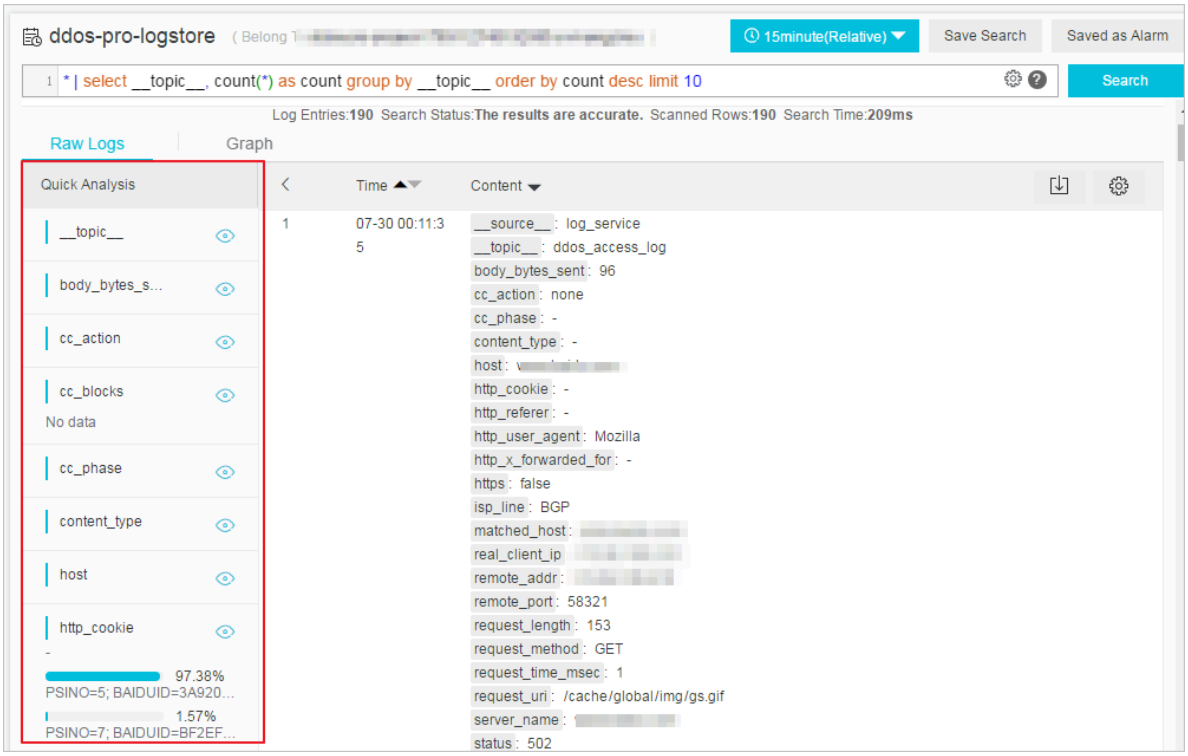
  図 4-10 : 統計グラフ



- クイック分析

  クイック分析機能は、ワンクリックの対話式クエリを提供します。これにより、一定期間にわたるフィールドの分布を迅速に分析し、重要なデータのインデックス付けにかかる時間コストを削減できます。 詳細は、クイック分析をご参照ください。

  図 4-11 : クイック分析



カスタムクエリ分析

ログクエリステートメントは、クエリ構文（Search）と分析構文（Analytics）の 2 つの部分で構成され、 | で区切られています。

```
$ Search  | $ Analytics
```

タイプ	説明
クエリ(Search)	クエリ条件は、キーワード、ファジー、数値、間隔範囲、および組み合わせ条件によって生成できます。 空白のまま、または * にすると、すべてのデータが表示されます。
分析 (Analytics)	クエリ結果または全データ量を計算し、統計します。

 注：
検索と分析は両方オプションです。 Search が空の場合、指定した期間内のすべてのデータはフィルター処理されず、結果は直接統計されます。 Analytics が空の場合、クエリ結果が返され、統計は収集されません。

クエリ構文

Log Service のクエリ構文は、フルテキストクエリとフィールドクエリをサポートしています。クエリボックスは、改行表示、構文の強調表示、及びその他の機能をサポートしています。

- フルテキストクエリ

  フィールドを指定することなく、キーワードクエリを直接入力することができます。 キーワードを二重引用符（""）で囲み、スペースで区切るか、複数のキーワードの間に `and` を挿入します。

  例

  - 複数キーワードクエリ

    `www . aliyun . com` と `error` を含むログを検索します。 例：

    ```
    www . aliyun . com   error
    ```

    または

    ```
    www . aliyun . com   and   error
    ```

  - 条件付きクエリ

    `www . aliyun . com` を含み、 `error` または `404` を含むログを検索します。 例：

    ```
    www . aliyun . com   and  ( error   or   404 )
    ```

  - プレフィックスクエリ

    `www . aliyun . com` を含み、 `failed_` で始まるすべてのキーワードを検索します。 例：

    ```
    www . aliyun . com   and   failed_ *
    ```

    注：
    クエリはプレフィックス + `*` をサポートしますが、`* _error` のようなプレフィックスが`*` になる形式をサポートしません。

- フィールドクエリ

  Log Service は、フィールドに基づくより正確なクエリをサポートしています。

  数値型フィールドの比較は、 `field : value` または `field >= value` のような形式で実装でき、 `and` または `or` を使用して組み合わせることができます。 また、 `and` と `or` の組み合わせを使用して、フルテキスト検索と組み合わせることもできます。

  DDoS Web サイトのアクセスログと攻撃ログもフィールドクエリに基づくことが可能です。各フィールドの意味、種類、形式、およびその他の情報については、DDoS ログフィールドをご参照ください。

  例

  - 複数フィールドクエリ

    CC に攻撃された `www . aliyun . com` を含むログを検索します。

    ```
    matched_ho  st :  www . aliyun . com  and  cc_blocks :  1
    ```

    Web サイト `www . aliyun . com` でクライアント `1 . 2 . 3 . 4` のエラー 404 を含むアクセスログを検索します。

    ```
    real_clien  t_ip :  1 . 2 . 3 . 4  and  matched_ho  st :  www .
    aliyun . com  and  status :  404
    ```

     注：

例 matched_ho st 、 cc_blocks 、 real_clien t_ip 、および status で使用されているフィールドは、DDoS アクセスおよび攻撃ログのフィールドです。フィールドに関する詳細情報は、 DDoS ログフィールドを参照してください。

- 数値フィールドクエリ

  応答時間が 5 秒を超えるすべてのスローリクエストログを検索します。

  ```
  request_ti me_msec > 5000
  ```

  間隔クエリをサポートしています。応答時間が 5 秒を超え 10 秒以下のログを検索します：

  ```
  request_ti me_msec in ( 5000 10000 ]
  ```

  クエリは、次のステートメントでも実行できます。

  ```
  request_ti me_msec > 5000 and request_ti me_msec <=
  10000
  ```

- 日本語が使用されているかどうかを確認します。

  特定のフィールドの存在をクエリします。

  ■ ua_browser フィールドに存在するログをクエリします： ua_browser : *

  ■ ua_browser フィールドに存在しないログをクエリします： not ua_browser : *

クエリ構文の詳細は、インデックスとクエリを参照してください。

分析構文

ログデータの分析と統計には SQL / 92 構文を使用できます。 Log Service でサポートされている構文と機能の詳細は、分析文法を参照してください。

注：

- 解析ステートメントでは、標準 SQL 構文の from table name ステートメント、つまり from log を省略できます。
- ログデータはデフォルトで最初の 100 のエントリを返します。戻り値の範囲はLIMIT 構文を参照して変更できます。

時間ベースのログクエリ分析

各 DDoS ログには、年-月-日- T 時：分：秒 + タイムゾーンの形式の time フィールドがあります。 たとえば、 2018 - 05 - 31T20 : 11 : 58 + 08 : 00 の場合、タイムゾーンは UTC + 8 、つまり北京の時間です。 同時に、各ログには組み込みフィールド：

`__time__` があります。これは、このログの時刻も示しているため、時間ベースの計算を統計で実行できます。 形式：Unix タイムスタンプ。 その本質は、1970 年 1 月 1 日 0 時 0 分 0 秒からの累積秒数です。 そのため、実際の使用では、時刻を表示する前に、まず計算とフォーマットが必要となります。

- 時間の選択及び表示

  特定の期間にわたって、CC に攻撃されたウェブサイト `www . aliyun . com` の最新の 10 のログを選択し、直接 `time` フィールドを使用して、時間、送信元 IP およびアクセスクライアントを表示します。

```
matched_ho st : www . aliyun . com and cc_blocks : 1
| select time , real_clien t_ip , http_user_ agent
order by time desc
limit 10
```

- 時間の計算

  CC攻撃後の日数をクエリするには、`__ time__` を使用して計算します：

```
matched_ho st : www . aliyun . com and cc_blocks : 1
| select time ,
round (( to_unixtim e ( now ()) - __time__ )/ 86400 ,
1 ) as " days_passe d ", real_clien t_ip , http_user_ agent
order by time desc
limit 10
```

  注：

  `round (( to_unixtim e ( now ()) - __time__ )/ 86400 , 1 )` を使用しています。まず `to_unixtim e` を使用して `now ()` より取得した時刻を Unix タイムスタンプに変換します、そして組込みの時刻フィールド `__time__` と減算して経過した秒数を取得します。 最後に、1 日の合計秒数である `86400` で割り、それから関数 `round ( data , 1 )` を使用して 10 進数に丸めます。 1桁の値は、各攻撃ログが何日かを過ぎたことを示します。

- 特定の時間に基づくグループ統計

  Web サイトが CC に攻撃されている毎日の様子を知りたい場合は、次の SQL を使用します。

```
matched_ho st : www . aliyun . com and cc_blocks : 1
| select date_trunc (' day ', __time__ ) as dt ,
count ( 1 ) as PV
group by dt
order by dt
```

  注：

この例では、時間調整のために組み込みの時間フィールド `__time__` を関数 `date_trunc ('day', ..)` に渡します。各ログは、統計の合計数（count（1））ごとに所属する日付のパーティションにまとめられ、パーティションの時間ブロックでソートされます。関数 `date_trunc` の最初の引数は、`second`、`miniute`、`hour`、`week`、`month`、`year` を含むその他の単位の配置を提供します。関数の詳細は、日付と時間の関数をご参照ください。

・時間ベースのグループ統計

より柔軟なグループ化時間ルールを使用するには、たとえば、5 分ごとにCCに攻撃されている Web サイトの傾向を把握するには、数学計算が必要です。次の SQL を実行します：

```
matched_host: www.aliyun.com and cc_blocks: 1
| select from_unixtime(__time__ - __time__ % 300) as dt,
         count(1) as PV
      group by dt
   order by dt
   limit 1000
```

注：

組み込みの時間フィールドを使用して `__time__ - __time__ % 300` を計算し、`from_unixtime` 関数を使用してフォーマットします。各ログは、5分（300秒）のパーティションにまとめられて合計数を統計します（count（1））、そしてパーティションの時間ブロックでソートされ、最初の 1000 のログを取得します（選択した時間内の最初の 83 時間のデータに相当します）。

時間形式の変換など、その他の時間分析関数では、`date_parse` と `date_format` を使用する必要があります。詳細は、日付と時間の関数をご参照ください。

クライアント IP ベースのクエリ分析

DDoS ログにはクライアントの実 IP を取得するための `real_client_ip` というフィールドがあります。ただし、ユーザーがプロキシによって実際の IP を取得できず、ヘッダー内の IP アドレスが正しくない場合は、`remote_addr` フィールドを使用してクライアント IP に直接接続できます。

・アタッカーの国分布

Web 上での CC 攻撃の発信国の分布

```
matched_host: www.aliyun.com and cc_blocks: 1
| SELECT ip_to_country(if(real_client_ip ='-', remote_addr, real_client_ip)) as country,
         count(1) as "number of attacks"
```

```
    group   by   country
```

注：

real_clien t_ip フィールドまたは real_clien t_ip フィールド（ real_clien t_ip が - の場合）を選択するには、関数 if (condition , option1 , option2) を使用します。取得した IP を関数 ip_to_coun try に渡して、この IP に対応する国情報を取得します。

- アクセス分布

  より詳細な省ベースの分布情報を取得するには、 ip_to_prov ince 関数を使用します。例えば：

```
 matched_ho  st :  www . aliyun . com   and   cc_blocks :  1
|  SELECT   ip_to_prov  ince ( if ( real_clien  t_ip ='-',
 remote_add  r ,  real_clien  t_ip ))  as   province ,
          count  ( 1 )  as  " number   of   attacks "
    group   by   province
```

注：

もう一つの IP 関数 ip_to_prov ince が IP の所属する省の情報を取得できます。IP アドレスが中国国外の場合でも、システムは州へ変換します。

- 攻撃者の熱分布

  攻撃者のヒートマップを取得するには、 ip_to_geo 関数を使用します。例えば：

```
 matched_ho  st :  www . aliyun . com   and   cc_blocks :  1
|  SELECT   ip_to_geo ( if ( real_clien  t_ip ='-',  remote_add  r
 ,  real_clien  t_ip ))  as   geo ,
          count  ( 1 )  as  " number   of   attacks "
    group   by   geo
    limit   10000
```

注：

IP の緯度と経度を取得して最初の 10,000 を取得するには、もう一つの IP 関数 ip_to_geo を使用します。

IP演算子 ip_to_prov ider の取得、IP がインターネットかイントラネットの ip_to_doma in かの判断など、その他の IP ベースの構文解析機能については、IP 機能をご参照ください。

4.4.4 ログレポート

ログレポートページは、ログサービスのダッシュボードに埋め込まれています。 このページには、デフォルトのダッシュボードが表示されます。 時間範囲を変更してフィルタを追加することにより、さまざまなフィルタ条件でダッシュボードデータを表示できます。

レポートの表示

1. Anti-DDoS Pro コンソールにログインし、左側のナビゲーションペインで Log > Full Log を選択します。 Full Log ページに入ります。
2. DDoS ログ収集機能を有効にするWebサイトを選択し、Status がオンになっていることを確認します。
3. ログレポートをクリックします。

   Log Service ダッシュボードページが現在のページに埋め込まれ、フィルタ条件が自動的に追加されます。 たとえば、選択された Web サイトに基づいてログレポートを表示するには、`matched_ho st : www . aliyun . com` を使用します。

図 4-12 : レポートの表示

Web サイトで DDoS ログ収集機能を有効にした後、 Log Service はオペレーションセンターとアクセスセンターの 2 つの既定のレポート作成ツールを自動的に作成します。 デフォルトダッシュボードの詳細については、デフォルトダッシュボードを参照してください。

ダッシュボード	ダッシュボード名	説明
ddos-pro-logstore_ddos_operation_center	DDoS 運用センター	有効なリクエストステータス、トラフィック、傾向、攻撃元の分布および CC によって攻撃されたトラフィック量とピークを含む、DDoS で保護された Web サイトの現在の全体的な動作ステータスを表示します。

ダッシュボード	ダッシュボード名	説明
ddos-pro-logstore_ddos_access_center	DDoS アクセスセンター	PV / UV トレンド、帯域幅のピーク、訪問者、トラフィック、クライアントの種類、 リクエスト 、訪問した Web サイトの分布など、DDoS で保護された Web サイトの現在の全体的な運用ステータスを表示します。

図 4-13 : デフォルトのダッシュボード

レポートの表示に加えて、次の操作を実行できます。

- 時間範囲を選択
- フィルタ条件の追加または編集
- グラフを見る

タイムピッカー

ダッシュボードページのすべてのグラフは、異なる期間の統計結果に基づいています。 たとえば、訪問のデフォルトの時間範囲は 1 日で、アクセス傾向は 30 日です。 現行のページのすべてのグラフを同じ時間範囲に表示するように設定するには、時間ピッカーを設定します。

1. 選択をクリックします。
2. ダイアログボックスで設定を行います。 相対時間、全体のポイント時間、またはカスタム時間を選択することができます。

> 注 :
>
> - 時間範囲が変更されると、すべてのチャートの時間がこの時間範囲に変更されます。
> - 時間ピッカーは、現在のページのチャートの一時的な表示のみを提供し、システムは設定を保存しません。 次にレポートを表示すると、システムはデフォルトの時間範囲を表示します。

図 4-14 : 時間範囲を設定する

フィルター条件

Web サイトを選択し、ログレポートをクリックしてダッシュボードページに入ります。 選択された Web サイトに基づいてログレポートを表示するには、`matched_ho st : www . aliyun . com` などのフィルタ条件が自動的に追加されます。

フィルタ条件を設定することによって、レポートのデータ表示範囲を変更できます。

- すべてのウェブサイトの全体的なレポートを表示する

  フィルタ条件をクリアして、全体レポートライブラリ `ddos - pro - logstore` を表示します。
- より多くのフィルタ条件を追加する

  キーと値を設定することにより、レポートデータをフィルタリングできます。 複数のフィルター間の `AND` 関係がサポートされています。

  たとえば、電気通信回線によるアクセス要求の全体的な状況を表示します。

  図 4-15 : フィルタ条件を追加する

> 注 :
> `isp_line` は DDoS ログのフィールドで、オペレータネットワークがポートに接続していることを示します。 フィールドの詳細については、DDoS ログフィールドを参照してください。

グラフの種類

レポート表示領域には、以下のタイプを含む定義済みのレイアウトに従って複数のレポートが表示されます。 グラフの種類の詳細については、グラフについてを参照してください。

グラフの種類	説明
数字	効果的なリクエスト率や攻撃のピークなどの重要なインジケータを表示します。
ライン/エリアマップ	着信帯域幅の傾向や攻撃の遮断率など、特定の期間内の特定の重要なインジケータの傾向グラフを表示します。
地図	CC 攻撃国、アクセスホットスポットなどの訪問者と攻撃者の地理的分布を表示します。

グラフの種類	説明
円グラフ	攻撃を受けた Web サイトのトップ 10、クライアントタイプの分布など、情報の分布を表示します。
表	通常複数の列に分割された攻撃者のリストなどの情報を表示します。
地図	データの地理的分布を表示します。

デフォルトのダッシュボード

- オペレーションセンター

  オペレーションセンターは、有効なリクエストステータス、トラフィック、トレンド、攻撃者の分布、および CC によって攻撃されたトラフィック量とピークを含む、DDoS で保護された Web サイトの現在の全体的な運用ステータスを表示します。

グラフ	タイプ	デフォルトの時間範囲	説明	例
有効リクエストパッケージレート	単一値	1 時間（相対）	有効なリクエスト、つまり、すべてのリクエストの総数に対する非 CC 攻撃または 400 エラーリクエストの数。	95%
有効リクエストの流量	単一値	1 時間（相対）	すべてのリクエストの合計流量に対する有効リクエストの割合。	95%
受信トラフィック	単一値	1 時間（相対）	受信した有効リクエストの合計。 単位：MB。	300 MB
攻撃トラフィック	単一値	1 時間（相対）	CC 攻撃の着信トラフィックの合計。 単位：MB。	30 MB

グラフ	タイプ	デフォルトの時間範囲	説明	例
発信トラフィック	単一値	1 時間（相対）	送信した有効トラフィックの合計。 単位：MB。	300 MB
ネットワーク in の帯域幅ピーク。	単一値	1 時間（相対）	Web サイトによってリクエスト済みの着信トラフィックレートの最高ピーク。 単位：bytes/s。	100 Bytes/s
ネットワーク out の帯域幅ピーク。	単一値	1 時間（相対）	Web サイトによってリクエスト済みの送信トラフィックレートの最高ピーク。 単位：bytes/s。	100 Bytes/s
受信データパケット	単一値	1 時間（相対）	有効なリクエスト（非 CC 攻撃）に対する着信リクエストの数。単位：個。	30, 000
攻撃データパケット	単一値	1 時間（相対）	CC 攻撃のリクエスト数の合計。 単位：個。	100
攻撃ピーク	単一値	1 時間（相対）	CC 攻撃のピーク。 単位は分あたりの数です。	100/ 分
受信帯域幅と攻撃のトレンド	二線図	1 時間（絶対）	1 分あたりの有効リクエスト数と攻撃リクエストのトラフィック帯域幅のトレンドグラフ。 単位：KB/s。	-

グラフ	タイプ	デフォルトの時間範囲	説明	例
リクエストと傍受のトレンド	二線図	1 時間（絶対）	1 分あたりのリクエストおよび傍受された CC 攻撃リクエストの傾向図。 単位は分あたりの数です。	-
有効リクエスト率のトレンド	二線図	1 時間（絶対）	1 分あたりの有効リクエスト数（非 CC 攻撃または 400 エラーリクエスト）の、全リクエストの総数に対する傾向図。	-
アクセス状況の分布トレンド	流れ図	1 時間（絶対）	1 分あたりのさまざまなリクエスト処理状況（400、304、20）の傾向図。 単位は分あたりの数です。	-
CC 攻撃分布	世界地図	1 時間（相対）	発信国における CC 攻撃数の合計。	-
CC 攻撃分布	中国地図	1 時間（相対）	発信元の省（中国）における CC 攻撃数の合計。	-
攻撃の一覧表	表	1 時間（相対）	IP、都市、ネットワーク、攻撃数、および総トラフィックを含む、最初の 100 の攻撃の攻撃者情報。	-

グラフ	タイプ	デフォルトの時間範囲	説明	例
攻撃アクセス回線の分布	円グラフ	1 時間（相対）	telecommunications、Unicom、BGP などの DDoS 回線にアクセスした CC 攻撃の分布。	-
攻撃されたウェブサイトトップ 10	ドーナツグラフ	1 時間（相対）	攻撃されたウェブサイトトップ 10	-

- アクセスセンター

  アクセスセンターは、PV / UV のトレンドと帯域幅のピーク、訪問者、トラフィック、クライアントの種類、リクエスト、訪問した Web サイトの分布など、DDoS で保護された Web サイトの現在の全体的な運用状況を表示します。

グラフ	タイプ	デフォルト時間範囲	説明	例
PV	単一値	1 時間（相対）	リクエストの合計数。	100,000
UV	単一値	1 時間（相対）	個別のアクセスクライアントの総数	100,000
受信トラフィック	単一値	1 時間（相対）	Web サイトの受信トラフィックの合計。単位：MB。	300 MB
ネットワーク in の帯域幅ピーク。	単一値	1 時間（相対）	Web サイトによってリクエスト済みの受信トラフィックレートの最高ピーク。 単位：bytes/s。	100 Bytes/s

グラフ	タイプ	デフォルト時間範囲	説明	例
ネットワーク out の帯域幅ピーク。	単一値	1 時間（相対）	Web サイトによってリクエスト済みの送信トラフィックレートの最高ピーク。 単位：bytes/s。	100 Bytes/s
トラフィック帯域幅トレンド	二線図	1 時間（絶対）	1 分あたりの Web サイトの受信トラフィックと送信トラフィックの傾向図。 単位：KB/s。	-
リクエストと傍受のトレンド	二線図	1 時間（絶対）	1 分あたりのリクエストおよび傍受された CC 攻撃リクエストの傾向図。 単位は分あたりの数です。	-
PV/UV アクセストレンド	二線図	1 時間（絶対）	1 分あたりの PV と UV の傾向図。 単位：個。	-
訪問者分布	世界地図	1 時間（相対）	送信元国における訪問者の分布（PV）。	-
訪問者ヒートマップ	Amap	1 時間（相対）	訪問者の地理的アクセスヒートマップ。	-
受信トラフィック分布	世界地図	1 時間（相対）	送信元国での受信トラフィック分布の合計。 単位：MB。	-

グラフ	タイプ	デフォルト時間範囲	説明	例
受信トラフィック分布	中国地図	1 時間（相対）	発信元の省（中国）における受信トラフィックの合計。 単位：MB。	-
アクセス回線分布	ドーナツグラフ	1 時間（相対）	telecommunications、Unicom、BGP などの DDoS 回線にアクセスした訪問者の分布。	-
受信トラフィックネットワーク事業者の分布。	ドーナツグラフ	1 時間（相対）	訪問者がネットワーク事業者によってアクセスする受信トラフィックの分布。 例：telecommunications 、 Unicom 、 mobile connections 、 education network。 単位：MB。	-
訪問回数の最も多いクライアント	表	1 時間（相対）	IP、都市、ネットワーク、リクエスト方式の分布、着信トラフィック、不正アクセス数、傍受された CC 攻撃数など、最も訪問回数が多いクライアントのトップ 100。	-
アクセスドメイン名	ドーナツグラフ	1 時間（相対）	最も訪問回数の多いドメイン名のトップ 20。	-

グラフ	タイプ	デフォルト時間範囲	説明	例
Referer	表	1 時間（相対）	最もリダイレクトされた参照元 URL、ホスト、および頻度のトップ 100。	-
クライアントタイプの分布	ドーナツグラフ	1 時間（相対）	iPhone、iPad、Windows IE、Chrome など、最も訪問回数の多いユーザーエージェントのトップ 20。	-
リクエストコンテンツタイプの分布	ドーナツグラフ	1 時間（相対）	HTML、フォーム、JSON、ストリーミングデータなど、最もリクエストの多いコンテンツタイプのトップ 20。	-

4.4.5 課金方法

DDoS ログ収集機能は、Log Service の課金項目に従って課金されます。 ログデータが生成されない場合、請求は行われません。 Log Service はリソース使用量によって請求され、DDoS ログストアの FreeTier クォータを提供します。

DDoS ログ収集機能は、ログ収集、保管、リアルタイムのクエリと分析、およびダッシュボードなどの機能を提供します。 ログデータのリアルタイムのクエリと分析は、Log Service に依存しているため、 この機能は Log Service の請求方法に従って課金されます。 Log Service はリソース使用量によって請求され、DDoS ログストアの FreeTier クォータを提供します。 具体的な料金は、ログデータの量によって異なります。 Log Service を有効にしたとしても、Web サイトのログ機能を有効にしていない場合は、料金は表示されません。

払い戻し及び未払い

Log Service はリソース使用量によって請求され、請求サイクルは 1 日です。 控除と未払いの詳細は、Deduction and outstanding payment を参照してください。

請求項目

請求項目	説明
読み書きトラフィック	・ 読み取りおよび書き込みトラフィックは、圧縮ログを送信するためのトラフィックによって計算されます。 DDoS ログは通常 5~10 倍圧縮されています。 ・ 読み取りおよび書き込みトラフィックは消費損失インターフェイスを含め、一般に API / SDK およびコンシューマグループ SDK を使用して読み取りトラフィックを生成します。 圧縮転送トラフィックにより計算し、ログは API / SDK モードで圧縮できます。 注 : Log Service コンソールのログ消費の下にあるプレビュー機能でも、マイクロフローのトラフィックを消費します。 ・ インデックスベースのクエリと分析によって生成されたデータには、読み書きのトラフィック料金はかかりません。 たとえば、コンソールのログクエリ分析、ダッシュボード、およびアラームは課金されません。
記憶領域	記憶領域は、圧縮後のデータサイズとインデックス付きデータサイズの合計です。
インデックストラフィック	・ インデックストラフィックは、実際のインデックスフィールドによって計算されます。 ストレージ料金は書き込み中に全額請求されます。 DDoS ログは、デフォルトでフルインデックスを有効にします。 ・ FullText と KeyValue の両方のインデックスを持つフィールドのトラフィックは、一度だけ計算されます。 ・ インデックスが記憶領域を占有するため、記憶領域料金が請求されます。
アクティブシャードレンタル	現行読み書き状況にあるシャードのみが統計されます。 マージ/分割されているシャードにはレンタル料金は請求されません。 注 : デフォルトでは、Log Service は 2 つのシャードを作成し、シャード自動分割機能を有効にします。 通常、各シャードは 1 日に 430 GB の書き込みデータ量を処理できます。
読み書き回数	Log Service に書き込まれるログの書き込み数は、ログの生成速度によって異なります。 バックグラウンド実装メカニズムは、読み書き回数をできる限り最小にします。
インターネット読取りトラフィック	インターネットプログラムが Log Service によって収集されたログデータを読み取るときに生成されるデータトラフィック。

FreeTier クォータ

次の場合では、Log Service は請求されません：

- Log Service が有効になっていますが、 Web サイトで DDoS ログ機能が有効になっていません。
- DDoS ログ機能を有効にする Web サイトログの量は、無料クォータの範囲内にあります。
- インデックスベースのクエリ分析、レポート、およびアラームは請求されません。

Log Service は、DDoS ログストアのための無料クォータを提供します。 データ量が無料クォータの制限より少ない場合、料金は表示されません。

請求項目	FreeTier クォータ
書き読みトラフィック	30 GB/日
記憶領域	3 日
インデックストラフィック	100 GB/日
アクティブシャードレンタル	4 日間/月
読み書き回数	100 万回/日
インターネット読取りトラフィック	0
消費を読み取るトラフィック	0
消費を読み取る回数	0

 注：
ログデータの保存期間はデフォルトで 3 日に設定されており、3 日以上に変更すると追加料金が発生する可能性があります。

請求方法

ログ分析機能を有効にしている Web サイトのログ容量が無料クォータを超えた場合、Log Service は超過したクォータを請求します。

請求項目	追加料金
読み書きトラフィック（USB/GB）	0.045
記憶領域（USD/GB/日）	0.002875
インデックストラフィック（USD/GB）	0.0875
アクティブシャードレンタル（USD/日）	0.01
読み回回数（USD/百万回）	0.03

請求項目	追加料金
インターネット読取りトラフィック（USD/GB）	0.2

例

- FreeTier クォータ：ログの平均容量は約1600バイトで、1 日に約 6000 万のログが生成され、保存期間は 3 日です。 合計ログ容量は 1 日あたり約 96 GB で、クォータを超えることはありません。
- インデックス：ログ容量は1日あたり 150 GB で、50 GB が請求されます（150 GB - 100 GB）。つまり 1 日あたり 0.0875 x 50 = 17.5 USD です。
- 書き込み送信：ログ容量は 1 日あたり 300 GB で、ログは 6 回圧縮されます。 実際の圧縮サイズは約 50 GB で、20 GB（50 GB - 30 GB）が請求されます。つまり一日あたり 0.045 x 20 = 0.9 USD です 。
- 記憶領域サイズ：
  - 1 日あたりのデータ量が 10 GB で、圧縮後で 2 GB になります、それ以外に 10 GBのインデックストラフィックがあります。 保存期間は 30 日で、30 日後の最大ストレージ容量は 30×（10 + 2）= 360 GB、 3 日間の無料クォータを加算すると、27×（10 + 2）= 324 GB 。よって1日のストレージの最大料金は 0.002875 × 324 = 0.9315 USD。
  - 1日あたりのデータ量が 1 GB で、圧縮後で 200 MB になります、それ以外に FullText およびインデックスフィールド（サイズ1 GB）があります。 30日後の累積最大ストレージ容量は 30×（1000 + 200）≒ 36 GB で、3日間の無料クォータを加算すると、27×（1000 + 200）≒32.4 GB。 よって1日のストレージの最大料金は 0.002875 × 32.4 = 0.09315 USD。
- アクティブシャードレンタル：現在、10 個のシャードが利用できます、そのうち 7 個が読み書き用で、3 個が読み取り専用です。 DDoS ログストアは 1 日あたりの請求のみサポートされます。 3（7 - 4）シャードのレンタル料金は 1 日あたり 0.03 USD です。
- 読み書き回数：Web サイトのログの数は1日あたり 100 億であり、書き込み回数は約 500,000（平均1時間あたり 2,000）で、無料です
- インターネットトラフィック：2 GB の Log Service データが Alibaba Cloud 以外のプロダクトに配信され、外部ネットワークの読み取りトラフィックで 0.4 USD 発生します。

請求に関するよくある質問

- Web サイトのログの保存期間を変更するにはどうすればよいですか？

  1. Log Service コンソールにログインし、対象プロジェクトをクリックして Logstore リストに入ります。 DDoS ログ用のデフォルトのプロジェクトは、`ddos-pro-project-Alibaba #########ID` です。
  2. 操作列の変更をクリックします。
  3. データ保存時間ページで、変更をクリックします。

- 現在のログ量を確認してコストを見積もるにはどうすればよいですか？

  - 日単位でコスト測定データを表示するには、Alibaba Cloud Expense Management Center にアクセスします。
  - 1. DDoS IP 保護コンソールにログインし、左側のフルログをクリックします。
    2. ログ量を表示したい Web サイトを選択して、右側のログ分析をクリックします。
    3. クエリボックスに次のクエリステートメントを入力します。時間範囲は昨日（全ポイントタイム）です。

    ```
    __topic__ : ddos_acces s_log | select count ( 1 ) as PV
    ```

    4. クエリをクリックし、グラフタイプがテーブルの統計グラフを選択します。

    前日のデータ量を取得できます、現在のログ保存時間に従ってコストを見積もることができます。

  - 大量のログが生成された際に、Log Service を構成してアラームをトリガーする方法は？

    大量の DDoS ログが収集されると、Log Service の無料クォータを超え、一定の料金が発生することがあります。 このようなリスクがあるときにアラーム通知を受け取りたい場合

は、大量のログが生成された際にアラームをトリガーするように Log Service を構成できます。

1. DDoS IP 保護コンソールにログインし、左側のフルログをクリックします。
2. ログ量を表示したい Web サイトを選択して、右側のログ分析をクリックします。
3. クエリボックスに次のクエリステートメントを入力して、検索をクリックします：

```
*|  select   count ( 1 )  as   PV
```

4. クエリページの右上隅にあるクイック検索として保存をクリックして、`ddos - metering - pv` など、クエリに関する情報を入力します。 OK をクリックします。
5. アラームとして保存をクリックしてアラーム構成を作成します。次の図を参照してください。 5 分ごとに過去 1 時間のログ容量を確認し、560 万を超えるログが生成された場合はアラームをトリガーします。

注：
1 日のログ容量を 100 GB の無料クォータ以内にした場合、1 時間あたりの平均インポート容量は 100 GB÷1600 バイト÷ 24 時間 ≒ 280 万と推定されます。 この例でのアラームがトリガーされる条件は、1時間当たりの量の 2 倍で、つまり 560 万となりますが、実際の状況とニーズに応じて調整できます。

4.5 WAF ログ

4.6 Anti-Bot ログ

4.7 ActionTrail のアクセスログ

4.7.1 概要

Alibaba Cloud の ActionTrail を Log Service と連動させることにより、ActionTrail のログを Log Service でリアルタイムに収集/分析することができます。ActionTrail に収集された操作ログは、リアルタイムに Log Service に送信されます。Log Service には、さまざまな機能が用意されており、ログをリアルタイムにクエリ、分析、ダッシュボード表示することができます。

効率とサービス品質の向上に、情報技術とクラウドコンピューティング技術を採用する企業は増えていますが、企業や組織のネットワーク、デバイス、データへの攻撃は止まることがありませ

ん。一般的に、攻撃者の目的は、損害を与えることにはなく、利益を得ることにあり、身を隠すことに長けています。 その結果、攻撃の発見と特定がますます困難になっています。

また、監査およびセキュリティにおける原因特定には、企業の IT およびデータリソースの操作ログは非常に重要な位置を占めています。 ネットワーク情報技術の発展に伴い、中国においては「サイバーセキュリティー法」の徹底的な実施により、各企業や組織は操作ログの保管と分析の徹底が図られています。クラウドコンピューティングでの、リソースに対して行った操作のログは非常に重要です。

ActionTrail には、クラウドアカウントのリソースに対して行った操作が記録され、そのログを照会することができます。なお、ログファイルは指定の OSS (Object Storage Service) または Log Service に保存されます。 ActionTrail に保存された操作ログをもとに、セキュリティ分析、リソース変更トラッキング、およびコンプライアンス監査を実施できます。

ActionTrail には、クラウドサービスの API 呼び出しログが収集されます (コンソール操作によってトリガーされた API 呼び出しログを含む)。正規化処理後、操作ログは JSON 形式に保存され、送信できる形になります。通常、コンソール操作または SDK 呼び出しを行うと、ActionTrail はその操作ログを 10 分以内に収集します。

ActionTrail は Log Service と連動しているため、Log Service でリアルタイムにログを収集/分析することができます。 ActionTrail の収集する操作ログは、リアルタイムに Log Service に送信されます。 Log Service には、さまざまな機能が用意されており、ログをリアルタイムにクエリ、分析、およびダッシュボード表示することができます。

利点

- 設定が容易: リアルタイムログの収集を難なく設定できます。設定手順とログフィールドの詳細については、手順をご参照ください。
- リアルタイム分析: Log Service を使用することにより、ログをリアルタイムに分析し、通知センターをすぐに使用し始めることができます。また、重要なクラウドリソースに対する操作ログの詳細データをリアルタイムに探し出すことができます。
- リアルタイムアラーム: 擬似リアルタイムモニタリングおよびアラームの指標を設定することができるため、重大なエラーに迅速に対応できるようになります。
- エコシステム: ストリーム処理、クラウドストレージ、可視化ソリューションといった他のエコシステムと連携させることで、データ価値を高めることができます。
- 無料枠: 毎月 500 MB のデータのインポートおよびストレージを無料で利用できます。なお、コンプライアンス、追跡、およびファイリングするために保管期間を延長することもできます。永久保管のサービスを月額 0.0875 GB/USD の低価格でご利用いただけます。 課金の詳細については、「課金方法」をご参照ください。

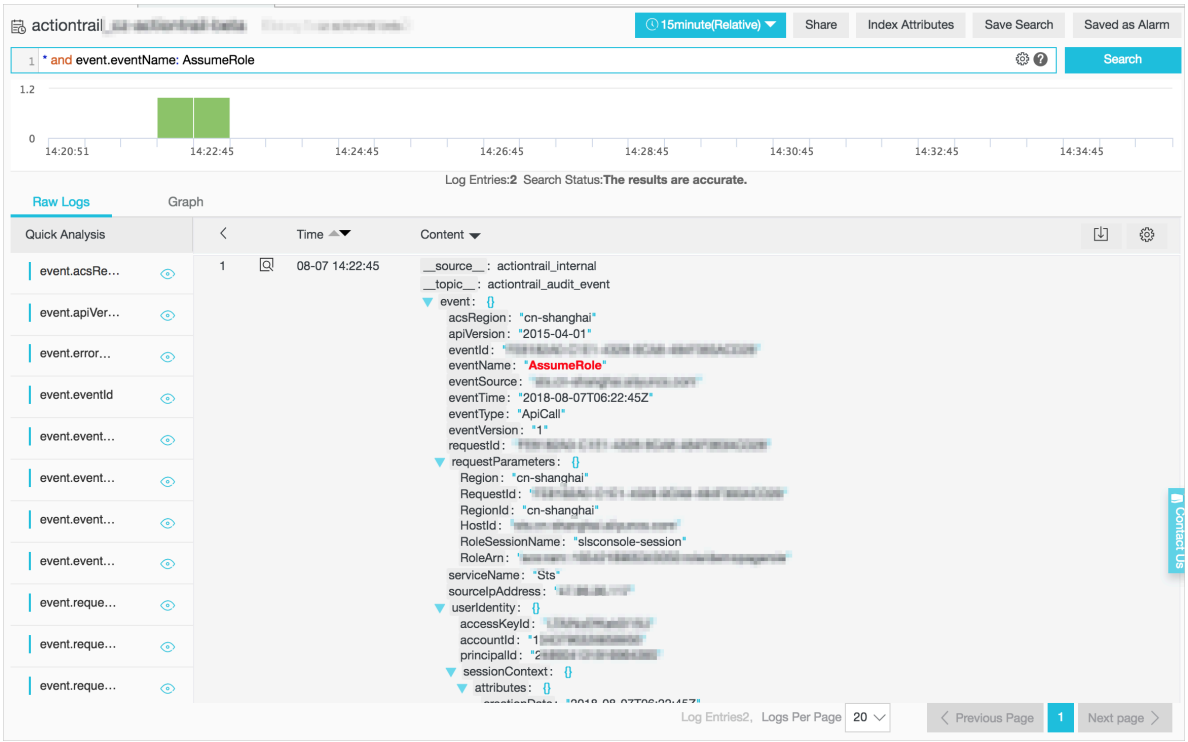
アプリケーションシナリオ

- 異常な操作のトラブルシューティングと分析

  アカウント下の全 Alibaba Cloud リソースに対する操作をモニタリングし、異常な操作を迅速にトラブルシューティングおよび分析できます。不注意による削除、リスクを伴う操作といった操作のログを残すことで、追跡できるようになります。

  例: Elastic Compute Service (ECS) リリース操作のログを表示

図 4-16 : ECS リリース操作のログを表示

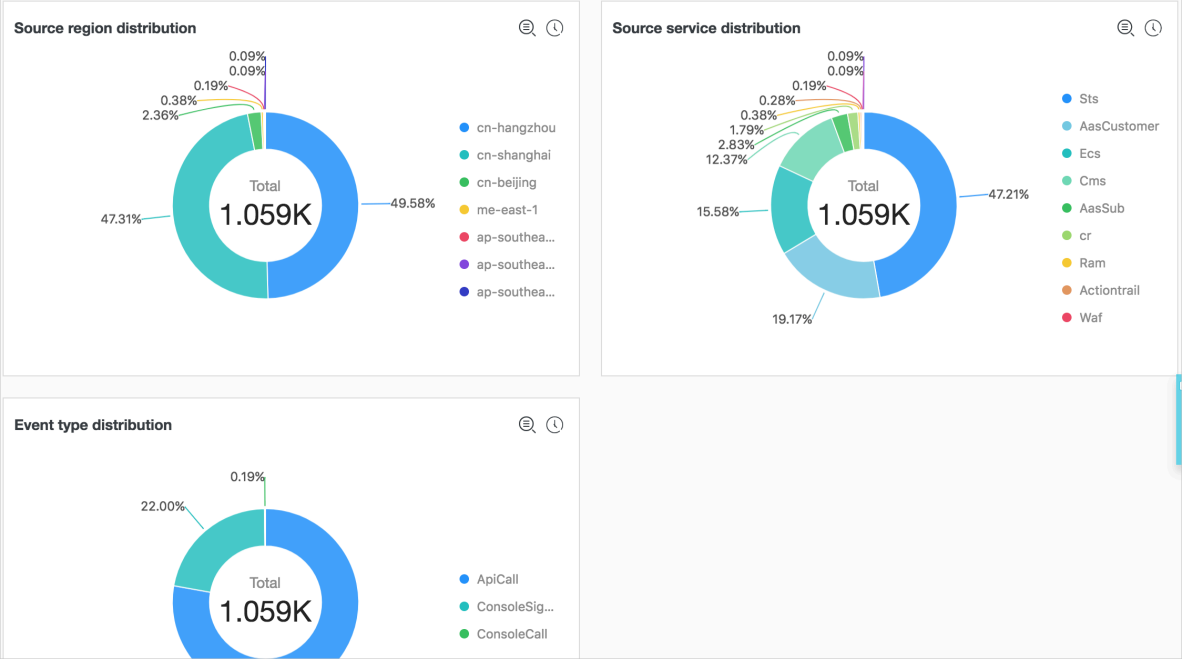


- 重要度の高いリソース操作の分布および発生源の追跡

  ログコンテンツを分析し、重要度の高いリソース操作の分布と発生源を追跡することができます。また、分析結果から解決方法を特定し、最適化することができます。

  例: リレーショナルデータベースサービス (RDS) を削除した事業者の国分布を表示

  図 4-17 : RDS 削除の分布を表示



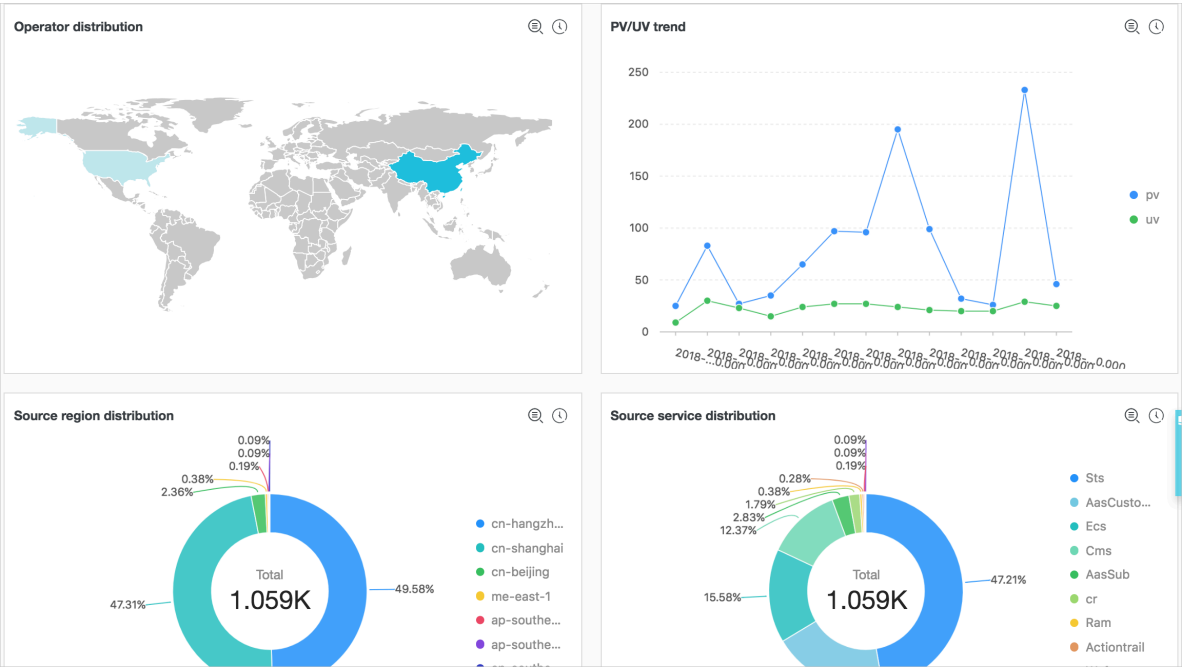
- リソース操作の分布を表示

  収集した ActionTrail 操作ログを SQL クエリステートメントでリアルタイムに照会/分析し、リソースに対するすべての操作、運用保守操作の分布と傾向を時系列に表示できます。管理者

はリソースの稼働状況をリアルタイムにモニタリングできるようになります。運用保守の信頼性がひと目でわかるようになります。

例: 失敗操作の分布を表示

図 4-18 : 失敗操作の分布



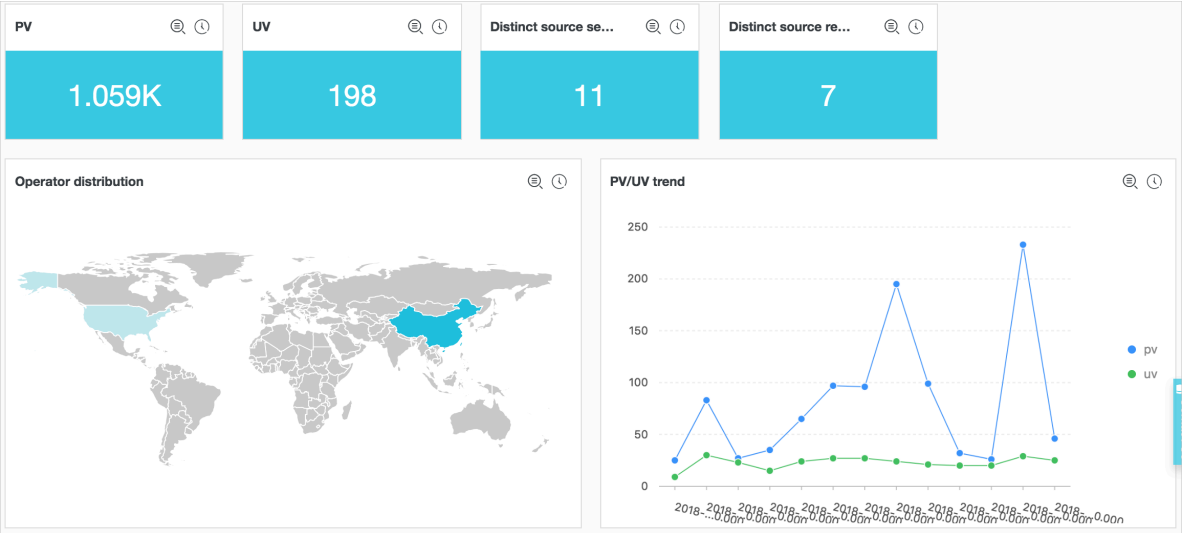
- 操作ログのリアルタイム分析

  操作ごとにクエリステートメントをさまざまにカスタマイズ、データごとにクイック検索/分析ダッシュボードを作成することができます。また、リソース使用状況やユーザーのログイン

ステータスといったデータをリアルタイムに表示するダッシュボードを作成することもできます。

例: アクセスのインターネットキャリア分布を表示

図 4-19 : アクセスのインターネットキャリア分布



4.7.2 手順

ActionTrail は Log Service と連動させることができます。 ActionTrail で収集された操作ログデータは、リアルタイムに Log Service に送信されます。 本ドキュメントは ActionTrail ログのログフィールドと収集手順を紹介します。

前提要件

1. Log Service を有効化していること。
2. ActionTrail を有効化していること。

手順

1. ActionTrail コンソールにログインします。
2. 左側のナビゲーションペインでトレイルリストをクリックし、トレイルリストページに移動します。
3. 右上隅のトレイルの作成をクリックしてトレイルの作成ページに移動します。

4. 各パラメータを設定します。

   a. トレイル名を入力します。

   b. 監査イベントを OSS バケットに送信します (オプション)。

      詳細は、トレイルの作成をご参照ください。

   c. Log Service のリージョンを選択します。

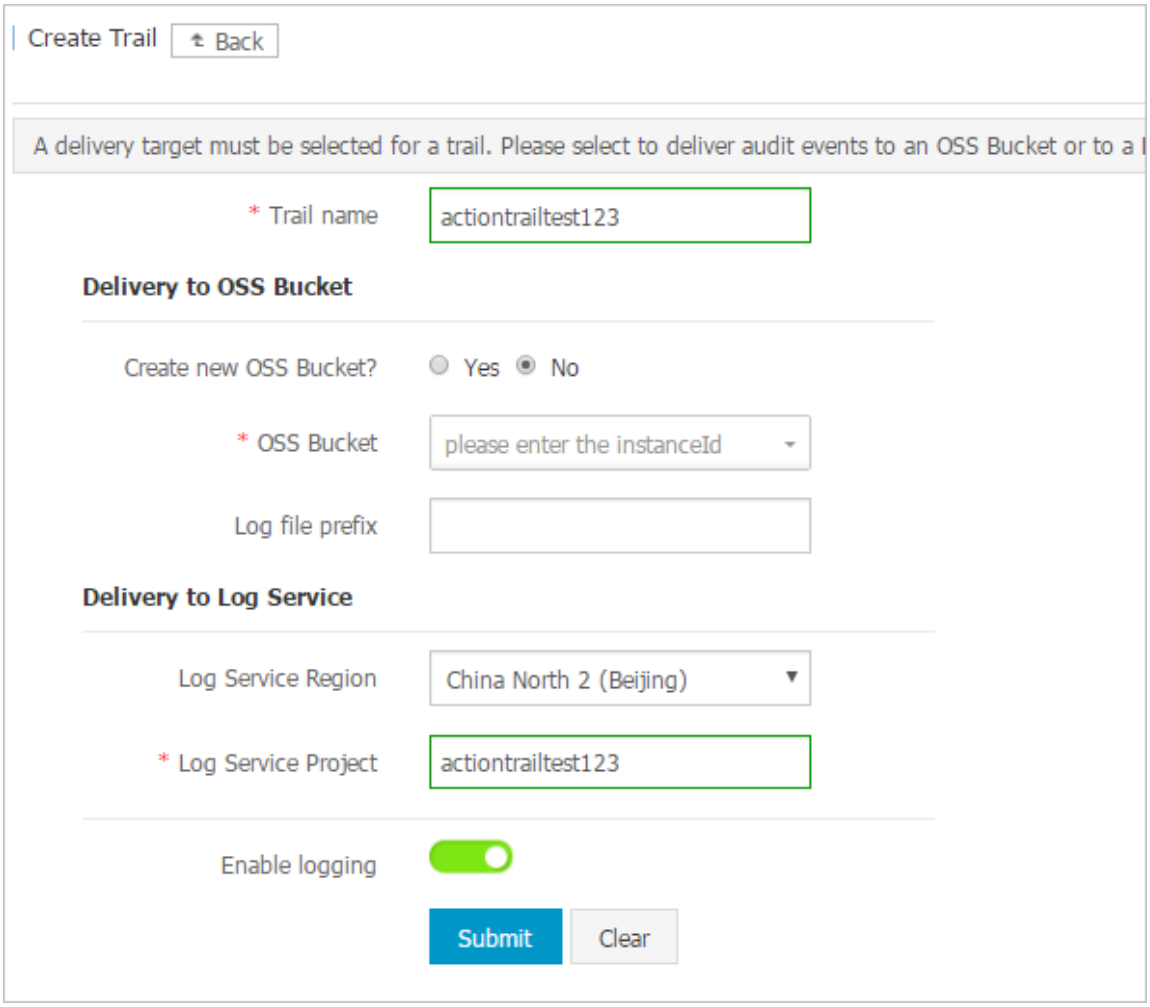
   d. Log Service プロジェクトを入力します。

      プロジェクトは ActionTrail ログの保存にに使用されます。 選択したリージョンに既存のプロジェクト名、または新しいプロジェクト名を入力してログを新しいプロジェクトに送信します。

   e. ログを有効にする。

      ログを有効にするをクリックします。 本機能を有効にすると、ActionTrail で記録したクラウドリソースの操作ログが Log Service に送信されます。

      図 4-20 : トレイルパラメータを設定します。

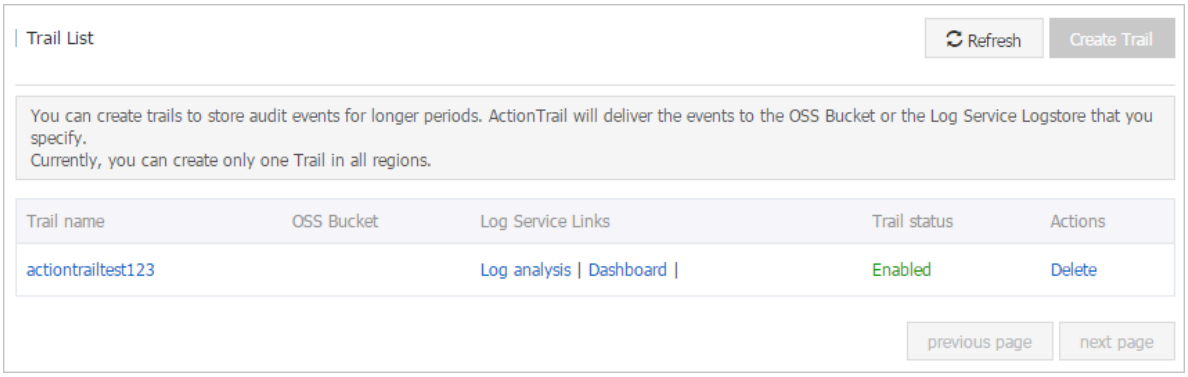


5. 送信をクリックして設定を完了します。

以上でトレイルの作成は完了です。作成したトレイルをトレイルリストに表示できます。

> 注：
> 初めて ActionTrail ログ収集を設定する場合は、ページの指示に従って ActionTrail に権限付与します。 権限付与により、ActionTrail は ActionTrail ログを Logstore に送信できるようになります。 権限付与を完了したら、もう一度送信をクリックして設定を終了します。

図 4-21 : トレイルリスト



制限

- 1 つのアカウントに作成できるトレイルは 1 つのみです。

  トレイルを使用すると、指定した OSS バケットまたは Log Service Logstore に監査イベントを送信できます。 現在、すべてのリージョンのアカウントに作成できるトレイルは 1 つのみです。 このトレイルは、OSS バケットと Logstore の両方またはいずれかに、すべてのリージョンにわたって監査イベントを送信します。
- トレイルを作成した場合は、トレイルが作成されたリージョンでのみトレイルを処理できます。

  トレイルを作成した場合は、そのトレイルが作成されたリージョンでのみトレイルを表示、変更、または削除できます。 例：OSS のトレイルを作成したときに Log Service のトレイルを構成する必要がある場合は、作成した OSS のトレイルに Log Service の構成を追加します。
- 専用 Logstore は追加データの書き込みをサポートしません。

  専用 Logstore は、Action Trail の操作ログだけを保存するために使用されます。そのため、専用 Logstore はその他のデータの書き込みをサポートしません。 クエリ、統計、アラーム、ストリーミングの消費など、その他の機能には制限がありません。

- 従量課金。

  ActionTrail のログ収集機能は、Log Service の請求方法を使用します。 Log Service には、従量課金をサポートし、一定量の無料クォータも提供しています。 詳細は、Billing methodをご参照ください。

クエリと分析

トレイルの構成が完了後、収集されたログデータをクエリおよび分析するには、トレイルリストページの Log Service リストでログ分析およびログレポートをクリックします。

- ログ分析：ログクエリおよび分析ページに入ります。

  Log Service はログのクエリと分析を提供します。 このページでは、収集した ActionTrail ログをリアルタイムでクエリおよび分析できます。

  クエリ構文と分析構文を定義することで、Log Service はさまざまな複雑なシナリオでログクエリを提供します。 クエリおよび分析の構文については、クエリ構文および分析構文を参照してください。

  重要なログデータを定期的にモニタリングし、異常な状態のアラーム通知を設定するには、現在のクエリ条件をクイック検索とアラームとして検索ページに保存します。詳細手順は、アラームの設定をご参照ください。
- ログレポート：ダッシュボードページに入ります。

  Log Service は、ActionTrail 専用の組み込みダッシュボードによって、イベントタイプやイベントソースなどのリアルタイムのダイナミクスの全体像を示します。

  専用ダッシュボードを変更したり、カスタムダッシュボードを作成したり、さまざまなシナリオのカスタム分析グラフをダッシュボードに追加したりできます。 ダッシュボードの詳細は、ダッシュボードをご参照ください。

デフォルト構成

構成が完了すると、Log Service は専用のプロジェクトと専用の Logstore を作成します。 ActionTrail で収集したクラウドリソースの操作ログは、リアルタイムで Logstore に送信されます。 さらに、Log Service は、クラウドリソースの運用状況をリアルタイムで表示するためのダッシュボードも作成します。 プロジェクトや Logstore などのデフォルト構成については、次の表をご参照ください。

表 4-2 : デフォルト構成

デフォルト構成項目	構成内容
Project	トレイルを作成するときに選択、またはカスタマイズするプロジェクト。
Logstore	Logstore はデフォルトで作成されます。 Logstore 名は `actiontrail_#####`となります。 ActionTrail のすべてのログはこの Logstore に保存されます。
リージョン	トレイルを作成するときに選択したリージョン。
シャード	デフォルトでは、2 つのシャードが作成され、シャード自動分割機能が有効になります。
ログの保存期間	デフォルトでは、ログは永続的に保存されます。 ログ保存期間は 1~3000 日の範囲の値にカスタマイズできます。詳細手順は、Logstore の管理をご参照ください。
ダッシュボード	ダッシュボードはデフォルトで作成されます： ・ 中国語環境：`actiontrail_#####_audit_center_cn` ・ 英語環境：`actiontrail_#####_audit_center_en`

ログフィールド

フィールド名	名前	例
__topic__	ログトピック。	このフィールドは `actiontrai l_audit_ev ent` に限定されています
event	イベントの本体。JSON形式です。 イベント本体の内容はイベントによって異なります。	イベントの例
event.eventId	イベントのID、イベントを一意に示します。	07F1234-3E1D-4BFF-AC6C-12345678
event.eventName	イベント名。	CreateVSwitch
event.eventSource	イベントの発生源。	http://account.aliyun.com:443/login/login_aliyun.htm
event.eventType	イベントの種類。	ApiCallApicall

フィールド名	名前	例
event.eventVersionEvent.eventversion	ActionTrail のデータ形式のバージョン。現在は 1 に限定されています。	1
event.acsRegion	イベントの存在するリージョン。	cn-hangzhou
event.requestId	クラウドサービス操作のリクエストID。	07F1234-3E1D-4BFF-AC6C-12345678
event.apiVersion	関連APIのバージョン。	2017-12-04
event.errorMessage	イベント失敗のエラーメッセージ。	unknown confidential
event.serviceName	イベント関連のサービス名。	Ecs
event.sourceIpAddress	イベントに関連付けられている送信元IP。	1.2.3.4
event.userAgent	イベント関連のクライアントエージェント。	Mozilla/5.0 (...)
event.requestParameters.HostId	リクエスト関連パラメータ内のホスト ID。	ecs.cn-hangzhou.aliyuncs.com
event.requestParameters.Name	リクエスト関連パラメータの名前。	ecs-test
event.requestParameters.Region	リクエスト関連パラメータ内のドメイン。	cn-hangzhou
event.userIdentity.accessKeyId	リクエストで使用された AccessKey ID。	25 **********
event.userIdentity.accountId	リクエストされたアカウントの ID。	123456
event.userIdentity.principalId	リクエストされたアカウントのバウチャー ID。	123456
event.userIdentity.type	リクエストされたアカウントのタイプ。	root-account
event.userIdentity.userName	リクエストされたアカウントの名前。	root

イベントの例

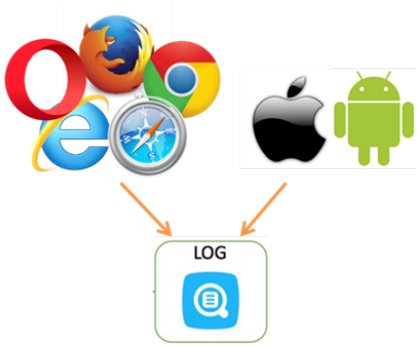
```
{
  " acsRegion ": " cn - hangzhou ",
  " additional  EventData ": {
    " isMFACheck  ed ": " false ",
```

```
    " loginAccou  nt ": " test1234 @ aliyun . com "
  },
  " eventId ": " 7be1e173 - 1234 - 44a1 - b135 - 1234 ",
  " eventName ": " ConsoleSig  nin ",
  " eventSourc  e ": " http :// account . aliyun . com : 443 / login /
 login_aliy  un . htm ",
  " eventTime ": " 2018 - 07 - 12T06 : 14 : 50Z ",
  " eventType ": " ConsoleSig  nin ",
  " eventVersi  on ": " 1 ",
  " requestId ": " 7be1e173 - 1234 - 44a1 - b135 - 1234 ",
  " serviceNam  e ": " AasCustome  r ",
  " sourceIpAd  dress ": " 42 . 120 . 75 . 137 ",
  " userAgent ": " Mozilla / 5 . 0  ( Macintosh ;  Intel   Mac   OS
 X   10_13_6 )  AppleWebKi  t / 537 . 36  ( KHTML ,  like   Gecko )
 Chrome / 67 . 0 . 3396 . 99   Safari / 537 . 36 ",
  " userIdenti  ty ": {
    " accessKeyI  d ": " 25 ***************",
    " accountId ": " 1234 ",
    " principalI  d ": " 1234 ",
    " type ": " root - account ",
    " userName ": " root "
  }
}
```

5 その他の収集方法

5.1 Web トラッキング

Web トラッキングを使用することにより、HTML、H5、iOS、または Android プラットフォームからデータを収集し、Log Service でディメンションおよび指標をカスタマイズできます。



上図に示されているように、Web トラッキング機能を使用することにより、あらゆるブラウザ、iOS アプリ、および Android アプリよりユーザー情報を収集できます (iOS / Android SDK を除く)。 たとえば、

- ユーザーの使用しているブラウザ、オペレーティングシステム、および解像度
- ユーザーの閲覧行動パターン (ユーザーのクリック行動や Web サイトの購入行動など)
- ユーザーのアプリケーション使用時間、アクティブ/非アクティブ

> 注 :
> Web トラッキングを使用すると、インターネットの匿名による Logstore への書き込みが可能になります。不正なデータが生成される危険性があります。

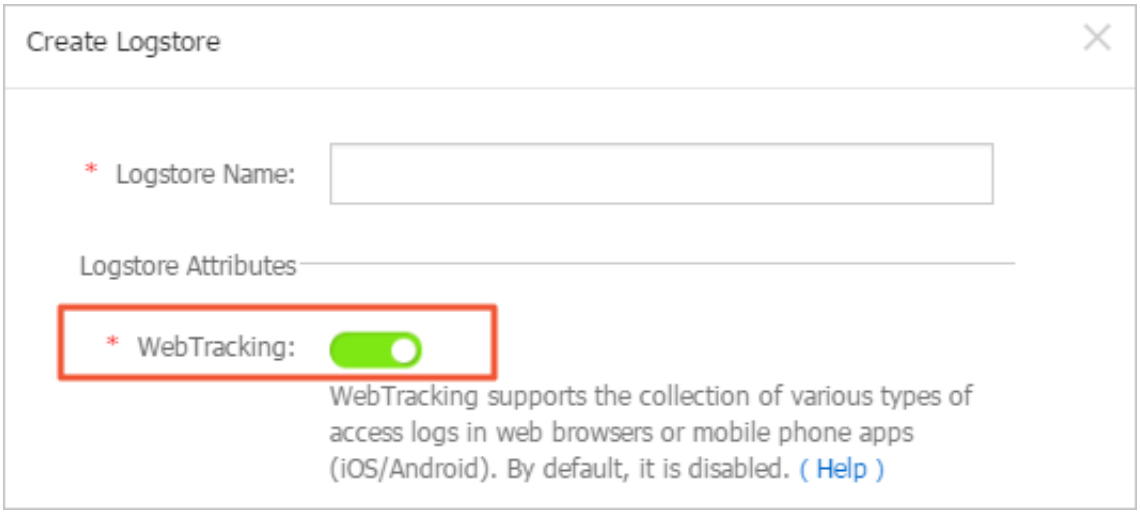
手順

ステップ1 Web トラッキングを有効化

Web トラッキングは、コンソールまたは Java SDK より有効にします。

- コンソールより Web トラッキングを有効化

  1. Logstore リストページで、 Web トラッキング機能を有効にする Logstore 名の右側の変更をクリックします。
  2. Webトラッキングをオンにします。



- Java SDK より Web トラッキングを有効化

Java SDK

```
import com . aliyun . openservic es . log . Client ;
import com . aliyun . openservic es . log . common . LogStore ;
import com . aliyun . openservic es . log . exception .
LogExcepti on ;
public class WebTrackin g {
  static private String accessId = " your accesskey id
";
  static private String accessKey = " your accesskey ";
  static private String project = " your project ";
  static private String host = " log service data
address ";
  static private String logStore = " your logstore ";
  static private Client client = new Client ( host ,
accessId , accessKey );
  public static void main ( String [] args ) {
      try {
         //作成した Logstore の Web Tracking 機能を有効化
          LogStore logSt = client . GetLogStor e ( project ,
logStore ). GetLogStor e ();
          client . UpdateLogS tore ( project , new LogStore (
logStore , logSt . GetTtl (), logSt . GetShardCo unt (), true
));
         // Web Tracking 機能を無効化
         // client . UpdateLogS tore ( project , new LogStore (
logStore , logSt . GetTtl (), logSt . GetShardCo unt (), false
));
         // Web Tracking 機能を有効にした Logstore を生成
         // client . UpdateLogS tore ( project , new LogStore (
logStore , 1 , 1 , true ));
     }
      catch ( LogExcepti on e ){
          e . printStack Trace ();
     }
```

```
  }
}
```

ステップ2 ログを収集

Logstore の Web トラッキング機能を有効にしたら、次の 3 つのいずれかの方法で、データを Logstore にアップロードします。

- HTTP GET リクエスト

```
curl  -- request  GET  ' http ://${ project }.${ host }/ logstores
/${ logstore }/ track ?  APIVersion = 0 . 6 . 0 & key1 = val1 &
key2 = val2 '
```

パラメータの定義

フィールド	定義
`$ { project }`	Log Service に作成したプロジェクトの名前
`${ host }`	Log Service のデプロイされているリージョンのドメイン名
`${ logstore }`	`${ project }`下で Web トラッキング機能が有効になっている Logstore の名前
`APIVersion = 0 . 6 . 0`	予約フィールド (必須)
`__topic__ = yourtopic`	ログトピックを指定します (オプションの予約フィールド)
`key1 = val1 , key2 = val2`	Logstore にアップロードされるキーと値のペア (複数ペア指定可、URL は 16 KB 未満)

- HTML の img タグ

```
< img   src =' http ://${ project }.${ host }/ logstores /${
 logstore }/ track . gif ?  APIVersion = 0 . 6 . 0 & key1 = val1 &
 key2 = val2 '/>
< img   src =' http ://${ project }.${ host }/ logstores /${
 logstore }/ track_ua . gif ?  APIVersion = 0 . 6 . 0 & key1 = val1
 & key2 = val2 '/>
```

パラメータの定義は、 HTTP GET リクエストと同様です。track_ua.gif は、カスタムパラメータをアップロードするだけでなく、HTTP ヘッダーの UserAgent および referer をサーバーのログフィールドとして送信します。

注：

HTTPS ページのリファラーを収集するには、前述の Webトラッキングのリンクが HTTPS タイプでなければなりません。

- JS SDK

1. *loghub - tracking . js* を *web* ディレクトリにコピーし、次のスクリプトをページに追加します。

   こちらをクリックしてダウンロード

```
< script   type =" text / javascript "  src =" loghub - tracking .
 js "  async ></ script >
```

   注：

   ページ読み込みを中断させることのないよう、スクリプトは HTTP リクエストを非同期に送信します。ページの読み込み中に何度もデータを送信する必要がある場合、前の HTTP リクエストは後続のリクエストに上書きされ、ブラウザには追跡リクエストを終了した旨が示されます。この問題は、同期リクエストを送信することにより、防ぐことができます。同期リクエストを送信するには、スクリプトの次の文を変更します。

   元のスクリプト

```
 this . httpReques  t_ . open (" GET ",  url ,  true )
```

   最後のパラメータを変更

```
 this . httpReques  t_ . open (" GET ",  url ,  false )
```

2. Tracker オブジェクトを作成します。

```
var   logger  =  new   window . Tracker ('${ host }','${ project
}','${ logstore }');
logger . push (' customer ', ' zhangsan ');
logger . push (' product ', ' iphone   6s ');
logger . push (' price ',  5500 );
logger . logger ();
logger . push (' customer ', ' lisi ');
logger . push (' product ', ' ipod ');
logger . push (' price ',  3000 );
logger . logger ();
```

   パラメータの意味は次のとおりです。

フィールド	定義
`${ host }`	Log Service のデプロイされているリージョンのドメイン名
`${ project }`	Log Service に作成したプロジェクトの名前

フィールド	定義
`${ logstore }`	`${ project }`下で Web トラッキング機能が有効になっている Logstore の名前

上記ステートメントが実行されると、Log Service に、以下の 2 つのログが表示されます。

```
customer : zhangsan
product : iphone   6s
price : 5500
```

```
customer : lisi
product : ipod
price : 3000
```

Log Service にデータがアップロードされたら、Log Service のLogSearch/Analyticsでリアルタイムにログデータを検索/分析し、さまざまな可視化ソリューションによって分析結果をリアルタイムに表示させることができます。 また、Log Service に用意されている LogHub クライアントライブラリ を使用して Log Service のデータを読み込むこともできます。

5.2 Logstash

5.2.1 カスタムインストール

Logstash は、クイックインストールまたはカスタムインストールによりインストールします。

logstroudsburg のインストールで詳細設定が必要な場合は、カスタムインストールを選択し、デフォルトのインストール構成を変更します。

1. Java をインストール

   a. インストールパッケージをダウンロード

   Java 公式 Web サイトより JDK をダウンロードします。

   b. 環境変数を設定

   システムの詳細設定で環境変数を追加または変更します。

   - PATH: *C :\ Program Files \ Java \ jdk1 . 8 . 0_73 \ bin*
   - CLASSPATH: *C :\ Program Files \ Java \ jdk1 . 8 . 0_73 \ lib ; C :\ Program Files \ Java \ jdk1 . 8 . 0_73 \ lib \ tools . jar*
   - JAVA_HOME: *C :\ Program Files \ Java \ jdk1 . 8 . 0_73*

   c. 確認

   `PowerShell` または `cmd . exe` を実行して確認します。

```
PS C :\ Users \ Administra tor > java - version
java version " 1 . 8 . 0_73 "
Java ( TM ) SE Runtime Environmen t ( build 1 . 8 . 0_73
- b02 )
Java HotSpot ( TM ) 64 - Bit Server VM ( build 25 . 73
- b02 , mixed mode )
PS C :\ Users \ Administra tor > javac - version
javac 1 . 8 . 0_73
```

2. Logstash をインストール

   a. 公式の Web サイトからインストールパッケージをダウンロードします。

   Logstashのホームページで Version 2.2 以降を選択します。

   b. Logstash をインストールします。

   logstash - 2 . 2 . 2 . zip を *C :\ logstash - 2 . 2 . 2* ディレクトリに展開します。

   Logstash 起動プログラムが *C :\ logstash - 2 . 2 . 2 \ bin \ logstash . bat* であることを確認します。

3. Logstash が Log Service にログを書き込むためプラグインをインストール

   マシンのネットワーク環境に合わせて、オンラインまたはオフラインでプラグインをインストールします。

   - オンラインインストール

     RubyGems がホストのプラグインです。 詳細は、こちらをご参照ください。

     `PowerShell` または `cmd . exe` を実行して Logstash インストールディレクトリに移動します。

     ```
     PS   C :\ logstash - 2 . 2 . 2 > .\ bin \ plugin   install
     logstash - output - logservice
     ```

   - オフラインインストール

     Logstash の公式 Web サイトからダウンロードします。 logstash-output-logservice ページに移動し、右下隅のダウンロードをクリックします。

     ログが収集されたマシンがインターネットにアクセスできない場合は、ダウンロードした gem パッケージをマシンの *C :\ logstash - 2 . 2 . 2* ディレクトリにコピーします。 *PowerShell* または *cmd . exe* を実行して Logstash インストールディレクトリに移動します。次のコマンドを実行して iLogstash をインストールします。

     ```
     PS   C :\ logstash - 2 . 2 . 2 > .\ bin \ plugin   install   C :\
     logstash - 2 . 2 . 2 \ logstash - output - logservice - 0 . 2 . 0
     . gem
     ```

   - 確認

     ```
     PS   C :\ logstash - 2 . 2 . 2 > .\ bin \ plugin   list
     ```

     マシンのプラグインリストに「logstash-output-logservice」があることを確認します。

4. NSSM のインストール

   Logstash の公式 Web サイトよりダウンロードします。NSSM のインストールパッケージをダウンロードするには、NSSM の公式 Web サイトにアクセスします。

   インストールパッケージをローカルマシンにダウンロード後、 *C :\ logstash - 2 . 2 . 2 \ nssm - 2 . 24* ディレクトリに展開します。

5.2.2 Logstash 収集の構成ファイル作成

プラグインパラメータ

- logstash-input-file

  本プラグインは、tail モードでログファイルを収集するときに使用します。 詳細は、logstash-input-fileをご参照ください。

   注：
  path にはファイルパスを Unix のファイル区切り文字を使用して指定します (例: `C :/ test / multiline /*. log`)。Unix のファイル区切り文字でない場合、ファジーマッチに対応しません。

- logstash-output-logservice

  本プラグインは、logstash-input-file プラグインによって収集されたログを Log Service に出力するために使用します。

パラメータ	説明
endpoint	Log Service のエンドポイント (例: `http :// regionid . example . com` 、詳細は「Log Service エンドポイント」を参照)
project	Log Service プロジェクトの名前
logstore	Logstore 名
topic	ログトピック名 (デフォルト値: null)
source	ログソース (本パラメータの設定が空の場合、ローカルマシンの IP アドレスがログソースになります)
access_key_id	Alibaba Cloud アカウントの AccessKeyID
access_key_secret	Alibaba Cloud アカウントのキーシークレット
max_send_retry	例外のためにデータパケットを Log Service に送信できない場合に実行される最大再試行回数 (200 ミリ秒ごとに再試行し、再試行に失敗したデータパケットは破棄されます)

1. 収集の構成ファイルを作成

   `C :\ logstash - 2 . 2 . 2 - win \ conf \` ディレクトリに構成ファイルを作成し、Logstash を再起動してファイルを適用します。

   ログタイプごとに構成ファイルを作成することができます。ファイルの拡張子は`*. conf` にします。構成ファイルは `C :\ logstash - 2 . 2 . 2 - win \ conf \` ディレクトリに作成し、管理しやすくすることをお勧めします。

    注：

構成ファイルは、BOM なしの UTF-8 でエンコードする必要があります。Notepad++ をダウンロードして、ファイルのエンコード形式を変更します。

- IIS ログ

  詳細は「Logstash を使用した IIS ログの収集」をご参照ください。

- CSV ログ

  ログ収集のシステム時間が、ログのアップロード時間になります。詳細は、CSV ログ設定をご確認ください。

- デフォルトのログ時間

  CSV ログの場合、 ログコンテンツ内の時間が、ログのアップロード時間になります。詳細については、「Logstash を使用した CSV ログの収集」をご参照ください。

- 一般的なログ

  デフォルトでは、ログ収集のシステム時間が、ログのアップロード時間になります。ログのフィールドは解析されません。一行および複数行のログの両方がサポートされています。詳細については「Logstash を使用したその他のログの収集」をご参照ください。

2. 構成の構文確認

   a. `PowerShell` または `cmd . exe` を実行して Logstash インストールディレクトリに移動します。

   ```
   PS   C :\ logstash - 2 . 2 . 2 - win \ bin > .\ logstash . bat
   agent  -- configtest  -- config   C :\ logstash - 2 . 2 . 2 - win
   \ conf \ iis_log . conf
   ```

   b. 収集の構成ファイルを変更します。コンソールに収集結果を出力するには、出力フェーズに rubydebug を一時的に追加します。必要に応じてタイプフィールドを設定します。

   ```
   output  {
   If  [ type ] = "***"{
     stdout  {  codec  =>  rubydebug  }
     logservice  {

    }
   ```

   c. `PowerShell` または `cmd . exe` を実行して Logstash インストールディレクトリに移動し、プロセスを起動します。

   ```
   PS   C :\ logstash - 2 . 2 . 2 - win \ bin > .\ logstash . bat
   agent  - f   C :\ logstash - 2 . 2 . 2 - win \ conf
   ```

   確認後、 *logstash . bat* プロセスを終了し、一時的な rubydebug 設定を削除します。

PowerShell で *logstash . bat* を起動すると、Logstash プロセスはフォアグラウンドで動作します。一般的に、Logstash は設定のテストやログ収集のデバッグに使用します。したがって、Logstash を電源投入時に自動的に起動し、バックグラウンドで実行させるには、デバッグ後に Logstash を Windows サービスに登録することをお勧めします。Logstash を Windows サービスに登録する方法については、「Logstash を Windows サービスに登録」をご参照ください。

5.2.3 Logstash を Windows サービスに登録

PowerShell で Logstash.bat が起動すると、Logstash プロセスはフロントエンドで実行されます。Logstash は通常、構成の確認や収集のデバッグに使用します。したがって、電源投入時に自動的に Logstash が起動し、バックエンドで実行させるよう、デバッグ後に Logstash を Windows サービスに設定されることをお勧めします。

Logstash を Windows サービスとして設定する以外にも、コマンドラインよりサービスを開始、停止、変更、および削除することもできます。NSSM の使用方法については、NSSM 公式文書をご参照ください。

Logstash を Windows サービスに追加

通常は、Logstash を初めてデプロイする際に追加します。Logstash を既に追加している場合は、この手順をスキップします。

Logstash を Windows サービスに追加するには、次のコマンドを実行します。

- 32 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win32 \ nssm .
exe   install   logstash  " C :\ logstash - 2 . 2 . 2 - win \ bin
\ logstash . bat " " agent  - f   C :\ logstash - 2 . 2 . 2 - win \
conf "
```

- 64 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win64 \ nssm .
exe   install   logstash  " C :\ logstash - 2 . 2 . 2 - win \ bin
\ logstash . bat " " agent  - f   C :\ logstash - 2 . 2 . 2 - win \
conf "
```

サービスの起動

Logstash の *conf* ディレクトリにある設定ファイルが更新されている場合は、Logstash サービスを一旦停止し、再起動します。

サービスを開始するには、次のコマンドを実行します。

- 32 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win32 \ nssm .
exe   start   logstash
```

- 64 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win64 \ nssm .
exe   start   logstash
```

サービスの停止

サービスを停止するには、次のコマンドを実行します。

- 32 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win32 \ nssm .
exe   stop   logstash
```

- 64 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win64 \ nssm .
exe   stop   logstash
```

サービスの修正

サービスを変更するには、次のコマンドを実行します。

- 32 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win32 \ nssm .
exe   edit   logstash
```

- 64 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win64 \ nssm .
exe   edit   logstash
```

サービスの削除

サービスを削除するには、次のコマンドを実行します。

- 32 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win32 \ nssm .
exe   remove   logstash
```

- 64 ビットシステム

```
C :\ logstash - 2 . 2 . 2 - win \ nssm - 2 . 24 \ win64 \ nssm .
exe   remove   logstash
```

5.2.4 高度な関数

Logstash には、あらゆる要件に対応できるようさまざまなプラグインが用意されています。たとえば、

- grok: 正規表現を使用してログを複数のフィールドに構造解析
- json_lines、json: JSON ログの構造解析
- date: ログの日付フィールドおよび時間フィールドを解析して変換
- multiline: さまざまな複数行のログをカスタマイズ
- kv: キーと値がペアになっているログの構造解析

5.2.5 Logstash エラー処理

Logstash によるログ収集に、以下の収集エラーが発生した場合は、そのガイドラインに従って、エラーを処理します。

Logstash によるログ収集に、以下の収集エラーが発生した場合は、そのガイドラインに従って、エラーを処理します。

- Log Service の文字化けデータ

  Logstash は、デフォルトでファイルを UTF-8 エンコードします。入力ファイルが正しくエンコードされているかどうかを確認します。

- コンソールのエラーメッセージ

  コンソールに、エラーメッセージ `io / console not supported ; tty will not be manipulate d` が表示されます。ただし、本エラーは機能自体には影響を及ぼさないため、無視して構いません。

  その他のエラーの発生時には、Google または Logstash フォーラムでの検索をお勧めします。

5.3 SDK 収集

5.3.1 Producer Library

LogHub Producer Library は、並行性の高い Java アプリケーション向けに作成された LogHub のクラスライブラリです。Producer Library およびConsumer Libraryは、LogHub がデータの収集および読み込みのしきい値を下げるための読み書きパッケージです。

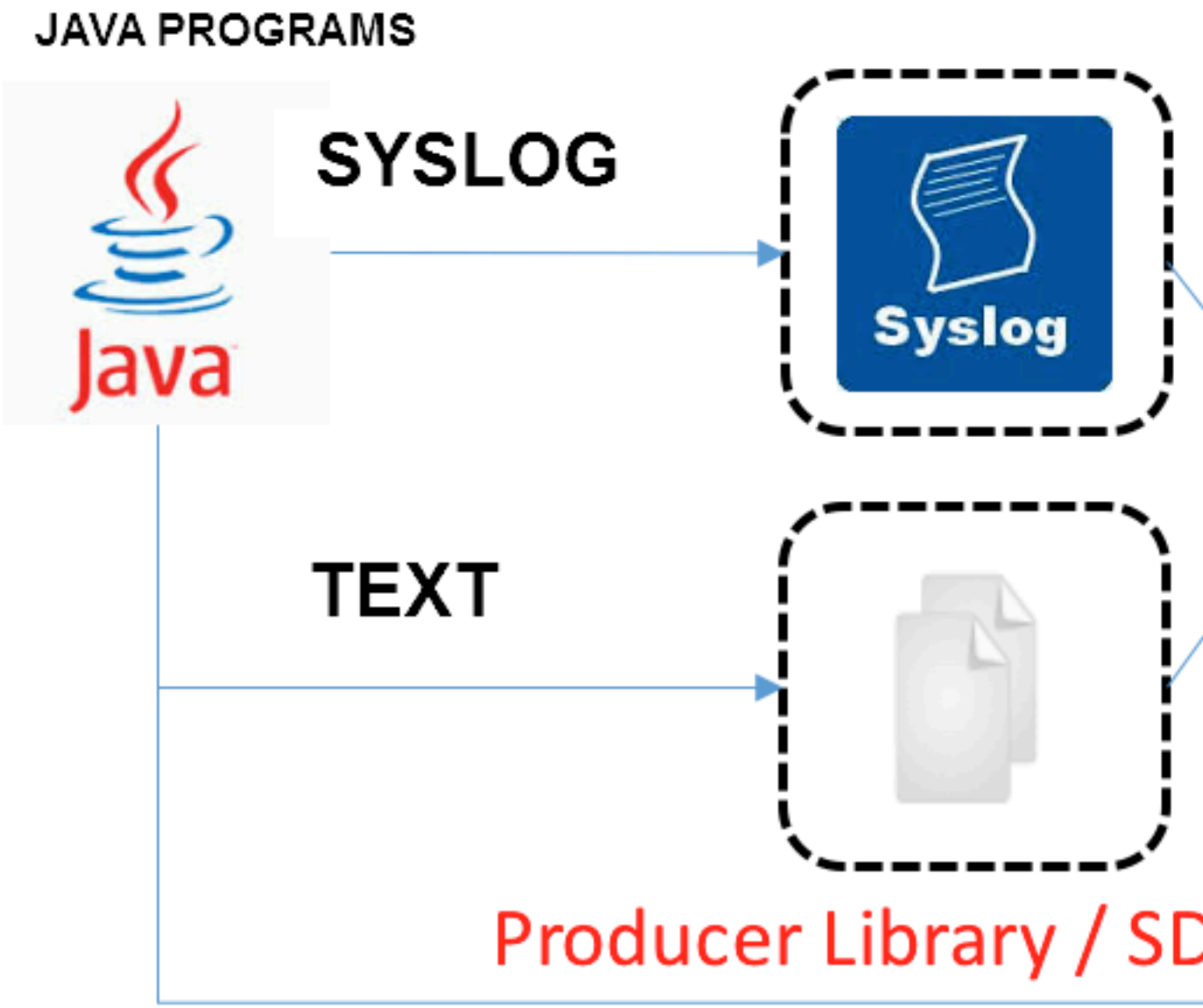
機能の特徴

- 非同期送信インターフェイスにより、セキュアなスレッドを保証します。
- プロジェクト構成を複数追加することができます。
- ログ送信のネットワーク I/O スレッド数を設定することができます。
- 統合パッケージのログの件数とサイズを設定することができます。
- メモリ使用量を調節できます。メモリ使用量が指定のしきい値に達すると、アイドルメモリが使用可能になるまで Producer の送信インターフェイスはブロックされます。

機能の利点

- クライアントからの収集ログは、ディスクには書き込まれません。ログデータが生成されると、ネットワークを介してそのまま Log Service に送信されます。
- 高い並行処理でクライアントに書き込まれます (100 件/秒を超える書き込み処理)。
- クライアントコンピューティングの I/O は論理的に分離されます。ログの書き込みは、処理時間に影響しません。

Producer Library を使用することにより、プログラム開発が簡素になります。書き込みリクエストは集約され、非同期に LogHub サーバーに送信されます。プログラムに集約パラメーター、また、サーバーにエラーが発生したときの処理を指定することができます。



上記のアクセス方法の比較一覧は下表のとおりです。

アクセス方法	利点/欠点	シナリオ
ログのディスクへの書き込み + Logtail	ログ収集とログは切り離されているため、コードを書き換える必要がありません。	一般的なシナリオ
Syslog + Logtail	高パフォーマンス (80 MB/s)。ログはディスクに書き込まれません。syslog プロトコルに対応している必要があります。	Syslog シナリオ

アクセス方法	利点/欠点	シナリオ
SDK による直接送信	ディスクに書き込まれず、直接サーバーに送信されます。ネットワーク I/O とプログラム I/O の切り替えは、適切に処理する必要があります。	ログはディスクに書き込まれません。
Producer Library	ディスクに書き込まれず、非同期にマージされ、サーバーに送信されるため、高スループットです。	ログはディスクに書き込まれず、クライアントのQPSは高くなります。

手順

- Java Producer
- Log4J1. Log4J1.XAppender (based on Java Producer)
- Log4J2. XAppender (based on Java Producer)
- LogBack Appender (based on Java Producer)
- C Producer
- C Producer Lite

5.3.2 LogHub Log4j Appender

Log4j は Apache のオープンソースプロジェクトであり、ログの出力先をコンソール、ファイル、GUI コンポーネント、ソケットサーバー、NT イベントレコーダー、または UNIX Syslog デーモンに設定できます。各ログの出力形式およびレベルを設定して、細かい粒度でログの生成を制御することもできます。構成ファイルに設定するため、アプリケーションコードを書き換える必要がなく、柔軟に対応できます。

Alibaba Cloud の Log4j Appender では、ログの出力先を Log Service に指定することができます。ダウンロードおよびユーザーガイドは、GitHubをご参照ください。

5.3.3 C Producer Library

LogHub は、Java Producer Library だけでなく、C Producer Library および Producer Lite Library にも対応しており、プラットフォーム間をシンプルかつ高性能に、最小限のリソースでワンストップのログ収集を実現できます。

GitHub プロジェクトの URL は、以下をご参照ください。

- C Producer Library (サーバー向け)、C Producer Library (サーバー向け)
- C Producer Lite Library (IOT およびスマートデバイス向け)

5.4 一般的なログフォーマット

5.4.1 Apache ログ

Apache のログフォーマットおよびディレクトリは、*/ etc / apache2 / httpd . conf* 設定ファイルに指定します。

ログフォーマット

Apache ログの設定ファイルには、出力フォーマットがデフォルトで 2 つ定義されています (結合フォーマットおよび一般フォーマット)。なお、必要に応じてログの出力フォーマットをカスタマイズすることもできます。

- 結合フォーマット

```
LogFormat  "% h  % l  % u  % t  \"% r \" %> s  % b  \"%{ Referer }
i \" \"%{ User - Agent } i \""  combined
```

- 一般フォーマット

```
LogFormat  "% h  % l  % u  % t  \"% r \" %> s  % b "
```

- カスタム化フォーマット

```
LogFormat  "% h  % l  % u  % t  \"% r \" %> s  % b  \"%{ Referer }
i \" \"%{ User - Agent } i \" % D  % f  % k  % p  % q  % R  % T  %
I  % O "  customized
```

Apache のログ設定ファイルに、ログの出力フォーマット、ファイルパス、およびログ名を指定する必要があります。たとえば、次のログ設定ファイルは、結合フォーマットを出力するように指定されており、ログのパスおよび名前は、`/ var / log / apache2 / access_log` です。

```
CustomLog  "/ var / log / apache2 / access_log "  combined
```

フィールドの説明

フォーマット	キー名	説明
%a	client_addr	クライアント IP アドレス
%A	local_addr	ローカルプライベート IP アドレス
%b	response_size_bytes	レスポンスのサイズ (バイト)。空の場合は、ハイフン (-) となります。
%B	response_bytes	レスポンスのサイズ (バイト)。空の場合は、ハイフン (-) となります。
%D	request_time_msec	リクエストの時間 (マイクロ秒単位)

%h	remote_addr	リモートホスト名
%H	request_protocol_supple	リクエストプロトコル
%l	remote_ident	identd のクライアント側のログ名
%m	request_method_supple	リクエストメソッド
%p	remote_port	サーバーのポート
%P	child_process	子プロセス ID
%q	request_query	クエリ文字列。クエリ文字列が存在しない場合は、このフィールドが空文字列となります。
"%r"	request	メソッド名、IP アドレス、および http プロトコルを含むリクエストのコンテンツ
%s	status	HTTP ステータスコード
%>s	status	最終の HTTP ステータスコード
%f	filename	ファイル名
%k	keep_alive	keep alive リクエストの数
%R	response_handler	サーバーのハンドラ
%t	time_local	サーバー時間
%T	request_time_sec	リクエストの時間 (秒単位)
%u	remote_user	クライアントのユーザー名
%U	request_uri_supple	リクエストした URL のパス (クエリを含まない)
%v	server_name	サーバー名
%V	server_name_canonical	UseCanonicalName 設定に従ったサーバー名
%I	bytes_received	サーバーの受信バイト数 (mod_logio モジュールが有効な場合のみ)
%O	bytes_sent	サーバーの送信バイト数 (mod_logio モジュールが有効な場合のみ)
%{User-Agent}i	http_user_agent	クライアント情報
"%{Rererer}i"	http_referer	ソースページ

サンプルログ

```
192 . 168 . 1 . 2  - - [ 02 / Feb / 2016 : 17 : 44 : 13  + 0800
] " GET  / favicon . ico  HTTP / 1 . 1 "  404  209  " http ://
localhost / x1 . html " " Mozilla / 5 . 0  ( Macintosh ;  Intel
Mac  OS  X  10_11_3 )  AppleWebKi  t / 537 . 36  ( KHTML ,  like
Gecko )  Chrome / 48 . 0 . 2564 . 97  Safari / 537 . 36 "
```

Logtail を使用した Apache ログの収集

1. Logstoreページで、データインポートウィザードアイコンをクリックします。
2. データタイプを選択します。

   APACHE アクセスログを選択します。
3. データソースを設定します。

   a. 構成名およびログパスを入力します。

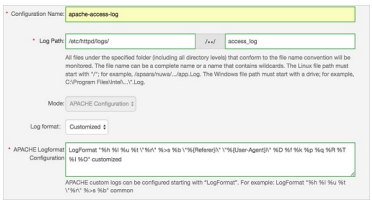
   b. ログフォーマットを選択します。

   c. ログフォーマットをカスタマイズする場合は、APACHE のログフォーマット構成に入力します。

   Apache 標準の設定ファイルのログフォーマット設定欄に入力します。通常、設定ファイルは「LogFormat」で始まります。

   > 注：
   > ログフォーマットドロップダウンリストより一般または結合を選択した場合に
   > は、APACHE のログフォーマット設定欄に該当するログフォーマットが自動的に入力され

ます。入力された内容が、ローカルの Apache 設定ファイルに定義されているフォーマットと同じであることを確認します。



d. APACHE キー名を確認します。

Log Service によって、お客様の Apache キーが自動的に読み込まれます。現在のページの APACHE キー名を確認します。

e. （オプション）詳細オプションを設定します。

構成項目	詳細
ローカルキャッシュ	ローカルキャッシュを有効にするかどうかを選択します。 この機能を有効にすると、ログサービスが利用できないときにログがマシンのローカルディレクトリにキャッシュされ、サービスの復旧後にログサービスに引き続き送信されます。 デフォルトでは、最大で1GBのログをキャッシュできます。
オリジナルログのアップロード	オリジナルログをアップロードするかどうかを選択します。有効にすると、デフォルトで新しいフィールドが追加され、オリジナルログがアップロードされます。

構成項目	詳細
トピック生成モード	· Null - トピックを生成しない：デフォルトオプションであり、トピックをヌル文字列として設定し、トピックを入力せずにログを照会できることを示します。 · マシングループのトピック属性：異なるフロントエンドサーバーで生成されたログデータを明確に区別するために使用されます。 · ファイルパスレギュラー：このオプションを選択すると、正規表現を使用してパスからコンテンツをトピックとして抽出するには、カスタム正規表現を入力する必要があります。 ユーザーとインスタンスによって生成されたログデータを区別するために使用されます。 ユーザーとインスタンスによって生成されたログデータを区別するために使用されます。
カスタム正規表現	トピック生成モードとしてファイルパスレギュラーを選択すると、カスタム正規表現を入力する必要があります。
ログファイルエンコーディング	· utf8: UTF-8エンコーディングを使用。 · gbk: GBKエンコーディングを使用
最大モニターディレクトリの深さ	ログソースからログを収集するときに監視するディレクトリの最大深度を指定します。要するに、最大でモニターできるログレベルを指定します。 範囲は0~1000で、0を指定した場合は、現行ディレクトリレベルのみをモニターすることになります。
タイムアウト	指定された時間内に更新がない場合、ログファイルはタイムアウトします。 タイムアウトの次の設定を構成できます。 · タイムアウトにならない：すべてのログファイルを永続的にモニターし、ログファイルがタイムアウトしないように指定します。 · 30分のタイムアウト：ログファイルが30分以内に更新がないければタイムアウトになり、モニターされなくなります。

構成項目	詳細
フィルター構成	フィルター条件に完全に準拠したログのみ、収集できます。 たとえば： ・ 条件に一致するログを収集します：Key: level Regex:WARNING|ERRORは、レベルがWARNINGまたはERRORのログのみを収集することを示します。 ・ 条件に適合しないログをフィルターします： - `Key : level Regex :^(?!. *( INFO | DEBUG ))`、INFOまたはDEBUGレベルのログを収集しないことを示します。 - `Key : url Regex :. *^(?!.*( healthchec k )). *`、urlにヘルスチェックを含むログをフィルターすることを示します。 keyがurlでvalueが`/ inner / healthchec k / jiankong . html` のログなどは収集されません。 同様の事例についてはregex-exclude-wordとregex-exclude-patternを参照してください。

4. 次へをクリックします。
5. マシングループを選択し、マシングループに適用をクリックします。

   マシングループをまだ作成していない場合は、マシングループの作成をクリックしてマシングループを作成します。

   Logtail 設定をマシングループに適用すると、Log Service は設定に従って Apache ログを収集します。データインポートウィザードの手順に従って、インデックスおよびログ送信を設定します。

5.4.2 Nginx ログ

Nginx のログフォーマットおよびディレクトリは/ etc / nginx / nginx . conf 設定ファイルに指定します。

Nginx のログフォーマット

ログ設定ファイルに、Nginx ログの出力フォーマットが定義されています (main フォーマット)。

```
log_format   main  '$ remote_add  r  - $ remote_use  r  [$ time_local
] "$ request " '
                '$ request_ti  me  $ request_le  ngth  '
                '$ status  $ body_bytes  _sent  "$ http_refer  er " '
                '"$ http_user_  agent "';
```

次の宣言文には、 main ログフォーマットおよびログデータの書き込み先ファイル名が定義されています。

```
access_log  / var / logs / nginx / access . log   main
```

フィールド説明

フィールド名	定義
remoteaddr	クライアントの IP アドレス
remote_user	クライアントのユーザー名
request	リクエスト URL および HTTP プロトコル
status	リクエストステータス
bodybytessent	クライアントへの送信バイト数 (応答ヘッダーを含まない。本変数は、Apache モジュールの modlogconfig で bytes_sent と併せて使用)
connection	接続のシリアル番号
connection_requests	接続で受信したリクエストの数
msec	ログ書き込み時間（ミリ秒単位）
pipe	リクエスト送信を HTTP パイプライン経由にするかどうか (HTTP パイプライン経由でリクエストを送信する場合は p 、パイプライン経由でリクエストを送信しない場合は.を指定)
httpreferer	Web ページの参照元
“http_user_agent”	クライアントのブラウザ情報 (http_user_agent は、二重引用符で囲むこと)

フィールド名	定義
requestlength	リクエストの長さ (リクエスト行、リクエストヘッダー、およびリクエスト本文を含む)
Request_time	リクエストの処理時間 (単位: ミリ秒単位、最初のバイトがクライアントに送信されてから、最後の文字がクライアントに送信されるまでの時間)
[$time_local]	一般的なログフォーマットが適用されるローカル時刻 (本変数は角かっこで囲むこと)

ログサンプル

```
192 . 168 . 1 . 2  - - [ 10 / Jul / 2015 : 15 : 51 : 09  + 0800 ] "
GET  / ubuntu . iso   HTTP / 1 . 0 "  0 . 000   129   404   168  "-"
" Wget / 1 . 11 . 4   Red   Hat   modified "
```

Logtail を使用した Nginx ログの収集

1. Logstoreリストページのデータインポートウィザードアイコンをクリックすると、データインポートウィザードが起動します。
2. データソースを選択します。

   テキストファイルを選択し、次へをクリックします。
3. データソースを選択します。

   a. 構成名およびログパスを入力します。

   b. Nginx のログフォーマットを入力します。

      Nginx 標準のログフォーマット設定を入力します。通常、 `log_format` で始まります。Log Service によって、ご利用の Nginx キーが自動的に読み込まれます。

   c. 必要に応じて詳細オプションを設定します。設定したら、次へをクリックします。

      詳細については、「詳細オプション」をご参照ください。

   Logtail を設定したら、設定をマシングループに適用します。Nginx ログの収集が開始されます。

5.4.3 Python ログ

Python のロギングモジュールは、サードパーティーのモジュールやアプリケーションで使用できる汎用のロギングシステムです。ロギングモジュールは、ファイル、HTTP GET/POST、SMTP、Socket といったさまざまな方法で、さまざまなログレベルおよびログを記録します。ロギング方法をカスタマイズすることもできます。ロギングモ

ジュールは基本的には Log4j と同じですが、細部が異なります。ロギングモジュールには、Logger、Handler、Filter、および Formatter 機能があります。

Python ログを収集するには、logging handler をそのまま使用されることをお勧めします。

- logging handler で Python ログを自動アップロード
- logging handler で KV 形式のログを自動解析
- logging handler で JSON 形式のログを自動解析

Python のログフォーマット

formatter のログ出力フォーマットがログフォーマットです。Formatter には、メッセージのフォーマット (String 型) および日付 (String 型) の 2 つのパラメーターを定義します。両パラメーターともオプションです。

Python のログフォーマット

```
import logging
import logging . handlers
LOG_FILE = ' tst . log '
handler = logging . handlers . RotatingFi leHandler ( LOG_FILE ,
maxBytes = 1024 * 1024 , backupCoun t = 5 ) # Instantiat e
the handler
fmt = '%( asctime ) s - %( filename ) s :%( lineno ) s - %( name )
s - %( message ) s '
formatter = logging . Formatter ( fmt ) # Formatter を初期化
handler . setFormatt er ( formatter ) # Handler に Formatter
を追加
logger = logging . getLogger (' tst ') # 「 tst 」 Logger を取得
logger . addHandler ( handler ) # Logger に Handler を追
加
logger . setLevel ( logging . DEBUG )
logger . info (' first info message ')
logger . debug (' first debug message ')
```

フィールドの説明

Formatter の設定書式は、`%( key ) s` です。属性辞書のキーワードを置き換えます。キーワードは、以下のとおりです。

フォーマット	意味
%(name)s	生成されたログの Logger 名
%(levelno)s	メッセージのログレベル (DEBUG、INFO、WARNING、ERROR、および CRITICAL) の番号
%(levelname)s	メッセージのログレベル (DEBUG、INFO、WARNING、ERROR、および CRITICAL) の文字列

フォーマット	意味
%(pathname)s	ロギングの呼び出しソースファイルの完全パス (取得可能な場合)
%(filename)s	ファイル名
%(module)s	ロギングの呼び出しモジュール名
%(funcName)s	ロギングの呼び出し関数名
%(lineno)d	ロギングの呼び出しコードの行 (取得可能な場合)
%(created)f	ログの生成時間 (UNIX タイムスタンプ)。1970-1-100 00:00:00 UTC からの秒数。
%(relativeCreated)d	ログの生成時間およびロギングモジュールのロード時間との差 (単位: ミリ秒)
%(asctime)s	ログの生成時間。書式: デフォルトで「2003-07-08 16：49：45,896」(コンマ (,) の後の数字はミリ秒数)
%(msecs)d	ログ生成時間 (単位: ミリ秒)
%(thread)d	スレッドID (取得可能な場合)
%(threadName)s	スレッド名 (取得可能な場合)
%(process)d	プロセス ID (オプション)
%(message)s	ログメッセージ

ログサンプル

ログサンプル

```
2015 - 03 - 04   23 : 21 : 59 , 682  -  log_test . py : 16  -  tst
-  first   info   message
2015 - 03 - 04   23 : 21 : 59 , 682  -  log_test . py : 17  -  tst
-  first   debug   message
```

一般的な Python ログおよび正規表現

- ログフォーマット

```
2016 - 02 - 19   11 : 03 : 13 , 410  -  test . py : 19  -  tst  -
first   debug   message
```

正規表現

```
(\ d +-\ d +-\ d +\ s \ S +)\ s +-\ s +([^:]+):(\ d +)\ s +-\ s +(\
w +)\ s +-\ s +(. *)
```

- ログフォーマット

```
%( asctime ) s  - %( filename ) s :%( lineno ) s  - %( levelno ) s
%( levelname ) s  %( pathname ) s  %( module ) s  %( funcName ) s
%( created ) f  %( thread ) d  %( threadName ) s  %( process ) d
%( name ) s  - %( message ) s
```

ログサンプル

```
2016 - 02 - 19   11 : 06 : 52 , 514  -  test . py : 19  -  10
  DEBUG   test . py   test  < module >  1455851212 . 514271
1398659966  87072   MainThread   20193   tst  -  first   debug
message
```

正規表現

```
(\ d +-\ d +-\ d +\ s \ S +)\ s -\ s ([^:]+):(\ d +)\ s +-\ s +(\ d
+)\ s +(\ w +)\ s +(\ S +)\ s +(\ w +)\ s +(\ S +)\ s +(\ S +)\ s
+(\ d +)\ s +(\ w +)\ s +(\ d +)\ s +(\ w +)\ s +-\ s +(. *)
```

Logtail を使用した Python ログの収集

Logtail を使って Python ログを収集する詳しい手順については、「5 分でクイックスタート」をご参照ください。ネットワーク構成とネットワーク設定をもとに、対応する構成を選択します。

1. プロジェクトおよび Logstore を作成します。手順の詳細については、「準備」をご参照ください。
2. Logstoreリストページで、データインポートウィザードアイコンをクリックします。
3. データソースを選択します。

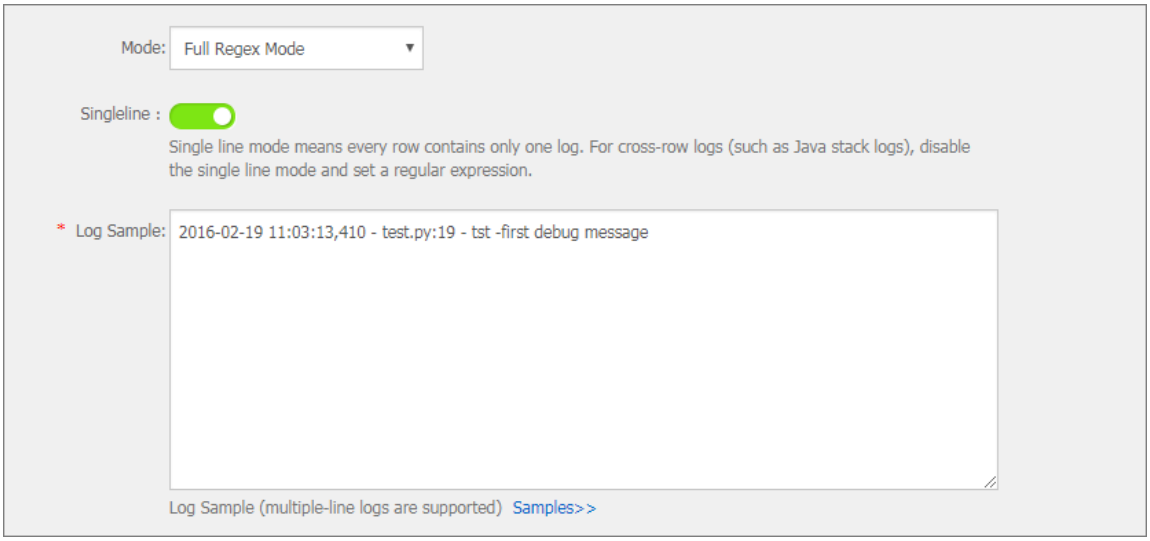
   テキストを選択します。

4. データソースを設定します。

   a. 構成名およびログパスを入力し、モードドロップダウンリストで完全正規表現モードを選択します。

   b. 単一行スイッチをオンにします。

   c. ログサンプルを入力します。



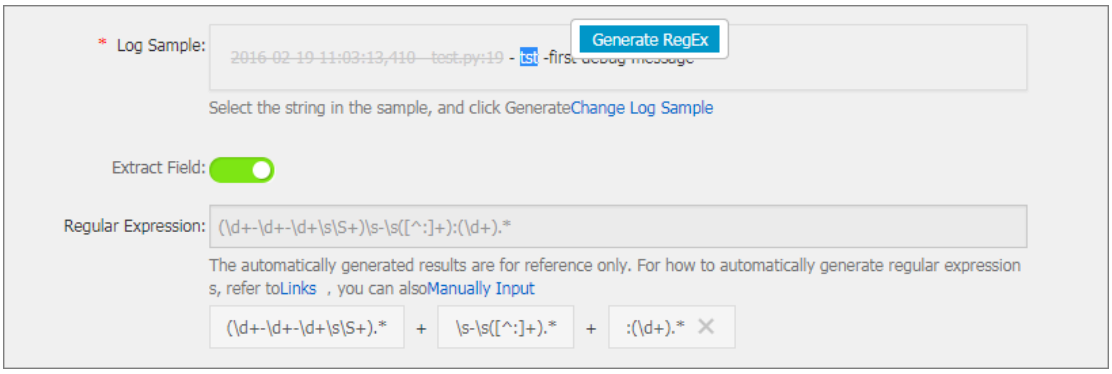
   d. フィールド抽出スイッチをオンにします。

   e. 正規表現を設定します。

      A. ログサンプルの文字列を選択して正規表現を生成します。

         自動生成された正規表現がログサンプルと異なる場合、ログサンプルの文字列を選択して正規表現を生成することができます。ログサンプルの文字列を選択すると、Log Service は選択した文字列からフィールドを自動解析し、正規表現を自動生成します。ログサンプルで、ログフィールドを選択し、Generate RegExをクリックします。選択

したフィールドの正規表現が正規表現列に表示されます。複数の選択することにより、ログサンプルの完全な正規表現が生成されます。



B. 正規表現を修正します。

ログフォーマットを修正する必要がある場合、手動入力をクリックして、自動生成された正規表現を修正して、収集処理に必要なログフォーマットに一貫性を持たせます。

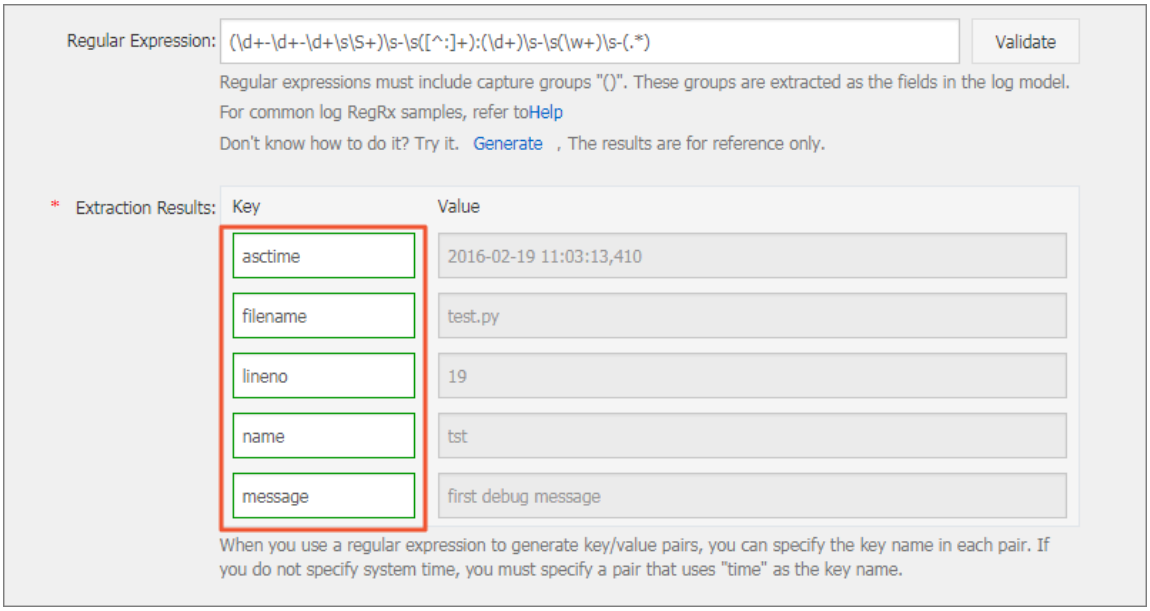
C. 正規表現を検証します。

正規表現を修正したら、検証をクリックします。正規表現が正しい場合、抽出結果が表示されます。エラーがあった場合には、正規表現を修正します。

f. 抽出結果を確認します。

ログフィールドの解析結果を表示し、ログ抽出結果に対応するキーを入力します。

各ログフィールドの抽出結果は、容易に判別できるフィールド名を割り当てます。たとえば、time に時間フィールドを割り当てます。システム時間を使用しない場合は、time キーに値が時間であるフィールドを割り当てます。



g. システム時間スイッチをオンにします。

システム時間を採用すると、Logtail クライアントがログを解析した時間がログ時間になります。

h. (オプション) 詳細オプションを設定します。

i. 次へをクリックします。

Logtail 設定が完了したら、設定をマシングループに適用して Python ログを収集します。

5.4.4 Log4j ログ

アクセスモード

Log Service では、次のプロダクトを使用して Log4j を収集します。

- LogHub Log4j Appender
- Logtail

LogHub Log4j Appender を使用した Log4j ログの収集

詳細については、「LogHub Log4j Appender」をご参照ください。

Logtail を使用した Log4j ログの収集

Log4j ログには第 1 世代および第 2 世代がありますが、本ドキュメントでは、第 1 世代のデフォルト設定を例に、Log4j の一般的な設定方法を説明します。第 2 世代の Log4j を使用している場合、完全な日付を出力するために、デフォルトの設定を修正する必要があります。

```
< Configurat  ion   status =" WARN ">
  < Appenders >
    < Console   name =" Console "  target =" SYSTEM_OUT ">
      < PatternLay  out   pattern ="% d { yyyy - MM - dd   HH : mm :
 ss : SSS   zzz } [% t ] %- 5level  % logger { 36 } - % msg % n "/>
    </ Console >
  </ Appenders >
  < Loggers >
    < Logger   name =" com . foo . Bar "  level =" trace ">
      < AppenderRe  f   ref =" Console "/>
    </ Logger >
    < Root   level =" error ">
      < AppenderRe  f   ref =" Console "/>
    </ Root >
  </ Loggers >
</ Configurat  ion >
```

Log4j のログを収集するための設定については、「Python ログ」をご参照ください。ネットワーク構成およびネットワーク設定に適した設定を選択します。

自動生成される正規表現は、ログサンプルにのみ基づいており、すべてのログフォーマットに対応しているわけではありません。したがって、正規表現が自動生成されてから、微修正する必要があります。

Log4j のログファイルに出力される Log4j のデフォルトのログフォーマットのサンプルは次のとおりです。

```
2013 - 12 - 25   19 : 57 : 06 , 954  [ 10 . 207 . 37 . 161 ]  WARN
  impl . PermanentT  airDaoImpl  -  Fail   to   Read   Permanent
Tair , key : e : 4702173193  19741_1 , result : com . example . tair
. Result @ 172e3ebc [ rc = code =- 1 ,  msg = connection   error   or
  timeout , value =, flag = 0 ]
```

複数行ログの行の先頭を一致させる (IP アドレスで行の先頭を示す)

```
\ d +-\ d +-\ d +\ s .
```

ログデータの抽出に使用される正規表現

```
(\ d +-\ d +-\ d +\ s \ d +:\ d +:\ d +,\ d +)\ s \[([^\]]*)\]\ s (\ S
 +)\ s +(\ S +)\ s -\ s (.
```

時間の変換書式

```
% Y -% m -% d  % H :% M :% S
```

ログサンプルの抽出結果

キー	値
time	2013-12-25 19:57:06,954
ip	10.207.37.161
level	WARN
class	impl.PermanentTairDaoImpl
message	Fail to Read Permanent Tair,key:e:470217319319741_1,result :com.example.tair.Result@172e3ebc[rc=code=-1, msg= connection error or timeout,value=,flag=0]

5.4.5 Node.js ログ

Node.js のログは、直接コンソールに出力されます。ログデータの収集およびトラブルシューティングには不便です。Log4js を使用すると、ログをファイルに出力し、ログフォーマットをカスタマイズすることができます。データの収集にも運用にも便利です。

```
var   log4js  =  require (' log4js ');
log4js . configure ({
  appenders : [
   {
      type : ' file ', // file   output
      filename : ' logs / access . log ',
      maxLogSize :  1024 ,
      backups : 3 ,
      category : ' normal '
   }
```

```
  ]
});
 var   logger  =  log4js . getLogger (' normal ');
 logger . setLevel (' INFO ');
 logger . info (" this   is   a   info   msg ");
 logger . error (" this   is   a   err   msg ");
```

ログフォーマット

Log4js を使用してログデータをテキストファイルに保存する場合、ファイルに出力されるログフォーマットは、次のとおりです。

```
[ 2016 - 02 - 24   17 : 42 : 38 . 946 ] [ INFO ]  normal  -  this   is
   a   info   msg
[ 2016 - 02 - 24   17 : 42 : 38 . 951 ] [ ERROR ]  normal  -  this
 is   a   err   msg
```

Log4js には、trace、debug、info、warn、error、および fatal の 6 つの出力ログレベルがあります (後者ほどより深刻)。

Logtail を使って Node.js ログを収集

Logtail を使用して Node.js ログを収集する方法については、「Python ログ」をご参照ください。ネットワーク構成およびネットワーク設定に適した設定を選択します。

自動生成される正規表現は、ログサンプルのみを基にしており、すべてのログフォーマットに対応しているわけではありません。したがって、正規表現が自動生成されたら、修正を加える必要があります。以下の Node.js ログサンプルを参考に、ログに必要な正規表現を記述します。

一般的な Node.js ログとその正規表現は、以下のとおりです。

- ログサンプル 1
  - ログサンプル

    ```
    [ 2016 - 02 - 24   17 : 42 : 38 . 946 ] [ INFO ]  normal  -  this
       is   a   info   msg
    ```

  - 正規表現

    ```
    \[([^]]+)]\ s \[([^\]]+)]\ s (\ w +)\ s -(. *)
    ```

  - 抽出フィールド

    `time` 、 `level` 、 `loggerName` 、および `message`

- ログサンプル 2
  - ログサンプル

    ```
    [ 2016 - 01 - 31   12 : 02 : 25 . 844 ] [ INFO ]  access  -  42 .
     120 . 73 . 203  - - " GET  / user / projects / ali_sls_lo  g ?
     ignoreErro  r = true   HTTP / 1 . 1 "  304  - " http ://
    ```

```
aliyun . com /" " Mozilla / 5 . 0  ( Macintosh ;  Intel   Mac
OS   X   10_10_3 )  AppleWebKi  t / 537 . 36  ( KHTML ,  like
Gecko )  Chrome / 48 . 0 . 2564 . 97   Safari / 537 . 36 "
```

- 正規表現

```
\[([^]]+)]\ s \[(\ w +)]\ s (\ w +)\ s -\ s (\ S +)\ s -\ s -\ s
 "([^"]+)"\ s (\ d +)[^"]+("[^"]+)"\ s "([^"]+). *
```

- 抽出フィールド

`time` 、 `level` 、 `loggerName` 、 `ip` 、 `request` 、 `status` 、 `referer` および `user_agent`

5.4.6 Wordpress ログ

WordPress のログフォーマット (デフォルト)

ログ (未加工) のサンプル

```
172 . 64 . 0 . 2  - - [ 07 / Jan / 2016 : 21 : 06 : 39  + 0800 ]
" GET  / wp - admin / js / password - strength - meter . min . js
?  ver = 4 . 4   HTTP / 1 . 0 "  200   776  " http :// wordpress .
c4a1a0aecd  b194316955  5231dcc4ad  fb7 . cn - hangzhou . alicontain
er . com / wp - admin / install . php " " Mozilla / 5 . 0  (
Macintosh ;  Intel   Mac   OS   X   10_10_5 )  AppleWebKi  t / 537 .
36  ( KHTML ,  like   Gecko )  Chrome / 47 . 0 . 2526 . 106   Safari
/ 537 . 36 "
```

複数行ログの行の先頭の正規表現 (行の始まりは IP アドレス):

```
\ d +\.\ d +\.\ d +\.\ d +\ s -\ s . *
```

ログ情報を抽出する正規表現:

```
(\ S +) - - \[([^\]]*)] "(\ S +) ([^"]+)" (\ S +) (\ S +) "([^"]+)"
 "([^"]+)"
```

時間の変換書式:

```
% d /% b /% Y :% H :% M :% S
```

ログサンプルの抽出結果

キー	値
ip	10.10.10.1
time	07/Jan/2016:21:06:39 +0800
method	GET
url	/wp-admin/js/password-strength-meter.min.js? ver=4.4 HTTP/1.0

キー	値
status	200
length	776
ref	http://wordpress.c4a1a0aecdb1943169555231dcc4adfb7.cn-hangzhou.alicontainer.com/wp-admin/install.php
user-agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36

5.4.7 区切り文字ログ

概要

区切り文字ログは、各ログが改行で区切られています。1 行 1 ログになります。各ログのフィールドは、タブ、スペース、縦線 (|)、カンマ (,)、セミコロン (;)、といった定義されている区切り文字 (単一文字) で接続されます。フィールドのデータに区切り文字を使用する場合は、その文字を二重引用符 (") で囲みます。

一般的な区切り文字ログには、CSV ログ (コンマ区切り) および TSV ログ (タブ区切り) があります。

ログフォーマット

区切り文字ログは、区切り文字で各フィールドは分割されます。なお、単一文字および文字列の 2 つのモードがあります。

- 単一文字モード

  単一文字モードでは、タブ、スペース、縦線 (|)、コンマ (,)、セミコロン (;) といった定義された単一の文字により、ログの各フィールドは分割されます。

  
  注 :
  二重引用符(") を区切り文字にすることはできません (単一区切り文字のエスケープ文字のため)。

  ログフィールドの値に区切り文字が含まれることはよくあります。ログフィールドを二重引用符"で括ることで、ログフィールドが不適切に分割されないようにします。また、フィールドの値に二重引用符"が含まれる場合には、""と記述してエスケープします。二重引用符("")をフィールドの括りに使用することも、フィールドの値に二重引用符を使用することもできます。それ以外の場合は、区切り文字ログモードではなく、シンプルモードまたは完全モードで

フィールドが解析されるようにします。区切り文字ログのログフォーマット定義では対応できません。

- フィールドを括るために二重引用符 (") を使用

  区切り文字を含むフィールドは、二重引用符 (") で括る必要があります。二重引用符は、区切り文字の隣に記述します。二重引用符と区切り文字の間には、スペース、タブ、およびその他の文字が含まれないようにします。

  区切り文字にコンマ (,)、フィールドを括るために二重引用符 (") を使用するとします。ログフォーマットが `1997 , Ford , E350ˌ" ac , abs , moon ", 3000 . 00` である場合、ログは、`1997`、`Ford`、`E350`、`ac , abs , moon`、および `3000 . 00` の 5 つのフィールドに解析されます。二重引用符で囲まれた `ac , abs , moon` は 1 つのフィールドとみなされます。

- ログフィールドの値に二重引用符 ("") を使用

  フィールドを括るためにではなく、ログフィールドの値に二重引用符 ("") が含まれる場合は、二重引用符"を""と記述してエスケープする必要があります。フィールドが構文解析される際、""は"に復元されます。

  区切り文字にコンマ (,) を使用し、ログフィールドの値にコンマ (,) および二重引用符 ("") が含まれているとします。フィールドの値のコンマは二重引用符で囲みます。また、二重引用符 (") は""と記述してエスケープします。ログ `1999 , Chevy ," Venture "" Extended Edition , Very Large """,""", 5000 . 00` は、`1999`、`Chevy`、`Venture " Extended Edition , Very Large "`、空白フィールド、および `5000 . 00` の 5 つのフィールドに解析されます。

・ 文字列モード

文字列モードでは、||、&&&、^ _ ^といった 2 ～ 3 文字を区切り文字に指定します。文字列区切りのログが解析されます。各ログを二重引用符で括る必要はありません。

> 注：
> ログフィールドの値に、区切り文字列が含まれないようにします。フィールドの値に区切り文字列が含まれる場合、ログの各フィールドは適切に分割されません。

区切り文字列が&&のログ `1997 && Ford && E350 && ac & abs & moon && 3000 . 00` は、`1997`、`Ford`、`E350`、`ac & abs & moon`、および `3000 . 00` の 5 つのフィールドに解析されます。

ログサンプル

- 単一文字区切りのログ

```
05 / May / 2016 : 13 : 30 : 28 , 10 . 10 . 10 . 1 ," POST
 / PutData ?  Category = YunOsAccou  ntOpLog & AccessKeyI
d = U0Ujpek **********& Date = Fri % 2C % 2028 % 20Jun %
202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature
=******************************  HTTP / 1 . 1 ", 200 , 18204 ,
aliyun - sdk - java
05 / May / 2016 : 13 : 31 : 23 , 10 . 10 . 10 . 2 ," POST
 / PutData ?  Category = YunOsAccou  ntOpLog & AccessKeyI
d = U0Ujpek **********& Date = Fri % 2C % 2028 % 20Jun %
202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature
=******************************  HTTP / 1 . 1 ", 401 , 23472 ,
aliyun - sdk - java
```

- 文字列区切りのログ

```
05 / May / 2016 : 13 : 30 : 28 && 10 . 200 . 98 . 220 && POST
 / PutData ?  Category = YunOsAccou  ntOpLog & AccessKeyI
d = U0Ujpek **********& Date = Fri % 2C % 2028 % 20Jun %
202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature
=******************************  HTTP / 1 . 1 && 200 && 18204 &&
aliyun - sdk - java
05 / May / 2016 : 13 : 31 : 23 && 10 . 200 . 98 . 221 && POST
 / PutData ?  Category = YunOsAccou  ntOpLog & AccessKeyI
d = U0Ujpek **********& Date = Fri % 2C % 2028 % 20Jun %
202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature
=******************************  HTTP / 1 . 1 && 401 && 23472 &&
aliyun - sdk - java
```

区切り文字ログを収集するように Logtail を設定

Logtail を使用して区切り文字ログを収集する手順については、「Python ログ」をご参照ください。ネットワーク構成およびネットワーク設定に合わせて構成を選択します。

1. Logstoreリストページで、データインポートウィザードのアイコンをクリックします。
2. データソースを選択します。

   テキストファイルを選択し、次へをクリックします。

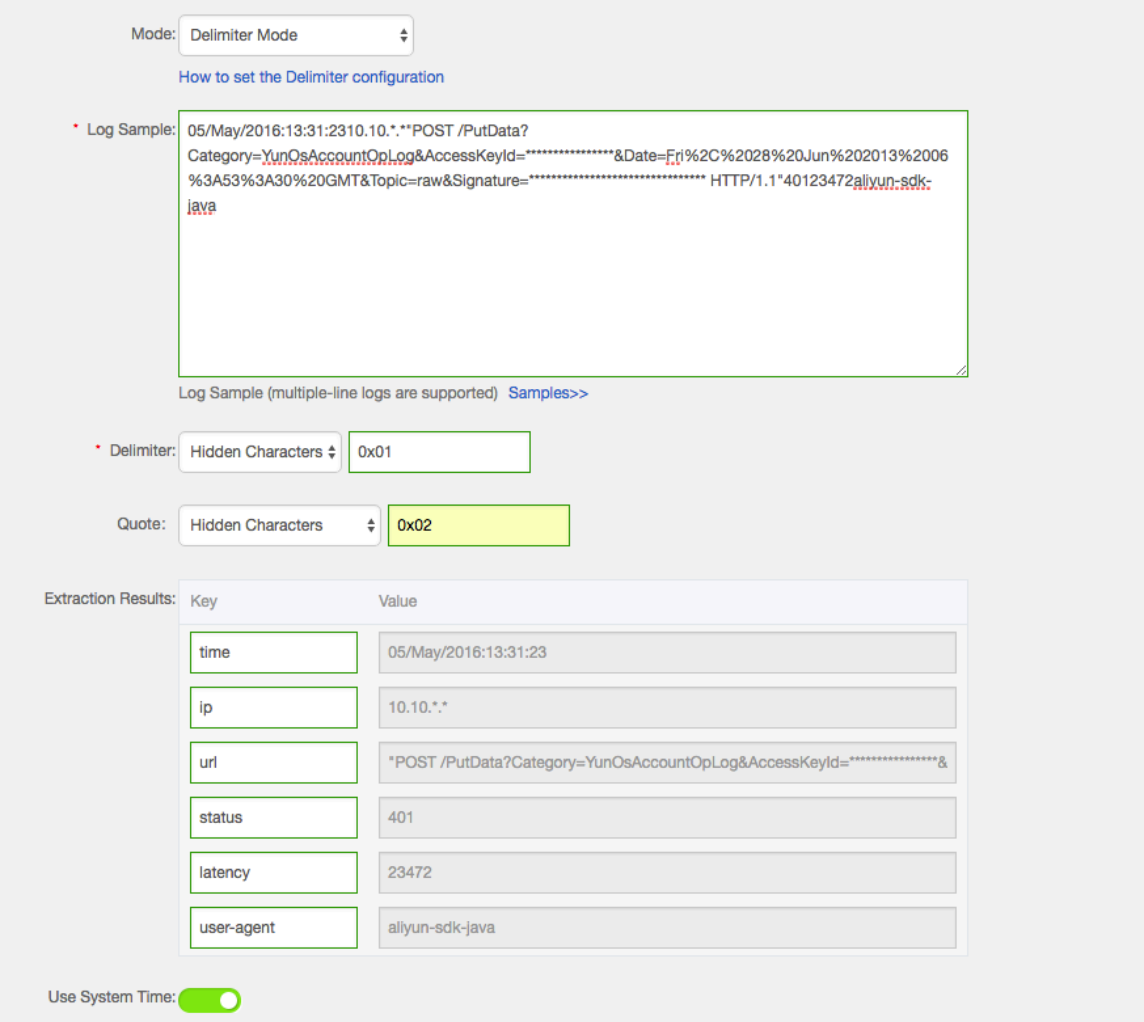
3. データソースを設定します。

   a. 構成名およびログパスを入力します。また、ログ収集モードには区切り文字モードを選択します。

   b. ログサンプルを入力し、区切り文字を選択します。

      ログフォーマットと同じ区切り文字を選択します。ログフォーマットと異なる区切り文字が選択された場合、ログの解析に失敗します。

      図 5-1 : データソースを選択



   c. ログ抽出フィールドにキーを割り当てます。

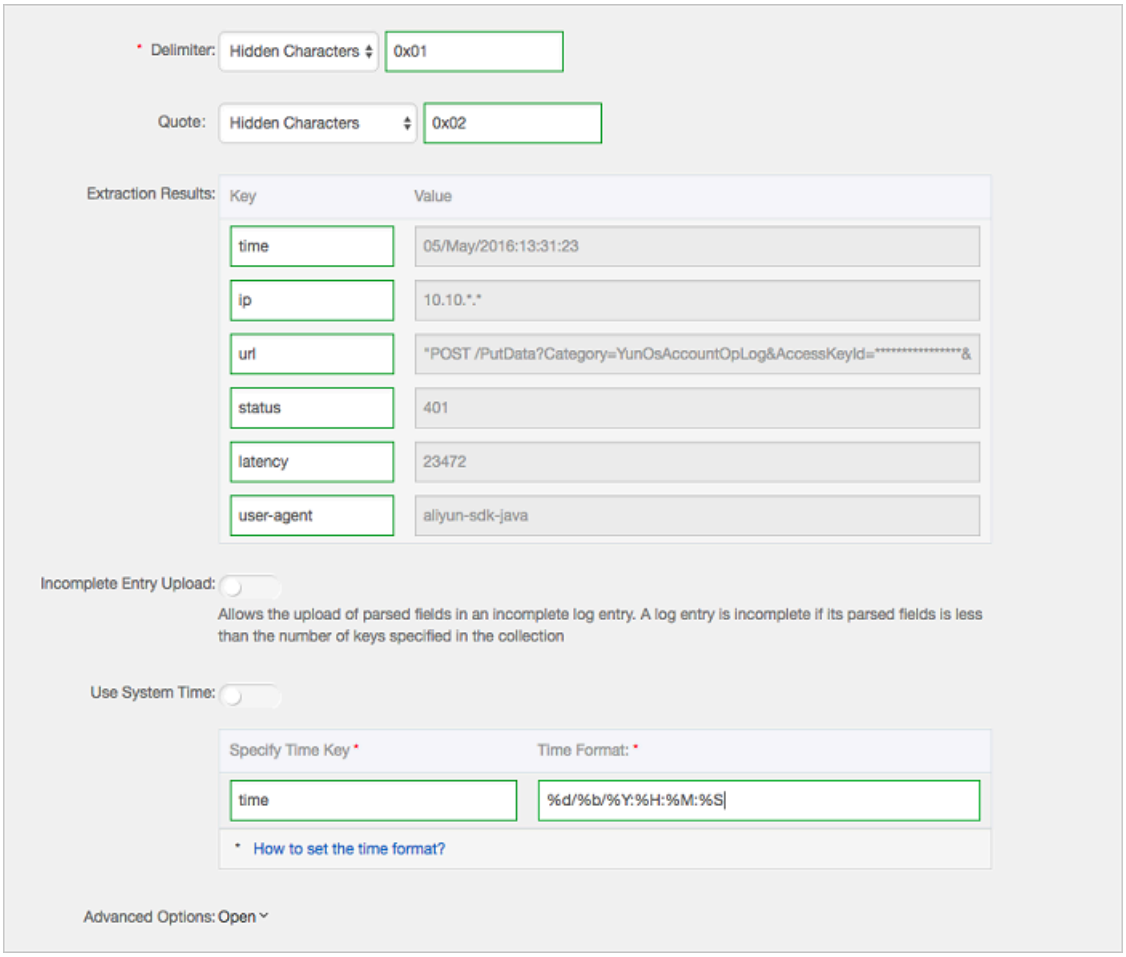
      ログサンプルを入力し、区切り文字を選択したら、選択した区切り文字に基づいてログの各フィールドが抽出されます。抽出された各フィールドは値です。値のキーを指定します。

      上記のログサンプルでは、区切り文字にコンマ (,) が用いられ、6 つのフィールドがログに記録されています。各フィールドのキーをそれぞれ time、ip、url、status、latency、user-agent に指定します。

d. ログ時間を指定します。

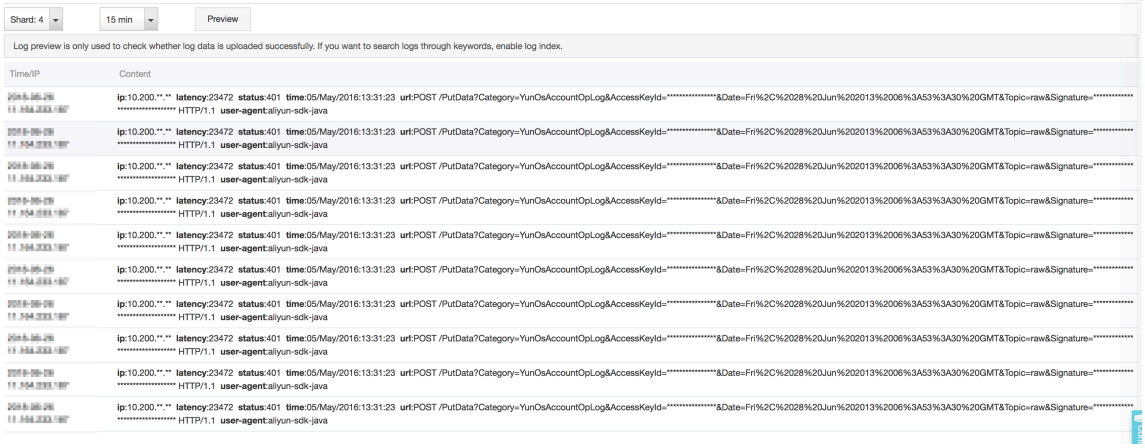
ログ時間には、システム時間またはログのフィールド (例: time フィールド 「05/May/2016:13:30:29」) のいずれかを指定します。時間の書式を設定する方法については、テキストログー時間書式を設定をご参照ください。

図 5-2 : ログ時間の指定



e. コンソールにログをプレビュー表示し、ログが正常に収集されているかどうかを確認します。

図 5-3 : ログのプレビュー



5.4.8 JSON ログ

JSON ログは、次の 2 種類の構造体にします。

- オブジェクト: キーと値のペアの集合
- 配列: 値のリスト

Logtail は、Object 型の JSON ログに対応しています。オブジェクトの第 1 階層からキーおよび値が抽出され、フィールド名および値になります。フィールド値には、オブジェクト型、配列型、および、文字列型や数値型といった単純なデータ型が入ることができます。また、JSON ログは、`\ n` で行は区切られます。抽出される各行が、1 つのログになります。

なお、JSON 配列といった Object 型でないデータは自動解析されません。正規表現でフィールドを抽出し、また、シンプルモードで行ごとにログを収集します。

ログサンプル

```
{" url ": " POST  / PutData ?  Category = YunOsAccou  ntOpLog &
 AccessKeyI  d =< yourAccess  KeyId >& Date = Fri % 2C % 2028 % 20Jun
 % 202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature =<
 yourSignat  ure >  HTTP / 1 . 1 ", " ip ": " 10 . 200 . 98 . 220 ", "
 user - agent ": " aliyun - sdk - java ", " request ": {" status ": "
 200 ", " latency ": " 18204 "}, " time ": " 05 / May / 2016 : 13 : 30
 : 28 "}
{" url ": " POST  / PutData ?  Category = YunOsAccou  ntOpLog &
 AccessKeyI  d =< yourAccess  KeyId >& Date = Fri % 2C % 2028 % 20Jun
 % 202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature =<
 yourSignat  ure >  HTTP / 1 . 1 ", " ip ": " 10 . 200 . 98 . 210 ", "
 user - agent ": " aliyun - sdk - java ", " request ": {" status ": "
```

```
200 ", " latency ": " 10204 "}, " time ": " 05 / May / 2016 : 13 : 30
: 29 "}
```

Logtail を使用して JSON ログを収集

Logtail を使用して JSON ログを収集する手順については、「5 分でクイックスタート」をご参照ください。 本ドキュメントでは、Logtail のLog 収集モードを設定する方法について説明します。

1. Logstore リストのデータインポートウィザードをクリックします。
2. データの型を選択します。

   テキストファイルを選択し、次へをクリックします。
3. データソースを設定します。

   a. 構成名およびログパスを入力し、ログ収集モードにJSON モードを選択します。

   b. ログ時間にシステム時間を使用するかどうかを選択します。システム時間を使用を有効または無効にします。

   - システム時間を使用を有効にする場合

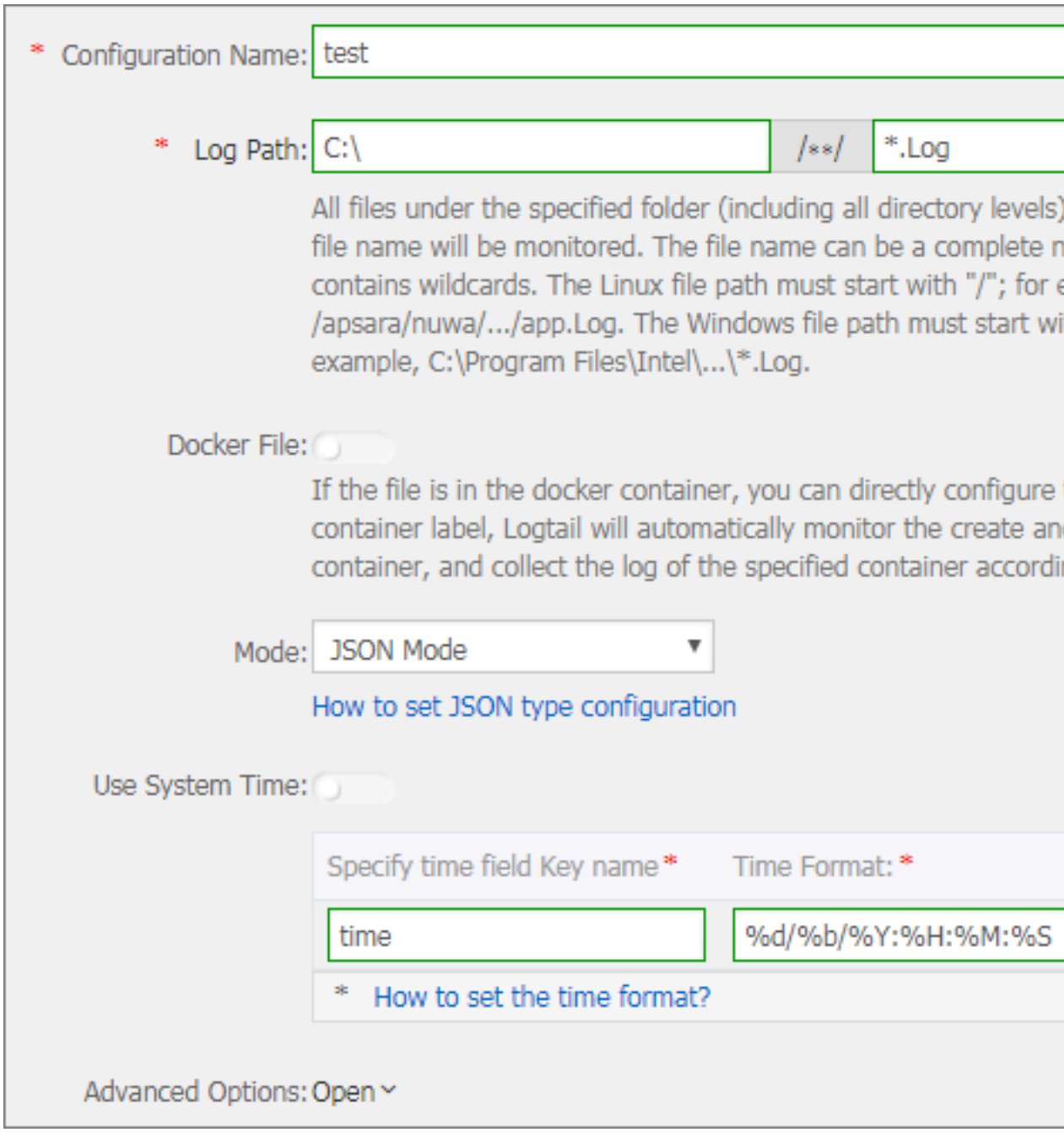
     システム時間を使用すると、ログ時間は、ログ内の time フィールドではなく、Log Service のログを収集した時間が採用されます。
   - システム時間を使用を無効にする場合

     システム時間を使用しない場合、ログ内の time フィールドがログ時間になります。

     システム時間を使用を無効にする場合は、time フィールドのキーおよび時間変換形式を定義する必要があります。たとえば、JSON オブジェクトの `time` フィールド「05/

May/2016:13:30:29」を抽出して、ログ時間にすることができます。ログ時間の書式設定については、テキストログー時間書式の設定をご参照ください。

図 5-4 : JSONログ



5.4.9 ThinkPHP ログ

ThinkPhP は PHP 言語に基づいた、Web アプリケーションの開発フレームワークです。

ログフォーマット

ThinkPhP では、以下のロギング方法が使用されています。

```
<?  php
 Think \ Log :: record (' D   method   instantiat  ion   does   not
 find   the   model   class ' );
```

ログサンプル

```
[  2016 - 05 - 11T21 : 03 : 05 + 08 : 00  ]  10 . 10 . 10 . 1  / index
 . php
 INFO : [  app_init  ] -- START --
 INFO :  Run   Behavior \ BuildLiteB  ehavior  [  RunTime : 0 .
 000014s  ]
 INFO : [  app_init  ] -- END -- [  RunTime : 0 . 000091s  ]
 Info : [ app_begin ] --  start  --
 INFO :  Run   Behavior \ ReadHtmlCa  cheBehavio  r  [  RunTime : 0 .
 000038s  ]
 INFO : [  app_begin  ] -- END -- [  RunTime : 0 . 000076s  ]
 INFO : [  view_parse  ] -- START --
 INFO :  Run   Behavior \ ParseTempl  ateBehavio  r  [  RunTime : 0 .
 000068s  ]
 INFO : [  view_parse  ] -- END -- [  RunTime : 0 . 000104s  ]
 INFO : [  view_filte  r  ] -- START --
 INFO :  Run   Behavior \ WriteHtmlC  acheBehavi  or  [  RunTime : 0 .
 000032s  ]
 INFO : [  view_filte  r  ] -- END -- [  RunTime : 0 . 000062s  ]
 INFO : [  app_end  ] -- START --
 INFO :  Run   Behavior \ ShowPageTr  aceBehavio  r  [  RunTime : 0 .
 000032s  ]
 INFO : [  app_end  ] -- END -- [  RunTime : 0 . 000070s  ]
 ERR :  D   method   instantiat  ion   does   not   find   the   model
   class
```

Logtail を使用して ThinkPhP ログを収集

Logtail を使用して ThinkPhP ログを収集する手順については、「Python ログ」をご参照ください。ネットワーク構成およびネットワーク設定に合わせて構成を選択します。

自動生成される正規表現は、ログサンプルのみを基にしており、すべてのログフォーマットに対応しているわけではありません。したがって、正規表現が自動生成されたら、修正を加える必要があります。

ThinkPhP ログは、複数行に渡り、定まった形式がありません。ThinkPhP ログより、時間、ユーザーの IP アドレス、アクセス URL、表示メッセージといったフィールドを抽出できます。メッセージフィールドの情報は複数行に渡り、定型化されていないため、1 つのフィールドとして扱われます。

Logtail より ThinkPhP ログの設定パラメータを取得

行の先頭の正規表現:

```
\[\ s \ d +-\ d +-\ w +:\ d +:\ d +\+\ d +:\ d +\ s .
```

正規表現:

```
\[\ s (\ d +-\ d +-\ w +:\ d +:\ d +)[^:]+:\ d +\ s ]\ s +(\ S +)\ s
 (\ S +)\ s +(.
```

時間書式:

```
% Y -% m -% dT % H :% M :% S
```

5.4.10 Logstash を使用した IIS ログの収集

Logstash を使用して IIS ログを収集する前に、IIS ログフィールドが解析されるよう、構成ファイルを変更する必要があります。

ログサンプル

IIS ログ設定を表示し、W3C 形式 (デフォルトのフィールド設定) を選択し、フォーマットを保存して有効にします。

```
 2016 - 02 - 25   01 : 27 : 04   112 . 74 . 74 . 124   GET  / goods /
 list / 0 / 1 . html  -  80  -  66 . 249 . 65 . 102   Mozilla / 5 . 0
 +( compatible ;+ Googlebot / 2 . 1 ;++ http :// www . google . com /
 bot . html )  404   0   2   703
```

ログ収集の設定

```
 input  {
   file  {
     type  => " iis_log_1 "
     path  => [" C :/ inetpub / logs / LogFiles / W3SVC1 /*. log "]
     start_posi  tion  => " beginning "
  }
}
 filter  {
   if  [ type ] == " iis_log_1 " {
  # ignore   log   comments
   if  [ message ] =~ "^#" {
     drop  {}
  }
   grok  {
    #  check   that   fields   match   your   IIS   log   settings
     match  => [" message ", "%{ TIMESTAMP_  ISO8601 : log_timest
 amp } %{ IPORHOST : site } %{ WORD : method } %{ URIPATH : page } %{
 NOTSPACE : querystrin  g } %{ NUMBER : port } %{ NOTSPACE : username
 } %{ IPORHOST : clienthost } %{ NOTSPACE : useragent } %{ NUMBER
 : response } %{ NUMBER : subrespons  e } %{ NUMBER : scstatus } %{
 NUMBER : time_taken }"]
  }
     date  {
```

```
      match  => [ " log_timest  amp ", " YYYY - MM - dd   HH : mm : ss
 " ]
        timezone  => " Etc / UTC "
  }
   useragent  {
     source => " useragent "
     prefix => " browser "
  }
   mutate  {
     remove_fie  ld  => [ " log_timest  amp "]
  }
  }
}
 output  {
   if  [ type ] == " iis_log_1 " {
   logservice  {
         codec  => " json "
         endpoint  => "***"
         project  => "***"
         logstore  => "***"
         topic  => ""
         source  => ""
         access_key  _id  => "***"
         access_key  _secret  => "***"
         max_send_r  etry  =>  10
    }
    }
}
```

注：

- 構成ファイルは、BOM なしの UTF-8 でエンコードする必要があります。Notepad++ をダウンロードして、ファイルのエンコード形式を変更します。
- *path* にはファイルパスを指定します。*C :/ test / multiline /*. log* のように Unix 形式のファイル区切り文字を使用します。Unix 形式以外のファイル区切り文字を使用すると、ファジー一致を使用できません。
- *type* フィールドは統一させ、ファイル全体で一貫性が保たれている必要があります。また、マシンに Logstash 構成ファイルが複数ある場合、各構成ファイルの type フィールドは統一させます。統一されていない場合、データは正しく処理されません。

関連プラグイン: file および grok

Logstash を再起動して設定を適用

conf ディレクトリに構成ファイルを作成します。Logstash を再起動すると構成ファイルが適用されます。詳細は Logstash を Windows サービスに登録 をご参照ください。

5.4.11 Logstash を使用した CSV ログの収集

Logstash を使用して CSV ログを取得する前に、CSV ログフィールドが解析されるよう、構成ファイルを修正する必要があります。 CSV ログの収集は、ログを収集した時点のシステム時間と

ログ内に記載される時間をログをアップロードする時間として使用できます。ログ時間の各定義方法においては、CSV ログを収集するための Logstash の構成方法が 2 種類あります。

ログのアップロード時間をシステム時間に

- ログサンプル

```
10 . 116 . 14 . 201 ,-, 2 / 25 / 2016 , 11 : 53 : 17 , W3SVC7
, 2132 , 200 , 0 , GET , project / shenzhen - test / logstore /
logstash / detail , C :\ test \ csv \ test_csv . log
```

- ログ収集の設定

```
 input  {
   file  {
     type  => " csv_log_1 "
     path  => [" C :/ test / csv /*. log "]
     start_posi  tion  => " beginning "
  }
}
 filter  {
   if  [ type ] == " csv_log_1 " {
   csv  {
     separator  => ","
     columns  => [" ip ", " a ", " date ", " time ", " b ", "
 latency ", " status ", " size ", " method ", " url ", " file "]
  }
  }
}
 output  {
   if  [ type ] == " csv_log_1 " {
   logservice  {
         codec  => " json "
         endpoint  => "***"
         project  => "***"
         logstore  => "***"
         topic  => ""
         source  => ""
         access_key  _id  => "***"
         access_key  _secret  => "***"
         max_send_r  etry  =>  10
    }
    }
}
```

注：

- 構成ファイルは、BOM なしの UTF-8 でエンコードする必要があります。Notepad++ をダウンロードして、ファイルのエンコード形式を変更します。
- *path* にはファイルパスを指定します。*C :/ test / multiline /*. log* のように Unix 形式のファイル区切り文字を使用します。Unix 形式以外のファイル区切り文字を使用すると、ファジー一致を使用できません。

- *type* フィールドは統一させ、ファイル全体で一貫性が保たれている必要があります。また、マシンに Logstash 構成ファイルが複数ある場合、各構成ファイルの type フィールドは統一させます。統一されていない場合、データは正しく処理されません。

関連するプラグイン：fileおよびcsv。

- Logstash を再起動して設定を適用

conf ディレクトリに構成ファイルを作成し、Logstashを再起動してファイルを適用します。 詳細については、「Logstash を Windows サービスに登録」をご参照ください。

ログフィールドの内容をアップロードされたログ時刻として使用する

- ログサンプル

```
10 . 116 . 14 . 201 ,-, Feb   25   2016   14 : 03 : 44 , W3SVC7
, 1332 , 200 , 0 , GET , project / shenzhen - test / logstore /
logstash / detail , C :\ test \ csv \ test_csv_w  ithtime . log
```

- 収集の設定

```
input  {
  file  {
    type  => " csv_log_2 "
    path  => [" C :/ test / csv_withti  me /*. log "]
    start_posi  tion  => " beginning "
 }
}
filter  {
  if  [ type ] == " csv_log_2 " {
  csv  {
    separator  => ","
    columns  => [" ip ", " a ", " datetime ", " b ", " latency ", "
status ", " size ", " method ", " url ", " file "]
 }
  date  {
    match  => [ " datetime " , " MMM   dd   YYYY   HH : mm : ss " ]
 }
 }
}
output  {
  if  [ type ] == " csv_log_2 " {
  logservice  {
        codec  => " json "
        endpoint  => "***"
        project  => "***"
        logstore  => "***"
        topic  => ""
        source  => ""
        access_key  _id  => "***"
        access_key  _secret  => "***"
        max_send_r  etry  =>  10
   }
   }
```

```
}
```

注：

- 構成ファイルは、BOM なしの UTF-8 でエンコードする必要があります。Notepad++ をダウンロードして、ファイルのエンコード形式を変更します。
- *path* にはファイルパスを指定します。 *C :/ test / multiline /*. log* のように Unix 形式のファイル区切り文字を使用します。Unix 形式以外のファイル区切り文字を使用すると、ファジー一致を使用できません。
- *type* フィールドは統一させ、ファイル全体で一貫性が保たれている必要があります。また、マシンに Logstash 構成ファイルが複数ある場合、各構成ファイルの type フィールドは統一させます。統一されていない場合、データは正しく処理されません。

関連プラグイン: fileおよびcsv

- Logstash の再起動で設定を適用

conf ディレクトリに構成ファイルを作成し、Logstash を再起動すると、構成ファイルが適用されます。 詳細については、「Logstash を Windows サービスに登録をご参照ください。

5.4.12 Logstash を使用して他のログを収集

Logstash を使用してログを収集する前に、ログフィールドが解析されるよう、構成ファイルを修正する必要があります。

ログのアップロード時間をシステム時間に

- ログサンプル

```
2016 - 02 - 25   15 : 37 : 01  [ main ]  INFO   com . aliyun . sls
. test_log4j  -  single   line   log
2016 - 02 - 25   15 : 37 : 11  [ main ]  ERROR   com . aliyun . sls
. test_log4j  -  catch   exception  !
 java . lang . Arithmetic  Exception : /  by   zero
    at   com . aliyun . sls . test_log4j . divide ( test_log4j .
java : 23 ) ~[ bin /:?]
    at   com . aliyun . sls . test_log4j . main ( test_log4j . java
: 13 ) [ bin /:?]
2016 - 02 - 25   15 : 38 : 02  [ main ]  INFO   com . aliyun . sls
. test_log4j  -  normal   log
```

- ログ収集の設定

```
input  {
  file  {
    type  => " common_log  _1 "
    path  => [" C :/ test / multiline /*. log "]
    start_posi  tion  => " beginning "
    codec  =>  multiline  {
```

```
        pattern  => "^\ d { 4 }-\ d { 2 }-\ d { 2 } \ d { 2 }:\ d { 2
 }:\ d { 2 }"
        negate  =>  true
        auto_flush  _interval  =>  3
        what  =>  previous
     }
   }
}
 output  {
   if  [ type ] == " common_log  _1 " {
   logservice  {
         codec  => " json "
         endpoint  => "***"
         project  => "***"
         logstore  => "***"
         topic  => ""
         source  => ""
         access_key  _id  => "***"
         access_key  _secret  => "***"
         max_send_r  etry  =>  10
     }
     }
}
```

注：

- 構成ファイルは、BOM なしの UTF-8 でエンコードする必要があります。Notepad++ をダウンロードして、ファイルのエンコード形式を変更します。
- *path* にはファイルパスを指定します。*C :/ test / multiline /*. log* のように Unix 形式のファイル区切り文字を使用します。Unix 形式以外のファイル区切り文字を使用すると、ファジー一致を使用できません。
- *type* フィールドは統一させ、ファイル全体で一貫性が保たれている必要があります。また、マシンに Logstash 構成ファイルが複数ある場合、各構成ファイルの type フィールドは統一させます。統一されていない場合、データは正しく処理されません。

関連プラグイン: fileおよびmultiline（ログファイルが 1 行のみの場合、codec => multiline 設定を削除してください）

- Logstash を再起動して設定を適用

  conf ディレクトリに構成ファイルを作成し、Logstash を再起動してファイルを適用します。詳細については、「Logstash を Windows サービスに登録」をご参照ください。

5.4.13 Unity3D

Unity3D は、あらゆるプラットフォーム向けの統合されたゲーム開発ツールです。Unity Technologies の開発した Unity3D では、3D ビデオゲーム、建築的視覚化、リアルタイム 3D アニメーションといったさまざまなインタラクティブコンテンツを簡単に作成できます。Unity3D は完全に統合されたプロフェッショナルなゲームエンジンです。

Log Service の Web トラッキング を使用することにより、Unity3D のログを簡単に収集できます。本ドキュメントでは、Webトラッキング機能を使用して Unity3D のログ *Unity Debug . Log* を Log Service に収集する方法を紹介します。

1. Web トラッキング機能の有効化

   詳細については、「Web トラッキング」をご参照ください。

2. Unity3D LogHandler の登録

   Unity エディタで C# ファイル *LogOutputH andler . cs* を作成します。以下のコードを入力し、コード内の次の 3 つのメンバー変数を変更します。

   - プロジェクト: ログプロジェクトの名前
   - Logstore: Logstore の名前
   - serviceAddr: ログプロジェクトのアドレス

     詳細については、「サービスエンドポイント」をご参照ください。

```
using UnityEngin e ;
using System . Collection s ;
public class LogOutputH andler : MonoBehavi our
{
    // Register the HandleLog function on scene start
to fire on debug . log events
    public void OnEnable ()
    {
        Applicatio n . logMessage Received += HandleLog ;
    }
    // Remove callback when object goes out of scope
    public void OnDisable ()
    {
        Applicatio n . logMessage Received -= HandleLog ;
    }
    string project = " your project name ";
    string logstore = " your logstore name ";
    string serviceAdd r = " http address of your log
service project ";
    // Capture debug . log output , send logs to Loggly
    public void HandleLog ( string logString , string
stackTrace , LogType type )
    {
        string parameters = "";
        parameters += " Level =" + WWW . EscapeURL ( type .
ToString ());
        parameters += "&";
        parameters += " Message =" + WWW . EscapeURL ( logString
);
        parameters += "&";
        parameters += " Stack_Trac e =" + WWW . EscapeURL (
stackTrace );
        parameters += "&";
       // Add any User , Game , or Device MetaData that
  would be useful to finding issues later
        parameters += " Device_Mod el =" + WWW . EscapeURL (
SystemInfo . deviceMode l );
```

```
        string   url  = " http ://" +  project  + "." +
 serviceAdd  r  + "/ logstores /" +  logstore  + "/ track ?
 APIVersion = 0 . 6 . 0 &" +  parameters ;
        StartCorou  tine ( SendData ( url ));
    }
     public   IEnumerato  r   SendData ( string   url )
    {
        WWW   sendLog  =  new   WWW ( url );
        yield   return   sendLog ;
    }
}
```

上記のコードにより、非同期に Log Service にログを送信することができます。収集するフィールドを上記コードに追加することもできます。

3. Unity ログの生成

   プロジェクトで、 *LogglyTest . cs* ファイルを作成し、 次のコードを追加してください。

```
 using   UnityEngin  e ;
 using   System . Collection  s ;
 public   class   LogglyTest  :  MonoBehavi  our  {
     void   Start  () {
        Debug . Log  (" Hello   world ");
    }
}
```

4. コンソールでログのプレビュー

   上記の手順を完了したら、Unity プログラムを実行します。送信されたログを Log Service コンソールでプレビューできます。

   上記の例では、 *Debug . Log* 、 *Debug . LogError* 、 *Debug . LogExcepti on* といったログを収集する方法です。Unity のコンポーネントオブジェクトモデル、Unity プログラムのクラッシュ API、およびその他のログ API を使用することで、クライアント上のデバイス情報を容易に収集できます。

6 インデックスとクエリ

6.1 インデックスの作成

Log Service は、大量のログをリアルタイムで照会および分析するための LogSearch/Analytics 機能を提供します。 この機能を使用するには、索引とフィールドの統計を使用可能にします。

機能上の利点

- リアルタイム：ログは、書かれた直後に分析することができます。
- 高速：
  - クエリ：数十億のデータを処理し、1 秒以内にクエリできます（条件は 5 つ）。
  - 分析：数百万のデータを 1 秒以内に集約して分析することができます（5 つのディメンションと GroupBy 条件による集計を使用）。
- フレキシブル：リアルタイムで結果を得るために、必要に応じてクエリおよび分析条件を変更できます。
- 生態学：そのようなコンソールで提供されるレポート、ダッシュボード、および迅速な分析などの機能のほかに、 Log Service シームレスな Grafana、などの製品との相互接続 DataV 、そしてイェーガー、そして、そのような RESTful な API や JDBC などのプロトコルをサポートしています。

基本概念

LogSearch/Analytics （インデックス）機能を有効にしなければ、生データはシャード内のシーケンスに従って消費されます。 これは Kafka に似ています。LogSearch/Analytics （インデックス）機能を有効にすると、順番に消費するだけでなく、ログを数えてクエリすることもできます。 ログ消費とログクエリーの違いについては、ログ使用とログ検索の違いをご参照ください。

インデックスを有効にする

1. Log Service コンソールにログインします。 プロダクトページでプロジェクト名をクリックします。

2. Logstore を選択し、 検索をクリックします。 次に、右上隅のインデックスを有効にするをクリックします。 以前にインデックスを有効にしている場合は、インデックス属性 > 変更をクリックします。

   ・ クエリと統計情報を有効にした後、データはバックエンドで索引付けされます。 索引のトラフィックと格納領域が必要です。

   ・ この機能が必要ない場合は、 インデックス属性 > 無効化をクリックして無効にします。

3. 設定を完了するには、設定メニューに入ります。

データ型

ログ内の各キーの型を設定できます（フルテキストインデックスは特別なキーで、その値はログです）。 現在、 Log Service は次のデータ型をサポートしています。

カテゴリー	型	説明	クエリの例
Basic	TEXT	キーワードとファジーマッチをサポートするテキスト型。	`uri :" login *"  method :" post "`
Basic	Long	インターバルクエリをサポートする型の値。	`status > 200 ,  status in  [ 200 ,  500 ]`
Basic	Double	float 型の値です。	`price > 28 . 95 ,  t  in [ 20 . 0 ,  37 ]`
Combination	JSON	コンテンツはデフォルトでテキスト型 JSON フィールドで、ネストされたモデルをサポートします。 ab のようなパスフォーマットを使用して a の下の b 要素のテキスト、long 型、double 型のインデックスを設定することができます。 設定後の項目型は設定の対象となります。	`level0 . key > 29 . 95 level0 . key2 :" action "`
Combination	Full text	クエリのテキストとしてログを使用します。	`error  and  " login fail "`

クエリと分析の構文

リアルタイムのクエリと分析は、検索と分析で構成され、それらは縦線(|)で区切られています。

```
$ Search  |$ Analytics
```

・ 検索：キーワード、ファジー一致条件、値、範囲、および組み合わせ条件を使用して生成されるクエリ条件。 検索が空またはアスタリスク（*）の場合、すべてのデータが照会されます。

・ 分析：クエリ結果または完全なデータを計算して数えます。

 注：

検索と分析の両方はオプションです。 検索が空の場合、指定した期間内のすべてのデータはフィルタリングされず、結果は直接カウントされます。 Analytics が空の場合、クエリ結果が返され、統計は収集されません。

 注：

詳細については、クエリ構文と分析文法を参照してください。

クエリの例

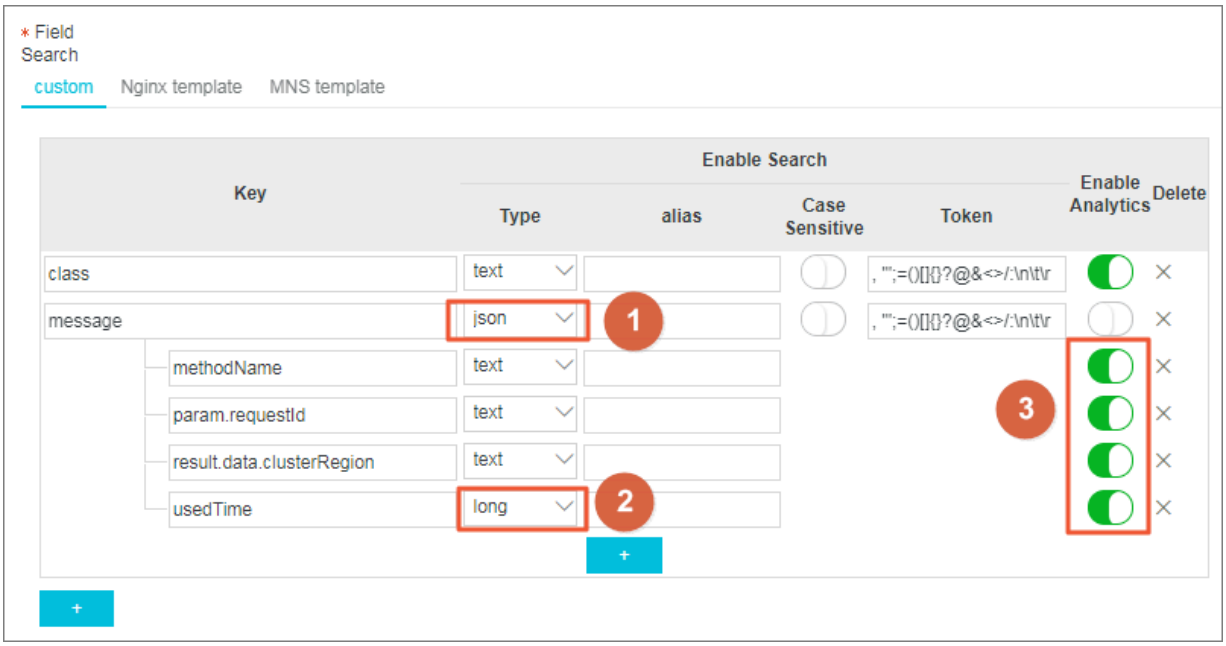
時間のほかに、次のログには 4 つのキー値も含まれています。

シーケンス番号	キー	型
0	time	-
1	class	text
2	status	Long
3	Latency	double
4	message	json

```
0 .  time : 2018 - 01 - 01   12 : 00 : 00
  1 .  class : central - log
  2 .  status : 200
  3 .  latency : 68 . 75
  4 .  message :

     " methodName ": " getProject  Info ",
     " success ":  true ,
     " remoteAddr  ess ": " 1 . 1 . 1 . 1 : 11111 ",
     " usedTime ":  48 ,
     " param ": {
           " projectNam  e ": " ali - log - test - project ",
           " requestId ": " d3f0c96a - 51b0 - 4166 - a850 -
f4175dde73  23 "

     " result ": {
         " message ": " successful ",
         " code ": " 200 ",
         " data ": {
             " clusterReg  ion ": " ap - southeast - 1 ",
             " ProjectNam  e ": " ali - log - test - project ",
             " CreateTime ": " 2017 - 06 - 08   20 : 22 : 41 "

         " success ":  true
     }
```

設定は次のとおりです。

図 6-1 : インデックス設定



場所：

- ① は、JSON フィールドの型とブール型すべてのデータを照会できることを示します。
- ② は、long 型データを照会できることを示します。
- ③ は、SQL 文を使用して構成済みのフィールドを分析できることを示します。

例 1：クエリ文字列、bool 型

```
class : cental *
message . traceInfo . requestId : 92 . 137_151813 9699935_55 99
message . param . projectNam e : ali - log - test - project
message . success : true
```

注：

- JSON フィールドの設定は必要ありません。
- JSON Map と Array は Auto Scaling あり、マルチレベルネストをサポートします。 各レイヤーはピリオド（.）で区切られています。

例 2： クエリ double、long 型

```
latency > 40
```

```
message . usedTime > 40
```

注：

JSON フィールドを個別に設定します。 フィールドは配列であってはなりません。

例 3： 複合クエリ

```
class : cental * and message . usedTime > 40 not message .
param . projectNam e : ali - log - test - project
```

その他の情報

大量のログデータ（100 億を超える長いクエリ時間など）を照会すると、1 つのリクエストすべてのデータを照会することができません。 この場合、 Log Service は既存のデータを返し、クエリ結果が不完全であることを通知します。

同時に、サーバーは 15 分以内に照会の結果を キャッシュします。 クエリ結果が部分的にキャッシュされると、サーバーはキャッシュされていないログデータをスキャンし続けます。 複数のクエリ結果をマージする際の負荷を軽減するため、 Log Service はキャッシュヒットの結果を新しいクエリの結果とマージして返します。

したがって、 Log Service 使用すると、同じパラメータを使用してインターフェイスを繰り返し呼び出すことで、最終的な結果を得ることができます。

6.2 分析文法

Log Service は、SQL 集計コンピューティングと同様の機能を提供します。 この関数は、クエリ関数と SQL 演算関数を統合してクエリ結果を計算します。

構文例：

```
status > 200 | select avg ( latency ), max ( latency ) , count ( 1
) as c GROUP BY method ORDER BY c DESC LIMIT
20
```

基本的な構文：

```
[ search query ] | [ sql query ]
```

SEARCH 条件と計算条件は垂直バー（ |）で区切られています。 この構文は、検索クエリを使用してログから必要なログをフィルタリングし、これらのログに対して SQL クエリの計算を実行することを示します。 検索クエリの構文は、 Log Service 固有のものです。 詳細は、クエリ構文を参照してください。

前提条件

分析機能を使用するには、 Search＆Analysis 設定の対応する SQL フィールドの Enable Analytics スイッチをオンにします。 詳細は、インデックスの作成 を参照してください。

- Enable Analytics スイッチがオフになっている場合、 Log Service はデフォルトで一部のデータに対してのみ計算機能を提供し、待ち時間は長くなります。
- Enable Analytics スイッチをオンにすると、 Log Service は数秒で迅速な分析を提供します。
- 機能が有効になっている場合にのみ、新しいデータに対して機能します。
- アナリティクスを有効にすると、追加料金は発生しません。

対応している SQL 構文

Log Service は、次の SQL 構文に対応しています。 詳細は、特定のリンクをクリックしてください。

- SELECT 集計関数：
  - 共通集計関数
  - セキュリティ検知関数
  - マッピング関数
  - 推定関数
  - 数学的統計関数
  - 数学計算関数
  - 文字列関数
  - 日付と時間の関数
  - URL 関数
  - 正規表現関数
  - JOSN 関数
  - 型変換関数
  - IP 機能
  - 配列
  - バイナリ文字列関数
  - ビット操作
  - 区間値比較および周期性値比較関数
  - 比較関数と演算子
  - Lambda 関数
  - 論理関数
  - 地理空間関数
  - Geo機能
- GROUP BY の構文
- ウィンドウ関数
- HAVING 構文
- ORDER BY 構文
- LIMIT 構文
- CASE WHEN 構文
- UNNEST 関数
- 列のエイリアス
- ネストされたサブクエリ

構文構造

SQL の構文構造は次のとおりです。

- FROM 句と WHERE 句は、SQL 文では必須ではありません。 デフォルトでは、FROM は現在の Logstore のデータを照会することを示し、WHERE 条件は検索照会です。
- サポートされる句には、SELECT、GROUP BY、ORDER BY [ASC、DESC]、LIMIT、および HAVING が含まれます。

注：

デフォルトでは、最初の 10 個の結果のみが返されます。 より多くの結果を返すには、 limit n を追加します。たとえば、`* | select count ( 1 ) as c , ip group by ip order by c desc limit 100` と選択します。

組み込みフィールド

Log Service は統計用のいくつかの組み込みフィールドがあります。 これらの組み込みフィールドは、有効な列を構成すると自動的に追加されます。

フィールド名	タイプ	説明
`__time__`	bigint	ログ時間。
`__source__`	varchar	ログのソース IP。 このフィールドはあなたが照会するとき source です。 下線（ __）は source の前後に SQL でのみ追加されます。
`__topic__`	varchar	ログトピック

制限

1. 各プロジェクトの最大同時実行数は15です。
2. 単一列の VARCHAR の最大長は 2048 で、長さが 2048 を超えると切り捨てられます。
3. デフォルトでは 100 行のデータが返され、ページめくりはサポートされていません。 より多くのデータを返すには、LIMIT 構文 を使用してください。

例

毎時の PV と UV、最高レイテンシのユーザー要求、および上位 10 の最高レイテンシに関する統計を作成します。

```
*| select   date_trunc (' hour ', from_unixt  ime ( __time__ ))  as
 time ,
      count ( 1 )  as   pv ,
```

```
approx_dis tinct ( userid ) as uv ,
max_by ( url , latency ) as top_latenc y_url ,
max ( latency , 10 ) as top_10_lat ency
group by 1
order by time
```

6.3 インデックスの有効化及び設定

Log Service の解析検索機能を使用する前に、ログのインデックスを有効化し、設定する必要があります。

そうしなければ、収集したログを検索できません。 ログフィールドと検索要件に基づいてインデックスを設定します。

 注：

- 解析検索機能が有効になった後、データはバックエンドサーバー上でインデックス付きされます。 そのため、インデックストラフィックが発生し、インデックスストレージスペースが必要になります。
- インデックス設定は、設定が有効または変更されてから、記録されたデータにのみ有効になります。
- フルテキストインデックスとキー/値インデックスのうち、少なくとも 1 つのインデックスをログに対して有効にする必要があります。
- SQL ステートメントを使用してフィールドの検索結果を分析するには、そのフィールドの分析機能を有効化する必要があります。
- インターネット IP アドレスや Unix タイムスタンプなどのタグフィールドにインデックスを設定する場合は、キー名を `__tag__:key` 形式の値に設定します。例：`_tag__:__receive_time__` 。 タグフィールドは数値型のインデックスをサポートしません。 その代わりに、すべてのタグフィールドのタイプをテキストに設定します。 例えば、キー名 `__tag__:__receive_time__` を持つフィールドを検索するには、`__tag__:__receive_time__:1537928*` などのファジー値、または `__tag__:__receive_time__:1537928404` などのフィールドの完全値を使用できます。

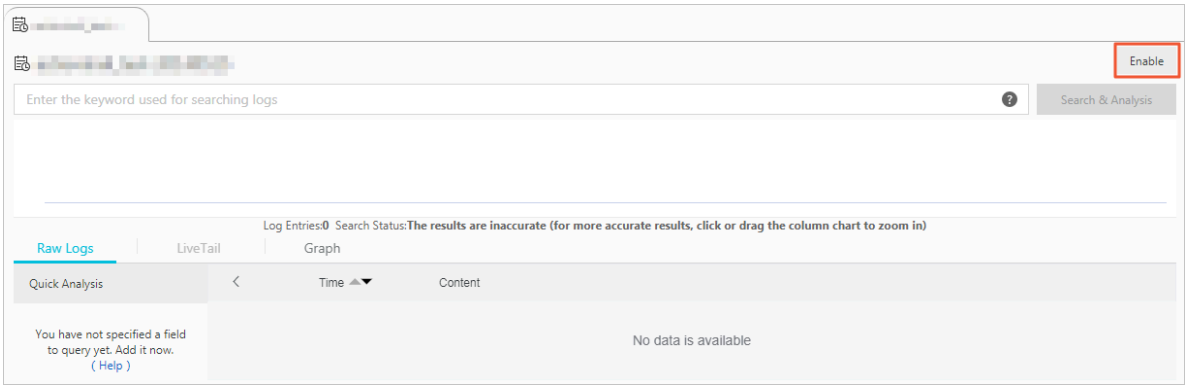
ログが収集されると、ログに関するソースや時間などの情報が、キーと値のペアとしてログに自動的に追加されます。 これらのフィールドは Log Service で予約されています。 ログのインデックスを有効化し、設定すると、これらのフィールドに対するインデックスと分析機能が自動的に有効になります。

> 注：
> `__topic__` および `__source__` フィールドの区切り文字はヌルです。 つまり、2 つのフィールドを検索するために使用されるキーワードはフィールド値と一致しなければなりません。

表 6-1 : Log Service の予約フィールド

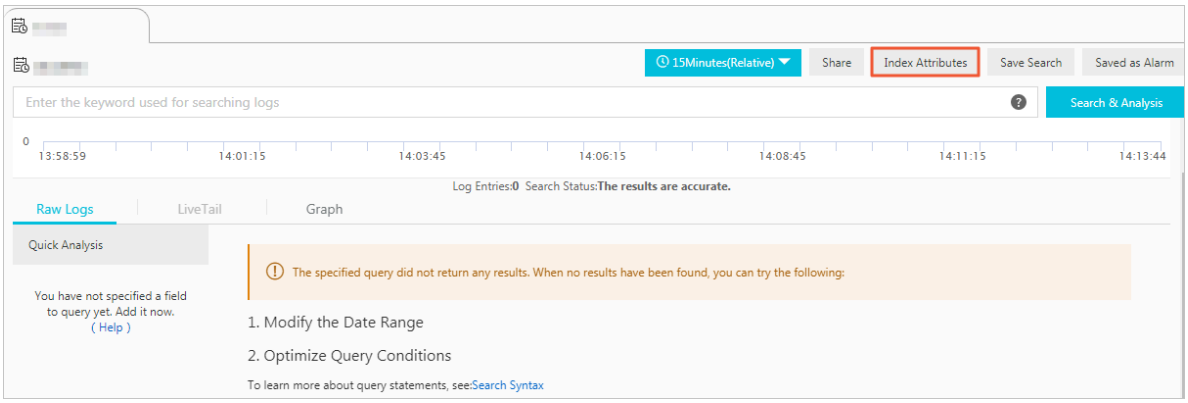
名前	説明
`__topic__`	ログのトピック。 ログにトピックを設定した場合、Log Service は自動的にトピックフィールドをログに追加します。フィールドのキーは `__topic__` であり、その値はログトピックです。
`__source__`	ログを生成するソース機器。
`__time__`	SDK を使用してログデータを書き込む際に指定した時間。

1. Log Service コンソールにログインし、プロジェクト名をクリックします。
2. 解析検索列で、検索をクリックします。
3. 右上隅の有効化をクリックします。



> 注：

作成済みのインデックスが存在する場合、インデックス属性 > 変更をクリックしてインデックスを変更することができます。



4. ログのインデックスを設定します。

Log Service は、フルテキストインデックスとキー/値インデックスの 2 種類のインデックスをサポートしています。 2 つのインデックスのうち、少なくとも 1 つをログに設定する必要があります。

注：
フルテキストインデックスとキー/値インデックスの両方がログに設定されている場合は、キー/値インデックスが優先されます。

インデックスタイプ	説明
フルテキストインデックス	ログ内のすべてのフィールドが、キー/値インデックスを持つテキストとして検索されます。 インデックスの key と値はテキストであり、どちらも検索可能です。 検索で key 名を特定する必要がありません。

インデックスタイプ	説明
キー/値インデックス	フィールドにキー/値インデックスを設定後、フィールドを検索するためのキー名を指定する必要があります。 フルテキストインデックスがログに設定され、キー/値インデックスがログ内のフィールドに設定されている場合、フルテキストインデックスはそのフィールドに対して有効になりません。 フィールドに複数のデータ型を設定できます： ・ Text型 ・ JSON型 ・ 数値型 (Long と Double)

a) ログにフルテキストインデックスを設定します。

ログの全部の内容のインデックスを設定できます。 ログを検索すると、ログ内のすべてのキーの値がデフォルトで検索されます。

パラメータ	説明	例
フルテキストインデックス	このオプションを有効にすると、ログの全部の内容に対してインデックスが有効になります。 ログ内のすべてのキーの値はデフォルトで検索されます。 いずれか一つのキーがキーワードに一致すると、検索結果に表示されます。	-
大文字と小文字を区別	検索で大文字と小文字が区別されるかどうかを指定します。 ・ このオプションを無効にすると、検索では大文字と小文字が区別されません。つまり、内部エラーログは、キーワード "INTERNALERROR"と"internalerror"の両方で検索できます。 ・ このオプションを有効にすると、検索では大文字と小文字が区別されます。つまり、 "internalError"を含むログは、キーワード "internalError"でのみ検索できます。	-

パラメータ	説明	例
漢字	英語と中国語を区別するかどうかを設定します。 ・ 有効にすると、ログに中国語が含まれる場合、中国語の単語分割は中国語の文法に従って実行され、英語の単語分割は単語分割文字に従って実行されます。 ・ 無効にすると、すべての内容を区切り文字で分割します。	-
区切り文字	ログを複数のキーワードに分けるために使用されるシングルバイト文字を指定します。 たとえば、ログの内容が `a , b ; c ; D - F` の場合、ログを a、b、c、D、および F の 5 つのキーワードに分けるために区、切り文字カンマ（、）、セミコロン（;）、およびハイフン（-）を指定できます。	`, '";=()[]{}? @&<>/:\ n \ t`

b) ログのキー/値インデックスを設定します。

指定した key にインデックスを設定できます。 ログにキー/値インデックスを設定後、指定した key を検索して検索範囲を絞り込むことができます。

注：

・ Log Service は自動的に予約済みフィールドのインデックスを作成し、フィールドの分析機能を有効にします。 予約フィールドには、`__topic__` 、`__ source__` 、および `__time__` があります。

・ このトピックでは、[カスタマイズ]タブページの設定を例として説明します。 Nginx テンプレートと MNS テンプレートは、Nginx ログと MNS ログを収集するためにのみ使用され、カスタマイズされたインデックス設定をサポートしません。

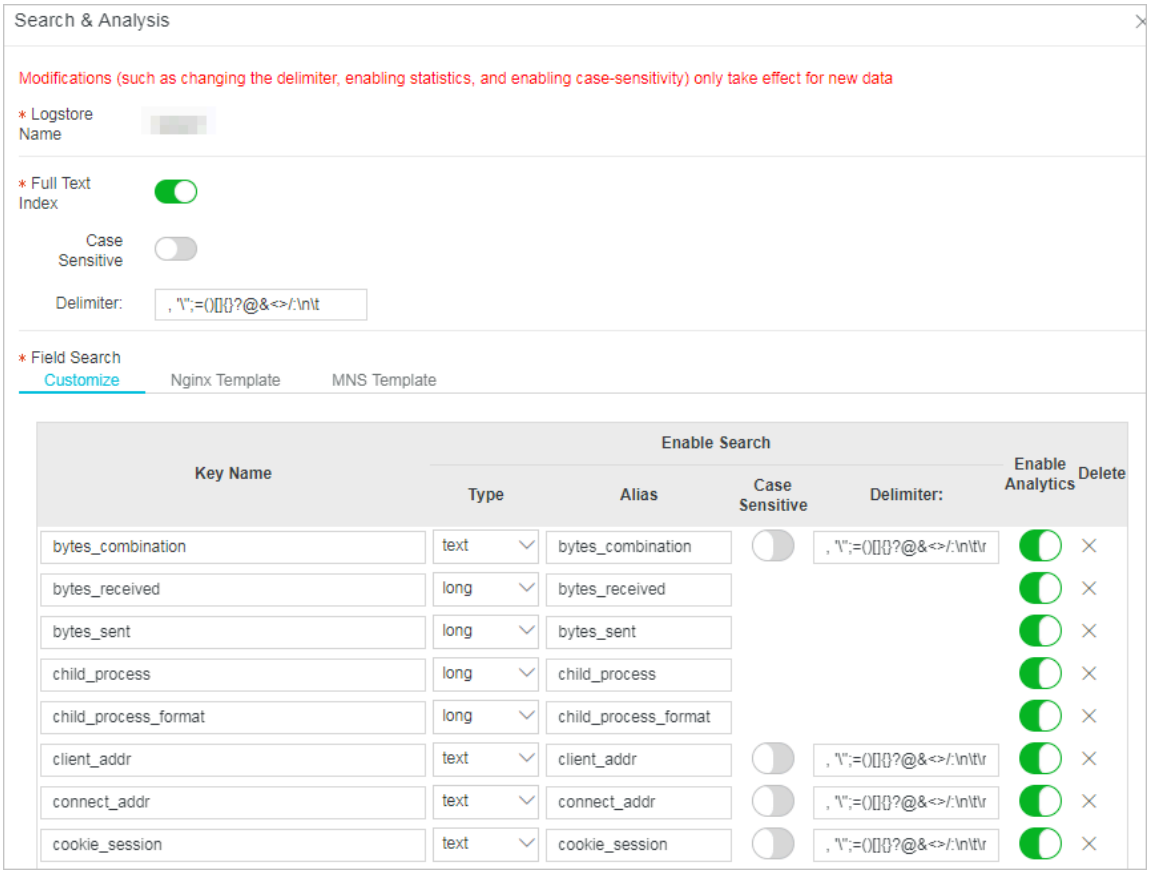
・ インターネット IP アドレスや Unix タイムスタンプなどのタグフィールドにインデックスを設定する場合は、キー名を `__tag__ : key` 形式の値に設定します。 例：`_tag__ : __receive_ time__` 。 タグフィールドは数値型のインデックスをサポートしません。 すべてのタグフィールドのタイプをテキストに設定します。 例えば、キー名 `__tag__ : __receive_ time__` を持つフィールドを検索するには、`__tag__ : __receive_ time__ : 1537928 *` などのファジー値、また

は `__tag__ : __receive_ time__ : 1537928404` などのフィールドの完全値を使用できます。

パラメータ	説明	例
キー名	ログ内のフィールドの名前を指定します。	`_address_`
タイプ	ログ内のフィールドのデータ型を指定します。 ・ text:フィールドの内容がテキスト型である。 ・ long:フィールドの内容が整数である。このフィールドは値の範囲で検索する必要があります。 ・ double:フィールドの内容が浮動小数点数である。 このフィールドは値の範囲で検索する必要があります。 ・ json:フィールドの内容がJSON形式である。 注 : 数値型（Long と Double）は、大文字と小文字を区別する文字や区切り文字をサポートしません。	-
エイリアス	列の別名。 別名は SQL 統計にのみ使用されます。フィールドは、基礎となるストレージ内の元の名前で識別されます。 そのため、フィールドを検索するにはフィールドの元の名前を使用する必要があります。 詳細は、列のエイリアス を参照してください。	`address`

パラメータ	説明	例
大文字と小文字を区別	クエリで大文字と小文字が区別されるかどうかを指定します。 このパラメータには 2 つの値があります ・ false:検索では大文字と小文字が区別されません。つまり、サンプルログはキーワード "INTERNALERROR"と"internalerror"の両方で検索できます。 ・ true:検索では大文字と小文字が区別されます。つまり、サンプルログはキーワード "internalError"でのみクエリできます。	-
区切り文字	ログを複数のキーワードに分けるために使用されるシングルバイト文字を指定します。 たとえば、ログの内容が `a , b ; c ; D - F` の場合、ログを a、b、c、D、および F の 5 つのキーワードに分けるために、区切り文字カンマ（、）、セミコロン（;）、およびハイフン（ - ）を指定できます。	`, '";=()[]{}? @&<>/:\ n \ t`

パラメータ	説明	例
分析機能の有効化	Analytics 機能を有効にするかどうかを指定します。 この機能はデフォルトで有効にしています。 分析機能を使用可能にしたら、検索と分析ステートメントを使用して検索結果を分析することができます。	-



5. OK をクリックします。

注：

- インデックス設定は 1 分以内に有効になります。
- インデックス設定は、設定が有効または変更されてから、記録されたデータにのみ有効になります。

6.4 ログを照会

インデックス機能を有効にして設定を行うと、収集されたログをコンソールより照会および分析できます。

- 既に収集されているログがあること
- インデックス機能の有効化および設定を実施していること

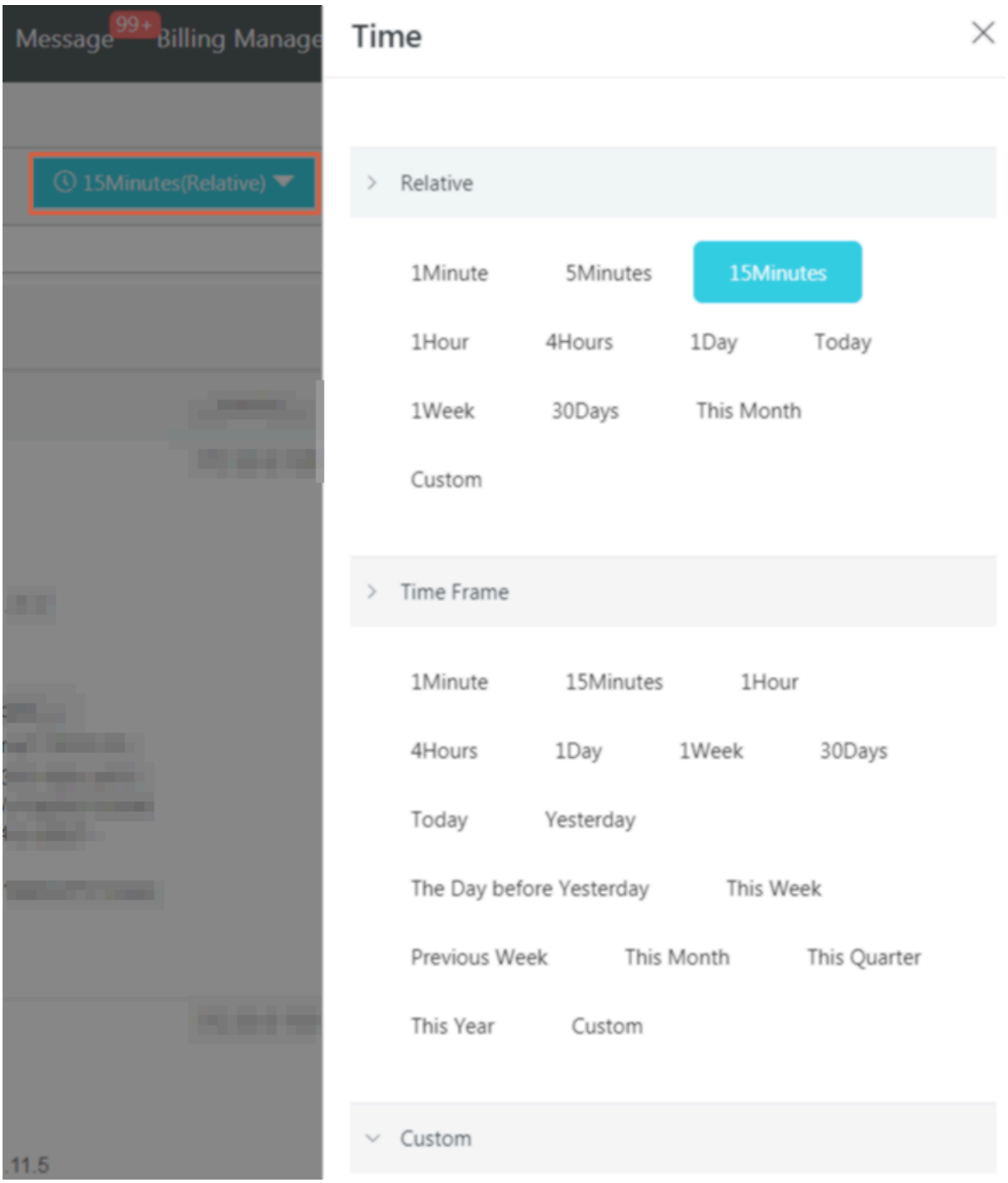
1. Log Service コンソールにログインし、プロジェクト名をクリックします。
2. 照会/分析列の照会をクリックします。
3. クエリ入力欄に、クエリ分析ステートメントを入力します。

   クエリ分析ステートメントは、`クエリステートメント|分析ステートメント`の形式のクエリステートメントおよび分析ステートメントを作成します。詳細については、「概要」をご参照ください。
4. 右上隅の15分 (相対)をクリックして、クエリ対象ログの時間を指定す。

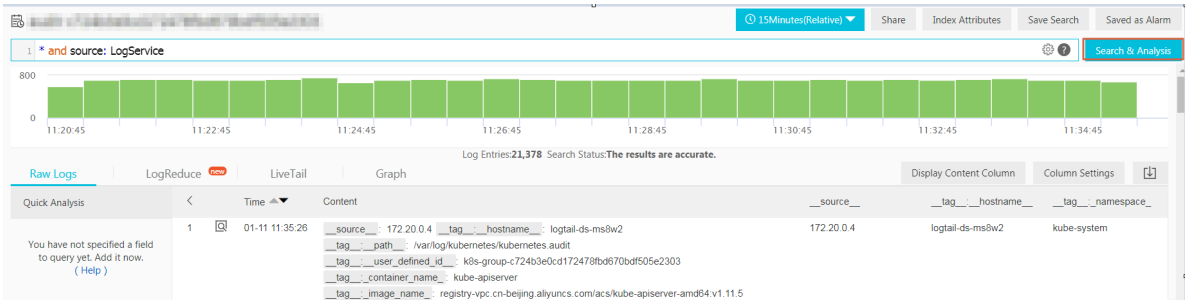
   相対的な期間、絶対的な期間のいずれかを選択するか、期間を指定することができます。

    注：

検索結果は、指定の時間より 1 分早いログや 1 分遅いログを含むことがあります。



5. 照会/分析をクリックしてクエリ結果を表示します。



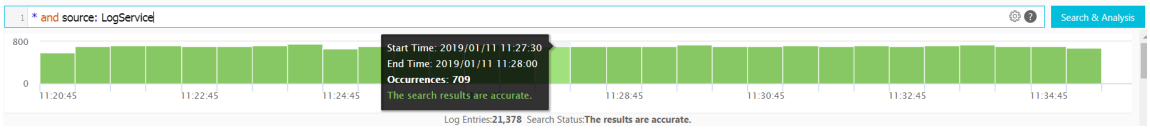
ログ分布ヒストグラム、Raw ログ、また、クエリ結果をさまざまなグラフに表示させることができます。

> **注：**
> デフォルトでは、100 件のクエリ結果が返されます。結果を 100 件以上表示するには、LIMIT 句をご参照ください。

・ ログ分布ヒストグラム

ログ分布ヒストグラムは、ログの時間分布を表示します。

- ポインタを緑色のデータブロックに移動させると、データブロックの期間とその期間のログ数が表示されます。
- データブロックをクリックすることにより、ログ分布の詳細を確認することができます。なお、Raw ログタブページには、クエリ結果のログが表示されます。



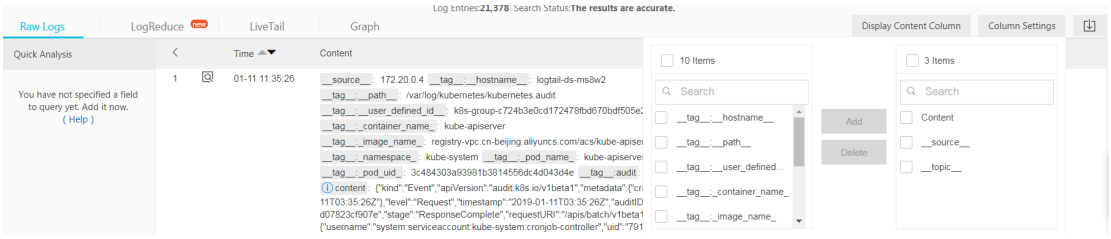
・ Raw ログ

Raw ログタブページには、検索条件を満たしたログが表示されます。

- クイック分析: クイック分析機能を使用して、一定期間におけるフィールドの時間分布をすばやく表示させることができます。 詳細は、「クイック分析」をご参照ください。
- ダウンロード: 右上隅のダウンロードアイコンをクリックし、ダウンロードの範囲を指定して OK をクリックします。
- カラム設定: 右上隅のカラム設定をクリックします。表示されるダイアログボックスの左側のリストより追加するフィールドを選択します。追加をクリックすると選択した

フィールドがタブページに表示されます。各ログのフィールド名がカラム名であり、そのフィールドの値が表示されます。

> **注：**
> タブページにログのコンテンツを表示するには、コンテンツを選択します。



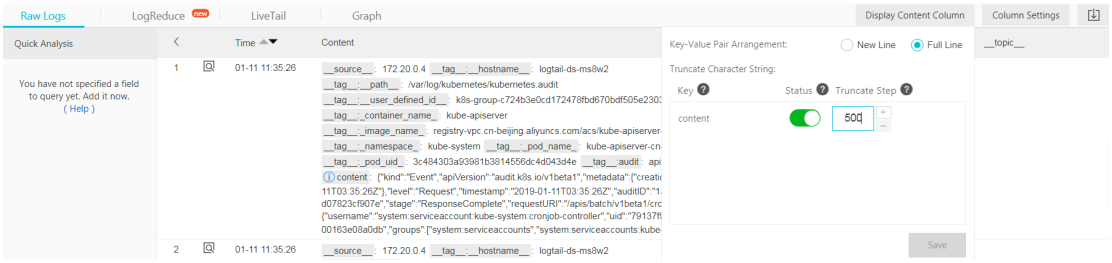
- コンテンツの表示設定: デフォルトでは、フィールドのコンテンツが 3,000 字を超える場合、コンテンツの一部は非表示になります。フィールド名の前に非表示部分があることを示すアイコンが表示されます。タブの右上隅のコンテンツの表示設定をクリックし、表示されるダイアログボックスにフィールドコンテンツの表示および文字列の省略を設定します。

> **注：**
> 表示するコンテンツの上限を 10,000 字に設定すると、10,000 字を超える部分はすべて省略されます。また、省略部分の表示/非表示アイコンは表示されず、省略部分を表示させることはできません。

パラメータ		説明
キーと値のペアの配置		折り返し または 行全体 に設定できます。
文字列の省略	キー	デフォルトでは、フィールドの値が 3,000 字を超える場合、3,000 字を超える部分は省略されます。ただし、この値に達しなければ、パラメータは空のままです。 このパラメータの値は、トランケート表示された値のキーです。

パラメータ		説明
	ステータス	値の省略の有効/無効を指定します (デフォルト: 有効)。 ■ 有効: コンテンツが、設定した字数を超えると、字数を超えた部分のコンテンツは省略されます。省略部分を表示するには、コンテンツ末尾の表示ボタンをクリックします。省略部分が表示されます。表示字数に設定された字数が増分表示されます。 ■ 無効: 表示字数に設定された字数を超えるコンテンツは、表示されません。

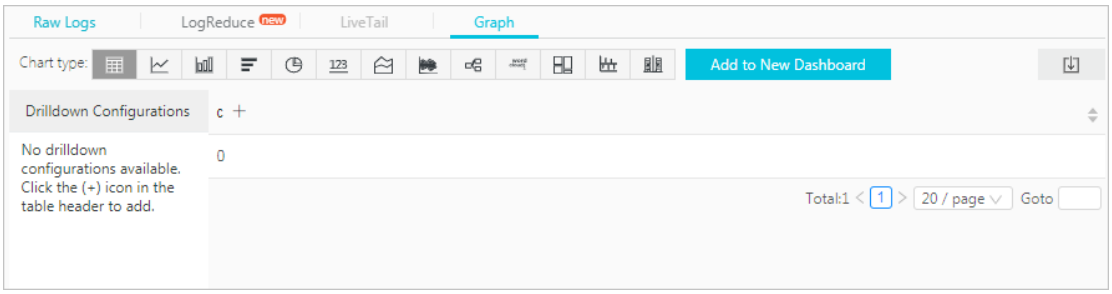
パラメータ		説明
	表示字数	展開表示させる字数 指定可能な値: 500~10,000 、デフォルト: 3,000



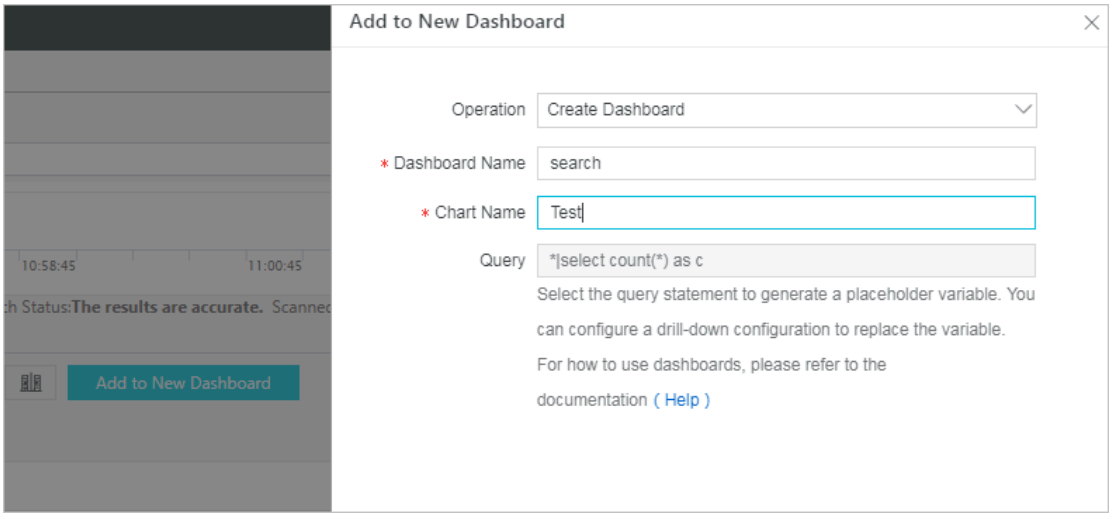
- グラフ

  統計機能を有効にし、クエリ分析ステートメントを指定すると、グラフタブページに分析結果が表示されます。

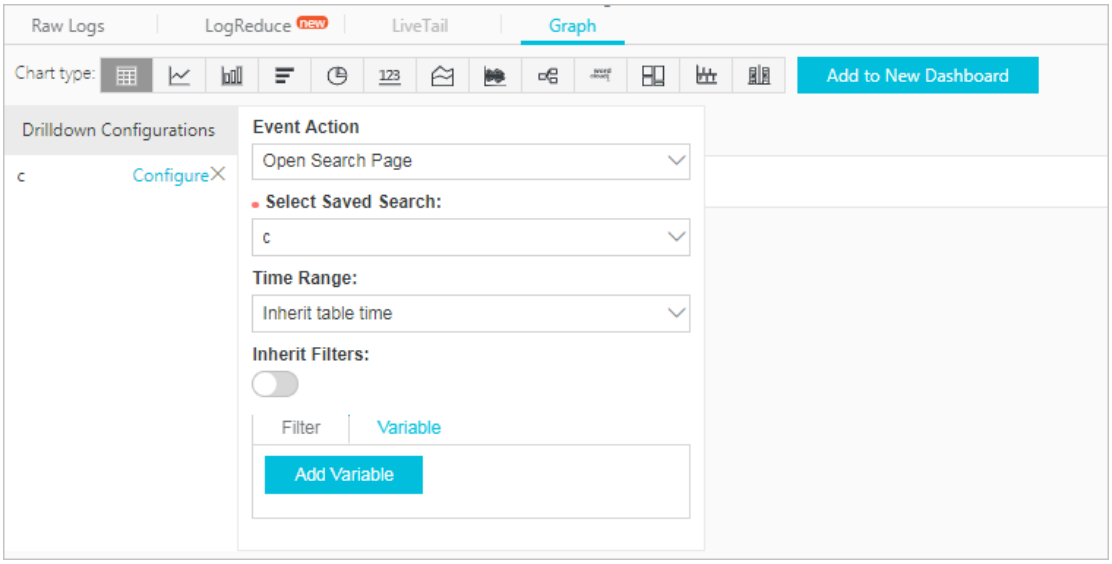
  - 分析結果の表示に適したグラフの種類を選択します。Log Service には、表、折れ線グラフ、棒グラフといったさまざまなグラフが用意されています。



  - ダッシュボードにグラフを追加すると、データ分析結果がリアルタイムに表示されます。 ダッシュボードに追加をクリックして、よく使用するクエリステートメントのグラフをダッシュボードに保存します。



- ドリルダウン構成を設定すると、分析結果の詳細情報が得られます。グラフ内の値をクリックすることにより、詳細な分析結果が表示されます。 詳細は、「ドリルダウン分析」をご参照ください。



なお、クイック検索機能およびアラーム機能を利用するには、右上のクイック検索として保存または新規アラームに保存をクリックします。

6.5 インデックスのデータ型

6.5.1 概要

Log Service を使用すると、収集したログの前文または一部のフィールドにインデックスを設定できます。 ログの全文にインデックスを設定した場合、このログの検索で使用される値はログ全体の内容です。 ログの一部のフィールドにインデックスを設定した場合、検索で使用される各 Key のデータ型を設定できます。

データ型

次の表に、サポートされているインデックスの種類を示します。

検索タイプ	インデックスタイプ	説明	例
基本検索	text	検索でキーワード + あいまい一致をサポートするテキスト型を示します。	`uri :" login *" method :" post "`
	long	間隔検索をサポートする数値型を示します。	`status > 200 , status in [ 200 , 500 ]`

検索タイプ	インデックスタイプ	説明	例
	double	浮動小数点数をサポートする数値型を示します。	`price > 28 . 95 , t in [ 20 . 0 , 37 ]`
コンビネーション検索	JSON	インデックスがネストされた検索をサポートする JSON フィールドであることを示します。 フィールド型はデフォルトでテキスト型となります。 a . b などのパス形式を使用して、要素 'a'の下の要素 'b'に Text 、Long、または Double 型のインデックスを設定できます。 フィールド型は、設定したインデックス型によって決まります。	`level0 . key > 29 . 95 level0 . key2 :" action "`
	フルテキスト	ログの全文がテキストとして検索されることを示します。	`error and " login fail "`

例

次のログには、時間とその他の 4 つの key が含まれています。

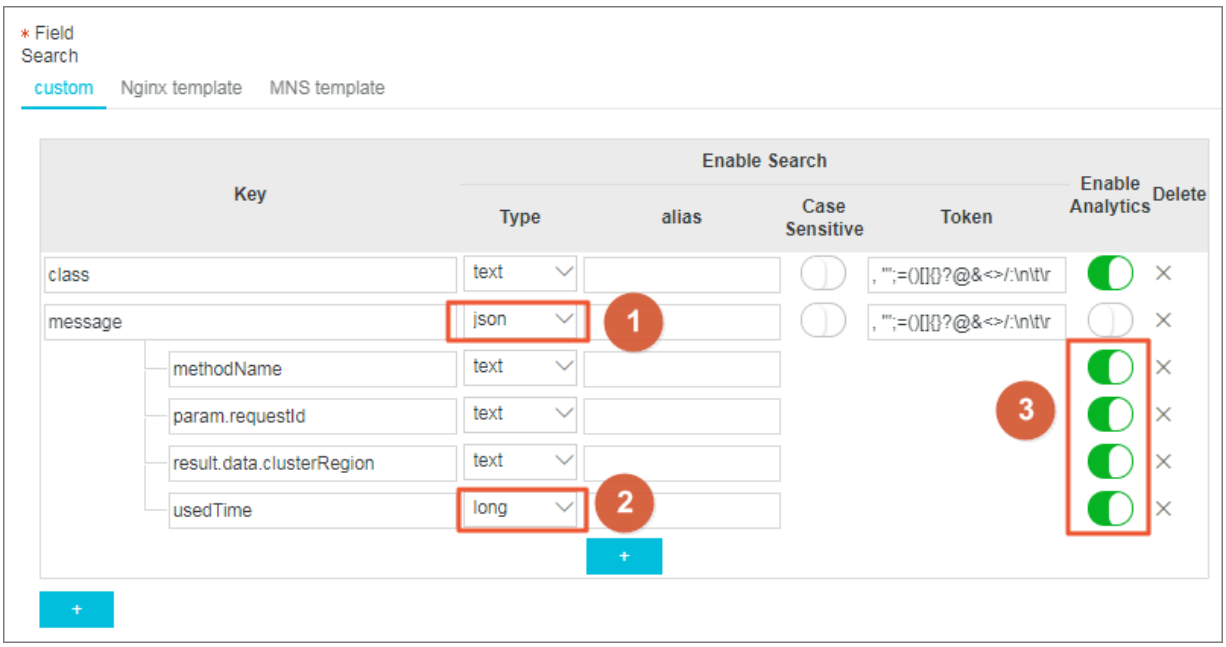
No.	Key	データ型
0	time	-
1	class	text
2	status	long
3	latency	double
4	message	json

```
0 .  time : 2018 - 01 - 01   12 : 00 : 00
  1 .  class : central - log
  2 .  status : 200
  3 .  latency : 68 . 75
  4 .  message :
 {
     " methodName ": " getProject  Info ",
     " success ":  true ,
     " remoteAddr  ess ": " 1 . 1 . 1 . 1 : 11111 ",
     " usedTime ":  48 ,
     " param ": {
           " projectNam  e ": " ali - log - test - project ",
           " requestId ": " d3f0c96a - 51b0 - 4166 - a850 -
f4175dde73  23 "
     },
```

```
      " result ": {
          " message ": " successful ",
          " code ": " 200 ",
          " data ": {
              " clusterReg  ion ": " ap - southeast - 1 ",
              " ProjectNam  e ": " ali - log - test - project ",
              " CreateTime ": " 2017 - 06 - 08   20 : 22 : 41 "
          },
          " success ":  true
      }
  }
```

次のようにログのインデックスを設定できます。

図 6-2 : インデックスの設定



上図では：

- ①は、このフィールドのインデックス型が JSON であり、フィールド内の文字列型とブール型のすべてのデータを検索できることを示します。
- ②は、このフィールドのインデックス型が Long で、フィールド内の Long 型のデータを検索できることを示します。
- ③は、SQL 文を使用してフィールドを分析できることを示します。

例：

1. 文字列型とブール型のデータを検索します。

   - JSON のフィールドを設定する必要はありません。
   - JSON map と array は自動的に展開されます。 複数レイヤでネストされているフィールドを検索できます、各レイヤの間にはピリオド（.）で区切られます。

```
class : cental *
message . traceInfo . requestId : 92 . 137_151813 9699935_55
99
message . param . projectNam e : ali - log - test - project
message . success : true
```

2. Double 型と Long 型のデータを検索します。

   JSON のフィールドを別々に設定する必要があり、array に含めることはできません。

```
latency > 40
message . usedTime > 40
```

3. データ型がコンビネーションされたデータを検索します。

```
class : cental * and message . usedTime > 40 not message
. param . projectNam e : ali - log - test - project
```

6.5.2 テキストタイプ

検索エンジンと同様に、テキストデータは用語に基づいて照会されます。 したがって、オプションを含む単語セグメンテーション、大文字と小文字の区別を設定する必要があります。

注意事項

大文字と小文字を区別

生ログを照会するときに大文字小文字を区別するかどうかを決定します。例えば、生ログは internalError です。

- 大文字小文字の区別スイッチをオフにすると、サンプルログは、キーワード INTERNALERROR または internalerror に基づいて照会することができます。
- 大文字小文字の区別スイッチをオンにすると、サンプルログはキーワード internalError に基づいてのみ照会できます。

トークン

トークンを使用すると、生ログの内容をいくつかのキーワードに分けることができます。

たとえば、生ログは

```
/ url / pic / abc . gif です。
```

- トークンが設定されていない場合、その文字列は個別の単語 `/ url / pic / abc . gif` と見なされます。 完全な文字列や `/ url / pic /*` のようなファジーマッチだけを使ってこのログを照会することができます。
- `/`がトークンとして設定されている場合、生ログは `url` 、 `pic` 、 `abc . gif` の 3 つの単語に分けられます。 このログは、 `url` 、 `abc . gif` 、 `pi *`のように、3 つの単語やファジーマッチのいずれかを使って照会することができます。 また、 `/ url / pic / abc . gif` を使ってこのログを照会することもできます（`/ url / pic / abc . gif` は url、pic、 abc.gif」）。
- `/ .`がトークンとして設定されている場合、生ログは `url` 、 `pic` 、 `abc` 、 `gif` の 4 つの単語に分かれています。

 注：
適切なトークンを設定することによって、クエリの範囲を広げることができます。

全文索引

既定では、フルテキストクエリ（インデックス）は、時間フィールドを除くログのすべてのフィールドとキーをテキストデータとみなし、キーを指定する必要はありません。 たとえば、次のログは 4 つのフィールド（時間/ステータス/レベル/メッセージ）で構成されます。

```
[ 20180102   12 : 00 : 00 ]  200 , error , some   thing   is   error
 in   this   field
```

- 時間：2018-01-02 12:00:00
- レベル：error
- ステータス：200
- メッセージ：an error occurred in this field

全文索引を有効にすると、次のテキストデータが「key：value + space」モードで組み立てられます。

```
 status : 200   level : error   message :" some   thing   is   error
 in   this   field "
```

 注：

- プレフィックスはフルテキストクエリには必要ありません。 キーワードとして error と入力すると、レベルフィールドとメッセージフィールドの両方がクエリ条件を満たします。
- フルテキストクエリのトークンを設定する必要があります。 スペースがトークンとして設定されている場合、 status：200 はフレーズと見なされます。 ：がトークンとして設定されている場合、 status と 200 は 2 つの独立したフレーズと見なされます。
- 数値はテキストとして処理されます。 たとえば、キーワード 200 を使用してこのログを照会することができます。 時間フィールドはテキストとして処理されません。
- status のようなキーを入力すると、このログを問い合わせることができます。

6.5.3 JSON データ型

JSON は、テキスト、boolean、数値、配列 (Array)、およびマップ (Map) で構成される複合型データです。

設定方法

テキストの場合

JSON 型フィールドの場合、テキスト型と boolean 型のフィールドは自動的に認識されます。

たとえば、次の jsonkey は `jsonkey . key1 :” text_value ”` などの条件を使用して照会できます。 .

```
 jsonkey : {
    key1 : text_value ,
    key2 : true ,
    key3 : 3 . 14
}
```

数値のデータ型

データ型を設定してパスを指定することによって、JSON 配列にない double 型または long 型のデータを照会することができます。

たとえば、jsonkey.key3 フィールドが double 型の場合、クエリ文は次のようになります。

```
  jsonkey . key3  >  3
```

非完全有効な JSON

非完全有効な JSON には、無効な部分が表示されるまで、有効な部分を解析します。

例：

```
" json_strin  g ":
{
     " key_1 " : " value_1 ",
     " key_map " :
```

```
{
        " key_2 " : " value_2 ",
        " key_3 " : " valu
```

key_3 の後のデータは切り捨てられ、失われます。 そのような一部が欠落しているログに対して、 `json_strin g . key_map . key_2` フィールドまでのログが正しく解析されます。

注意事項

- JSON オブジェクト型と JSON 配列型はサポートされていません。
- JSON 配列にフィールドを含めません。
- boolean 型フィールドはテキスト型に変換できます。

クエリ構文

指定した Key でクエリする場合、クエリ文に JSON の親パスの接頭辞を追加する必要があります。 テキスト型と数値型のクエリ構文は他の型と同じです。 詳細については、クエリ構文 をご参照ください。

6.5.4 値型

索引を設定する際には、フィールドを値型として設定し、値の範囲を使用してキーを照会できます。

設定方法注意事項

サポートされる型： `long` (long 整数型) and `double` (decimal 型) フィールドを値型として設定すると、値の範囲を使用したキーの照会のみ可能です。

例

キー範囲が [1000 2000] の longkey を照会するには、以下のメソッドを使用します。

- 値を使用して longkey を照会する場合

```
longKey > 1000 and longKey <= 2000
```

- インターバルを使用して longkey を照会する場合

```
longKey in ( 1000 2000 ]
```

構文の詳細については、「クエリ構文」をご参照ください。

6.6 クエリ

6.6.1 クエリ構文

ログサービスは、クエリ条件を表現するために使用される一連のクエリ構文を提供し、ログを簡単に検索するのに役立ちます。 ログサービス API のGetLogsおよびGetHistogramsインターフェイスまたは ログサービスコンソールのクエリページでクエリ条件を指定できます。 このセクションでは、クエリ条件の構文について詳しく説明します。

インデックスタイプ

2 つのモードでログストアのインデックスを作成できます。

- フルテキストインデックス：キーと値（キー、値）を区別することなく、ログの行全体が全体として照会されます。
- キー/値インデックス：Key が指定されたときにクエリが実行されます。 たとえば、 `FILE : app , Type : action` などです。 このキーに含まれるすべての文字列が照会されます。

構文キーワード

ログサービスのクエリ条件では、次のキーワードがサポートされています。

名前名前	説明
and	バイナリ演算子 。 query1 and query2 の形式で、 query1 と query2 のクエリ結果の共通部分を示します。 単語の間に構文キーワードがない場合、単語間の関係はデフォルトでは and です。
or	バイナリ演算子 。 形式： `query1 or query2` query1 or query2 のクエリ結果の和集合を示す `query1` と `query2` の形式の二項演算子。
not	バイナリ演算子 。 `query1 not query2` の形式で、 `query1` にマッチし、 `query2` にマッチしない、すなわち `query1 - query2` を示すバイナリ演算子です。 `not query1` だけが存在する場合、 `query1` のクエリ結果を含まないログが選択されていることを示します。
(,)	左右の角かっこは、1つまたは複数のサブクエリを1つのクエリにマージして、角括弧内のクエリ優先度を向上させるために使用されます。
:	キーと値のペアを照会するために使用されます。 `term1 : term2` は、キーと値のペアを形成します。 キーまたは値に スペースが含まれている場合は、キーまたは値全体を含めるために引用符を使用する必要があります。

名前名前	説明
“	キーワードを共通のクエリ文字に変換します。 左右の引用符で囲まれた語はすべて照会され、構文キーワードとしては使用されません。 または、左と右の引用符のすべての用語は、 キーと値のクエリで全体と見なされます。
\	エスケープ文字。 引用符をエスケープするために使用します。 エスケープされた引用符はシンボル自体を示し、 `"\""`のようにエスケープ文字として使用することはできません。
|	パイプライン演算子は、以前の計算に基づいてより多くの計算を示します。たとえば、query1 | timeslice 1h | count。
timeslice	time-slice 演算子は、データが全体としてどのくらい計算されるかを示します。 Timeslice 1h、1m、1s はそれぞれ 1 時間、1 分、1 秒を示します。 たとえば、query1 | timeslice1h | count はクエリのクエリ条件を表し、1 時間で割った合計時間に戻ります。
count	count 演算子は、ログ行の数を示します。
*	あいまいクエリーキーワード。 0 または複数の文字を置き換えるために使用されます。 たとえば、 `que *`のクエリ結果では、 `que` で始まるすべてのヒットワードが返されます。 注 : およそ 100 のクエリの結果が返されます。
?	あいまいクエリキーワード。 1 つの文字を置き換えるために使用されます。 例えば、 `qu ? ry` ; 全てのヒットワードは `qu` で始まり、 `ry` で終わり、中間の文字で始まります。
`__topic__`	トピックデータクエリ。 新しい構文では、クエリ内で 0 個以上のトピックのデータを クエリできます。 例えば、 `__topic__ : mytopicnam e` 。
`__tag__`	タグキーのタグ値を照会します。 たとえば、 `__tag__ : tagkey : tagvalue` のようになります。
Source	IP のデータをクエリします。 例えば、 `source : 127 . 0 . 0 . 1` 。
>	特定の数値より大きいフィールドの値でログを照会します。 例えば、 `latency > 100` .
>=	特定の数値以上のフィールドの値でログを照会します。 例えば、 `latency >= 100` .

名前名前	説明
<	特定の番号よりも小さいフィールドの値を持つログを照会します。 例えば、 `latency < 100` .
<=	特定の数値以下のフィールドの値でログを照会します。 例えば、 `latency <= 100` .
=	特定の数値に等しいフィールドの値でログを照会します。 例えば、 `latency = 100` .
in	特定の範囲内のフィールドでログを照会します。 中括弧([])は閉じた間隔を示すのに使用され、かっこ(())は開いた間隔を示すために使用されます。 中括弧([])またはかっこ(())で 2 つの数字を囲み、数字を複数のスペースで区切ります。 たとえば、 `latency in [ 100 200 ]` または `latency in ( 100 200 ]` のレイテンシや（100 200）のレイテンシ。

注：

- 構文キーワードは大文字と小文字を区別しません。
- 構文キーワードの優先順位は、: > " > ( ) > and not > or で降順でソートされます。
- ログサービスは、以下のキーワードを使用する権利を有します： `sort asc desc group by avg sum min max limit` 。次のキーワードを使用する必要がある場合は、引用符で囲みます。
- 全文索引とキー/値索引が構成されているときに異なる単語区切り文字を持つ場合、全文問合せ方式を使用してデータを照会することはできません。
- 数値問合せを実行するには、問合せ列のデータ型を double または long として設定する必要があります。 データ型が設定されていないか、数値範囲クエリに使用される構文が正しくない場合、ログサービスはクエリ条件をフルテキストインデックスとして変換し、予期しない結果につながります。
- 列のデータ型をテキストから数値に変更すると、変更前のデータに対しては = クエリのみがサポートされます。

クエリの例

1. a と b を同時に含むログ: `a and b` or `a b` 。
2. a または b を含むログ: `a or b` 。
3. a を含むが b を含まないログ: `a not b` 。
4. a を含まないすべてのログ: `not a` 。

5. a と b を含むが、c を含まないログ: `a and b not c` 。
6. a または b を含み、c を含むログ: `( a or b ) and c` 。
7. a または b を含み、c を含まないログ: `( a or b ) not c` 。
8. a と b を含み、または c を含むログ: `a and b or c` 。
9. FILE フィールドに apsara が含まれているログ: `FILE : apsara` 。
10. FILE フィールドに apsara と shennong が含まれているログ: `FILE :" apsara shennong "` or `FILE : apsara FILE : shennong` or `FILE : apsara and FILE : shennong` 。
11. and を含むログ: `and` 。
12. apsara または shennong を含むFILEフィールドのログ: `FILE : apsara or FILE : shennong` .
13. 情報フィールドに apsara を含むログ:`" file info ": apsara` 。
14. 引用符を含むログ:`\"`。
15. shen で始まるログ: `shen *`。
16. FILEフィールドの shen で始まるすべてのログ： `FILE : shen *`。
17. shen で始まり、ong で終わるログ: `shen ? ong` 。
18. shen と aps で始まるすべてのログ: `shen * and aps *`。
19. aps で始まるすべてのログを20分ごとにクエリ： `shen *| timeslice 20m | count` 。
20. topic1 と topic2 のすべてのデータ: `__topic__ : topic1 or __topic__ : topic2` 。
21. tagkey1 の tagvalue2 のすべてのデータ: `__tag__ : tagkey1 : tagvalue2` 。
22. 100 以上 200 未満のレイテンシですべてのデータ: `latency >= 100 and latency < 200` or `latency in [ 100 200 )`。
23. 待ち時間が 100 を超えるすべてのリクエスト: `latency > 100` 。
24. http_referer にスパイダーを含まず、opx を含まないログ: `not spider not bot not http_refer er : opx` .
25. 空の cdnIP フィールドを使用してログ: `cdnIP :""`.
26. cdnIP フィールドのないログ: `not cdnIP :*`.
27. cdnIP フィールドのあるログ: `cdnIP :*`.

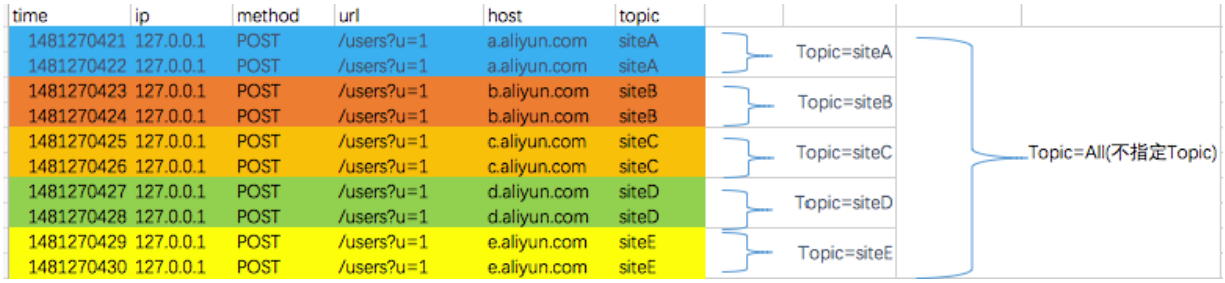
指定されたトピックまたはクロストピックのクエリ

トピックごとに、各LogStoreを1つ以上の部分空間に分割できます。 therfhfrgクエリ中にトピックを指定すると、クエリの範囲が制限され、速度が向上する可能性があります。 したがって、LogStore のセカンダリ分類要件がある場合は、 topic を使用して LogStore を分割することをお勧めします。

1 つまたは複数のトピックが指定されている場合、クエリは条件を満たすトピックでのみ実行されます。 ただし、トピックが指定されていない場合は、デフォルトですべてのトピックのデータが照会されます。

たとえば、異なるドメイン名でログを分類するには、topic を使用します。

図 6-3 : ログトピック



トピッククエリ構文：

- すべてのトピックのデータを照会できます。 クエリの構文とパラメータにトピックが指定されていない場合は、すべてのトピックのデータがクエリされます
- トピックごとにクエリをサポートします。 クエリ構文は `__topic__ : topicName` です。古いモード（URL パラメータでトピックを指定）は引き続きサポートされています。
- 複数のトピックを照会することができます。 たとえば、 `__topic__ : topic1 or __topic__ : topic2` は、Topic1 と Topic2 からのデータの結合クエリを示します 。

ファジー検索

ログサービスはファジー検索をサポートしています。 64文字以内の単語を指定し、`*`や`?`などのファジー検索キーワードを単語の文中または後尾に追加します。 100 件の適格な単語が検索され、その間に 100 語を含むすべてのログが返されます。

制限：

- クエリログが`*`または`?`で始めることができない場合は、接頭辞を指定する必要があります。
- 特定の単語を正確にすると、より正確な結果が得られます。

- ファジー検索を使用して 64 文字を超える単語を検索することはできません。 64文字以下の単語を指定することをお勧めします。

6.6.2 LiveTail

LiveTail は、Log Service によってコンソールに提供される対話型の機能で、リアルタイムでログをモニタリングし、重要なログ情報を抽出するのに役立ちます。

シナリオ

オンラインの O&M のシナリオでは、ログキューのインバウンドデータをリアルタイムでモニタリングし、最新のログデータから重要な情報を抽出して例外の原因をすばやく見つける必要がある状況がよくあります。 従来の O&M 方法を使用すると、サーバー上のログファイルに対して `tail -f` コマンドを実行して、ログファイルをリアルタイムでモニタリングする必要があります。 必要なログ情報が十分に明らかでない場合は、キーワードをフィルタリングするためにコマンドに `grep` または `grep -v` を追加する必要があります。 Log Service はコンソールに LiveTail を提供します。これはオンラインのログデータをリアルタイムでモニタリングおよび分析するインタラクティブな機能です。

利点

- リアルタイムのログ情報をモニタリングし、キーワードをマークおよびフィルターできます。
- 収集構成を通じてインデックスを使用して、収集されたログを区別できます。
- ログフィールドの単語分割を実行して、セグメント化された単語を含むコンテキストログを検索できます。
- サーバーに接続することなしに、単一のログエントリに従ってログファイルをトラッキングし、リアルタイムモニタリングを実行できます。

制限

- LiveTail は Logtail によって収集されたログにのみ適用できます。
- LiveTail はログが収集されたときにのみ利用可能です。

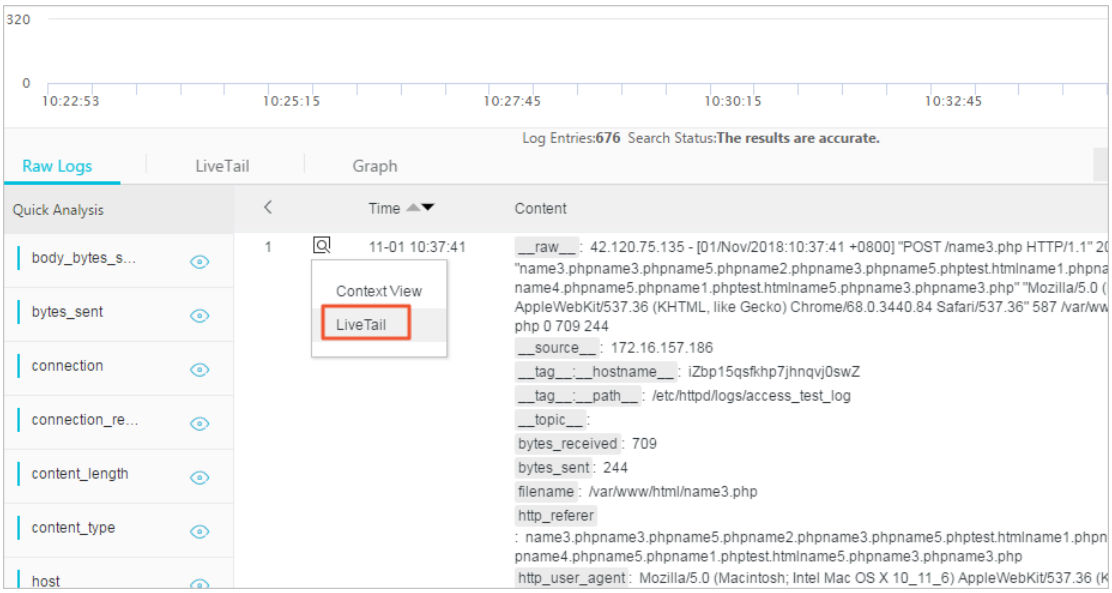
LiveTail を使用してリアルタイムでログをモニタリングする

1. 登#日志服#控制台，##Project名称。
2. 解析検索列で検索をクリックします。

3. LiveTail は、次の 2 つの方法のいずれかで使用できます。

   - LiveTail をクイックスタートします。

     a. Raw データタブで、生ログのシーケンス番号の右側にある アイコンをクリックし、LiveTail を選択します。

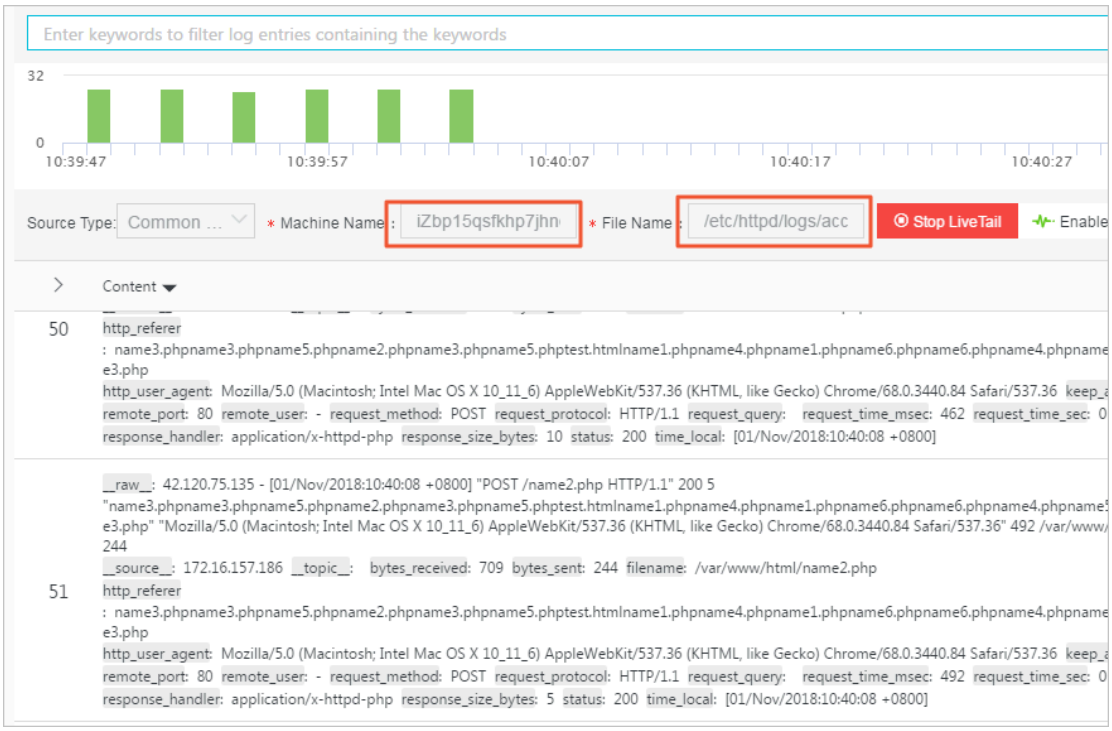


     b. システムは自動的に LiveTail を起動し、計時を開始します。

     ソースタイプ、マシン名、およびファイル名は、生ログを指定するように事前設定されています。

     LiveTail の起動後、Logtail によって収集されたログデータがページ上に順番に表示されます。 最新のログデータは常にページの下部に表示されます。 デフォルトではスクロールバーはページの最下部にあり、最新のデータをすぐに確認することができます。 このページには最大 1000 件のログエントリが表示されます。 1000 件のログエントリ

が表示されると、ページが自動的に更新されて最新のログエントリがページの下部に表示されます。



c. （オプション）検索ボックスにキーワードを入力します。

モニタリングリストに表示できるのは、キーワードを含むログエントリのみです。 キーワードを含むログをフィルタリングすることで、ログの内容をリアルタイムでモニタリングできます。

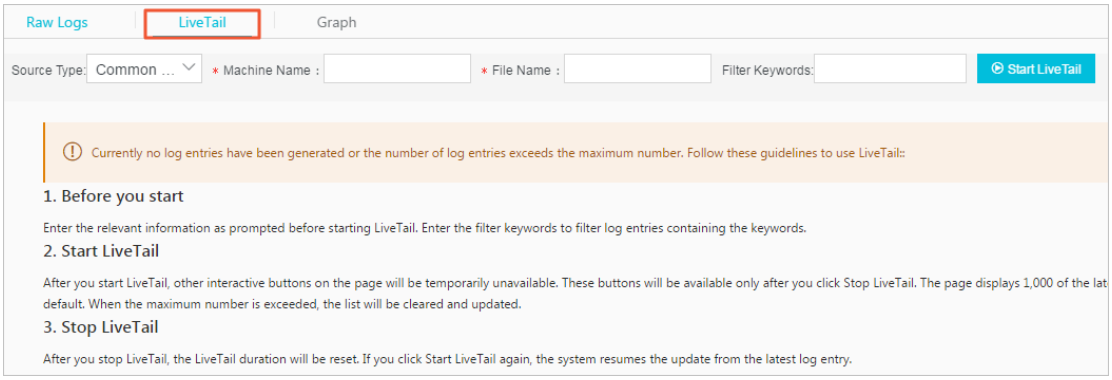
d. リアルタイムログモニタリングプロセス中に例外が存在する可能性があるログを分析するには、LiveTail の停止をクリックします。

LiveTail を停止すると、LiveTail のタイミングとログデータのリアルタイム更新も停止します。

ログモニタリングの過程で見つかった検出した例外に対して、Log Service は複数の分析方法を提供します。 詳細は、LiveTail を使用してログを分析するを参照してください。

- LiveTail 設定をカスタマイズします。

a. LiveTail タブをクリックします。



b. LiveTail を構成します。

構成項目	必須項目	説明
ソースタイプ	はい	ログのソース、次を含めています： - 一般的なログ - Kubernetes - Docker
マシン名	はい	ログソースサーバーの名前。
ファイル名	はい	ログファイルのフルパスとファイル名。
フィルターキーワード	いいえ	キーワード。 キーワードを構成後、そのキーワードを含むログのみをリアルタイムモニタリングウィンドウに表示できます。

c. LiveTail を起動をクリックします。

LiveTail の起動後、Logtail によって収集されたログデータはページ上に順番に表示されます。 最新のログデータは常にページの下部に表示されます。 スクロールバーはデフォルトではページの最下部にあり、最新のデータを確認することができます。 このページには最大1000件のログエントリが表示されます。 1000件のログエントリが表示されると、ページが自動的に更新されて最新のログエントリがページの下部に表示されます。

d. リアルタイムログモニタリングプロセス中に例外が存在する可能性があるログを分析するには、LiveTail を停止をクリックします。

LiveTail を停止すると、LiveTail のタイミングとログデータのリアルタイム更新も停止します。

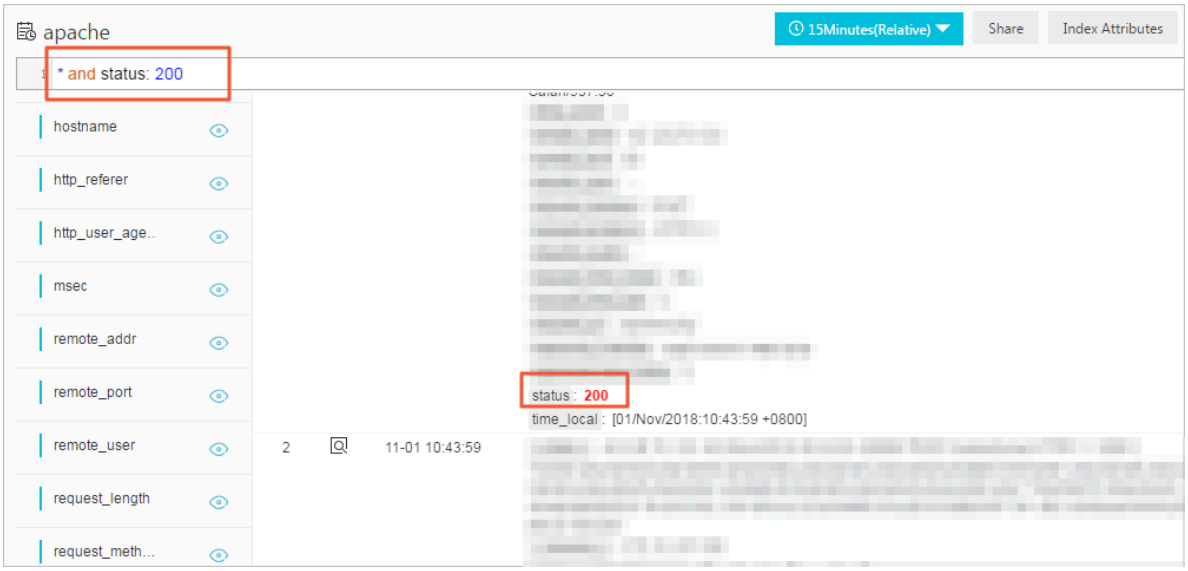
ログモニタリングの過程で検出した例外に対して、Log Service は複数の分析方法を提供します。 詳細は、LiveTail を使用してログを分析するを参照してください。

LiveTail を使用してログを分析する

LiveTailを停止すると、リアルタイムモニタリングウィンドウはログの更新を停止し、モニタリングプロセスで検出した例外を分析してトラブルシューティングできます。

- 特定したフィールドを含むログを表示します。

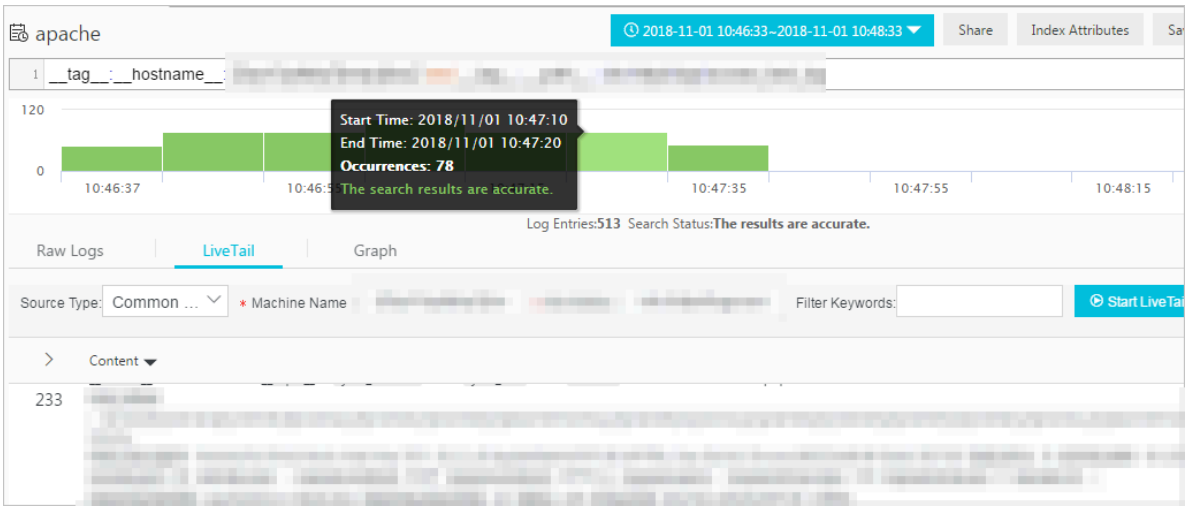
  単語分割は全てのフィールドに対して行われています。 例外フィールドの内容、つまりキーワードをクリックすると、ページは自動的に Raw データタブにジャンプし、システムはすべてのログをフィルタリングしてキーワードを含むログを表示します。 さらに、コンテキストビュー、統計グラフ、およびその他の分析方法を使用して、キーワードを含むログを分析することもできます。



- ログ分布ヒストグラムに従って検索の時間範囲を絞込みます。

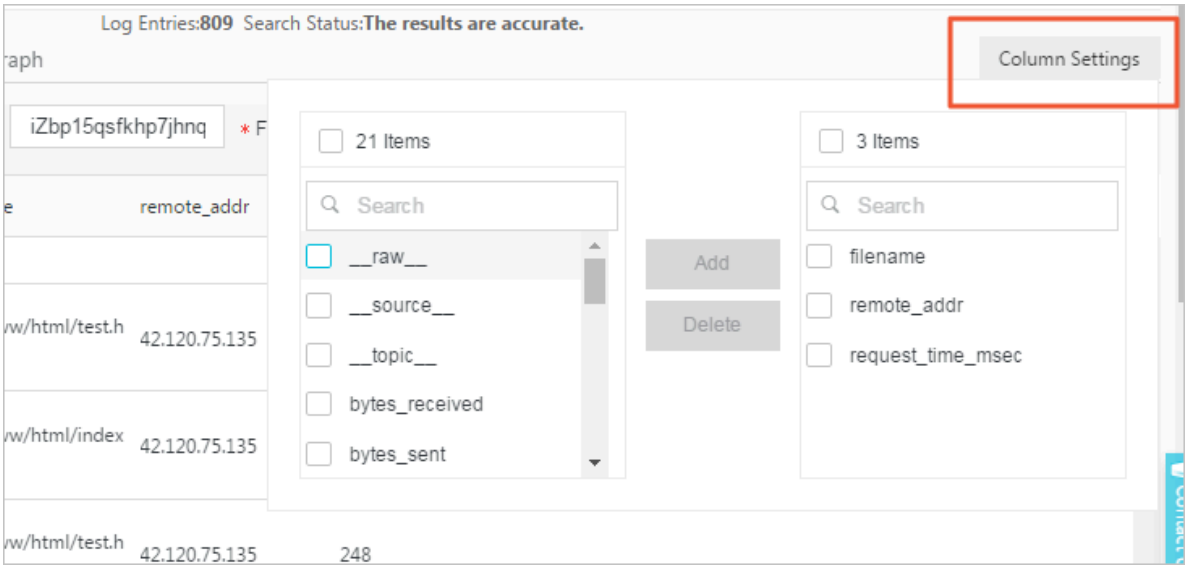
  LiveTail を起動すると、ログ配布ヒストグラムも同期的に更新されます。 ある期間におけるログ配布の例外、たとえばログ数の大幅な増加が検出された場合は、その期間の緑色の長方形をクリックして、検索の期間を絞り込むことができます。 LiveTail ページからリダイレクト

された生ログのタイムラインは、LiveTail でクリックされたタイムラインに関連付けられています。 この期間中は、すべての未加工ログと詳細なログ配布を経時的に表示できます。



- 列設定で重要な情報を強調表示します。

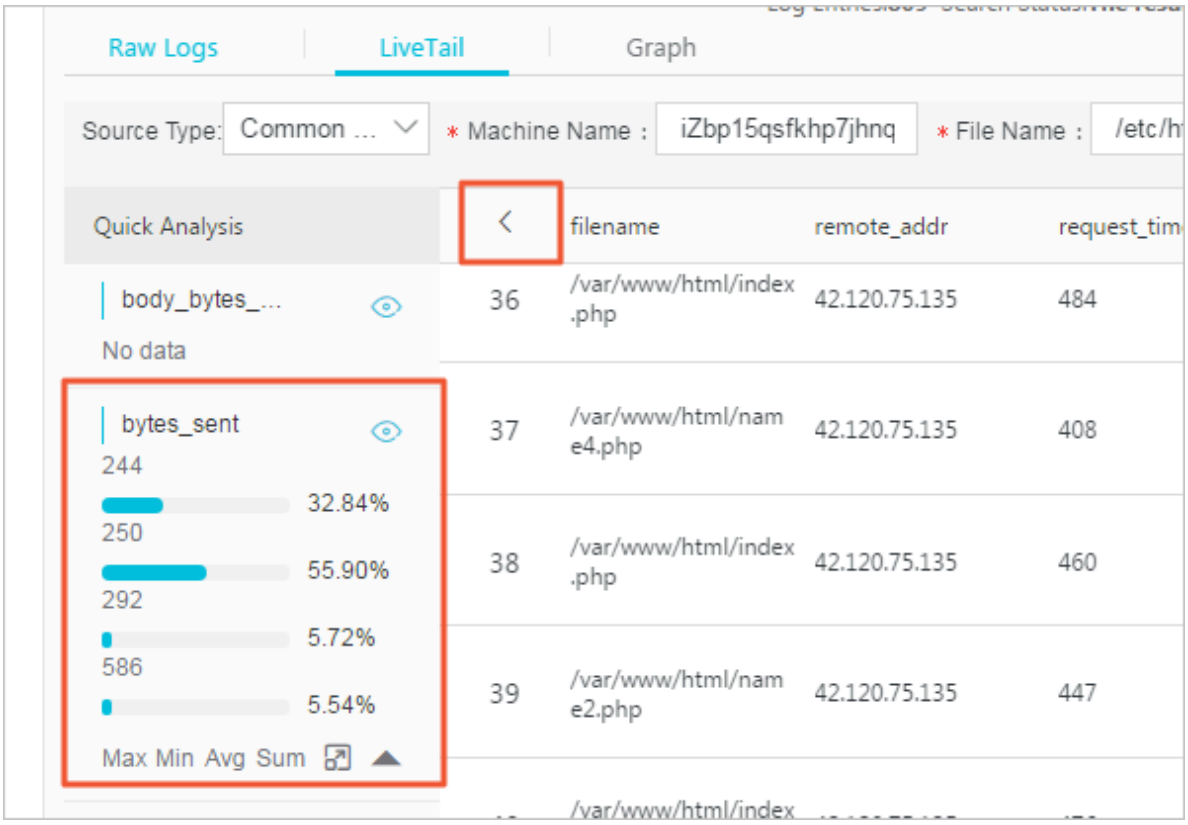
  LiveTail タブで、ログリストの右上隅にある列の設定をクリックします。この列のデータをわかりやすくするために、指定したフィールドを別の列として設定できます。 注意を必要とするデータを 1 つの列として構成して、例外を表示および認識しやすくすることができます。



- ログデータのクイック分析。

  LiveTail タブで、ログリストの左上隅にある矢印をクリックすると、クイック分析エリアを拡張できます。 クイック分析の時間間隔は、LiveTail が起動してから停止するまでの期間

です。 LiveTail で提供されるクイック分析は、生ログで提供されるものと同じです。 詳細は、クイック解析を参照してください。



6.6.3 コンテキストクエリ

ログファイルを展開すると、各ログにイベントが記録されます。 一般に、ログは互いに独立しているわけではありません。 いくつかの連続したログにより、イベント全体のプロセスを順番に見ることができます。

ログ・コンテキスト・クエリは、ログ・ソース（マシン + ファイル）とログ・ソース内のログを指定し、 元のログ・ファイルのログの前（前の部分）および後（次の部分）にいくつかのログを照会します。これにより、DevOps シナリオで問題を簡単にトラブルシューティングする方法が提供されます。 提供されます。

ログサービスコンソールには、クエリ用の特定のページが用意されています。コンテキスト情報は、コンソールの指定されたログの元のログファイルに表示できます。 これは元のログファイルのページングの上または下に似ています。 指定したログのコンテキスト情報を表示することで、ビジネス上の問題をすばやく解決できます。

シナリオ

たとえば、O2O テイクアウトウェブサイトは注文書のトランザクショントラックをサーバー上のプログラムログに記録します。

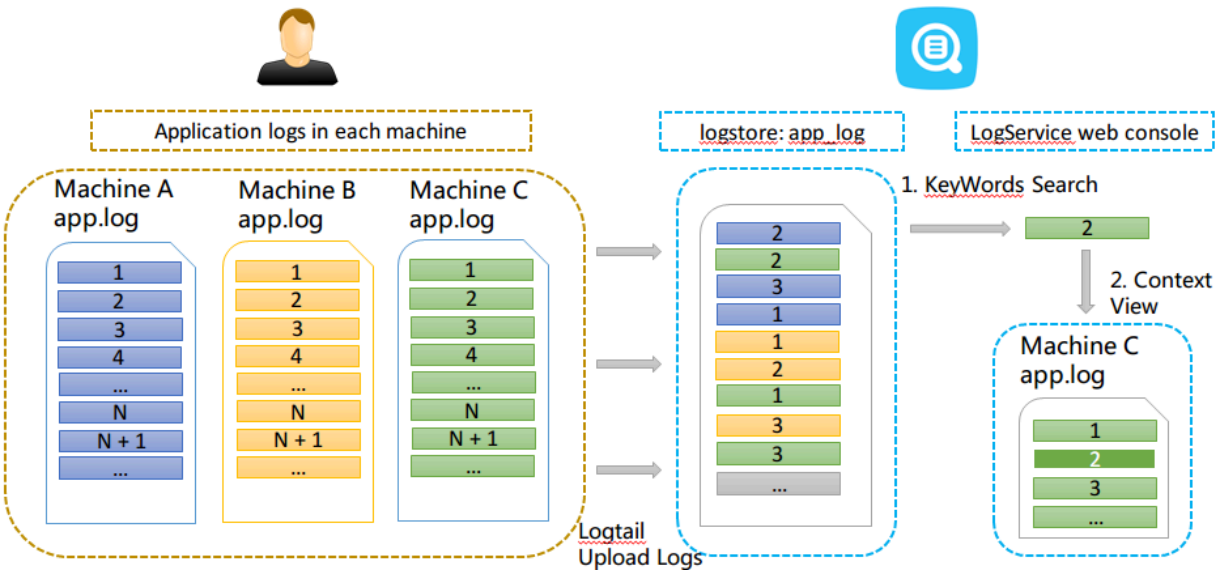
ユーザログオン> 商品を閲覧 > 商品をクリックする> 買い物カゴに追加 > 注文> 注文の支払い> 注文方法から引き落とし> 注文完了

注文を行うことができない場合、運営責任者と顧客サービス担当者は、問題の原因をすばやく突き止めなければなりません。 従来のコンテキストクエリでは、管理者は関連するメンバにマシンログイン許可を追加し、次にアプリケーションが展開される各コンピュータにログインし、 注文 ID をキーワードとして使用してアプリケーションログファイルを検索します。

Log Service、次の手順で問題のトラブルシューティングを行うことができます。

1. ログ収集クライアント Logtail をサーバーにインストールし、マシン・グループとログ収集構成をコンソールに追加します。 その後、Logtail は増分ログのアップロードを開始します。 Log4J、LogBack、C-Producer などのプロデューサ関連の SDK のアップロードを使用することもできます
2. Log Service コンソールのログクエリページで、時間範囲を指定し、注文 ID に基づいて注文失敗ログを検索します。
3. 見つかったエラーログに基づいて、他の関連するログが見つかるまでページングします（たとえば、クレジットカードでの支払いが失敗した場合など）。

図 6-4 : シナリオ



利点

- アプリケーションへの接続はありませんし、 ログファイル形式を変更する必要もありません。
- Log Service ログファイルを表示するために、各マシンにログオンせずに、任意のマシンまたはファイルのログコンテキスト情報を表示することができます。

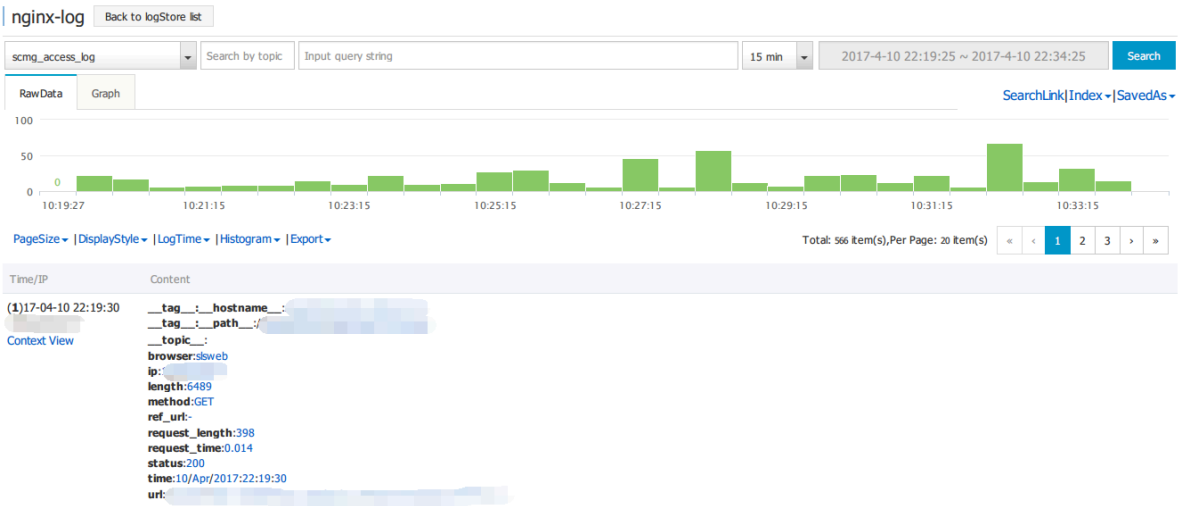
- イベントが発生した時刻と合わせて疑わしいログをすばやく探し出し、Log Service コンソールでコンテキスト情報を効率的に照会するための時間範囲を指定することができます。
- サーバーの記憶域の不足やログファイルのローテーションによるデータの損失を心配する必要はありません。 Log Service コンソールで履歴データをいつでも参照できます。
- Logtail によるログの収集。 Logstore にデータをアップロードします。 マシングループとコレクション構成を作成します。 他の構成は必要ありません。 プロデューサライブラリなど、プロデューサ関連の SDK アップロードを使用することもできます。
- ログのインデックスとクエリ関数を有効にします。

> 注 :
> 現在、syslog データのコンテキスト情報を照会することはできません。

1. Log Service コンソールにログインします。
2. プロダクトページでプロジェクト名をクリックします。
3. Logstore List ページで、Logstore の右側にある Search をクリックします。
4. クエリと解析文を入力し、時間範囲を選択します。 Search をクリックします。

   クエリ結果では、 Context View リンクがログの左側に存在する場合、ログがコンテキストクエリ機能をサポートしていることを示します。

   図 6-5 : ログを検索



5. ログの左側にある Context View をクリックします。 このログのコンテキストログを右側に表示します。
6. 上下にスクロールすると、指定したログのコンテキストログが表示されます。 より多くのコンテキストログを表示するには、前の または後のをクリックします。

6.6.4 クイック検索

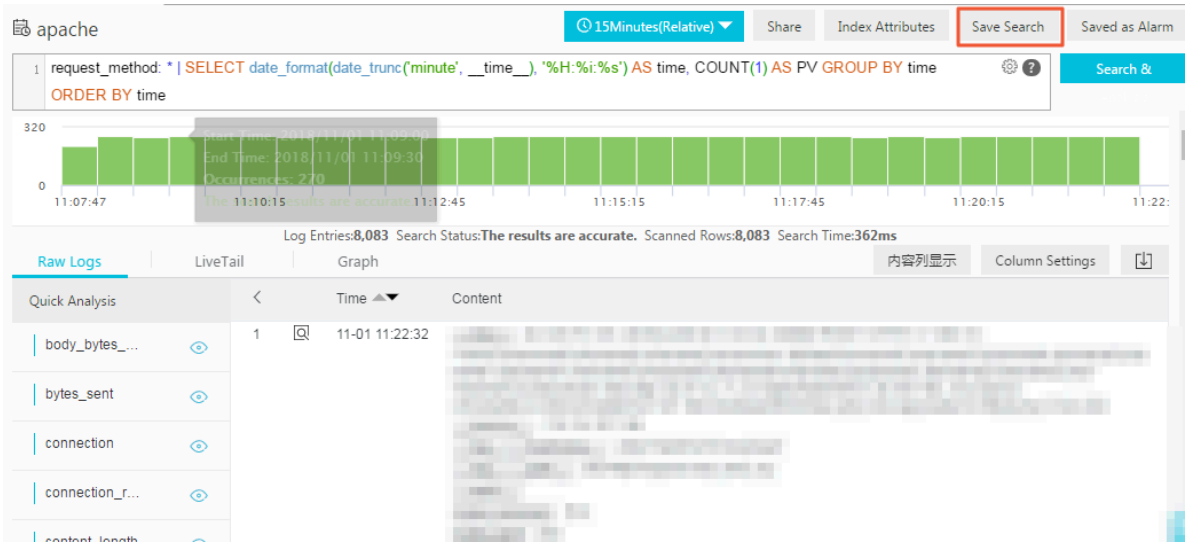
クイック検索は、Log Service が提供するワンクリックの検索と分析機能です。

インデックスを有効にして構成しました。

検索と分析ステートメントの結果を頻繁に確認する必要がある場合は、そのステートメントをクイック検索として保存してください。 次回の検索では、ページの左側にある保存済みの検索名をクリックするだけで済みます。 保存した検索条件をアラームルールで使用することもできます。 Log Service は保存された検索のステートメントを定期的に実行し、検索結果がステートメントの事前設定された条件を満たしたときにアラーム通知を送信します。

ドリルダウン分析を構成時に、ドリルダウンイベントをクイック検索にジャンプするように設定するには、クイック検索を事前に設定し、検索ステートメントにプレースホルダを設定する必要があります。

1. Log Service コンソールにログインし、プロジェクト名をクリックします。
2. Logstore ページの解析検索列で検索をクリックします。
3. 検索分析ステートメントを入力して時間範囲を設定し、検索と分析をクリックします。
4. ページの右上隅にあるクイック検索として保存をクリックします。



5. クイック検索の属性を構成します。

   a) クイック検索の名前を設定します。

      - 名前には、小文字、数字、ハイフン（ - ）、およびアンダースコア（_）のみを含めることができます。
      - 名前は、小文字または数字で始めて終わりにする必要があります。
      - 名前は 3 から 63 文字の string でなければなりません。

   b) Logstore、トピック、および検索ステートメントを確認します。

   Logstore と トピックが要件を満たしていない場合は、検索ページに戻って適切な Logstore にアクセスして検索ステートメントを入力し、もう一度クイック検索として保存をクリックします。

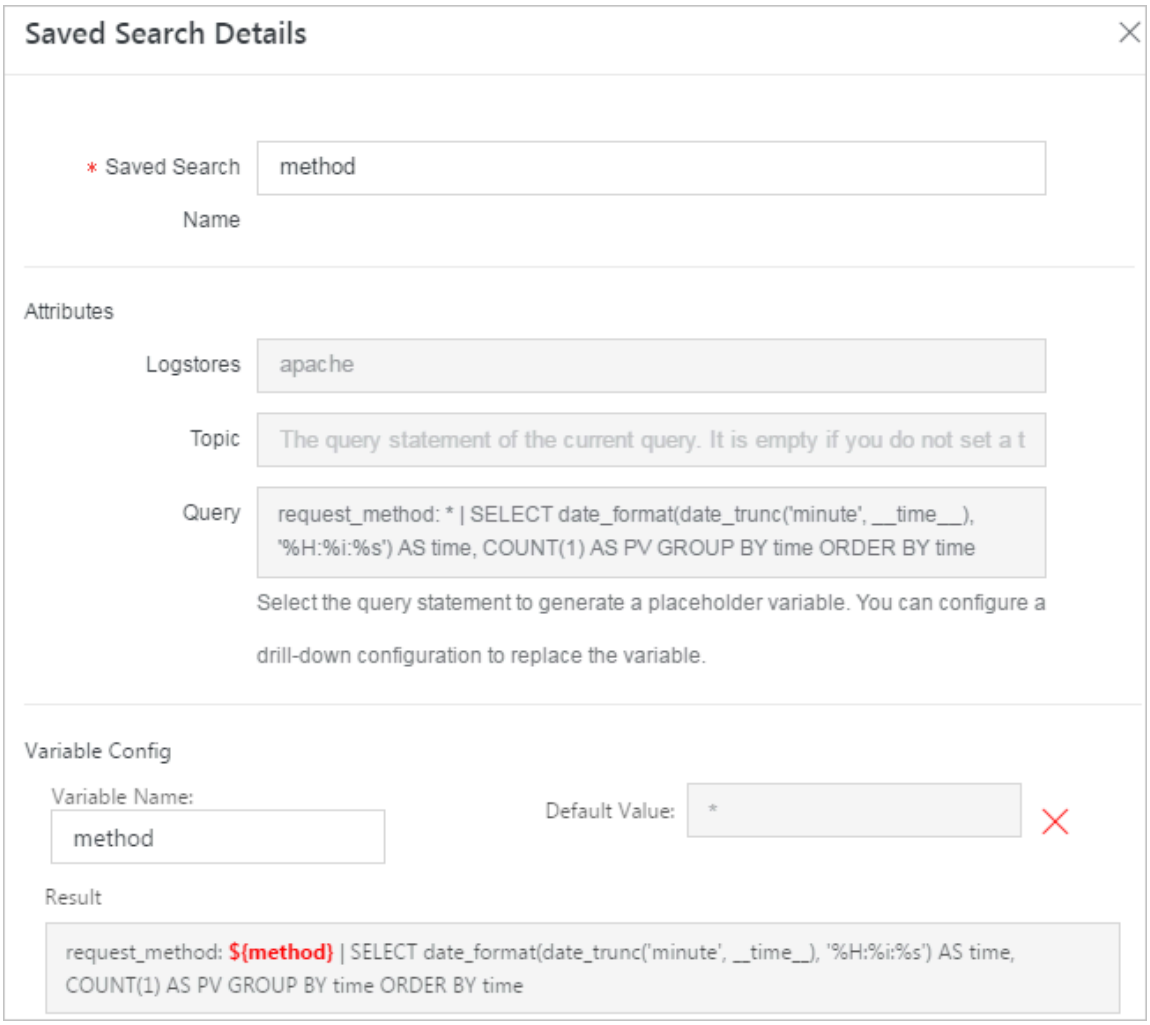
   c) オプション: 検索ステートメントの一部を選択して変数の生成をクリックします。

   生成された変数はプレースホルダ変数です。 変数名ボックスでプレースホルダに名前をつけます。 デフォルト値は選択されたステートメントです。

    注 :

   グラフのドリルダウンイベントがクイック検索にジャンプすることで、グラフにこのクイック検索と同じ変数がある場合は、グラフをクリックするとジャンプがトリガーされます。 さらに、プレースホルダ変数のデフォルト値がドリルダウンイベントをトリガする

グラフ値に置き換えられ、変数が置き換えられた検索ステートメントが検索に使用されます。 詳細は、ドリルダウン分析を参照してください。



6. OK をクリックして構成を終了します。

6.6.5 クイック解析

Log Service のクイック解析機能は、 1 回のクリックでインタラクティブなクエリをサポートし、一定期間にわたるフィールドの分布を迅速に分析し、キーデータのインデックス作成コストを削減します。

機能と特徴

- テキストフィールドの最初の 10,000 個のデータの最初の 10 個の統計をグループ化することをサポートします。
- `Text` フィールドのために `approx_dis tinct` クエリ文を素早く生成することをサポートします。
- `long` または `double` フィールドの近似分布のヒストグラム統計をサポートします。

- `long` または `double` フィールドの最大、最小、平均、または合計のクイック検索をサポートします。
- 迅速な分析とクエリに基づいてクエリ文を生成するサポート。

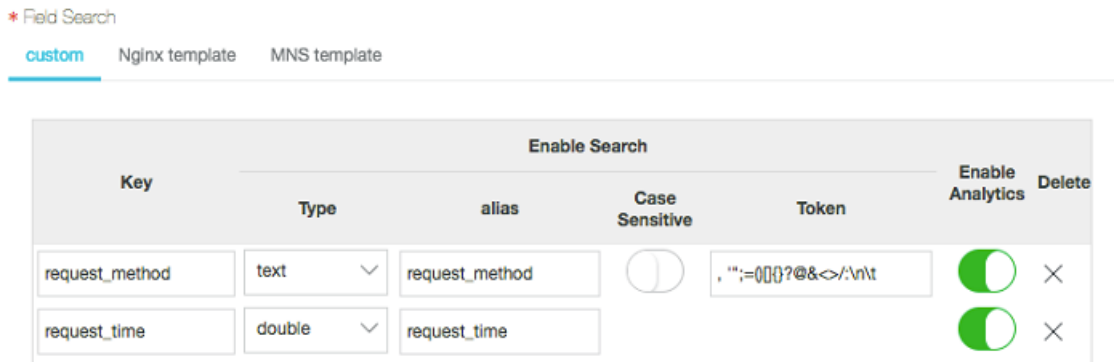
前提条件

クイック分析を使用する前に、フィールドクエリプロパティを指定する必要があります。

1. 指定されたフィールド照会では、照会および分析機能を活動化するために索引を使用可能にする必要があります。 インデックスを有効にする方法については、クエリと分析を参照してください。
2. ログに `key` をフィールド名として設定し、型、エイリアス、セパレータを設定します。

   アクセスログに `request_me thod` と `request_ti me` フィールドがある場合、以下の設定を行うことができます。

   図 6-6 : 前提条件

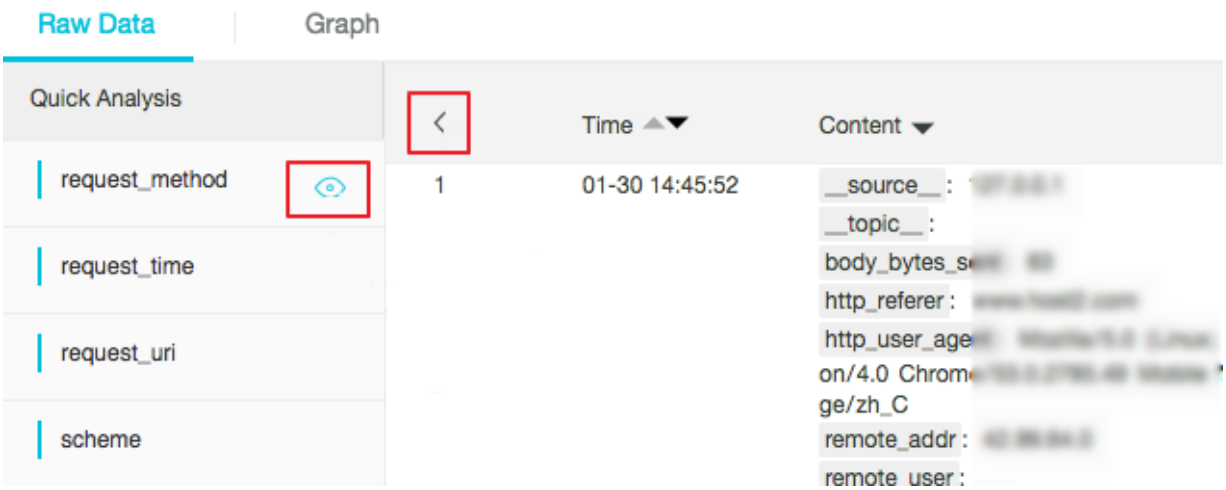


ユーザーガイド

指定したフィールドクエリを設定すると、クエリページのRaw Dataタブの Quick Analysis フィールドが表示されます。 1 ボタンをクリックするとページを折り畳むことができます。 目ボ

タンを使用すると、現在の時間間隔および現在の $Search 条件に基づいたクイック解析を行います。

図 6-7 : オリジナルログ



テキスト

- テキストフィールドの統計情報のグループ化

  ファイルの右側にある目ボタンをクリックすると、この Text フィールドの最初の 1000 個のデータをグループ化し、最初の 10 個の割合を返します。

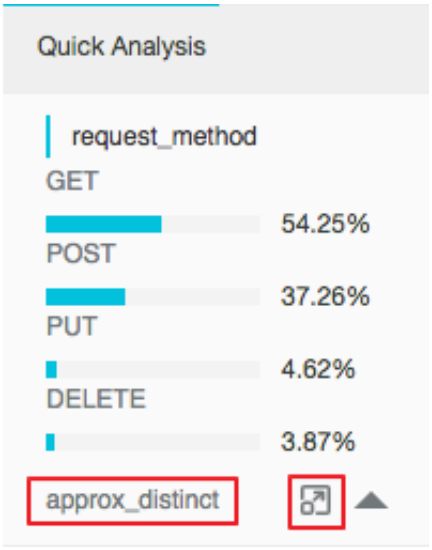
  クエリ文：

```
$ Search | select ${ keyName } , pv , pv * 1 . 0 / sum ( pv )
 over () as percentage from ( select count ( 1 ) as pv ,
 "${ keyName }" from ( select "${ keyName }" from log limit
```

```
    100000 )  group   by  "${ keyName }"  order   by   pv   desc )
 order   by   pv   desc   limit   10
```

`request_me thod` は、GET リクエストが多数存在するグループ化統計に基づいて次の結果を返します。

図 6-8 : グループ化統計



- フィールドのユニークなエントリの数を確認する

  Quick Analysis の対象フィールドの下で、approx_distinct をクリックして、`${ keyName }` の一意のエントリの数を確認します。

  `request_me thod` は、統計をグループ化することによって次の結果を得ることができ、`GET` リクエストは多数を占めます。

- グループ化統計のクエリステートメントを検索ボックスに拡張する

  approx_distinct の右側にあるボタンをクリックして、グループ化統計のクエリステートメントを検索ボックスに拡張し、その後の操作を行います。

long/double

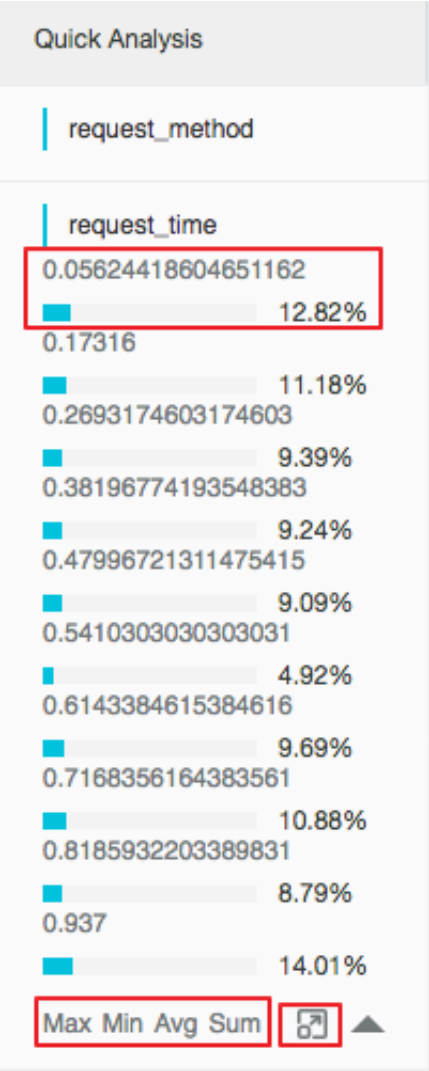
- 近似分布のヒストグラム統計情報

  統計のグループ化は、複数の型の値を持つ `long / double` フィールドにとってはほとんど意味がありません。 したがって、近似分布のヒストグラム統計は、 10 個のバケットを使用して採用されます。

```
$ Search | select numeric_histogram ( 10 , ${ keyName })
```

  `request_time` はおおよその分布のヒストグラム統計に基づいて次の結果を返します。ここから要求時間はほとんど 0.056 の周りに分布しています。

  図 6-9 : リクエスト配信



- `Max` , `Min` , `Avg` , `Sum` ステートメントの簡単な分析

  ターゲットフィールドの `Max` 、 `Min` 、 `Avg` 、 `Sum` をそれぞれクリックすると、${keyName} の最大値、最小値、平均値、合計値をすばやく検索できます。

- グループ化統計のクエリステートメントを検索ボックスに拡張する

  `Sum` の右側にあるボタンをクリックすると、近似分布のヒストグラム統計の照会ステートメントを検索ボックスに拡張し、以降の操作を行うことができます。

6.6.6 その他機能

ステートメントベースのクエリ機能に加えて、Log Service のクエリおよび分析機能は、クエリ最適化のために次の拡張機能を提供します。

- 生ログ
- グラフ
- コンテキスト検索
- クイック分析
- クイック検索
- タグ
- ダッシュボード
- アラームとして保存

生ログ

インデックスが有効になったら、検索ボックスにキーワードを入力して検索時間範囲を選択します。 次に、検索をクリックして、ログ量、生ログ、および統計グラフのヒストグラムを表示します。

ログ量のヒストグラムは、ログ検索のヒット数の時間ベースの分布を表示します。 ヒストグラムを使用することで、一定期間におけるログ量の変化を確認できます。 長方形領域をクリックして時間範囲を絞り込むと、指定した時間範囲内のログヒットに関する情報を表示して、ログ検索結果の表示を絞り込むことができます。

[生データ]タブでは、ヒットログを時間順で表示できます。

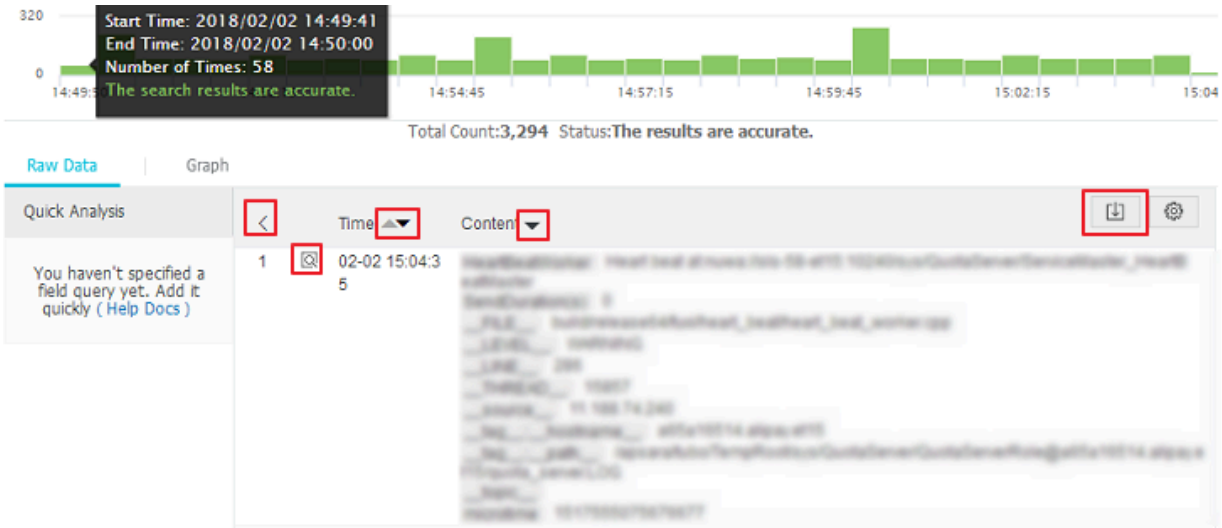
- 時間の横にある三角形の記号をクリックすると、時間順と逆時間順を切り替えることができます。
- コンテンツの横にある三角形の記号をクリックすると、改行で表示と 1行で表示を切り替えることができます。
- ログコンテンツの value キーワードをクリックすると、このキーワードを含むすべてのログを表示できます。
- [生データ]タブの右上隅にあるダウンロードボタンをクリックすると、クエリ結果を CSV 形式でダウンロードできます。 構成ボタンをクリックすると、生ログの表示結果に表示列として

フィールドを追加できるため、新しい生ログの各生ログの対象フィールドの内容をより直感的に確認できます。

- コンテキストをクリックすると、現在のログエントリの前後のそれぞれ15個のログを表示できます。 詳細は、コンテキストクエリを参照してください。

> **注：**
> 現在、コンテキスト検索機能は Logtail でアップロードされたデータのみをサポートしています。

図 6-10 : 生ログ



グラフ

インデックスを有効にして検索と分析のためのステートメントを入力すると、グラフタブでログの統計を表示できます。

- データは、表、折れ線グラフ、縦棒グラフ、横棒グラフ、円グラフ、数値、面グラフ、およびマップの方法で表示できます。

  実際の統計分析の必要に応じて適切な統計グラフタイプを選択できます。
- ニーズに合った表示結果を得るために、X 軸と Y 軸の表示内容を調整できます。

・ 分析結果をダッシュボードに追加します。 詳細は、ダッシュボードを参照してください。

図 6-11 : ダッシュボード

コンテキスト検索

Log Service コンソールには検索ページがあり、コンソールで元のファイルで指定したログのコンテキスト情報を表示できます。 元のログファイルを上下にページをめくるのと似ています。 指定されたログのコンテキスト情報を表示することで、ビジネスのトラブルシューティング中に障害情報を素早く見つけることができます。 詳細は、コンテキストクエリを参照してください。

クイック分析

Log Service のクイック分析機能は、 ワンクリックで対話式の検索をサポートします。これにより、一定期間にわたるフィールドの分布を素早く分析し、キーデータの索引付けのコストを削減することができます。 詳細は、クイック解析を参照してください。

クイック検索

検索ページの右上隅にあるクイック検索として保存をクリックすると、現在の検索アクションをクイック検索として保存できます。 この検索を再度実行するには、検索ステートメントを手動で入力せずに、左側のクイック検索タブですばやく実行できます。

このクイック検索条件をアラームルールで使用することもできます。 このクイック検索を タグに追加した場合は、タグで直接アクセスできます。

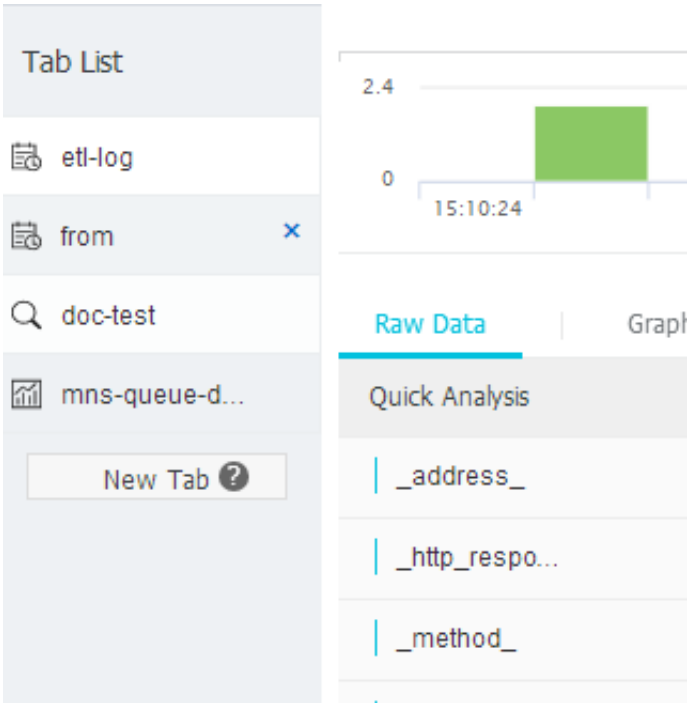
タグ

Log Service 検索ホームページの左側にあるタグリストには、次の 3 種類のデータページを追加できます。

- Logstore
- クイック検索
- ダッシュボード

タグリストを使用すると、簡単かつ迅速にページを開くことができます。 直接クリックして Logstore、クイック検索、およびダッシュボードをタグリストに開くことができます。 Logstore、クイック検索、またはダッシュボードをタグとして追加するには、タグリストのタグの追加をクリックして、ページの右側に表示されるメニューで追加する Logstore、クイック検索、またはダッシュボードを選択します。 タグを削除するには、タグリストで削除するタグ名の右側にある削除（X）ボタンをクリックします。

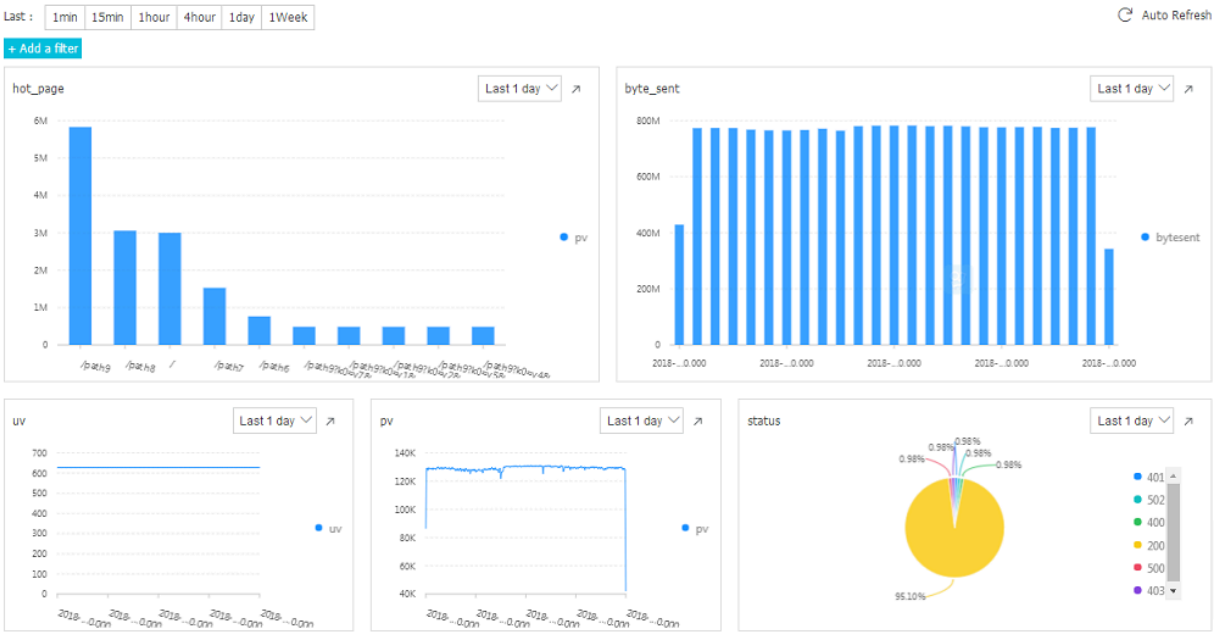
図 6-12 : タグ



ダッシュボード

Log Service はダッシュボード機能を提供します。これは検索と分析ステートメントを可視化することができます。 詳細は、ダッシュボードを参照してください。

図 6-13 : ダッシュボード



アラームとして保存

Log Service は LogSearch Results に基づいてアラームを生成することができます。 アラームルールを構成して、サイト内の通知または DingTalk メッセージを使用して特定のアラームコンテンツを送信できるようにします。

基本的な手順は次の通りです：

1. クイック検索を設定する。
2. アラームルールを設定する。
3. 通知タイプを設定する。
4. アラームの結果を確認できるように、システムは SMS / E メールを送信します。

詳細は、アラームの設定を参照してください。

6.7 SQL分析文法及び機能

6.7.1 共通集計関数

Log Service のクエリと分析機能は、一般的な集計関数を使用してログを分析することをサポートします。 具体的な記述と意味は次のとおりです。

ステートメント	説明	例
`arbitrary ( x )`	列 x の値をランダムに返します。	`latency > 100 | select arbitrary ( method )`
`avg ( x )`	列 x のすべての値の算術平均を計算します。	`latency > 100 | select avg ( latency )`
`checksum ( x )`	列のすべての値のチェックサムを計算し、base64 でエンコードされた値を返します。	`latency > 100 | select checksum ( method )`
`count (*)`	列の行数を計算します。	-
`count ( x )`	列内の NULL 以外の値の数を計算します。	`latency > 100 | count ( method )`
`count_if ( X )`	X = true の数を計算します。	`latency > 100 | count ( url like '%abc ')`
`geometric_ mean ( x )`	列内のすべての値の幾何平均を計算します。	`latency > 100 | select geometric_ mean ( latency )`
`max_by ( x , y )`	カラム y が最大値を持つときのカラム x の値を返します。	最大レイテンシのメソッド: `latency > 100 | select max_by ( method , latency )`
`max_by ( x , y , n )`	列 y の最大値を持つ n 行に対応する列 x の値を返します。	最大待ち時間を持つ上位3行のメソッド: `latency > 100 | select max_by ( method , latency , 3 )`
`min_by ( x , y )`	列 y が最小値を持つときの列 x の値を返します。	最小待ち時間のメソッド: `* | select min_by ( x , y )`
`min_by ( X , Y , n )`	列 y の最小値を持つ n 行に対応する列 x の値を返します。	最小待ち時間を持つ上位 3 行のメソッド:`* | select min_by ( method , latency , 3 )`

ステートメント	説明	例
`max ( x )`	最大値を返します。	`latency > 100 | select max ( inflow )`
`min ( x )`	最小値を返します。	`latency > 100 | select min ( inflow )`
`sum ( x )`	列 x のすべての値の合計を返します。	`latency > 10 | select sum ( inflow )`
`bitwise_an d_agg ( x )`	列のすべての値に対して AND 演算を実行します。	-
`bitwise_or _agg ( x )`	列内のすべての値に対して OR 演算を行います。	-

6.7.2 セキュリティ検知関数

グローバルホワイトハット共有セキュリティ資産ライブラリに基づき、Log Service はセキュリティ検出関数を提供します。 セキュリティ検出機能にログの IP アドレス、ドメイン名、または URL を渡すだけで、安全であるかどうかを検出することができます。

シナリオ

1. インターネット、ゲーム、情報などの企業のような、サービスの運用と保守に強い需要がある企業や機関。 これらの業界の IT およびセキュリティ運用保守（O&M）担当者は、セキュリティ検出関数を使用して、疑わしいアクセス、攻撃、および侵入をタイムリーにフィルタリングできます。 セキュリティ検出関数は、さらなる詳細な分析とそれらに対する防御策もサポートします。
2. 銀行、証券、電子商取引など、内部資産保護に対する強い需要がある企業および機関。 IT とセキュリティの保守人員は、危険な Web サイトへの内部アクセス、トロイの木馬のダウンロードなどの行為を即座に発見し、すぐに行動を起こすことができます。

機能の特徴

- 信頼性が高い：グローバル共有ホワイトハットセキュリティ資産ライブラリに依存し、タイムリーな更新を行っています。
- 高速：数百万の IP アドレス、ドメイン名、または URL を検出するのにわずか数秒かかります。

- 簡単：あらゆるネットワークログをシームレスにサポートします。 security_check_ip、security_check_domain、およびsecurity_check_urlの3つのSQL関数を呼び出すことによってすぐに結果を取得できます。
- 柔軟性：対話式クエリとレポートビューの構築の両方をサポートします。 アラームを構成して、さらに対策を講じることができます。

関数一覧

関数名	説明	例
security_check_ip	IPアドレスが安全かどうかを確認します。その中でも： - 1 が返される場合：命中しました、安全ではないことを示します - 0 が返される場合：ミス、安全であることを示します	`select security_check_ip ( real_client_ip )`
security_check_domain	ドメインが安全かどうかを確認します。その中でも： - 1 が返される場合：命中しました、安全ではないことを示します - 0 が返される場合：ミス、安全であることを示します	`select security_check_domain ( site )`
security_check_url	URL が安全かどうかを確認します。その中でも： - 1 が返される場合：命中しました、安全ではないことを示します - 0 が返される場合：ミス、安全であることを示します	`select security_check_domain ( concat ( host , url )`

例

- 外部の不審なアクセス行動を確認してレポートを生成する

  とある電子商取引は、それが運営する Nginx サーバーのログを収集し、安全でないクライアント IP アドレスが存在するかどうかを確認するためにサーバーにアクセスするクライアントをスキャンします。 この場合、Ngnix ログの ClientIP フィールドを `security_c`

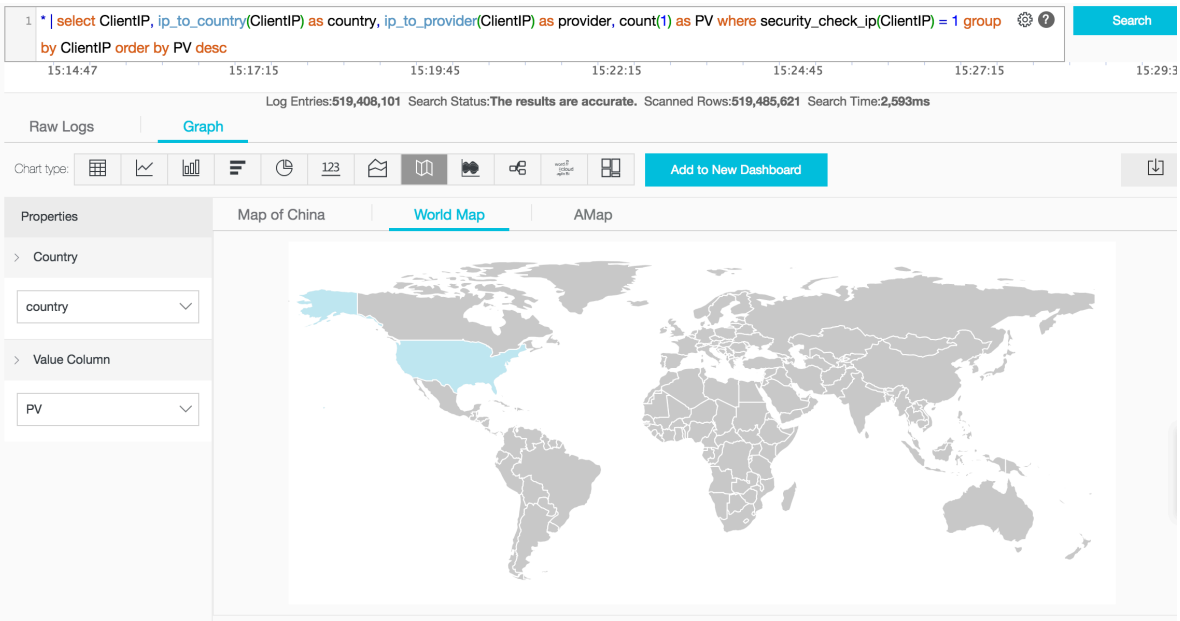
heck_ip 関数に渡し、戻り値が 1 である IP アドレスを表示し、国、ネットワーク事業者、およびその他の IP アドレスの関連情報を表示します。

検索分析ステートメントは次のとおりです：

```
* | select ClientIP , ip_to_coun try ( ClientIP ) as country , ip_to_prov ider ( ClientIP ) as provider , count ( 1 ) as PV where security_c heck_ip ( ClientIP ) = 1 group by ClientIP order by PV desc
```

ClientIP	sec	country	provider	PV
[redacted]	1	中国	电信	575
[redacted]	1	中国	联通	241
[redacted]	1	中国	电信	185
[redacted]	1	中国	联通	179
[redacted]	1	中国	联通	32
[redacted]	1	中国	电信	28

マップビュー表示に設定します:



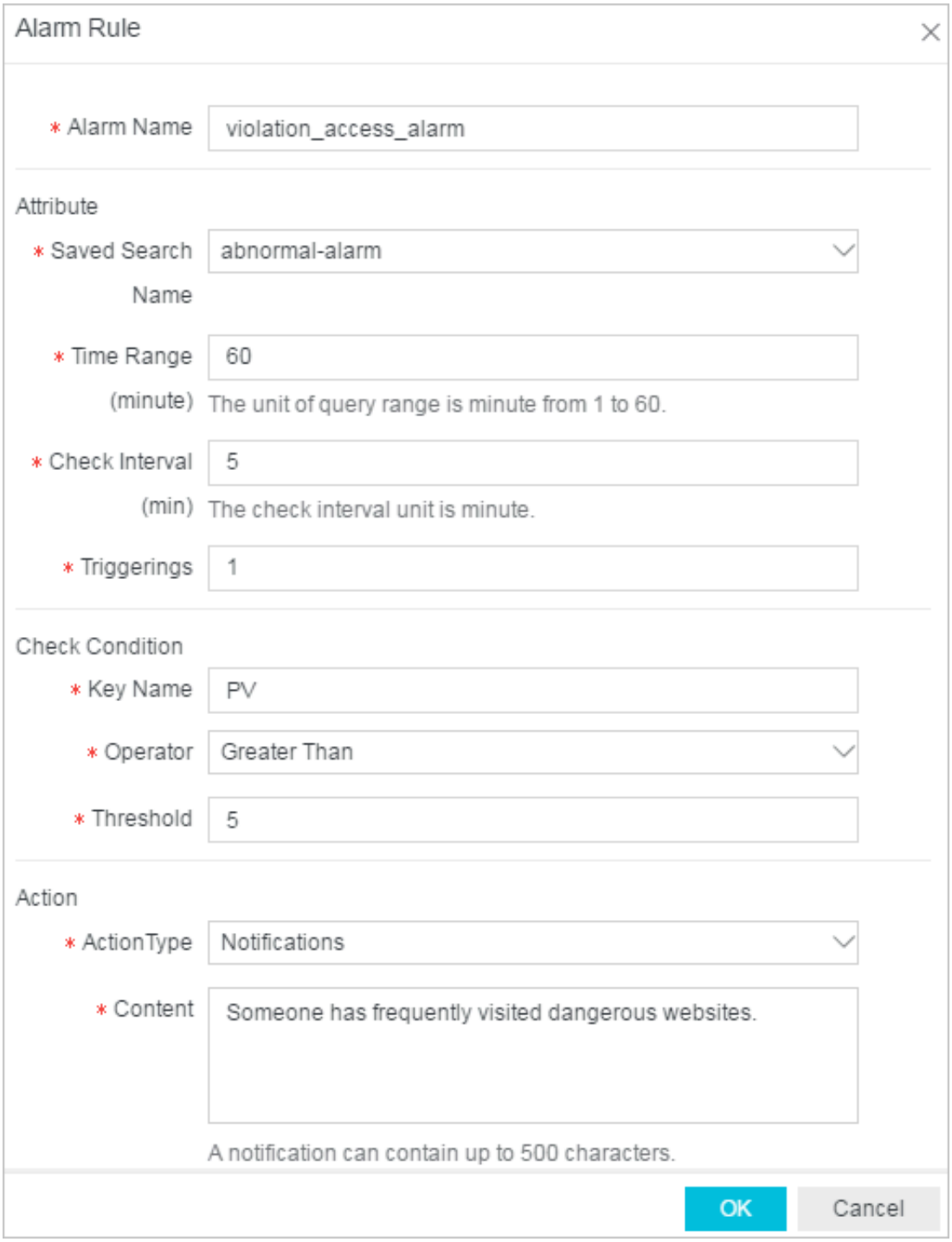
- 内部の不審なアクセス動作を確認してアラームを構成する

  たとえば、とある証券運用者は、その内部デバイスがゲートウェイプロキシを通じて外部ネットワークにアクセスしたときに記録されたネットワークトラフィックログを収集します。 誰かが問題のある Web サイトにアクセスしたかどうかを確認するには、次の検索を実行します。

```
* | select client_ip , count ( 1 ) as PV where security_c heck_ip ( remote_add r ) = 1 or security_c
```

```
heck_site ( site ) =  1   or   security_c  heck_url ( concat ( site
,  url )) =  1   group   by   client_ip   order   by   PV   desc
```

このステートメントをクイック検索として保存し、セキュリティアラームを構成することもできます。 とあるクライアントが危険な Web サイトに頻繁にアクセスすると、アラームが発生します。 過去 1 時間に誰かが危険な Web サイトに頻繁に（5 回以上）アクセスしたかどうかを確認するために 5 分ごとにチェックするようアラームを設定します。 必要に応じてパラメータを変更します。 次のとおりに構成します：



6.7.3 マッピング関数

Log Service のクエリと分析機能は、マッピング機能を使用してログの分析をサポートします。具体的な記述と意味は次のとおりです。

ステートメント	意味	例
Subscript operator []	マップ内のキーの結果を取得します。	-
histogram(x)	列 x の各値に従って GROUP BY を実行し、カウントを計算します。 構文は `select count group by x` の形となります。。	`latency > 10 | latency > 10 | histogram ( status )`と `latency > 10 | select count ( 1 ) group by status` は同じです。
map_agg(Key,Value)	key/value 配列を使用して作成されたマッピングを返します。	`latency > 100 | select map_agg ( method , latency )`
multimap_agg(Key,Value)	key/value 配列を使用して作成された複数値マッピングを返します。	`latency > 100 | select multimap_a gg ( method , latency )`
cardinality(x) → bigint	マップのサイズを取得します。	-
element_at(map`< K , V >`, key) → V	キーに対応する値を取得します。	-
map() → map`< unknown , unknown >`	空のマップを返します。	-
map(array`< K >`, array`< V >`) → map`< K , V >`	2 つの配列を 1 対 1 のマップに変換します。	`SELECT map ( ARRAY [ 1 , 3 ], ARRAY [ 2 , 4 ]); — { 1 -> 2 , 3 -> 4 }`
map_from_entries(array`< row < K , V >>`) → map`< K , V >`	多次元配列をマップに変換します。	`SELECT map_from_e ntries ( ARRAY [( 1 , ‘ x ’), ( 2 , ‘ y ’)]); — { 1 -> ‘ x ’, 2 -> ‘ y ’}`

ステートメント	意味	例
map_entries(map< K , V >) → array< row < K , V >>	1つの要素を1つの配列に変換します。	SELECT map_entrie s (MAP (ARRAY [1 , 2], ARRAY [‘ x ’, ‘ y ’])); — [ROW (1 , ‘ x ’), ROW (2 , ‘ y ’)]
map_concat(map1< K , V >, map2< K , V >, …, mapN< K , V >) → map< K , V >	複数のマップの結合が必要です。キーが複数のマップに存在する場合は、最初のものを使ってください。	-
map_filter(map< K , V >, function) → map< K , V >	lambda map_filter 関数を参照してください。	-
transform_keys(map< K1 , V >, function) → MAP< K2 , V >	lambda transform_keys 関数を参照してください。	-
transform_values(map< K , V1 >, function) → MAP< K , V2 >	lambda transform_values 関数を参照してください。	-
map_keys(x< K , V >) → array< K >	マップ内のすべてのキーを取得し、配列を返します。	-
map_values(x< K , V >) → array< V >	マップ内のすべての値を取得し、配列を返します。	-
map_zip_with(map< K , V1 >, map< K , V2 >, function< K , V1 , V2 , V3 >) → map< K , V3 >	lambda の指数関数を参照してください。	-

6.7.4 推定関数

Log Service の検索と分析機能は、推定関数を使用したログの分析をサポートしています。 具体的な記述と意味は次の通りです。

ステートメント	説明	例
approx_dis tinct (x)	列 x の固有値の数を推定します。	-

ステートメント	説明	例
`approx_per centile ( x , percentage )`	列 x をソートし、おおむね percentage に位置する値を返します	半分の位置の値を返します。`approx_per centile ( x 、 0 . 5 )`
`approx_per centile ( x , percentage s )`	前のステートメントと似ていますが、複数の percentage を特定して、それらの percentage 位置の値を返します。	`approx_per centile ( x , array [ 0 . 1 , 0 . 2 ])`
`numeric_hi stogram ( buckets , Value )`	複数バケット内で値列の統計情報を作成します。 値列をバケット数に相応した数に分割し、各バケットの key と count を返します。その値は `select count group by` と相当します。	POST リクエストの場合、遅延を10個のバケットに分割し、各バケットのサイズを返します：`method : POST | select numeric_hi stogram ( 10 , latency )`

6.7.5 数学的統計関数

LogService のクエリと解析機能は、数学統計関数を使用してログの解析をサポートします。 具体的な記述と意味は次のとおりです。

ステートメント	意味	例
`corr ( y , x )`	2 つの列の相関係数を返します。 結果は 0 から 1 です。	`latency > 100 | select corr ( latency , request_si ze )`
`covar_pop ( y , x )`	母集団共分散を計算します。	`latency > 100 | select covar_pop ( request_si ze , latency )`
`covar_samp ( y , x )`	サンプルの共分散を計算します。	`Latency > 100 | select covar_samp ( request_si ze , latency )`
`regr_inter cept ( y , x )`	入力値の線形回帰切片を返します。 y は従属値です。 x は独立した値です。	`latency > 100 | select regr_inter cept ( request_si ze , latency )`

ステートメント	意味	例
`regr_slope ( y , x )`	入力値の線形回帰勾配を返します。 y は従属値です。 x は独立した値です。	`latency > 100 | select regr_slope ( request_si ze , latency )`
`stddev ( x )` または `stddev_sam p ( x )`	列 x の標本標準偏差を返します。	`latency > 100 | select stddev ( latency )`
`stddev_pop ( x )`	列xの母集団標準偏差を返します。	`latency > 100 | select stddev_pop ( latency )`
`variance ( x )`または `Var_samp ( X )`	列 x のサンプル分散を計算します。	`latency > 100 | select variance ( latency )`
`var_pop ( x )`	列 x の母集団の分散を計算します。	`latency > 100 | select variance ( latency )`

6.7.6 数学計算関数

Log Service のクエリと解析機能は、数学的計算機能を使用してログの解析をサポートします。クエリ文と数学的計算関数を組み合わせることで、ログクエリ結果に対して数学的計算を実行できます。

数学演算子

数学演算子は、SELECT 句で使用できるプラス記号（+）、マイナス記号（-）、乗算記号（*）、除算記号（/）、パーセント記号（%）をサポートします。

例：

```
*| select avg ( latency )/ 100 , sum ( latency )/ count ( 1 )
```

数学関数

Log Service は、以下の操作機能をサポートしています。

関数名	意味
`abs ( x )`	列 x の絶対値を返します。
`Cbrt ( X )`	列 x の立方根を返します。

関数名	意味
`ceiling ( x )`	列 x の最も近い整数に切り上げられた数値を返します。
`cosine_sim ilarity ( x , y )`	スパースベクトル x と y のコサイン類似度を返します。
`degrees`	ラジアンを度に変換します。
`e ()`	自然な定数を返します。
`exp ( x )`	自然数の指数を返します。
`floor ( x )`	列xの最も近い整数に切り捨てられた数値を返します。
`from_base ( string , radix )`	ベース - 基数表記で解釈される文字列を返します。
`ln ( x )`	自然対数を返します。
`log2 ( x )`	x の底 2 の対数を返します。
`log10 ( x )`	x の底 10 の対数を返します。
`log ( x , b )`	x のベース b の対数を返します。
`pi ()`	π を返します。
`pow ( x , b )`	x を b の累乗に戻します。
`radians ( x )`	度をラジアンに変換します。
`rand ()`	乱数を返します。
`random ( 0 , n )`	[0、n]の範囲の乱数を返します。
`round ( x )`	最も近い整数に丸められたxを返します。
`round ( x , y )`	最も近い整数に丸められたxを返します。
`sqrt ( x )`	x の平方根を返します。
`to_base ( x , radix )`	x の基数 - 基数表現を返します。
`truncate ( x )`	小数点以下を切り捨てて、x を整数に丸めます。
`acos ( x )`	アークコサインを返します。
`Asin ( X )`	逆正弦を返します。
`atan ( x )`	アークタンジェントを返します。
`atan2 ( y , x )`	y / x の逆正接を返します。

関数名	意味
cos (x)	コサインを返します。
sin (x)	サインを返します。
cosh (x)	双曲線コサインを返します。
tan (x)	タンジェントを返します。
tanh (x)	双曲線タンジェントを返します。
Infinity ()	double の最大値を返します。
is_infinit y (x)	それが最大値であるかどうかを決定します。
is_finity (x)	それが最大値であるかどうかを決定します。
is_nan (x)	数値であるかどうかを判定します。

6.7.7 文字列関数

Log Service のクエリと分析機能は、文字列関数を使用してログを分析することをサポートします。 具体的な記述と説明は次のとおりです。

関数名	説明
chr (x)	int 型を対応する unicode 文字列に変換します。たとえば、chr(65)=’A’ です。
length (x)	フィールドの長さを返します。
levenshtei n_distance (string1 , string2)	2 つの文字列間の最小編集距離を返します。
lower (string)	文字列を小文字に変換します。
lpad (string , size , padstring)	文字列をサイズに合わせます。 サイズより小さい場合は、左側から塗りつぶしを使用します。 サイズよりも大きければ、サイズに合わせて切り取ってください。
rpad (string , size , padstring)	lpad と似ていますが、右側から文字列が入ります。
ltrim (string)	左の空白文字を削除します。
replace (string , search)	文字列から検索を削除します。
replace (string , search , rep)	文字列の検索を rep に置き換えます。
reverse (string)	逆順で文字列を返します。
rtrim (string)	文字列の最後にある空白文字を削除します。

関数名	説明
`split ( string , delimeter , limit )`	文字列を配列に分割します。多くても限界値が得られます。 生成された結果は、添字が 1 から始まる配列です。
`split_part ( string , delimeter , offset )`	文字列を配列に分割し、番号オフセット文字列を取得します。 生成された結果は、添字が 1 から始まる配列です。
`split_to_m ap ( string , entryDelim iter , keyValueDe limiter ) → map < varchar , varchar >`	entryDelemiter によれば、文字列はいくつかの項目に分割され、各項目は keyValueDe limiter に従ってキー値に分割されます 最後に、マップが返されます。
`position ( substring IN string )`	部分文字列の先頭を探します。
`strpos ( string , substring )`	文字列中の部分文字列の開始位置を見つけます。 返される結果は 1 から始まります。 見つからなければ、0 が返されます。
`substr ( string , start )`	添え字が 1 から始まる文字列の部分文字列を返します。
`substr ( string , start , length )`	添字が 1 から length までの文字列の部分文字列を返します。
`trim ( string )`	文字列の先頭と末尾の空白文字を削除します。
`upper ( string )`	文字列を大文字に変換します。
`concat ( string , string ......)`	2 つ以上の文字列を 1 つの文字列に分割します。
`hamming_di stance ( string1 , string2 )`	2 つの文字列のハミング距離を返します。

注：

文字列は一重引用符、列名は二重引用符で囲んでいます。 たとえば、 a=’abc’ は列 a = 文字列 abc を意味し、a = “abc” は列 a = 列 abc を意味します。

6.7.8 日付と時間の関数

Log Service は、時間関数、日付関数、及び間隔関数をサポートしています。 このドキュメントで紹介した日付、時刻、および間隔関数を分析構文で使用できます。

日時タイプ

1. unixtime：int 型で 1970 年 1 月 1 日からの秒数を示します。 例えば、 `1512374067` は、 `Mon Dec 4 15 : 54 : 27 CST 2017` の時刻を示し, 各ログの Log Service 組み込み時刻 `__time__` は上記タイプのものです。
2. timestamp type: 時刻を文字列形式で示します。 たとえば、 `2017 - 11 - 01 13 : 30 : 00` のようになります。

日付関数

Log Service サポートされている共通の日付機能関数は、次のとおりです。

関数	説明	例
`current_da te`	現在の日付を返します。	`latency > 100 | select current_da te`
`current_ti me`	現在の時間を返します。	`latency > 100 | select current_ti me`
`current_ti mestamp`	current_date と current_time を組み合わせた結果を返します。	`latency > 100 | select current_ti mestamp`
`current_ti mezone ()`	タイムゾーンを返します。	`latency > 100 | select current_ti mezone ()`
`from_iso86 01_timesta mp ( string )`	iso8601 時刻をタイムゾーンの時刻に変換します。	`latency > 100 | select from_iso86 01_timesta mp ( iso8601 )`
`from_iso86 01_date ( string )`	iso8601 時刻を日付に変換します。	`latency > 100 | select from_iso86 01_date ( iso8601 )`
`from_unixt ime ( unixtime )`	UNIX の時刻をタイムスタンプに変換します。	`latency > 100 | select from_unixt ime ( 1494985275 )`

関数	説明	例
from_unixt ime (unixtime , string)	文字列をタイムゾーンとして使用して、UNIX の時刻をタイムスタンプに変換します。	latency > 100 | select from_unixt ime (1494985275 ,' Asia / Shanghai ')
localtime	現在の時刻を返します。	latency > 100 | select localtime
localtimes tamp	現在のタイムスタンプを返します。	latency > 100 | select localtimes tamp
now ()	current_ti mestamp に相当	-
to_unixtim e (timestamp)	タイムスタンプを UNIX 時刻に変換します。	*| select to_unixtim e (' 2017 - 05 - 17 09 : 45 : 00 . 848 Asia / Shanghai ')

時間関数

MySQL の時刻形式

Log Service は、%a、%b、%y などの MySQL の時刻形式をサポートしています。

関数	意味	例
date_forma t (timestamp , format)	形式を使用して表現されるタイムスタンプを変換します。	latency > 100 | select date_forma t (date_parse (' 2017 - 05 - 17 09 : 45 : 00 ','% Y -% m -% d % H :% i :% S '), '% Y -% m -% d ') group by method
date_parse (string , format)	形式を使用して文字列をタイムスタンプに解析します。	latency > 100 | select date_parse (' 2017 - 05 - 17 09 : 45 : 00 ','% Y -% m -% d % H :% i :% S ') group by method

表 6-2 : 説明

形式	説明
%a	1 週間のうちの 1 日の省略形（日..土）.
%b	月の略語（Jan..Dec）。
%c	数字のタイプの月（1 .. 12）[4]。
%D	接尾辞付きの月の日（0 番目、1 番目、2 番目、3 番目、…）。
%d	日（01 .. 31）[4]。
%e	日（01 .. 31）[4]。
%H	時（00 .. 23）。
%h	時（01 .. 12）。
%I	12 時間形式の時間（01 … 12）。
%i	分（00 .. 59）。
%j	年の日（001 .. 366）。
%k	時（0 .. 23）。
%l	時（1 .. 12）。
%M	英語で表示される月 (January .. December).
%m	月の数字（01 .. 12）[4]。
%p	午前または午後。
%r	12 時間形式の時刻。形式は、`hh : mm : ss 、 AM / PM` 。
% S	秒(00 .. 59).
%s	秒(00 .. 59).
%T	24 時間形式の時刻（`hh : mm : ss`）。
%U	一年の週（00 .. 53）。 毎週の最初の日は日曜日です。 値の範囲：00~53。
%u	一年の週（00 .. 53）。 毎週の最初の日は日曜日です。 値の範囲：00~53。
%V	一年の週（00 .. 53）。 毎週の最初の日は日曜日です。 値の範囲：01~53。 %X と組み合わせてこの形式を使用してください。

形式	説明
%v	一年の週（00 .. 53）。 毎週の最初の日は日曜日です。 値の範囲：01~53。 %x と組み合わせてこの形式を使用してください。
%W	曜日の名前（日曜日..土曜日）。
%w	曜日（0 .. 6）。 日曜日は 0 日です。
%Y	年。（4 桁）
%y	年。（2 桁）
%%	% エスケープ文字

時間軸合わせ機能

Log Service は、秒、分、時、日、月、年順に沿って期間の整列機能をサポートしています。 期間整列機能は、統計は時間に応じた際に、時間軸合わせ機能が使用されます

関数の構文:

```
 date_trunc ( unit ,  x )
```

パラメータ:

Unit のオプションの値は次のとおりです (x は `2001 - 08 - 22  03 : 04 : 05 . 000`):

単位	変換された結果
second	2001-08-22 03:04:05.000
minute	2001-08-22 03:04:00.000
hour	2001-08-22 03:00:00.000
day	2001-08-22 00:00:00.000
week	2001-08-20 00:00:00.000
month	2001-08-01 00:00:00.000
quarter	2001-07-01 00:00:00.000
year	2001-01-01 00:00:00.000

x は、タイムスタンプ型または UNIX の 時刻型にすることができます。

date_trunc 一定の期間ごとにしか統計を作成できません。 たとえば、柔軟な時間ディメンションに従って統計を作成する必要がある場合は、統計を 5 分ごとにわたって作成し、数学的モジュラス法に従って GROUP BY を実行します。

```
* | SELECT count ( 1 ) as pv , __time__ - __time__ % 300
 as minute5gro upby minute5 limit 100
```

`% 300` は 5 分ごとにモジュラスとアライメントを行うことを示します。

日付関数の例

時刻形式を使用した例：

```
*| select date_trunc (' minute ' , __time__ ) as t ,
        truncate ( avg ( latency ) ) ,
        current_da te
        group by t
        order by t desc
        limit 60
```

間隔関数

間隔関数は、間隔関連の計算を実行するために使用されます。 たとえば、日付の間隔を追加または削除するか、または 2 つの日付間の時間を計算します。

関数	説明	例
`date_add (unit, value, timestamp)`	`timestamp` に `value` `unit` を追加します。 マイナス計算を実行するには、負の `value` を使用します。	`date_add (' day ', - 7 , ' 2018 - 08 - 09 00 : 00 : 00 ')`は、8 月 9 日の 7 日前を示します。
`date_diff(unit, timestamp1, timestamp2)`	`timestamp1` と `timestamp2` の間の `unit` の数。	`date_diff (' day ', ' 2018 - 08 - 02 00 : 00 : 00 ', ' 2018 - 08 - 09 00 : 00 : 00 ') = 7`

この関数は、次の間隔単位をサポートします。

単位	説明
millisecond	ミリ秒
second	秒
minute	分
hour	時間
day	日

単位	説明
week	週
month	月
quarter	四半期、すなわち 3 ヶ月。
year	年

6.7.9 URL 関数

URL 関数は、標準の URL パスからフィールドを抽出することをサポートしています。 標準の URL は次のとおりです。

```
[ protocol :][// host [: port ]][ path ][?  query ][# fragment ]
```

一般的な URL 関数

関数名	意味	例
`url_extrac t_fragment ( url )`	URL からフラグメントを抽出し、結果は varchar 型です。	`*| select url_extrac t_fragment ( url )`
`url_extrac t_host ( url )`	URL からホストを抽出し、結果は varchar 型です。	`*| select url_extrac t_host ( url )`
`url_extrac t_paramete r ( url , name )`	クエリの name パラメータの値を URL から抽出し、結果は varchar 型です。	`*| select url_extrac t_paramete r ( url )`
`url_extrac t_path ( url )`	URL からパスを抽出し、結果は varchar 型です。	`*| select url_extrac t_path ( url )`
`url_extrac t_port ( url )`	URL からポートを抽出し、結果は bigint 型です。	`*| select url_extrac t_port ( url )`
`url_extrac t_protocol ( url )`	URL からプロトコルを抽出し、結果は varchar 型です。	`*| select url_extrac t_protocol ( url )`
`url_extrac t_query ( url )`	URL からクエリを抽出し、結果は varchar 型です。	`*| select url_extrac t_query ( url )`
`url_encode ( value )`	URL をエンコードします。	`*| select url_encode ( url )`
`url_decode ( value )`	URL をデコードします。	`*| select url_decode ( url )`

6.7.10 正規表現関数

正規表現関数は文字列を解析し、必要な部分文字列を返します。

共通正規表現関数とその意味は次のとおりです。

関数名	説明	例
`regexp_extract_all(string, pattern)`	文字列内の正規表現に一致するすべての部分文字列を文字列配列として返します。	`*| SELECT regexp_extract_all('5a 67b 890m', '\d+')`の結果：`['5','67','890']`, `*| SELECT regexp_extract_all('5a 67a 890m', '(\d+)a')`の結果：`['5a','67a']`.
`regexp_extract_all(string, pattern, group)`	グループの正規（）部分に当たる文字列の部分を返し、その結果を文字列の配列として返します。	`*| ` SELECT regexp_extract_all(‘5a 67a 890m’, ‘(\d+)a’, 1)` の結果：`[‘5’,’67’]`
`regexp_extract(string, pattern)`	文字列の正規表現に一致する最初の部分文字列を返します。	`*| SELECT regexp_extract('5a 67b 890m', '\d+')` の結果：`'5'`
`regexp_extract(string, pattern, group)`	文字列にヒットした通常の group（）内の最初の部分文字列を返します。	`*| SELECT regexp_extract('5a 67b 890m', '(\d+)([a-z]+)', 2)` の結果：`'b'`
`regexp_like(string, pattern)`	文字列が正規表現に一致するかどうかを判定し、bool 結果を返します。 正規表現は、文字列の一部に一致するように許可されています。	`*| SELECT regexp_like('5a 67b 890m', '\d+m')` true が返されます。
`regexp_replace(string, pattern, replacement)`	文字列内の正規表現に一致する部分を replacement に置き換えます。	`*| SELECT regexp_replace('5a 67b 890m', '\d+','a')` の結果：`'aa ab am'`
`regexp_replace(string, pattern)`	`regexp_replace(string, patterm,'')`に相当する文字列の正規表現にマッチする部分を削除します。	`*| SELECT regexp_replace('5a 67b 890m', '\d+')` の結果：`'a b m'`

関数名	説明	例
`regexp_spl it ( string , pattern )`	正規表現を使用して文字列を配列に分割します。	`*| SELECT regexp_spl it ('5a 67b 890m ', '\ d +')` の結果：`[' a ',' b ',' m ']`

6.7.11 JOSN 関数

JSON 関数は文字列を JSON 型として解析し、JSON でフィールドを抽出できます。 JSON の主な構造は、マップと配列の 2 つです。 文字列が JSON 型として解析されない場合、返される値は null です。

JSON を複数の行に分割するには、UNNEST 関数を参照してください

Log Service は、以下の共通の JSON 関数をサポートしています。

関数名	説明	例
`json_parse ( string )`	文字列を JSON 型に変換します。	`SELECT json_parse ('[ 1 , 2 , 3 ]')` returns a JSON array
`json_forma t ( json )`	JSON タイプを文字列に変換します。	`SELECT json_forma t ( json_parse ('[ 1 , 2 , 3 ]'))` returns a string
`json_array _contains ( json , value )`	JSON 型の値または文字列（内容が JSON 配列）に値が含まれているかどうかを判別します。	`SELECT json_array _contains ( json_parse ('[ 1 , 2 , 3 ]'), 2 ) or SELECT json_array _contains ('[ 1 , 2 , 3 ]', 2 )`
`json_array _get ( json_array , index )`	JSON 配列の添字の要素を取得するために使用される `json_array _contains` と同じです。	`SELECT json_array _get ('[" a ", " b ", " c "]', 0 )` returns `' a '`
`json_array _length ( json )`	JSON 配列のサイズを返します。	`SELECT json_array _length ('[ 1 , 2 , 3 ]')` Returns 3

関数名	説明	例
`json_extra ct ( json , json_path )`	JSON オブジェクトから値を抽出します。JSON のパス構文は`$. store . book [ 0 ]. title` に似ています。返される結果は JSON オブジェクトです。	`SELECT json_extra ct ( json , '$. store . book ');`
`json_extra ct_scalar ( json , json_path )`	`json_extra ct` に似ていますが、文字列を返します。	-
`json_size ( json , json_path )`	JSON オブジェクトまたは配列のサイズを取得します。	`Select json_size ('[ 1 , 2 , 3 ]') returns 3`

6.7.12 型変換関数

Log Service は、構成内の long 、 double 、および text 型、クエリの bigint 、 double 、 varchar 、 timestamp 、および int 型をサポートします。

型変換関数は、列を指定された型に強制的に変換します。

```
cast ( value AS type ) → type
try_cast ( value AS type ) → type
```

6.7.13 IP 機能

IP 認識機能は、IP がイントラネット IP であるかインターネット IP であるかを認識し、IP が属する国、地域、都市を決定することができます。

関数名	説明	例
`ip_to_doma in ( ip )`	IP が存在するドメインと、その IP がイントラネット IP かインターネット IP かを決定します。戻り値は、イントラネットまたはインターネットです。	`SELECT ip_to_doma in ( ip )`
`ip_to_coun try ( ip )`	IP が存在する国を決定します。	`SELECT ip_to_coun try ( ip )`
`ip_to_prov ince ( ip )`	IP が存在する省を決定します。IP が海外に存在する場合、国名が返されます。	`SELECT ip_to_prov ince ( ip )`

関数名	説明	例
ip_to_city (ip)	IP が存在する都市を決定します。IP が海外に存在する場合、国名が返されます。	SELECT ip_to_city (ip)
ip_to_geo (ip)	IP が存在する都市の経度と緯度を決定します。範囲の結果は緯度と経度の形式になります。	SELECT ip_to_geo (ip)
ip_to_city _geo (ip)	IP が存在する都市の経度と緯度を決定します。 都市の緯度と経度を返します。各都市は1つの緯度と経度しか持ちません。 範囲の結果は緯度と経度の形式になります。	SELECT ip_to_city _geo (ip)
ip_to_prov ider (ip)	IP のネットワークオペレータを取得します。	SELECT ip_to_prov ider (ip)
ip_to_coun try (ip ,' en ')	IP が存在する国を特定し、国際コードを返します。	SELECT ip_to_coun try (ip ,' en ')
ip_to_coun try_code (ip)	IP が存在する国を特定し、国際コードを返します。	SELECT ip_to_coun try_code (ip)
ip_to_prov ince (ip ,' en ')	IP が存在する地域を決定し、英語の州名または中国語のアルファベットを返します。	SELECT ip_to_prov ince (ip ,' en ')
ip_to_city (ip ,' en ')	IP が存在する都市を決定し、英語の都市名または中国語アルファベットを返します。	SELECT ip_to_city (ip ,' en ')

例

- クエリ内のイントラネットアクセスリクエストを除外し、リクエストの総数を表示する

```
* | selectcoun t ( 1 ) whereip_to _domain ( ip )! =' intranet '
```

- トップ 10 のアクセス地域を表示

```
* | SELECT count ( 1 ) as pv , ip_to_prov ince ( ip ) as
 province GROUP BY province order by pv desc limit
 10
```

応答結果の例:

```
[
    {
```

```
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " Zhejiang   province ",
        " pv ": " 4045 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " Shanghai   city ",
        " pv ": " 3727 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " Beijing   city ",
        " pv ": " 954 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " intranet   IP ",
        " pv ": " 698 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " Guangdong   Province  ",
        " pv ": " 472 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ":  Fujian   Province  ",
        " pv ": " 71 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " United   ArabEmirat  es  ( UAE )",
        " pv ": " 52 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " United   States  ",
        " pv ": " 43 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " Germany  ",
        " pv ": " 26 "
    }, {
        " __source__ ": "",
        " __time__ ": " 1512353137 ",
        " province ": " Kuala   Lumpur  ",
        " pv ": " 26 "
    }
]
```


前述の結果には、イントラネット IP が含まれます。 開発者がイントラネットからテストを行うことがあります。 これらのアクセスリクエストを除外するには、次の分析構文を使用します。

- イントラネットリクエストをフィルタリングし、トップ 10 のネットワークアクセスの省を表示する

```
* | SELECT count ( 1 ) as pv , ip_to_prov ince ( ip ) as province WHERE ip_to_doma in ( ip ) ! = ' intranet ' GROUP BY province ORDER BY pv desc limit 10
```

- 各国の平均応答レイテンシ、最大応答レイテンシ、最大レイテンシリクエストを確認する

```
* | SELECT AVG ( latency ), MAX ( latency ), MAX_BY ( requestId , latency ), ip_to_coun try ( ip ) as country group by country limit 100
```

- 異なるネットワーク事業者の平均待ち時間を確認する

```
* | SELECT AVG ( latency ), ip_to_prov ider ( ip ) as provider group by provider limit 100
```

- IP の緯度と経度を表示し、地図を作成する

```
* | select count ( 1 ) as pv , ip_to_geo ( ip ) as geo group by geo order by pv desc
```

応答結果の例:

pv	geo
100	35.3284,-80.7459

6.7.14 GROUP BY の構文

GROUP BY は複数の列をサポートし、SELECT 列の別名を使用して対応する KEY を示します。

例：

```
method : PostLogsto reLogs | select avg ( latency ), projectNam e , date_trunc (' hour ', __time__ ) as hour group by projectNam e , hour
```

エイリアス hour は、3 番目の SELECT 列 `date_trunc (' hour ', __time__ )`を表します。 このような使用法は、非常に複雑な query には非常に役立ちます。

GROUP BY は GROUPING SETS, CUBE, 及び ROLLUP をサポートしています。

例：

```
method : PostLogsto reLogs | select avg ( latency ) group by cube ( projectNam e , logstore )
method : PostLogsto reLogs | select avg ( latency ) group by GROUPING SETS ( ( projectNam e , logstore ), ( projectNam e , method ))
```

```
method : PostLogsto reLogs | select avg ( latency ) group by
rollup ( projectNam e , logstore )
```

実践例

時刻に従って GROUP BY を実行する

各ログには組込みの時間列 `__time__` があります。 任意の列の統計機能が有効になっていると、統計は自動的に時間列に対して作成されます。

`date_trunc` 関数を使用して、時間列を時、分、日、月、および年に合わせます。

`date_trunc` は、整列単位と、UNIX の時刻または timestamp 型の列 `__time__` などを受け入れます。

・ 1 時間または 1 分ごとに PV を数えて計算する

```
* | SELECT count ( 1 ) as pv , date_trunc (' hour ',
__time__ ) as hour group by hour order by hour
limit 100
* | SELECT count ( 1 ) as pv , date_trunc (' minute ',
__time__ ) as minute group by minute order by minute
limit 100
```

注 :

limit 100 は、最大 100 行を取得することを示します。 LIMIT ステートメントを追加しなければ、デフォルトで最大 10 行のデータを取得します。

・ 柔軟な時間ディメンションに従って統計を作成します。 たとえば、5 分ごとに統計を作成します。 date_trunc は一定期間ごとにのみ統計を作成できるため、 数学的係数法に従って GROUP BY を 実行します。

```
* | SELECT count ( 1 ) as pv , __time__ - __time__ % 300
as minute5 group by minute5 limit 100
```

%300 は、5 分ごとにモジュラスとアライメントを作成することを示します。

GROUP BY で非 agg 列を抽出する

標準 SQL では、GROUP BY 構文を使用すると、SELECT を実行する際に、SELECT GROUP BY 列の元の内容のみ選択できます。あるいは、任意の列で集約計算を実行する際に非 GROUP BY 列の内容を 取得することはできません。

たとえば、次の構文は無効です それは b が非 GROUP BY 列であり、a に従って GROUP BY を実行すると、b の複数行が使用可能になるためです。システムは、どの行の出力を選択すれば良いのかを認識できません。

```
*| select   a ,  b   ,  count ( c )  group   by   a
```

上記を達成するには、arbitrary 関数を使用して b を出力します。

```
*| select   a ,  arbitrary ( b ),  count ( c )  group   by   a
```

6.7.15 ウィンドウ関数

ウィンドウ関数は、行間の計算に使用されます。 一般的な SQL 集計関数は、1 つの行のみの結果を計算するか、すべての行を 1 つの行に集約して計算します。 ウィンドウ関数は行間計算をサポートし、各行に計算結果を入力します。

ウィンドウ関数の構文：

```
SELECT   key1 ,  key2 ,  value ,
       rank ()  OVER  ( PARTITION   BY   key2
                    ORDER   BY   value   DESC )  AS   rnk
FROM   orders
ORDER   BY   key1 , rnk
```

コア部分は次のとおりです。

```
rank ()  OVER  ( PARTITION   BY   KEY1   ORDER   BY   KEY2   DESC )
```

rank() は集合関数です。 分析構文や本書に記載されている関数内の関数を使用することができます。 PARTITION BY は、どの値が計算されるかに基づいてバケットを示します。

Windows で使われる特別な集計関数

関数名	説明
rank()	ウィンドウ内の特定の列に基づいてデータをソートし、ウィンドウ内のシリアル番号を返します。
row_number()	ウィンドウ内の行番号を返します。
first_value(x)	ウィンドウ内の最初の値を返します。 一般的に、値がウィンドウ内でソートされた後に最大値を取得するために使用されます。
last_value(x)	first_value の反対です。
nth_value(x, offset)	ウィンドウ内の x 番目の列の No. オフセット行の値。

関数名	説明
lead(x,offset,defaut_value)	ウィンドウの x 番目の列の特定の行の後にあるオフセット行の値。その行が存在しない場合は、default_value を使用します。
lag(x,offset,defaut_value)	ウィンドウ内の x 番目の列の特定の行の前のオフセット行の値。その行が存在しない場合は、default_value を使用します。

例

- それぞれの部門の従業員の給与をランク付けする

```
* | select department , persionId , sallary , rank () over
 ( PARTITION BY department order by sallary desc ) as
 sallary_ra nk order by department , sallary_ra nk
```

応答結果：

部門	個人 ID	給料	給料_ランク
dev	john	9000	1
dev	Smith	8000	2
dev	Snow	7000	3
dev	Achilles	6000	4
Marketing	Blan Stark	9000	1
Marketing	Rob Stark	8000	2
Marketing	Sansa Stark	7000	3

- 従業員の給与を各部門のパーセンテージとして計算する

```
* | select department , persionId , sallary * 1 . 0 / sum (
 sallary ) over ( PARTITION BY department ) as sallary_pe
 rcentage
```

応答結果：

department	persionId	sallary	sallary_percentage
dev	john	9000	0.3
dev	Smith	8000	0.26
dev	Snow	7000	0.23
dev	Achilles	6000	0.2
Marketing	Blan Stark	9000	0.375

department	persionId	sallary	sallary_percentage
Marketing	Rob Stark	8000	0.333
Marketing	Sansa Stark	7000	0.29

- 前日より日次 UV 増加を計算する

```
* | select day , uv , uv * 1 . 0 /( lag ( uv , 1 , 0 ) over
 () ) as diff_perce ntage from

 select approx_dis tinct ( ip ) as uv , date_trunc (' day ',
 __time__ ) as day from log group by day order by
 day asc
```

応答結果：

day	uv	diff_percentage
2017-12-01 00:00:00	100	null
2017-12-02 00:00:00	125	1.25
2017-12-03 00:00:00	150	1.2
2017-12-04 00:00:00	175	1.16
2017-12-05 00:00:00	200	1.14
2017-12-06 00:00:00	225	1.125
2017-12-07 00:00:00	250	1.11

6.7.16 HAVING 構文

Log Service のクエリと分析機能は、標準 SQL の構文をサポートしています。これは、GROUP BY 構文と一緒に使用され、GROUP BY 結果をフィルタリングします。

形式：

```
method : PostLogsto reLogs | select avg ( latency ), projectNam
e group by projectNam e HAVING avg ( latency ) > 100
```

HAVING と WHERE の区別

HAVING は、GROUP BY の実行後に集計および計算結果をフィルタリングするために使用されます。 WHERE は、集計計算中に元のデータをフィルタリングするために使用されます。

例

気温が 10℃ を超える各州の平均降水量を計算し、最終降水量が 100mL を超える地域のみを表示する：

```
* | select avg ( rain ) , province where temperatur e > 10
group by province having avg ( rain ) > 100
```

6.7.17 ORDER BY 構文

ORDER BY は出力結果をソートするために使用されます。 現在、結果は 1 つの列でのみソートできます。

構文形式：

```
order by Column name [ desc | asc ]
```

例：

```
method : PostLogsto reLogs | select avg ( latency ) as
avg_latenc y , projectNam e group by projectNam e
HAVING avg ( latency ) > 5700000
order by avg_latenc y desc
```

6.7.18 LIMIT 構文

LIMIT の後に数字が続き、出力結果の最大行数を示すために使用されます。

構文の形式：

Log Service は次の 2 種類の LIMIT 構文の形式に対応しています。

- 最初の N 行のみを読取る：

  ```
  limit N
  ```

- S 行から始まり、N 行を読み取る：

  ```
  limit S , N
  ```

注：

- LIMIT 構文を使用してページ間で結果を読み取る場合は、最終結果のみを取得するために使用され、SQL の途中で結果を取得するためには使用できません。

- サブクエリでは LIMIT 構文を使用できません。例えば：

```
* | select count ( 1 ) from ( select distinct ( url ) from
   limit 0 , 1000 )
```

例

- 結果を 100 行のみ取得します：

```
* | select distinct ( url ) from log limit 100
```

- 0 行~ 999 行、合計 1000 行の結果を取得します：

```
* | select distinct ( url ) from log limit 0 , 1000
```

- 1,000 行~ 1,999 行、合計 1,000 行の結果を取得します：

```
* | select distinct ( url ) from log limit 1000 , 1000
```

6.7.19 CASE WHEN 構文

Log Service 構文が連続したデータを分類する場合をサポートしています。例えば、HTTP_USER_AGENT から情報を抽出し、二つのタイプに情報を分類：Android と iOS。

```
SELECT
 CASE
 WHEN http_user_ agent like '% android %' then ' android '
 WHEN http_user_ agent like '% ios %' then ' ios '
 ELSE ' unknown ' END
 as http_user_ agent ,
   count ( 1 ) as pv
   group by http_user_ agent
```

例

- 要求の合計数に対する演算ステータスコードとして 200 とリクエストの割合：

```
* | SELECT
  sum (
  CASE
  WHEN status = 200 then 1
  ELSE 0 end
 ) * 1 . 0 / count ( 1 ) as status_200 _percentag e
```

- 異なる待ち時間の間隔の分布の統計を作成します

```
* | SELECT `
  CASE
  WHEN latency < 10 then ' s10 '
  WHEN latency < 100 then ' s100 '
  WHEN latency < 1000 then ' s1000 '
  WHEN latency < 10000 then ' s10000 '
  else ' s_large ' end
  as latency_sl ot ,
  count ( 1 ) as pv
```

```
  group  by  latency_sl  ot
```

構文 IF

構文は、CASE WHEN 構文と論理的に等価である場合。

```
Case
     WHEN  condition  THEN  true_value
    [  ELSE  false_valu  e  ]
 END
```

- IF（条件、 true_value）

  条件が true の場合、列 true_value が、そうでない場合は null を返します。

- IF（条件、true_value、 false_value）

  条件が true の場合、列 true_value は、そうでない場合は、列 false_value が返され、返されます。

構文 COALESCE

合体は、複数の列の最初の非ヌル値を戻します。

```
Coalesce  ( value1 ,  value2  [,...])
```

NULLIF 構文

value1 と value2 が等しい場合、null がそう VALUE1 が返され、返されます。

```
nullif ( value1 ,  value2 )
```

TRY 構文

トライ構文は null 値を返すために、このよう 0 エラーなど基礎となる例外のいくつかをキャッチすることができます。

```
try ( expression )
```

6.7.20 ネストされたサブクエリ

複雑なクエリのシナリオでは、SQL ネストされたクエリを使用して、1 レベルの SQL が要件を満たせない場合の複雑な要件を満たすことができます。

ネストされたサブクエリとネストされていないクエリの違いは、SQL 文で from 条件を指定する必要があることです。 クエリのキーワード `from  log` を指定すると、ログから元のデータを読み取ることができます。

例：

```
* |  select  sum ( pv )  from
```

```
(
 select   count ( 1 )  as   pv   from   log   group   by   method
)
```

6.7.21 配列

ステートメント	意味	例
添字演算子[]	[] は、配列内の特定の要素を取得するために使用されます。	-

連結演算子 ||	|| は、2 つの配列を 1 つに連結するために使用されます。	`SELECT ARRAY [ 1 ] || ARRAY [ 2 ]; — [ 1 , 2 ]`
		`SELECT ARRAY [ 1 ] || 2 ; — [ 1 , 2 ]`
		`SELECT 2 || ARRAY [ 1 ]; — [ 2 , 1 ]`

array_distinct	配列の重複排除を使用して、配列内の異なる要素を取得します。	-
array_intersect(x, y)	配列 x と y の交点を取得します。	-
array_union(x, y) → array	配列 x と y の和集合を取得します。	-
array_except(x, y) → array	配列 x と y の減算を取得します。	-
array_join(x, delimiter , null_replacement) → varchar	文字列配列を区切り文字で連結し、null 値をnull_replacement に置き換えます。	-
array_max(x) → x	配列 x の最大値を取得します。	
array_min(x) → x	配列 x の最小値を取得します。	-
array_position(x, element) → bigint	配列 x の要素の添え字を取得します。添字は 1 から始まります。添え字が見つからない場合は 0 が返されます。	-

ステートメント	意味	例
Array_remove (x, element)-array	配列から要素を削除します。	-
array_sort(x) → array	配列をソートし、ヌル値を最後まで移動します。	-
cardinality(x) → bigint	配列のサイズを取得します。	-
concat(array1, array2, ···, arrayN) → array	配列を連結します。	-
contains(x, element) → boolean	配列 x に要素が含まれている場合は TRUE を返します。	-
これはラムダ関数です。Lambda の filter() をご参照ください。	2 次元配列を 1 次元配列に連結します。	-
flatten(x) → array	2 次元配列を 1 次元配列に連結します。	-
reduce(array, initialState, inputFunction, outputFunction) → x	Lambda 関数の reduce() 関数をご参照ください。	-
reverse(x) → array	配列 x を逆順に並べ替えます。	-
sequence(start, stop) → array	シーケンスを start からstop まで生成し、各ステップを 1 ずつインクリメントします。	-
sequence(start, stop, step) → array	シーケンスを start からstop まで生成し、各ステップを指定されたステップ値だけインクリメントします。	-
sequence(start, stop, step) → array	start から stop までタイムスタンプ配列を生成します。Start と stop はタイムスタンプタイプです。 step はインターバルタイプで、DAY から SECOND までで、YEAR または MONTH でもかまいません。	-
shuffle(x) → array	配列をシャッフルします。	-

ステートメント	意味	例
slice(x, start, length) → array	配列 x の start から length 要素を持つ新しい配列を作成します。	-
transform(array, function) → array	Lambda 関数の transform() をご参照ください	-
zip(array1, array2[, …]) → array	複数の配列をマージします。結果の M 番目の要素の N 番目のパラメーターは、元の N 番目の配列の M 番目の要素です。これは、複数の配列を転置することと同じです	`SELECT zip ( ARRAY [ 1 , 2 ], ARRAY [‘ 1b ’, null , ‘ 3b ’]); – [ ROW ( 1 , ‘ 1b ’), ROW ( 2 , null ), ROW ( null , ‘ 3b ’)]`
zip_with(array1, array2, function) → array	Lambda の zip_with() をご参照ください。	-

6.7.22 バイナリ文字列関数

バイナリ文字列型 varbinary は、文字列型 varchar とは異なります。

ステートメント	説明
連結関数 ||	`a || b` の結果は `ab` です。
length(binary) → bigint	長さをバイナリで返します。
concat(binary1, …, binaryN) → varbinary	|| に相当するバイナリ文字列を連結します。
to_base64(binary) → varchar	バイナリ文字列を Base64 文字列に変換します。
from_base64(string) → varbinary	Base64 文字列をバイナリ文字列に変換します。
to_base64url(binary) → varchar	文字列を URL セーフ Base64 文字列に変換します。
from_base64url(string) → varbinary	URL セーフ Base64 文字列をバイナリ文字列に変換します。
to_hex(binary) → varchar	バイナリ文字列を 16 進文字列に変換します。
from_hex(string) → varbinary	16 進文字列をバイナリ文字列に変換します。
to_big_endian_64(bigint) → varbinary	ビッグエンディアンモードで数値をバイナリ文字列に変換します。
from_big_endian_64(binary) → bigint	ビッグエンディアンモードのバイナリ文字列を数値に変換します。

ステートメント	説明
md5(binary) → varbinary	バイナリ文字列の MD5 値を計算します。
sha1(binary) → varbinary	バイナリ文字列の SHA1 値を計算します。
sha256(binary) → varbinary	バイナリ文字列の SHA256 ハッシュ値を計算します。
sha512(binary) → varbinary	バイナリ文字列の SHA512 値を計算します。
xxhash64(binary) → varbinary	バイナリ文字列の xxhash64 値を計算します。

6.7.23 ビット操作

ステートメント	説明	例

bit_count(x, bits) → bigint	x の 2 進式での 1 の個数を数えます。	`SELECT bit_count ( 9 , 64 ); — 2`
		`SELECT bit_count ( 9 , 8 ); — 2`
		`SELECT bit_count (- 7 , 64 ); — 62`
		`SELECT bit_count (- 7 , 8 ); — 6`

bitwise_and(x, y) → bigint	バイナリ形式の x と y に対して AND 演算を実行します。	-
bitwise_not(x) → bigint	バイナリ形式で x のすべてのビットの反対の値を計算します。	-
bitwise_or(x, y) → bigint	バイナリ形式で x と y の OR 演算を実行します。	-
bitwise_xor(x, y) → bigint	バイナリ形式の x と y に対して XOR 演算を実行します。	-

6.7.24 区間値比較および周期性値比較関数

区間値比較および周期性値比較関数は、現在期間の計算結果と指定された前期間の計算結果を比較するために使用されます。

関数	説明	例
`compare ( value , time_windo w )`	この関数は、現在の期間において計算された値と time_window によって計算された値を比較します。 値のデータ型は double または long です。 time_window の単位は秒です。 戻り値は配列形式です。 可能な戻り値には、現在の値、time_window 前の値、および time_window 前の値に対する現在の値の比率が含まれます。	`*| select compare ( pv , 86400 ) from ( select count ( 1 ) as pv from log )`
`compare ( value , time_windo w1 , time_windo w2 )`	この関数は、現在の値を time_window1 と time_window2 の前の期間の値と比較します。 比較結果は JSON 配列形式であり、値は次の順序で配置する必要があります。[current value, value before time_window1, value before time_window2 , current value/value before time_window1, current value/value before time_window2].	`* | select compare ( pv , 86400 , 172800 ) from ( select count ( 1 ) as pv from log )`

関数	説明	例
`compare ( value , time_windo w1 , time_windo w2 , time_windo w3 )`	この関数は、現在の値を time_window1、time_window2、および time_window3 より前の期間の値と比較します。 比較結果は JSON 配列形式であり。値は次の順序で配置する必要があります。[current value, value before time_window1, value before time_window2, value before time_windo w3, current value/Value before time_window1 , current value/value before time_window2, current value/value before time_window3]。	`* | select compare ( pv , 86400 , 172800 , 604800 ) from ( select count ( 1 ) as pv from log )`
`compare_re sult ( value , time_windo w )`	この関数は `compare ( value , time_windo w )`と同様に機能します。 ただし、戻り値は `" Current value ( Increased percentage %)"`の形式の文字列型の値です。 増加したパーセント値は、デフォルトで小数点以下 2 桁に丸められます。	`* | select compare_re sult ( pv , 86400 ) from ( select count ( 1 ) as pv from log )`

関数	説明	例
`compare_re sult ( value , time_windo w1 , time_windo w2 )`	この関数は `compare ( value , time_windo w1 , time_windo w2 )`と同様に機能します。 ただし、戻り値は `" Current value ( Increased percentage compared with the first period %) ( Increased percentage compared with the second period )"`という形式の文字列型の値です。 増加したパーセント値は、デフォルトで小数点以下 2 桁に丸められます。	`* | select compare_re sult ( pv , 86400 , 172800 ) from ( select count ( 1 ) as pv from log )`

関数	説明	例
`ts_compare ( value , time_windo w )`	この関数は、現在の値を `time_windo w1` と `time_windo w2` より前の期間の値と比較します。 比較結果はJSON配列形式であり、値は次の順序で配置する必要があります。[current value, value before time_window1, current value/value before time_window1, unix timestamp at the start time of the previous period]。 この関数は時系列関数を比較するために使用されます。 これには、GROUP BY 操作を time 列の SQL ステートメントに含める必要があります。	例：`* | select t , ts_compare ( pv , 86400 ) as d from ( select date_trunc (' minute ', __time__ ) as t , count ( 1 ) as pv from log group by t order by t ) group by t` は、現在の期間の 1 分ごとの計算結果と最後の期間の 1 分ごとの計算結果を比較することを示します。 比較結果：`d :[ 1251 . 0 , 1264 . 0 , 0 . 9897151898 734177 , 1539843780 . 0 , 1539757380 . 0 ] t : 2018 - 10 - 19 14 : 23 : 00 . 000` 。

例

- 昨日と同じ期間に対する現在の時間の PV の比率を計算します。

  開始時間： 2018-07-25 14:00:00、終了時間： 2018-07-25 15:00:00。

  クエリと分析のステートメント：

```
* | select compare ( pv , 86400 ) from ( select count ( 1 ) as pv from log )
```

  86400 は、現在の期間から 86400 秒が減算されることを示します。

  戻り値：

```
[ 9 . 0 , 19 . 0 , 0 . 4736842105  2631579 ]
```

  そのうち：

  - 9.0 は、2018-07-25 14:00:00 から 2018-07-25 15:00:00 までの PV 値です。
  - 19.0 は、2018-07-24 14:00:00 から 2018-07-24 15:00:00 までの PV 値です。
  - 0.47368421052631579 は、前期間の PV 値に対する現在期間の PV 値の比率です。

  配列を 3 列の数値に展開するには、分析ステートメントは次のようになります：

```
* | select diff [ 1 ], diff [ 2 ], diff [ 3 ] from ( select compare ( pv , 86400 ) as diff from ( select count ( 1 ) as pv from log ))
```

- 現在の 1 時間ごとの PV と昨日と同じ期間の PV の比率を計算し、その結果を折れ線グラフで表示します。

  1. 昨日と同じ期間に対する現在の時間の毎分の PV の比率を計算します。 開始時間：2018-07-25 14:00:00、終了時間： 2018-07-25 15:00:00。

     クエリと分析のステートメント：

```
*| select t , compare ( pv , 86400 ) as diff from ( select count ( 1 ) as pv , date_forma t ( from_unixt ime ( __time__ ), '% H :% i ') as t from log group by t ) group by t order by t
```

     戻り値：

t	diff
14:00	[9520.0,7606.0,1.2516434393899554]
14:01	[8596.0,8553.0,1.0050274757395066]
14:02	[8722.0,8435.0,1.0340248962655603]

t	diff
14:03	[7499.0,5912.0,1.2684370771312586]

そのうち、t は `Hour : Minute` の形式で時刻を示します。 diff 列の内容は、次のものを含む配列です。

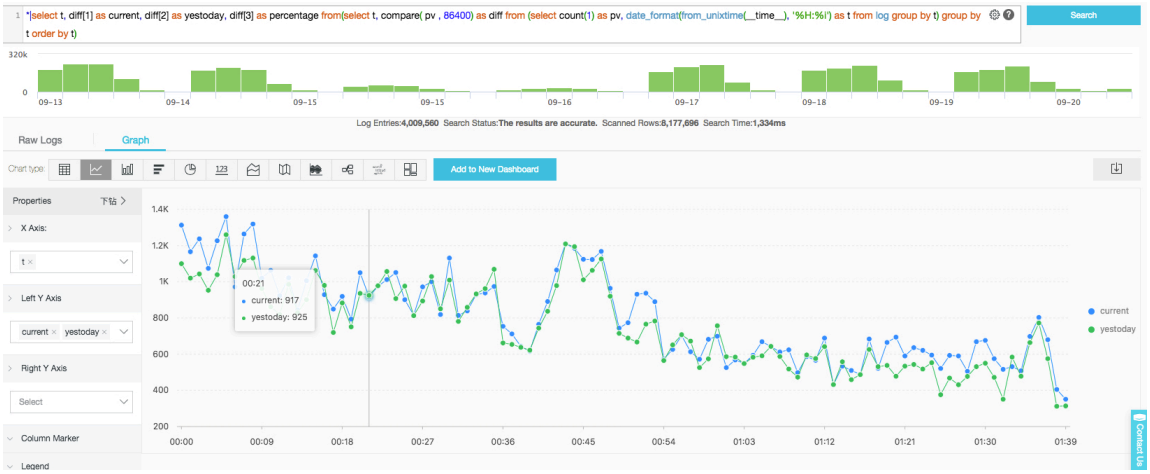
- 当期間の PV 値。
- 前期間の PV 値。
- 前期間の PV 値に対する当期間の PV 値の比率。

2. クエリ結果を折れ線グラフで表示するには、次のステートメントを使用します。

```
*| select t , diff [ 1 ] as current , diff [ 2 ] as
 yestoday , diff [ 3 ] as percentage from ( select t ,
 compare ( pv , 86400 ) as diff from ( select count ( 1
 ) as pv , date_forma t ( from_unixt ime ( __time__ ), '% H
 :% i ') as t from log group by t ) group by t
 order by t )
```

2 本の線は、今日と昨日の PV 値を示しています。

図 6-14 : 折れ線グラフ



6.7.25 比較関数と演算子

比較関数と演算子

比較演算は、int、bigint、double、および text などの比較可能な型に使用できる 2 つのパラメーターの値を比較します。

比較演算子

比較演算子を使用して、2 つのパラメーター値を比較します。比較の間、ロジックが真であれば TRUE が返されます。 それ以外の場合は、FALSE が返されます。

演算子	意味
<	より小さい
>	より大きい
<=	以下
>=	以上
=	等しい
<>	等しくない
！=	等しくない

範囲演算子 BETWEEN

BETWEEN は、パラメーター値が 2 つの他のパラメーターの値の間にあるかどうかを判定するために使用されます。 範囲は閉区間です。

- ロジックが真の場合、 TRUE が返されます。 それ以外の場合は、FALSE が返されます。

  例： `SELECT 3 BETWEEN 2 AND 6 ;` ロジックは真で TRUE が返されます。

  上記の例は、 `SELECT 3 >= 2 AND 3 <= 6 ;` と等価です 。
- BETWEEN は反対のロジックを判定するために NOT に続けることができます。

  例: `SELECT 3 NOT BETWEEN 2 AND 6 ;`. ロジックは偽で FALSE が返されます。

  上記の例は、 `SELECT 3 < 2 OR 3 > 6 ;` と等価です。
- パラメーターの値が NULL の場合、NULL が返されます。

IS NULL と IS NOT NULL

これらの演算子は、パラメーター値が NULL かどうかを判定するために使用されます。

IS DISTINCT FROM と IS NOT DISTINCT FROM

2 つの値が等しいかどうかを判定するのと同様ですが、これらの演算子は NULL 値が存在するかどうかを判定できます。

例：

```
SELECT NULL IS DISTINCT FROM NULL ; -- false
SELECT NULL IS NOT DISTINCT FROM NULL ; -- true
```

次の表からわかるように、ほとんどの場合、DISTINCT 演算子を使用してパラメーター値を比較できます。

a	b	a = b	a <> b	a DISTINCT b	NOT DISTINCT b
1	1	TRUE	FALSE	FALSE	TRUE
1	2	FALSE	TRUE	TRUE	FALSE
1	NULL	NULL	NULL	TRUE	FALSE
NULL	NULL	NULL	NULL	FALSE	TRUE

GREATEST と LEAST

これらの演算子は、複数の列間で最大値または最小値を取得するために使用されます。

例：

```
select   greatest ( 1 , 2 , 3 ) ; --  3   is   returned .
```

比較条件：ALL、ANY、および SOME

比較条件は、パラメーターが指定された条件を満たすかどうかを判定するために使用されます。

- ALL は、パラメーターがすべての条件を満たすかどうかを判定するために使用されます。 ロジックが真の場合、TRUE が返されます。 それ以外の場合は、 FALSE が返されます。
- ANY は、パラメーターがいずれかの条件を満たすかどうかを判定するために使用されます。 ロジックが真の場合、TRUE が返されます。 それ以外の場合は、 FALSE が返されます。
- ANY と同様に、SOME は、パラメーターがいずれかの条件を満たすかどうかを判定するために使用されます。
- ALL、ANY、および SOME は比較演算子の直後に続けなければなりません。

次の表に示すように、ALL と ANY は多くの場合、比較と判定をサポートします。

表現	意味
A = ALL (…)	A がすべての値と等しくない場合、TRUE が返されます。
A <> ALL (…)	A がすべての値と等しくない場合、TRUE が返されます。
A < ALL (…)	A がすべての値より小さい場合、TRUE が返されます。
A = ANY (…)	A が A IN (…) の任意の値と等しい場合、TRUE が返されます。
A <> ANY (…)	A が任意の値と等しくない場合、TRUE が返されます。

表現	意味
A < ANY (…)	A が最大値よりも小さい場合、TRUE が返されます。

例：

```
SELECT  ' hello ' =  ANY  ( VALUES  ' hello ', ' world '); --  true
SELECT   21  <  ALL  ( VALUES   19 ,  20 ,  21 ); --  false
SELECT   42  >=  SOME  ( SELECT   41   UNION   ALL   SELECT   42
UNION   ALL   SELECT   43 ); --  true
```

6.7.26 Lambda 関数

Lambda 式

Lambda 式は`->`で記述されます。

例：

```
 x  ->  x  +  1
( x ,  y ) ->  x  +  y
 x  ->  regexp_lik  e ( x , ' a +')
 x  ->  x [ 1 ] /  x [ 2 ]
 x  ->  IF ( x  >  0 ,  x , - x )
 x  ->  COALESCE ( x ,  0 )
 x  ->  CAST ( x   AS   JSON )
 x  ->  x  +  TRY ( 1  /  0 )
```

ほとんどの MySQL 式は Lambda で使用できます。

filter(array<T>, function<T, boolean>) → ARRAY<T>

配列からデータをフィルタリングし、関数が TRUE を返す要素だけを取得します。

例：

```
SELECT   filter ( ARRAY  [],  x  ->  true ); -- []
SELECT   filter ( ARRAY  [ 5 , - 6 ,  NULL ,  7 ],  x  ->  x  >  0 );
-- [ 5 ,  7 ]
SELECT   filter ( ARRAY  [ 5 ,  NULL ,  7 ,  NULL ],  x  ->  x   IS
NOT   NULL ); -- [ 5 ,  7 ]
```

map_filter(map<K, V>, function<K, V, boolean>) → MAP<K,V>

マップからデータをフィルタリングし、関数が TRUE を返す要素のペアのみを取得します。

例：

```
SELECT   map_filter ( MAP ( ARRAY [],  ARRAY []), ( k ,  v ) ->  true
); -- {}
SELECT   map_filter ( MAP ( ARRAY [ 10 ,  20 ,  30 ],  ARRAY [' a ',
NULL , ' c ']), ( k ,  v ) ->  v   IS   NOT   NULL ); -- { 10  ->  a
,  30  ->  c }
```

```
SELECT map_filter ( MAP ( ARRAY [' k1 ', ' k2 ', ' k3 '], ARRAY [
20 , 3 , 15 ]), ( k , v ) -> v > 10 ); -- { k1 -> 20 , k3 -
> 15 }
```

reduce(array<T>, initialState S, inputFunction<S, T, S>, outputFunction<S, R>) → R

reduce() 関数は、配列内の各要素を初期状態から順に走査し、状態 S に基づいて inputFunction(S,T) を計算し、新しい状態を生成します。 最後に outputFunction を適用して、最終状態 S を出力結果 R に変換します。

1. 初期状態 S。
2. 各要素 T を走査します。
3. inputFunction(S,T) を計算し、新しい状態 S を生成します。
4. 最後の要素が走査され、新しい状態が生成されるまで、手順 2 と 3 を繰り返します。
5. 最終状態 S を使用して最終出力結果 R を取得します。

例：

```
SELECT reduce ( ARRAY [], 0 , ( s , x ) -> s + x , s -> s
); -- 0
SELECT reduce ( ARRAY [ 5 , 20 , 50 ], 0 , ( s , x ) -> s +
x , s -> s ); -- 75
SELECT reduce ( ARRAY [ 5 , 20 , NULL , 50 ], 0 , ( s , x ) -
> s + x , s -> s ); -- NULL
SELECT reduce ( ARRAY [ 5 , 20 , NULL , 50 ], 0 , ( s , x ) -
> s + COALESCE ( x , 0 ), s -> s ); -- 75
SELECT reduce ( ARRAY [ 5 , 20 , NULL , 50 ], 0 , ( s , x ) -
> IF ( x IS NULL , s , s + x ), s -> s ); -- 75
SELECT reduce ( ARRAY [ 2147483647 , 1 ], CAST ( 0 AS
BIGINT ), ( s , x ) -> s + x , s -> s ); -- 2147483648
SELECT reduce ( ARRAY [ 5 , 6 , 10 , 20 ], -- calculates
arithmetic average : 10 . 25
             CAST ( ROW ( 0 . 0 , 0 ) AS ROW ( sum DOUBLE ,
count INTEGER )),
            ( s , x ) -> CAST ( ROW ( x + s . sum , s . count
+ 1 ) AS ROW ( sum DOUBLE , count INTEGER )),
             s -> IF ( s . count = 0 , NULL , s . sum / s .
count ));
```

transform(array<T>, function<T, U>) → ARRAY<U>

配列内の各要素の関数を呼び出し、新しい結果 U を生成します。

例：

```
SELECT transform ( ARRAY [], x -> x + 1 ); -- []
SELECT transform ( ARRAY [ 5 , 6 ], x -> x + 1 ); -- [ 6 ,
7 ] Increment each element by 1 .
SELECT transform ( ARRAY [ 5 , NULL , 6 ], x -> COALESCE ( x
, 0 ) + 1 ); -- [ 6 , 1 , 7 ]
SELECT transform ( ARRAY [' x ', ' abc ', ' z '], x -> x || '
0 '); -- [' x0 ', ' abc0 ', ' z0 ']
```

```
SELECT transform ( ARRAY [ ARRAY [ 1 , NULL , 2 ], ARRAY [ 3
, NULL ]], a -> filter ( a , x -> x IS NOT NULL )); --
[[ 1 , 2 ], [ 3 ]]
```

transform_keys(map<K1, V>, function<K1, V, K2>) → MAP<K2,V>

マップ内の各キーの関数を順番に適用して、新しいキーを生成します。

例：

```
SELECT transform_ keys ( MAP ( ARRAY [], ARRAY []), ( k , v ) -
> k + 1 ); -- {}
SELECT transform_ keys ( MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY ['
a ', ' b ', ' c ']), ( k , v ) -> k + 1 ); -- { 2 -> a , 3 -
> b , 4 -> c } Increment each key by 1 .
SELECT transform_ keys ( MAP ( ARRAY [' a ', ' b ', ' c '],
ARRAY [ 1 , 2 , 3 ]), ( k , v ) -> v * v ); -- { 1 -> 1 , 4
-> 2 , 9 -> 3 }
SELECT transform_ keys ( MAP ( ARRAY [' a ', ' b '], ARRAY [ 1
, 2 ]), ( k , v ) -> k || CAST ( v as VARCHAR )); -- { a1 -
> 1 , b2 -> 2 }
SELECT transform_ keys ( MAP ( ARRAY [ 1 , 2 ], ARRAY [ 1 . 0
, 1 . 4 ]), -- { one -> 1 . 0 , two -> 1 . 4 }
                    ( k , v ) -> MAP ( ARRAY [ 1 , 2 ], ARRAY ['
one ', ' two '])[ k ]);
```

transform_values(map<K, V1>, function<K, V1, V2>) → MAP<K, V2>

マップ内のすべての値に対して関数を適用し、V1 を V2 に変換し、新しいマップ `< K , V2 >` を生成します。

```
SELECT transform_ values ( MAP ( ARRAY [], ARRAY []), ( k , v )
-> v + 1 ); -- {}
SELECT transform_ values ( MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY [
10 , 20 , 30 ]), ( k , v ) -> v + 1 ); -- { 1 -> 11 , 2 ->
22 , 3 -> 33 }
SELECT transform_ values ( MAP ( ARRAY [ 1 , 2 , 3 ], ARRAY
[' a ', ' b ', ' c ']), ( k , v ) -> k * k ); -- { 1 -> 1 , 2
-> 4 , 3 -> 9 }
SELECT transform_ values ( MAP ( ARRAY [' a ', ' b '], ARRAY [
1 , 2 ]), ( k , v ) -> k || CAST ( v as VARCHAR )); -- { a
-> a1 , b -> b2 }
SELECT transform_ values ( MAP ( ARRAY [ 1 , 2 ], ARRAY [ 1 .
0 , 1 . 4 ]), -- { 1 -> one_1 . 0 , 2 -> two_1 . 4 }
                      ( k , v ) -> MAP ( ARRAY [ 1 , 2 ], ARRAY
[' one ', ' two '])[ k ] || ' _ ' || CAST ( v AS VARCHAR ));
```

zip_with(array<T>, array<U>, function<T, U, R>) → array<R>

2 つの配列をマージし、関数を使用して新しく生成された配列の要素を指定します。第 1 の配列の要素 T と、第 2 の配列の要素 U は、新しい結果 R を生成するために使用されます。

例：

```
SELECT zip_with ( ARRAY [ 1 , 3 , 5 ], ARRAY [' a ', ' b ', '
c '], ( x , y ) -> ( y , x )); -- Transpose the elements of
```

```
   the   two   arrays   to   generate   a   new   array .  Result : [
 ROW (' a ',  1 ),  ROW (' b ',  3 ),  ROW (' c ',  5 )]
 SELECT   zip_with ( ARRAY [ 1 ,  2 ],  ARRAY [ 3 ,  4 ], ( x ,  y ) -
>  x  +  y ); --  Result : [ 4 ,  6 ]
 SELECT   zip_with ( ARRAY [' a ', ' b ', ' c '],  ARRAY [' d ', ' e
 ', ' f '], ( x ,  y ) ->  concat ( x ,  y ));  Concatenat  e   the
 elements   of   the   two   arrays   to   generate   a   new   string
 .  Result : [' ad ', ' be ', ' cf ']
```

map_zip_with(map<K, V1>, map<K, V2>, function<K, V1, V2, V3>) → map<K, V3>

2 つのマップをマージし、値 V1 と V2 を使用して 各キーに基づいて V3 を生成し、新しいマップ `K , V3` を生成します。

```
SELECT   map_zip_wi  th ( MAP ( ARRAY [ 1 ,  2 ,  3 ],  ARRAY [' a ',
' b ', ' c ']),
                    MAP ( ARRAY [ 1 ,  2 ,  3 ],  ARRAY [' d ', ' e
', ' f ']),
                   ( k ,  v1 ,  v2 ) ->  concat ( v1 ,  v2 ));  Merge
  values   which   have   the   same   map   keys . -- { 1  ->  ad ,
2  ->  be ,  3  ->  cf }
SELECT   map_zip_wi  th ( MAP ( ARRAY [' k1 ', ' k2 '],  ARRAY [ 1 ,
2 ]),
                    MAP ( ARRAY [' k2 ', ' k3 '],  ARRAY [ 4 ,  9 ]),
                   ( k ,  v1 ,  v2 ) -> ( v1 ,  v2 ));  Generate   an
  array   by   using   the   two   values . -- { k1  ->  ROW ( 1 ,
null ),  k2  ->  ROW ( 2 ,  4 ),  k3  ->  ROW ( null ,  9 )}
SELECT   map_zip_wi  th ( MAP ( ARRAY [' a ', ' b ', ' c '],  ARRAY [
1 ,  8 ,  27 ]),
                    MAP ( ARRAY [' a ', ' b ', ' c '],  ARRAY [ 1 ,
2 ,  3 ]),
                   ( k ,  v1 ,  v2 ) ->  k  ||  CAST ( v1 / v2   AS
VARCHAR )); -- Concatenat  es   the   key   values   and   division
results   of   the   two   values  -- { a  ->  a1 ,  b  ->  b4 ,  c
->  c9 }
```

6.7.27 論理関数

論理演算子

表 6-3 : 論理演算子

オペレーター	説明	例
AND	左と右の両方のオペランドが TRUE の場合のみ TRUE を返します。	a AND b
OR	左または右のオペランドが TRUE の場合は TRUE を返します。	a OR b
NOT	右のオペランドが FALSE のときのみ TRUE を返します。	NOT a

論理演算に関連する NULL

次の表は、a および b の値がそれぞれ TRUE、FALSE および NULL の場合の真の値を示しています。

表 6-4 : Truth Table 1

a	b	a AND b	A or B
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	NULL	NULL	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	NULL	FALSE	NULL
NULL	TRUE	NULL	TRUE
NULL	FALSE	FALSE	NULL
NULL	NULL	NULL	NULL

表 6-5 : Truth Table 2

a	NOT a
TRUE	FALSE
FALSE	TRUE
NULL	NULL

6.7.28 列のエイリアス

SQL 標準では、列名は英字、数字、アンダーバー（_）で構成され、英字で始まる必要があります。

SQL 標準に準拠していないカラム名（User-Agent など）がログ収集設定で設定されている場合は、統計プロパティの設定ページでクエリに使用されるエイリアスをカラムに指定します。 エイリアスは、SQL 統計にのみ使用されます。 基礎となるストレージでは、列名が元の名前です。 元の列名を使用して照会します。

また、列名が非常に長い場合は、元の列名を置き換える別名を列に付けることができます。

表 6-6 : エイリアスの例：

元の列名	エイリアス
User-Agent	ua
User.Agent	ua
123	col
abceefghijklmnopqrstuvw	a

6.7.29 Logstore と RDS の結合

Log Service は、Logstore 内のログデータと RDS データベースのクエリ結合をサポートし、クエリ結果をRDS に保存します。

1. RDS VPC を作成し、ホワイトリストを設定します。
   a) RDS を作成し、VPC 環境を指定します。 作成が完了後、VPC ID と RDS インスタンス ID を取得します。
   b) RDS ホワイトリストに `100 . 104 . 0 . 0 / 16` を設定します。

      詳細な手順については、 『*RDS* ユーザーガイド』 のホワイトリストの設定に関する章をご参照ください。
2. External Store を作成します。

   次のステートメントで External Store を作成し、パラメータを実際のパラメータ値で置き換えます。

```
{
" externalSt  oreName ":" storeName ",
" storeType ":" rds - vpc ",
" parameter ":
   {
   " region ":" cn - qingdao ",
   " vpc - id ":" vpc - m5eq4irc1p  ucp *******"
   " instance - id ":" i - m5eeo2whsn *******"
   " host ":" localhost ",
   " port ":" 3306 ",
   " username ":" root ",
   " password ":"****",
   " db ":" scmc "
   " table ":" join_meta "
   }
}
```

表 6-7 : 説明

パラメータ	説明：
region	サービスを提供するリージョン。

パラメータ	説明：
vpc-id	VPC の ID。
instance-id	RDS インスタンス ID。
host	ECS インスタンス ID。
port	ECS インスタンスポート。
username	ユーザー名。
password	パスワード。
db	データベースの名前。
table	テーブル。

 注：

現在は、北京（cn-beijing）、青島（cn-qingdao）および杭州（cn-hangzhou）リージョンのみがサポートされています。

3. JOIN をつかってクエリします。

   Log Service コンソールの クエリページで JOIN ステートメントを実行します。

   サポートされている JOIN 構文：

   - INNER JOIN
   - LEFT JOIN
   - RIGHT JOIN
   - FULL JOIN

```
[  INNER  ]  JOIN
 LEFT  [  OUTER  ]  JOIN
 RIGHT  [  OUTER  ]  JOIN
 Full  [ outer ]  Join
```

 注：

- JOIN 操作は、Logstore と小さいテーブルの結合のみサポートします。
- JOIN の順序では、Logstore は前に、External Store は後に記述する必要があります。

- External Store の名前は JOIN 文に記述する必要があります。 その名前を自動的に RDS データベースとテーブル名に置き換えることができます。

JOIN 構文の例：

```
Method : Maid | select count ( 1 ), histogram ( logstore
) from log l join join_meta m on L . projectid =
cast ( M . ikey as varchar )
```

4. クエリ実行結果を RDS に保存します。

Insert 構文を使用して RDS にクエリ結果を挿入できます。

```
method : postlogsto relogs | insert into method_out
put select cast ( methodasva rchar ( 65535 )), count ( 1 )
fromloggro upbymethod
```

Python の例

```
# encoding : utf - 8
from __future__ import print_func tion
From aliyun . log import *
from aliyun . log . util import base64_enc
odestring
from random import randint
import time
Import OS
from datetime import datetime
    endpoint = os . environ . get (' ALIYUN_LOG
_SAMPLE_EN DPOINT ', ' cn - chengdu . log . aliyuncs . com
')
    Accesskeyi d = OS . internon . Get (' aliyun _
  Porter ','')
    accessKey = os . environ . get (' ALIYUN_LOG
_SAMPLE_AC CESSKEY ', '')
    logstore = os . environ . get (' ALIYUN_LOG
_SAMPLE_LO GSTORE ', '')
    Project = " Ali - yunlei - Chengdu "
    Client = logclient ( endpoint , accesskeyi d ,
accesskey , token )
# Create external store
    res = client . create_ext ernal_stor e ( project
, ExternalSt oreConfig (" rds_store "," region "," rds -
vpc "," vpc id "," instance id "," instance ip ","
instance port "," username "," paassword "," database ","
data table "));
    res . log_print ()
   # Get external store details
    res = client . get_extern al_store ( project ,"
rds_store ");
    res . log_print () ;
    res = client . list_exter nal_store ( project ,"");
    res . log_print () ;
   # JOIN query
    req = GetLogStor eLogsReque st ( project , logstore
, From , To ,"","," select count ( 1 ) from "+ logstore
+" s join meta m on s . projectid = cast ( m .
ikey as varchar )");
    res = client . get_logs ( req )
```

```
    res . log_print () ;
    # Query results insert into RDS
    req = GetLogStor eLogsReque st ( project , logstore
, From , To ,"","" insert into rds_store select
count ( 1 ) from "+ logstore );
    res = client . get_logs ( req )
    res . log_print ();
```

6.7.30 地理空間関数

地理空間コンセプト

地理空間関数は、Well-Known Text（WKT）形式のジオメトリをサポートします。

表 6-8 : ジオメトリ形式

ジオメトリ	WKT 形式
Point	`POINT ( 0 0 )`
Line string	`LINESTRING ( 0 0 , 1 1 , 1 2 )`
Polygon	`Polygon`
Multi-point	`MULTIPOINT ( 0 0 , 1 2 )`
Multi-line string	`MULTILINES TRING (( 0 0 , 1 1 , 1 2 ), ( 2 3 , 3 2 , 5 4 ))`
Multi-polygon	`MULTIPOLYG ON ((( 0 0 , 4 0 , 4 4 , 0 4 , 0 0 ), ( 1 1 , 2 1 , 2 2 , 1 2 , 1 1 )), ((- 1 - 1 , - 1 - 2 , - 2 - 2 , - 2 - 1 , - 1 - 1 )))`
Geometry collection	`GEOMETRYCO LLECTION ( POINT ( 2 3 ), LINESTRING ( 2 3 , 3 4 ))`

コンストラクタ

表 6-9 : コンストラクタ説明

関数	説明
ST_Point(double, double) → Point	指定された座標値を持つジオメトリ型ポイントを返します。

関数	説明
ST_LineFromText(varchar) → LineString	WKT 表現からのジオメトリタイプのライン String 返します。
ST_Polygon(varchar) → Polygon	WKT 表現からジオメトリタイプポリゴンを返します。
ST_GeometryFromText(varchar) → Geometry	WKT 表現からジオメトリ型オブジェクトを返します。
ST_AsText(Geometry) → varchar	ジオメトリの WKT 表現を返します。

操作

関数	説明
ST_Boundary(Geometry) → Geometry	このジオメトリのコンビナトリアル境界のクロージャを返します。
ST_Buffer(Geometry, distance) → Geometry	指定されたジオメトリからの距離が指定された距離以下であるすべての点を表すジオメトリを返します。
ST_Difference(Geometry, Geometry) → Geometry	指定されたジオメトリのポイントセット差を表すジオメトリ値を返します。
ST_Envelope(Geometry) → Geometry	ジオメトリの境界のある四角形ポリゴンを返します。
ST_ExteriorRing(Geometry) → Geometry	入力ポリゴンの外部リングを表す行 String 返します。
ST_Intersection(Geometry, Geometry) → Geometry	2 つのジオメトリのポイントセット交差を表すジオメトリ値を返します。
ST_SymDifference(Geometry, Geometry) → Geometry	2 つのジオメトリの点集合対称差を表すジオメトリ値を返します。

関係テスト

関数	説明
ST_Contains(Geometry, Geometry) → boolean	第 2 のジオメトリの点が第 1 のジオメトリの外部に存在せず、第 1 のジオメトリの内部の少なくとも 1 つの点が第 2 のジオメトリの内部にある場合にのみ、true を返します。2 つのジオメトリが少なくとも内部ポイントを共有する場合、false を返します。

関数	説明
ST_Crosses(Geometry, Geometry) → boolean	提供されたジオメトリに共通点があるがすべてではない内部点がある場合は true を返します。
ST_Disjoint(Geometry, Geometry) → boolean	指定されたジオメトリが空間的に交差していない場合は true を返します。
ST_Equals(Geometry, Geometry) → boolean	指定されたジオメトリが同じジオメトリを表す場合は true を返します。
ST_Intersects(Geometry, Geometry) → boolean	指定されたジオメトリが 2 次元で空間的に交差する（空間の任意の部分を共有する）場合は true を返し、そうでない場合はfalseを返します（これらは互いに素です）。
ST_Overlaps(Geometry, Geometry) → boolean	指定されたジオメトリが空間を共有し、同じ次元であるが、互いに完全には含まれていない場合、 true を返します。
ST_Relate(Geometry, Geometry) → boolean	最初のジオメトリが 2 番目のジオメトリに空間的に関連している場合は true を返します。
ST_Touches(Geometry, Geometry) → boolean	指定されたジオメトリに少なくとも 1 つのポイントが共通しているが、その内部が交差していない場合は true を返します。
ST_Within(Geometry, Geometry) → boolean	最初のジオメトリが完全に 2 番目のジオメトリの内側にある場合は true を返します。 2 つのジオメトリに少なくとも 1 つのポイントが共通する場合は、false を返します。

アクセサリー

関数	説明
ST_Area(Geometry) → double	投影された単位で 2 次元平面上のユークリッド測定を使用してポリゴンの面積を返します。
ST_Centroid(Geometry) → Geometry	ジオメトリの数学的重心である点の値を返します。
ST_CoordDim(Geometry) → bigint	ジオメトリの座標ディメンションを返します。
ST_Dimension(Geometry) → bigint	このジオメトリの固有次元を返します。この次元は座標次元以下でなければなりません。
ST_Distance(Geometry, Geometry) → double	2 つのジオメトリ間の最小距離を返します。

関数	説明
ST_IsClosed(Geometry) → boolean	行 String 開始点と終了点が一致する場合はtrueを返します。
ST_IsEmpty(Geometry) → boolean	このジオメトリが空のジオメトリコレクション、ポリゴン、またはポイントの場合はtrueを返します。
ST_IsRing(Geometry) → boolean	行が閉じていて単純な場合にのみtrueを返します。
ST_Length(Geometry) → double	投影された単位で（空間的な参考文献に基づいて）2 次元平面上でユークリッド測定を使用して、ラインストリングまたはマルチラインストリングの長さを返します。
ST_XMax(Geometry) → double	ジオメトリのバウンディングボックスの X 最大値を返します。
ST_YMax(Geometry) → double	ジオメトリの境界ボックスの Y 最大値を返します。
T_XMin(Geometry) → double	ジオメトリのバウンディングボックスの X 最小値を返します。
ST_YMin(Geometry) → double	ジオメトリの境界ボックスの Y 最小値を返します。
ST_StartPoint(Geometry) → point	ライン String ジオメトリの最初の点を返します。
ST_EndPoint(Geometry) → point	ライン String ジオメトリの最後のポイントを返します。
ST_X(Point) → double	点の X 座標を返します。
ST_Y(Point) → double	点の Y 座標を返します。
ST_NumPoints(Geometry) → bigint	ジオメトリ内の点の数を返します。
ST_NumInteriorRing(Geometry) → bigint	ポリゴンの内部リングの数を返します。

6.7.31 Geo機能

国、州、市、ISP、および指定された IP アドレスの経度と緯度を決定する関数の詳細については、IP 機能 を参照してください

表 6-10 : Geo機能

機能	説明	例
geohash(string)	指定された地理座標のgeohash 値を返します。 地理座標は「緯度、経度」の形式の文字列で表されます（緯度と経度の値はコンマで区切られます）。	`select geohash (' 34 . 1 , 120 . 6 ')= ' wwjcbrdnzs '`
geohash(lat, lon)	指定された地理座標のgeohash 値を返します。 地理座標は、緯度と経度を示す 2 つの個別のパラメータで表されます。	`select geohash ( 34 . 1 , 120 . 6 )= ' wwjcbrdnzs '`

6.7.32 Join 構文

Join は、複数のテーブルのフィールドを結合するために使用されます。 また、1 つの Logstore の Join の他に、Logstore と RDS の Join、および複数の Logstore の Join にも対応しています。 このドキュメントでは、Logstore 間で Join 機能を使用する方法について説明します。

手順

1. Python SDK の最新バージョンをダウンロードします。
2. クエリには GetProjectLogs インターフェイスを使用します。<li class="li">クエリには GetProjectLogs インターフェイスを使用します。

SDK サンプル

```
/ usr / bin / env   python
# encoding :  utf - 8
 import   time , sys , os
 From   aliyun .  log .  logexcepti  on   import   logexcepti  on
 from   aliyun . log . logitem   import   LogItem
 from   aliyun . log . logclient   import   LogClient
 From   aliyun .  log .  getlogsreq  uest   import   getlogsreq  uest
 from   aliyun . log . getproject  logsreques  t   import
 GetProject  LogsReques  t
 from   aliyun . log . putlogsreq  uest   import   PutLogsReq  uest
 from   aliyun . log . listtopics  request   import   ListTopics
 Request
 from   aliyun . log . listlogsto  resrequest   import   ListLogsto
 resRequest
 from   aliyun . log . gethistogr  amsrequest   import   GetHistogr
 amsRequest
 from   aliyun . log . index_conf  ig   import  *
 from   aliyun . log . logtail_co  nfig_detai  l   import  *
 from   aliyun . log . machine_gr  oup_detail   import  *
 from   aliyun . log . acl_config   import  *
 if   __name__ ==' __main__ ':
```

```
    token  =  None
    endpoint  = " http :// cn - hangzhou . log . aliyuncs . com "
    accessKeyI  d  = ' LTAIvKy7U '
    accessKey =' 6gXLNTLyCf  dsfwrewrfh  dskfdsfuiw  u '
    client  =  LogClient ( endpoint ,  accessKeyI  d ,  accessKey ,
token )
    logstore  = " meta "
   #  In  the  query  statement ,  specify  two  Logstores .
For  each  Logstore  specify  its  time  range  and  the
key
    req  =  GetProject  LogsReques  t ( project ," select  count
( 1 )  from  sls_operat  ion_log  s  join  meta  m  on  s .
__date__  >' 2018 - 04 - 10  00 : 00 : 00 '  and  s . __date__  <
' 2018 - 04 - 11  00 : 00 : 00 '  and  m . __date__  >' 2018 - 04 -
23  00 : 00 : 00 '  and  m . __date__  <' 2018 - 04 - 24  00 : 00
: 00 '  and  s . projectid  =  cast ( m . ikey  as  varchar )");
    Res  =  client .  Fig  ( req )
    res . log_print ();
    exit ( 0 )
```

6.7.33 UNNEST 関数

シナリオ

ログデータの列は通常、文字列や数値などのプリミティブデータ型です。 より複雑なデータ構造を持つ特定のシナリオでは、ログデータの列に、配列、マップ、JSON オブジェクトなどの複雑なデータ型が含まれることがあります。 このような特別なログフィールドに対して、より簡単に検索と分析を実行するため、 UNNEST 関数を使用できます。

例：

```
__source__ :  1 . 1 . 1 . 1
__tag__ : __hostname  __ :  vm - req - 1701032323  16569850 -
tianchi111  932 . tc
__topic__ :  TestTopic_  4
array_colu  mn : [ 1 , 2 , 3 ]
double_col  umn :  1 . 23
map_column : {" a ": 1 ," b ": 2 }
text_colum  n :  Product
```

`array_colu mn` フィールドは配列型です。 `array_colu mn` 配列内のすべての要素の集計を取得するには、各行について配列のすべての要素を列挙する必要があります。

UNNEST 関数

構文	説明
`unnest ( array ) as table_alia s ( column_nam e )`	指定された配列を複数の行に分割します。 行の名前は、 `column_nam e` となります。
`unnest ( map ) as table ( key_name , value_name )`	指定されたマップを複数の行に分割します。 Key の名前は `key_name` で、Value の名前は `value_name` となります。

 注：

注：UNNEST 関数は array 型や、map 型だけを受け入れます。 string 型を指定した場合、まず JSON 型に変換し、 `cast ( json_parse ( array_colu mn ) as array ( bigint ))` 構文を使用して、array 型や map 型に変換する必要があります。

配列の要素の列挙

SQL を使用して配列を複数の行に分割します:

```
* | select array_colu mn , a from log , unnest ( cast (
 json_parse ( array_colu mn ) as array ( bigint ) ) ) as t ( a
 )
```

UNNEST 関数は、配列を複数の行に分割し、各行を t と名づけた新しいテーブルに保存します、そして展開された列を参照するために a を使用します。

- 配列内の要素の合計を計算します：

```
* | select sum ( a ) from log , unnest ( cast ( json_parse
 ( array_colu mn ) as array ( bigint ) ) ) as t ( a )
```

- a（配列の各要素）を使って配列に対して GROUP BY 操作を実行します。

```
* | select a , count ( 1 ) from log , unnest ( cast (
 json_parse ( array_colu mn ) as array ( bigint ) ) ) as t (
 a ) group by a
```

map の要素を列挙する

- map の要素を列挙する：

```
* | select map_column , a , b from log , unnest ( cast (
 json_parse ( map_column ) as map ( varchar , bigint ) ) ) as
 t ( a , b )
```

- キーで map に対して GROUP BY 操作を実行します。

```
* | select key , sum ( value ) from log , unnest ( cast (
 json_parse ( map_column ) as map ( varchar , bigint ) ) ) as
 t ( key , value ) GROUP BY key
```

ヒストグラム、numeric_histogram 検索の結果を可視化する

- ヒストグラム

  ヒストグラム関数は、COUNT GROUP BY 構文に似ています。 構文の詳細については、マッピング関数を参照してください。

  ヒストグラム関数は通常、直接可視化できない JSON 文字列を返します。 例は次となります：

```
* | select histogram ( method )
```

  UNNEST 関数を使用して JSON データを複数の行に分割することができます。例は次となります：

```
* | select key , value from ( select histogram ( method )
 as his from log ) , unnest ( his ) as t ( key , value )
```

- Numeric_histogram

numeric_histogram 構文は、数値列の値を複数のバケットにソートします。 この操作は、数値列に対して GROUP BY 操作を実行するのと似ています。 構文の詳細については、推定関数を参照してください。

```
* |  select   numeric_hi  stogram ( 10 , Latency )
```

結果を可視化するには、次の検索ステートメントを使用します：

```
* |  select   key , value   from ( select   numeric_hi  stogram ( 10
 , Latency )  as   his   from   log ) ,  unnest ( his )  as   t (
 key , value )
```

6.7.34 電話番号機能

電話番号機能は、中国本土で登録されている電話番号の地域情報を照会するために使用されます。

関数リスト

関数名	説明	例:
mobile_province	この関数は、電話番号の属する省を照会するために使用されます。 電話番号は数値タイプである必要があります。 電話番号が文字列型の場合は、 `try_cast` を使用して型を数値に変換することができます。	`* |  select   mobile_pro vince ( 12345678 )` `* |  select   mobile_pro vince ( try_cast (' 12345678 '  as   bigint ) )`
mobile_city	この関数は、電話番号の属する都市を照会するために使用されます。 電話番号は数値タイプである必要があります。 電話番号が文字列型の場合は、 `try_cast` を使用して型を数値に変換することができます。	`* |  select   mobile_cit  y ( 12345678 )` `* |  select   mobile_cit  y ( try_cast (' 12345678 ' as   bigint ) )`

関数名	説明	例:
mobile_carrier	この関数は、電話番号が属する通信事業者を照会するために使用されます。 電話番号は数値タイプである必要があります。 電話番号が文字列型の場合は、 `try_cast` を使用して型を数値に変換することができます。	`* | select mobile_carrier ( 12345678 )` `* | select mobile_carrier ( try_cast ( ' 12345678 ' as bigint ) )`

シナリオ

・ 電話番号属性を照会してレポートを生成します。

電子商取引会社が顧客のイベントに関するログを収集するとします。 会社は、電話番号を含むフィールドを抽出してから、以下のクエリステートメントを使用して電話番号の地域情報を収集することができます。

```
SELECT mobile_cit y ( try_cast (" mobile " as bigint )) as " city ", mobile_pro vince ( try_cast (" mobile " as bigint )) as " province ", count ( 1 ) as " number of requests " group by " province ", " city " order by " number of request " desc limit 100
```

このステートメントでは、 `mobile_cit y` および `mobile_pro vince` 関数の入力フィールドとして `mobile` を使用して、電話番号が属する省と都市を示します。 このクエリから戻された情報を次の図に示します。

次の図に示すように、電話番号の地域情報を地図で表示することもできます。

・ 電話番号の地域情報を確認し、異常なログイン情報を報告します。

通信事業者が、平日の居場所がその電話番号の地域情報とは異なる（追加属性や頻繁にアクセスされる IP アドレスにより）顧客をフィルタリングする場合は、次のステートメントを使用できます。

```
* | select mobile , client_ip , count ( 1 ) as PV where mobile_cit y ( try_cast (" mobile " as bigint )) ! =
```

```
ip_to_city ( client_ip )  and   ip_to_city ( client_ip ) ! = ''
group   by   client_ip ,  mobile   order   by   PV   desc
```

さらに、電話番号属性を使用するアラームルールを作成することもできます。 詳細は、Log Service アラームをご参照ください。

6.8 マシンラーニングの構文と機能

6.8.1 はじめに

Log Service は、複数のアルゴリズムとメソッド呼び出しをサポートする機械学習機能を提供します。 ログの照会および分析中に、SELECT ステートメントおよび機械学習機能を使用して、機械学習アルゴリズムを呼び出し、ある期間に 1 つまたは複数のフィールドの特性を分析することができます。

特に、 Log Service は、時系列予測、時系列異常検出、シーケンス分解、および複数の時系列クラスタリングに関連する問題を迅速に解決するのに役立つ、多様な時系列分析アルゴリズムを提供します。 さらに、アルゴリズムは標準 SQL インターフェイスと互換性があり、アルゴリズムの使用を大幅に簡素化し、トラブルシューティングの効率を向上させます。

機能

- 単一時系列シーケンス上の、多数のスムーズ化操作がサポートされています。
- 予測、異常検出、変化点検出、変曲点検出、および単一時系列シーケンスの多期間推定に関連するアルゴリズムがサポートされています。
- 単一時系列シーケンスの分解操作がサポートされています。
- 複数時系列シーケンスのさまざまなクラスタリングアルゴリズムがサポートされています。
- 複数フィールドパターンマイニング (数値またはテキストの列に基づく) がサポートされています。

制限

- 入力時系列データは、同一間隔からサンプル抽出する必要があります。
- 入力時系列データに、同じ時点から繰り返しサンプル抽出されたデータを含めることはできません。

項目	制限
時系列データ処理の有効性能	最大 150,000 の連続した時点からの収集データ 制限を超過している場合、データを集約するか、サンプルデータ量を減らす必要があります。

項目	制限
密度ベースのクラスタリングアルゴリズムのクラスタリング性能	最大 5,000 の時系列曲線で、それぞれ最大 1,440 以下の時点を含められます。
階層的クラスタリングアルゴリズムのクラスタリング性能	最大 2,000 の時系列曲線で、それぞれ最大 1,440 以下の時点を含められます。

関数

	カテゴリー	関数	説明
時系列	平滑関数	ts_smooth_simple	Holt Winters アルゴリズムを使用して時系列データを平滑化します。
		ts_smooth_fir	FIR フィルターを使用して時系列データを平滑化します。
		ts_smooth_iir	IIR フィルターを使用して時系列データを平滑化します。
	多期間推定関数	ts_period_detect	ある時系列の期間情報を推定します。
	変化点検出関数	ts_cp_detect	ある時系列で異なる統計的特性を持つ期間を検出します。期間のエンドポイントは変化点です。
		ts_breakout_detect	ある時系列で統計が急激に増減する時点を検出します。
	予測と異常検出の関数	ts_predicate_simple	デフォルトのパラメーターを使用して時系列データをモデル化し、簡単な時系列予測と異常検出を実行します。
		ts_predicate_ar	自己回帰モデルを使用して時系列データをモデル化し、簡単な時系列予測と異常検出を実行します。
		ts_predicate_arma	自己回帰移動平均モデルを使用して時系列データをモデル化し、簡単な時系列予測と異常検出を実行します。

	カテゴリー	関数	説明
		ts_predicate_arima	自己回帰和分移動平均 (ARIMA) モデルを使用して時系列データをモデル化し、単純な時系列予測と異常検出を実行します。
	シーケンス分解関数	ts_decompose	STL アルゴリズムを使用して時系列データシーケンスを分解します。
	時系列クラスタリング関数	ts_density_cluster	密度ベースのクラスタリング手法を使用して、時系列データをクラスタリングします。
		ts_hierarchical_cluster	階層的クラスタリング方法を用いて時系列データをクラスタリングします。
		ts_similar_instance	指定された曲線に類似する曲線を照会します。
パターンマイニング	頻出パターン統計	pattern_stat	統計パターンの頻出パターンを示します。 これは任意の複数属性フィールドのサンプル間で、属性の代表的な組み合わせをマイニングするために使用されます。
	差分パターン統計	pattern_diff	指定された条件下の 2 セット間で、相違が生じるパターンを検出します。

6.8.2 平滑関数

平滑関数は、入力値の時系列曲線を滑らかにしてフィルタリングするために使用します。 フィルタリングは、時系列曲線の形を見つけるための第一歩です。

関数リスト

関数	説明
`ts_smooth_simple`	この関数は、Holt Winters アルゴリズムを使用して、時系列データを平滑化する、デフォルトの平滑関数です。
`ts_smooth_fir`	この関数は、FIR フィルターを使用して、時系列データを平滑化します。

関数	説明
ts_smooth_iir	この関数は、IIR フィルターを使用して、時系列データを平滑化します。

ts_smooth_simple

関数の形式：

```
select ts_smooth_simple(x, y)
```

パラメータについて、次の表に説明します。

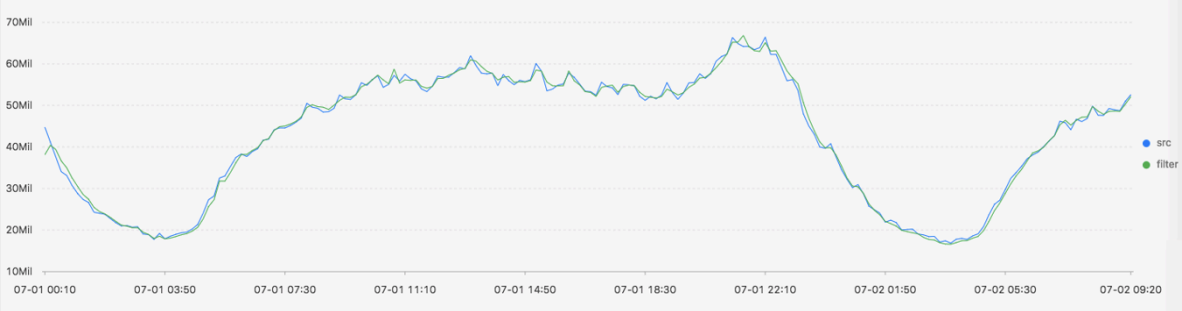
パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点のデータに対応する数値列	--

例：

・照合と分析の命令文：

```
* | select ts_smooth_simple(stamp, value) from (select __time__ - __time__ % 120 as stamp, avg(v) as value from log GROUP BY stamp order by stamp)
```

・結果：



次の表に、表示項目について説明します。

表示項目		説明
横軸	unixtime	データの timestamp (秒)
縦軸	src	フィルタリング前のデータ
	filter	フィルタリング後のデータ

ts_smooth_fir

関数の形式：

- フィルターパラメーターが定義されていない場合、組み込みウィンドウのパラメーターを使用して、時系列データをフィルタリングできます。

```
select ts_smooth_fir(x, y,winType,winSize,samplePeriod,sampleMethod)
```

- フィルターパラメーターが指定されている場合は、必要に応じて設定できます。

```
select ts_smooth_fir(x, y,array[],samplePeriod,sampleMethod)
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点でのデータに対応する数値列	-
winType	フィルターウィンドウの種類	値の範囲： ・ rectangle：矩形窓 ・ hanning：ハニング窓 ・ hamming：ハミング窓 ・ blackman：ブラックマン窓 注： よく表示するために、このパラメーターを「rectangle (矩形窓)」に設定することが推奨されます。
winSize	フィルターウィンドウ長	2～15 の long 型の値
array[]	特定の FIR フィルターパラメータ	配列形式の値、配列の合計は 1.0 です。たとえば [0.2, 0.4, 0.3, 0.1]。 For example, [0.2, 0.4, 0.3, 0.1].
samplePeriod	現在の時系列データをサンプル抽出する期間	1 から 86399 秒までの long 型の値

パラメータ	説明	値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

例：

・ 照合と分析の命令文：

```
* | select ts_smooth_fir ( stamp , value , ' rectangle ', 4 , 1 , ' avg ') from ( select __time__ - __time__ % 120 as stamp , avg ( v ) as value from log GROUP BY stamp order by stamp )
```

結果：

・ 照合と分析の命令文：

```
* | select ts_smooth_fir ( stamp , value , array [ 0 . 2 , 0 . 4 , 0 . 3 , 0 . 1 ], 1 , ' avg ') from ( select __time__ - __time__ % 120 as stamp , avg ( v ) as value from log GROUP BY stamp order by stamp )
```

結果：

次の表に、表示項目について説明します。

表示項目		説明
横軸	unixtime	Unixtime timestamp (秒)
縦軸	src	フィルタリング前のデータ
	filter	フィルタリング後のデータ

ts_smooth_iir

関数の形式：

```
select
 ts_smooth_iir(x, y, array[], array[], samplePeriod, sampleMethod)
```

次の表に、パラメーターについて説明します。

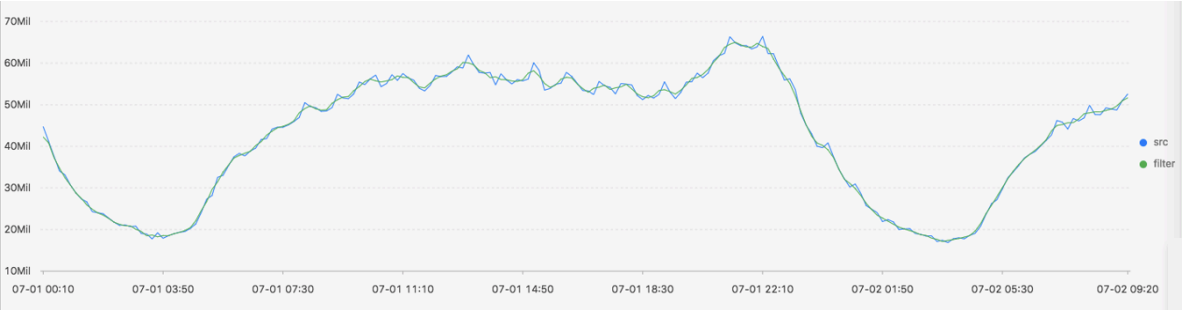
パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点でのデータに対応する数値列	-
array[]	IIR フィルターアルゴリズムにおける xi の特定パラメーター	長さが 2〜15 の配列値。 配列の合計は 1.0 です。 たとえば、配列 [0.2、0.4、0.3、0.1]。
array[]	IIR フィルターアルゴリズムにおける yi−1 の特定パラメーター	長さが 2〜15 の配列値。 配列の合計は 1.0 です。 たとえば、配列 [0.2、0.4、0.3、0.1]。
samplePeriod	現在の時系列データをサンプル抽出する期間	1 から 86399 秒までの long 型の値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計値

例：

・照合と分析の命令文：

```
* | select ts_smooth_iir(stamp, value, array[0.2, 0.4, 0.3, 0.1], array[0.4, 0.3, 0.3], 1, 'avg') from (select __time__ - __time__ % 120 as stamp, avg(v) as value from log GROUP BY stamp order by stamp)
```

・結果：



表示項目について、次の表に説明します。

表示項目		説明
横軸	unixtime	Unixtime timestamp (秒単位)
縦軸	src	フィルタリング前のデータ
	filter	フィルタリング後のデータ

6.8.3 多期間推定関数

多期間評価関数では、異なる期間の時系列データを推定し、フーリエ変換などの一連の動作を実行して期間データを抽出することができます。

ts_period_detect

この関数は、時系列データを期間ベースで推定します。

関数の形式：

```
select
 ts_period_detect(x, y,minPeriod,maxPeriod,samplePeriod,sampleMethod)
```

パラメーターについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点のデータに対応する数値列	-

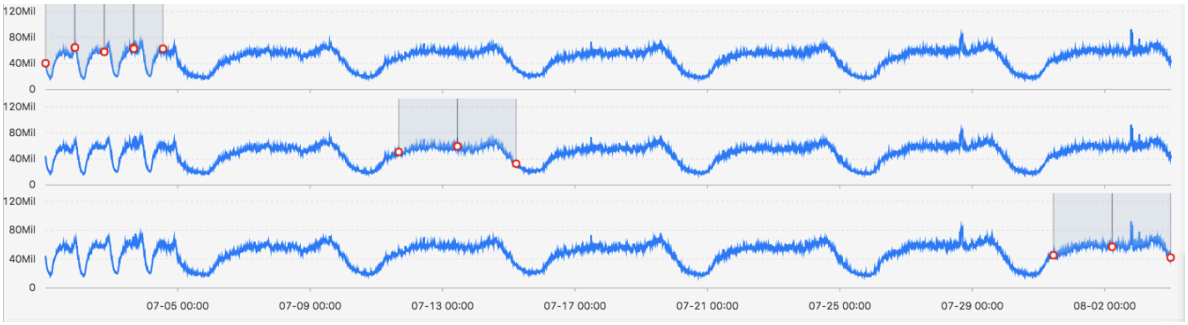
パラメータ	説明	値
minPeriod	時系列データの全長に対する事前推定期間の最小長の比率	値の範囲：(0、1)
maxPeriod	時系列データの全長に対する事前推定期間の最大長の比率 注：*maxPeriod*値は、*minPeriod*値より大きくなければなりません。	値の範囲：(0、1)
samplePeriod	現在の時系列データをサンプル抽出する期間	間 1 から 86399 秒までの long 型の値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

例：

・ 照合と分析の命令文：

```
* | select ts_period_detect ( stamp , value , 0 . 2 , 1 . 0 , 1 , ' avg ') from ( select __time__ - __time__ % 120 as stamp , avg ( v ) as value from log GROUP BY stamp order by stamp )
```

・ 結果：



次の表に、表示項目について説明します。

表示項目	説明
period_id	配列の長さが 1 の期間 ID で構成される配列。値 0.0 は元の系列を示します。
time_series	Timestamp シーケンス
data_series	各 timestamp の結果 ・ period_id が 0.0 の場合の元のシーケンスを示します。 ・ period_id が 0.0 ではない場合、フィルタリング後の期間推定結果を示します。

6.8.4 変化点検出関数

変化点検出関数は、時系列データの変化点を検出します。

変化点検出関数は、2 種類の変化点をサポートしています。

・ 指定期間内の統計的特性の変化

・ シーケンス内の明らかな障害

関数リスト

関数	説明
ts_cp_detect	この関数は、ある時系列で異なる統計的特性を持つ間隔を検出します。 間隔のエンドポイントは変化点です。
ts_breakout_detect	この関数は、時系列で統計が急激に増減する時点を検出します。

ts_cp_detect

関数の形式：

・ ウィンドウサイズが不明な場合は、次の形式で tscpdetect 関数を使用してください。 次に、関数によって呼び出されるアルゴリズムは、変化点を検出するために長さ "10" のウィンドウを使用します。

```
select ts_cp_detect(x, y, amplePeriod,sampleMethod)
```

・ サービス曲線の表示効果を調整する必要がある場合は、次の形式で ts_cp_detect 関数を使用します。 次に、minSize パラメーターを設定して、表示効果を最適化することができます。

```
select ts_cp_detect(x, y, minSize, samplePeriod, sampleMethod)
```

パラメータについて、次の表に説明します。

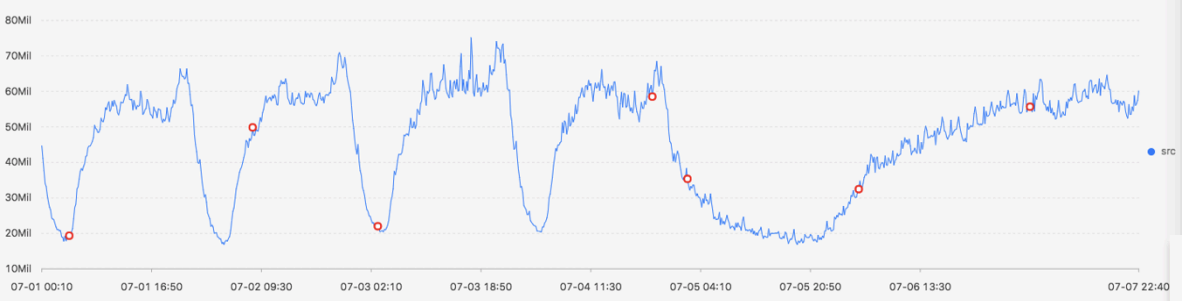
パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定時点のデータに対応する数値列	-
minSize	連続間隔の最小長	最小値は "3" で、最大値は現在の入力データ長の 1/10 までです。
samplePeriod	現在の時系列データをサンプリングする期間	1～86399 秒の long 型の値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

例：

・ 照会と分析の命令文：

```
* | select ts_cp_dete ct ( stamp , value , 3 , 1 , ' avg ') from ( select __time__ - __time__ % 10 as stamp , avg ( v ) as value from log GROUP BY stamp order by stamp )
```

・ 結果：



表示項目について、次の表に説明します。

表示項目		説明
横軸	unixtime	データの timestamp (秒)。(例：1537071480)
縦軸	src	フィルタリング前のデータ (例：1956092.7647745228)

表示項目		説明
	prob	ある時点が変化点である確率。 値の範囲は 0〜1 です。

ts_breakout_detect

関数の形式：

```
select ts_breakout_detect(x, y, winSize, samplePeriod, sampleMethod)
```

パラメーター、ついて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定した時点のデータに対応する数値列	-
winSize	連続する間隔の最小長	最小値は "3" で、最大値は現在の入力データ長の 1/10 までです。
samplePeriod	現在の時系列データをサンプリングする期間	1〜86399 秒の long 型の値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

例：

・ 照会と分析の命令文：

```
* | select ts_breakou t_detect ( stamp , value , 3 , 1 , '
 avg ') from ( select __time__ - __time__ % 10 as stamp
```

```
, avg ( v ) as value from log GROUP BY stamp order
  by stamp )
```

・結果：

表示項目について、次の表に説明します。

表示項目		説明
横軸	unixtime	データの timestamp (秒)。(例：1537071480)
縦軸	src	フィルタリング前のデータ (例：1956092.7647745228)
	prob	ある時点が変化点である確率。 値の範囲は 0〜1 です。

6.8.5 予測と異常検出の関数

予測と異常検出の関数は、時系列曲線を予測し、予測された曲線と実際の曲線との間の誤差の Ksigma および分位を特定することによって、異常を検出します。

関数リスト

関数	説明
ts_predicate_simple	デフォルトのパラメーターを使用して時系列データをモデル化し、簡単な時系列予測と異常検出を実行します。
ts_predicate_ar	自己回帰モデルを使用して時系列データをモデル化し、簡単な時系列予測と異常検出を実行します。
ts_predicate_arma	自己回帰移動モデルを使用して時系列データをモデル化し、簡単な時系列予測と異常検出を実行します。
ts_predicate_arima	差分のある自己回帰移動モデルを使用して時系列データをモデル化し、簡単な時系列予測と異常検出を実行します。

注：

予測関数と異常検出関数の表示項目は一致しています。 結果と説明の詳細については「tspredicatesimple 出力結果の例」をご参照ください。

ts_predicate_simple

関数の形式：

```
select ts_predica te_simple ( x , y , nPred , samplePeri od , sampleMeth od )
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点のデータに対応する数値列	-
nPred	予測ポイント数	1～5 x*p*の long 型の値
samplePeriod	現在の時系列データをサンプリングする期間	1～86399 秒の long 型の値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

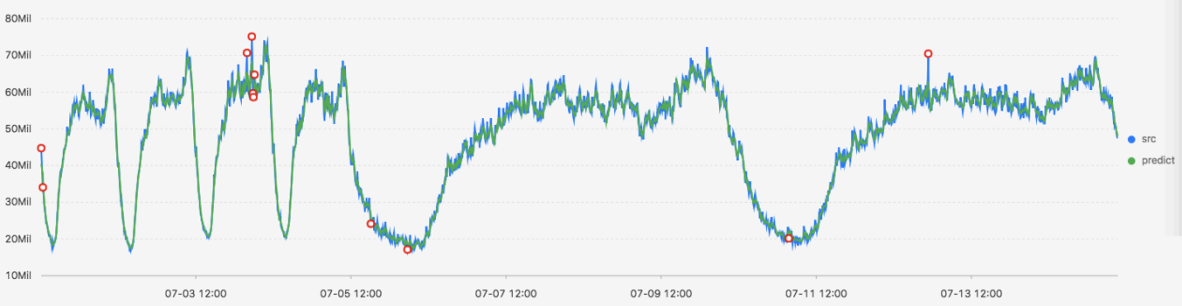
例：

・ 照合と分析の命令文：

```
* | select ts_predica te_simple ( stamp , value , 6 , 1 , ' avg ') from ( select __time__ - __time__ % 60 as stamp
```

```
, avg ( v ) as value from log GROUP BY stamp order
  by stamp )
```

・ 結果：



表示項目について、次の表に説明します。

表示項目		説明
横軸	unixtime	Unixtime timestamp (秒)
縦軸	src	生データ
	predict	フィルタリング後のデータ
	upper	予測の上限。 デフォルトで、現在の信頼度は "0.85" で、変更することはできません。
	lower	予測の下限。 デフォルトで、現在の信頼度は "0.85" で、変更することはできません。
	anomaly_prob	ポイントが異常ポイントである確率で、 値の範囲は 0～1 です。

ts_predicate_ar

関数の形式：

```
select ts_predica te_ar ( x , y , p , nPred , samplePeri od
, sampleMeth od )
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点のデータに対応する数値列	-
p	自己回帰モデルの順序	2～8 までの long 型の値
nPred	予測ポイント数	1～5 xp の long 型の値

パラメータ	説明	値
samplePeriod	現在の時系列データをサンプリングする期間	1～86399 の long 型の値
sampleMethod	サンプリングウィンドウデータのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

照合と分析の命令文：

```
* | select ts_predica te_ar ( stamp , value , 3 , 4 , 1 , '
 avg ') from ( select __time__ - __time__ % 60 as stamp ,
 avg ( v ) as value from log GROUP BY stamp order by
   stamp )
```

ts_predicate_arma

関数の形式：

```
 select ts_predica te_arma ( x , y , p , q , nPred ,
 samplePeri od , sampleMeth od )
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点のデータに対応する数値列	-
p	自己回帰モデルの順序	2～8 の long 型の値
q	移動平均 モデルの順序	2～8 の long 型の値
nPred	予測ポイント数	1～5 x*p* の long 型の値
samplePeriod	現在の時系列データをサンプリングする期間	1～86399 秒の long 型の値

パラメータ	説明	値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

照合と分析の命令文：

```
* | select ts_predica te_arma ( stamp , value , 3 , 2 , 4 , 1 , ' avg ') from ( select __time__ - __time__ % 60 as stamp , avg ( v ) as value from log GROUP BY stamp order by stamp )
```

ts_predicate_arima

関数の形式：

```
select ts_predica te_arima ( x , y , p , d , qnPred , samplePeri od , sampleMeth od )
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点のデータに対応する数値列	-
p	自己回帰モデルの順序	2〜8 の long 型の値
d	差分モデルの順序	1〜3 の long 型の値
q	移動平均 モデルの順序	2〜8 の long 型の値
nPred	予測ポイント数	1〜5 x*p* の long 型の値
samplePeriod	現在の時系列データをサンプリングする期間	1〜86399 秒の long 型の値

パラメータ	説明	値
sampleMethod	サンプリングウィンドウ内データのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

照合と分析の命令文：

```
* | select ts_predica te_arima ( stamp , value , 3 , 1 , 2 , 4 , 1 , ' avg ') from ( select __time__ - __time__ % 60 as stamp , avg ( v ) as value from log GROUP BY stamp order by stamp )
```

6.8.6 シーケンス分解関数

シーケンス分解関数は、サービス曲線を分解し、曲線の傾向と期間とに関する情報を強調表示することができる。

ts_decompose

関数の形式：

```
select ts_decompose(x, y, samplePeriod, sampleMethod)
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定された時点のデータに対応する数値列	-
samplePeriod	現在の時系列データをサンプル抽出する期間	1～86399 秒の long 型の値

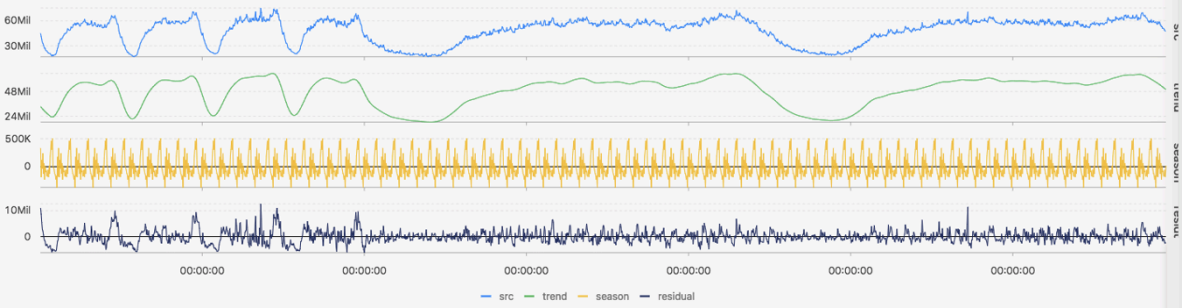
パラメータ	説明	値
sampleMethod	サンプリングウィンドウデータのサンプリング方法	値の範囲： ・ avg：ウィンドウ内データの平均値 ・ max：ウィンドウ内データの最大値 ・ min：ウィンドウ内データの最小値 ・ sum：ウィンドウ内データの合計

例：

・ 照合と分析の命令文：

```
* | select ts_decompose(stamp, value, 1, 'avg') from
  (select __time__ - __time__ % 60 as stamp, avg(v)
  as value from log GROUP BY stamp order by stamp)
```

・ 結果：



表示項目について、次の表に説明します。

表示項目		説明
横軸	unixtime	Unixtime timestamp (秒)
縦軸	src	生データ
	trend	分解後の曲線の傾向
	season	分解後の曲線の期間
	residual	分解後の残存データ

6.8.7 時系列クラスタリング関数

時系列クラスタリング関数は、入力時系列データを異なる曲線形状に自動的にクラスタリングするために使用されます。 その後、既存の曲線形状とは異なる形状の該当するクラスターセンターおよび曲線を迅速に見つけることができます。

関数リスト

関数	説明
ts_density_cluster	この関数は、密度ベースのクラスタリング手法を使用して、時系列データをクラスタリングします。
ts_hierarchical_cluster	この関数は、階層的クラスタリング方法を用いて時系列データをクラスタリングします。
ts_similar_instance	この関数は、指定された曲線に類似する曲線を照会します。

ts_density_cluster

関数の形式：

```
select ts_density_cluster(x, y, z)
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定した時点のデータに対応する数値列 -	-
z	指定した時点のデータに対応するメトリック名	文字列型の値 (例：machine01.cpuusr)

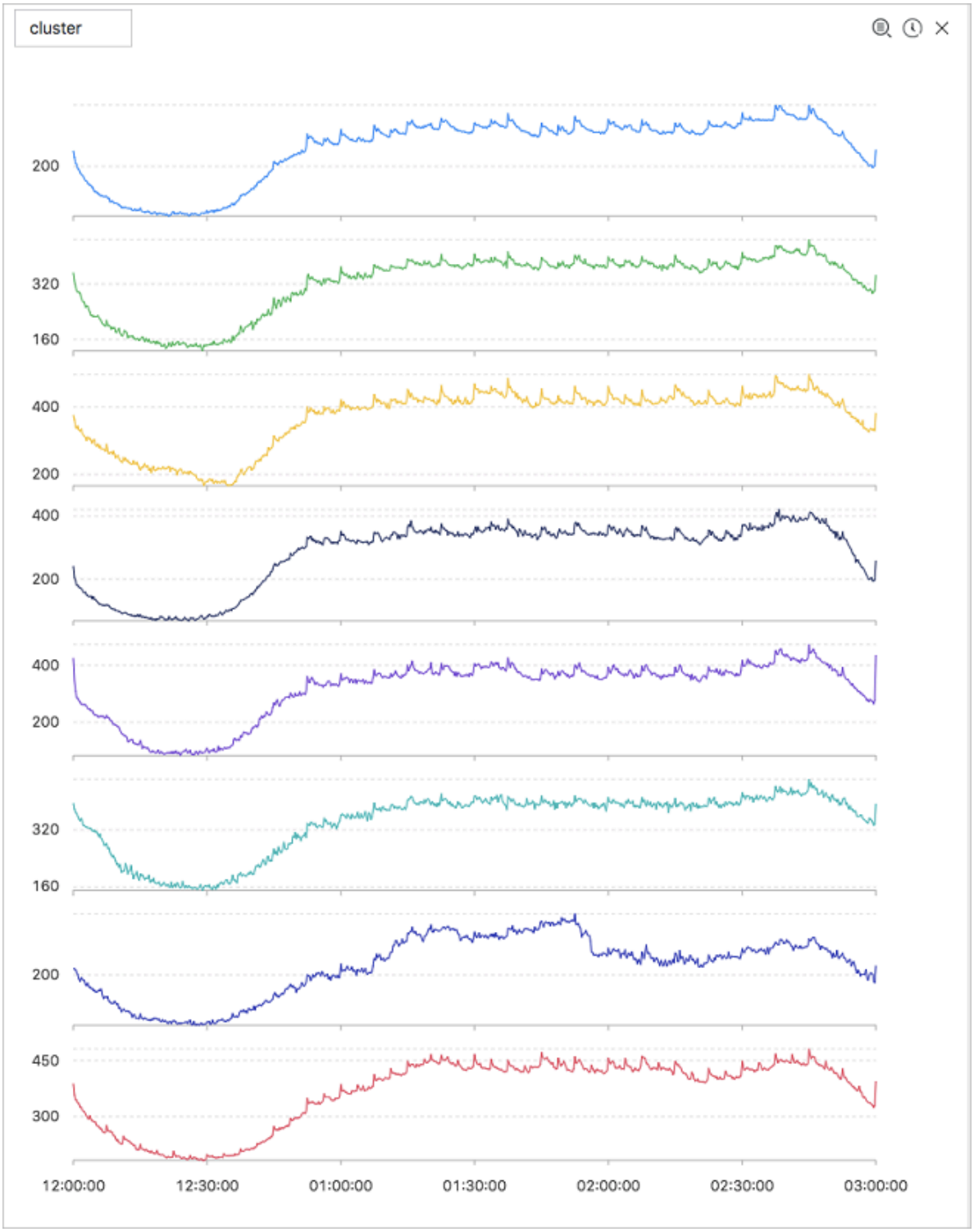
例：

・ 照合と分析の命令文：

```
*  and  ( h : " machine_01 "  OR   h : " machine_02 "  OR   h
 : " machine_03 ") |  select   ts_density  _cluster ( stamp ,
 metric_val  ue , metric_nam  e  )  from  (  select   __time__
 -  __time__  %  600   as   stamp ,  avg ( v )  as   metric_val
```

```
ue , h as metric_nam e from log GROUP BY stamp ,
metric_nam e order BY metric_nam e , stamp )
```

・結果：



表示項目について、次の表に説明します。

表示項目	説明
cluster_id	クラスタータイプ。 値 "-1" は、そのクラスタリングがいずれのクラスターセンターにも分類できないことを示します。
rate	クラスタリングにおけるインスタンスの割合
time_series	クラスターセンターの Timestamp シーケンス
data_series	クラスターセンターのデータシーケンス
instance_names	クラスターセンターに含まれるインスタンスのセット
sim_instance	クラスタリングにおけるインスタンス名

ts_hierarchical_cluster

関数の形式：

```
select ts_hierarchical_cluster(x, y, z)
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定した時点のデータに対応する数値列	-
z	指定した時点のデータに対応するメトリック名	文字列型の値 (例：machine01.cpu_usr)

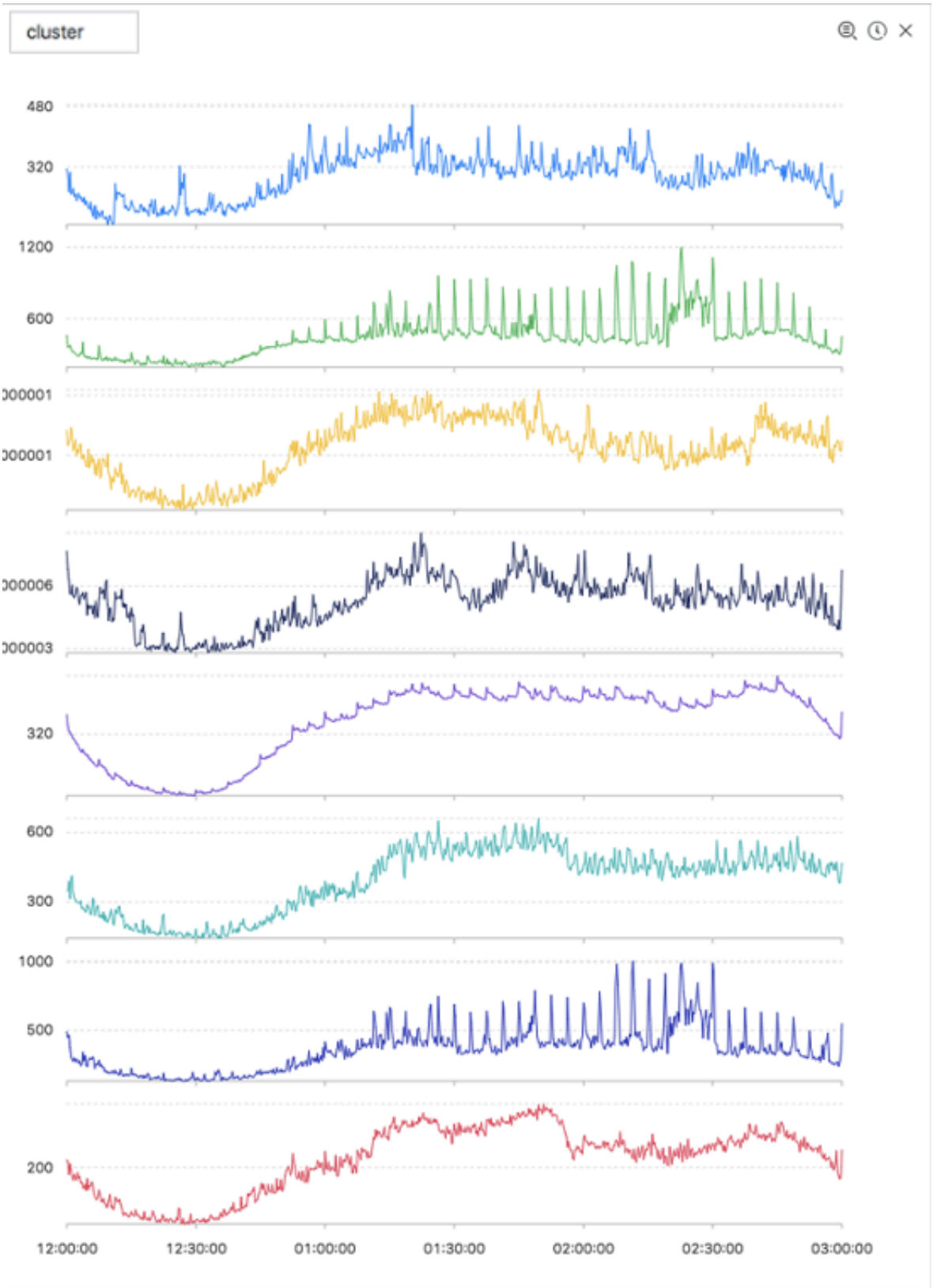
例：

・照合と分析の命令文：

```
*  and  ( h : " machine_01 "  OR   h : " machine_02 "  OR   h  :
 " machine_03 ") |  select   ts_hierarc  hical_clus  ter ( stamp
 ,  metric_val  ue ,  metric_nam  e )  from  (  select   __time__
  -  __time__  %  600   as   stamp ,  avg ( v )  as   metric_val
```

```
ue , h as metric_nam e from log GROUP BY stamp ,
metric_nam e order BY metric_nam e , stamp )
```

- 結果：



表示項目について、次の表に説明します。

表示項目	説明
cluster_id	クラスタータイプ。 値 "-1" は、クラスタリングがいずれのクラスターセンターにも分類できないことを示します。
rate	値 "-1" は、クラスタリングがいずれのクラスターセンターにも分類できないことを示します。
time_series	クラスターセンターの Timestamp シーケンス
data_series	クラスターセンターのデータシーケンス
instance_names	クラスターセンターに含まれるインスタンスのセット
sim_instance	クラスタリングにおけるインスタンス名

ts_similar_instance

関数の形式：

```
select ts_similar_instance(x, y, z, instance_name)
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
x	時間の列 (昇順)	Unixtime timestamp (秒)
y	指定した時点のデータに対応する数値列	-
z	指定した時点のデータに対応するメトリック名	文字列型の値 (例：machine01.cpu_usr)
instance_name	照会する指定メトリック名	z セット内の文字列値 (例：machine01.cpu_usr) **注：** メトリックは既存のメトリックでなければなりません。

照会と分析のステートメント例：

```
*  and  ( h : " machine_01 "  OR  h : " machine_02 "  OR  h
 : " machine_03 ") |  select  ts_similar  _instance ( stamp ,
 metric_val  ue ,  metric_nam  e , ' nu4e01524 . nu8 ' )  from  (
 select  __time__  -  __time__  %  600  as  stamp ,  avg ( v )  as
   metric_val  ue ,  h  as  metric_nam  e  from  log  GROUP  BY
   stamp ,  metric_nam  e  order  BY  metric_nam  e ,  stamp  )
```

表示項目について、次の表に説明します。

表示項目	説明
instance_name	指定したメトリックに類似するメトリックを含む結果リスト
time_series	クラスターセンターの Timestamp シーケンス
data_series	クラスターセンターのデータシーケンス

6.8.8 頻出パターン統計関数

頻出パターン統計関数は、現在のログを要約するために、指定された複数属性フィールドのサンプルから属性の代表的な組み合わせをマイニングします。

pattern_stat

関数の形式：

```
select pattern_stat(array[col1, col2, col3], array['col1_name',
 'col2_name', 'col3_name'], array[col5, col6], array['col5_name',
 'col6_name'], supportScore, sample_ratio)
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
array[col1, col2, col3]	文字型の値で 構成される入力列	配列形式の値 (例：array[clientIP, sourceIP, path, logstore])
array['col1_name', 'col2_name', 'col3_name']	文字型の値で構成される入力列に対応する名前	配列形式の値 (例：array[‘clientIP’, ‘sourceIP’, ‘path’, ‘logstore’])
array[col5, col6]	数値で構成される入力列	配列形式の値 (例：array[Inflow, OutFlow])
array['col5_name', 'col6_name']	数値で構成される入力列に対応する名前	配列形式の値 (例：array[‘Infl
supportScore	パターンマイニングの正例と負例のサポートレベル	double 型の値。 範囲：(0, 1)。
sample_ratio	サンプリング比はデフォルト値 "0.1" で、サンプル全体の 10% のみが使用されることを示します。	double 型の値。 範囲：(0, 1)。

例：

- 照会と分析の命令文：

```
* |  select   pattern_st  at ( array [  Category ,  ClientIP ,
 ProjectNam  e ,  LogStore ,  Method ,  Source ,  UserAgent  ],
 array [ ' Category ', ' ClientIP ', ' ProjectNam  e ', ' LogStore
```

```
', ' Method ', ' Source ', ' UserAgent ' ], array [ InFlow , OutFlow ], array [ ' InFlow ', ' OutFlow ' ], 0 . 45 , 0 . 3 ) limit 1000
```

・ 結果：

count	supportscore	pattern
468235	0.9880626809484018	InFlow >= 0.0 and InFlow <= 60968.7 and OutFlow >= 0.0 and OutFlow <= 15566.4
459356	0.9693263443991458	Status = '200' and OutFlow >= 0.0 and OutFlow <= 15566.4
458757	0.9680623433187309	Status = '200' and InFlow >= 0.0 and InFlow <= 60968.7
456228	0.9627256843331392	InFlow >= 0.0 and InFlow <= 60968.7 and Status = '200' and OutFlow >= 0.0 and OutFlow <= 15566.4
417662	0.8813442725346703	InFlow >= 0.0 and InFlow <= 60968.7 and UserAgent = 'sls-cpp-sdk v0.6' and Status = '200'
417662	0.8813442725346703	UserAgent = 'sls-cpp-sdk v0.6' and InFlow >= 0.0 and InFlow <= 60968.7
415133	0.8760076135490787	OutFlow >= 0.0 and OutFlow <= 15566.4 and InFlow >= 0.0 and InFlow <= 60968.7 and UserAgent = 'sls-cpp-sdk v0.6' and Status = '200'
415133	0.8760076135490787	OutFlow >= 0.0 and OutFlow <= 15566.4 and UserAgent = 'sls-cpp-sdk v0.6' and InFlow >= 0.0 and InFlow <= 60968.7
415133	0.8760076135490787	OutFlow >= 0.0 and OutFlow <= 15566.4 and UserAgent = 'sls-cpp-sdk v0.6' and Status = '200'
415133	0.8760076135490787	UserAgent = 'sls-cpp-sdk v0.6' and OutFlow >= 0.0 and OutFlow <= 15566.4
414167	0.8739691744110473	InFlow >= 0.0 and InFlow <= 60968.7 and Method = 'PullData' and Status = '200'
414167	0.8739691744110473	Method = 'PullData' and InFlow >= 0.0 and InFlow <= 60968.7

表示項目について、次の表に説明します。

表示項目	説明
count	現在のパターン例の数
supportScore	現在のパターンのサポートレベル
pattern	条件付き照会の形式で構成されたパターンコンテンツ

6.8.9 差分パターン統計関数

差分パターン統計関数は、与えられた複数の属性フィールドの例および条件に基づき、条件に影響を及ぼす差分パターンセットを分析します。 これにより、条件間の差分の原因を迅速に診断できます。

pattern_diff

関数の形式：

```
select
 pattern_diff(array_char_value, array_char_name, array_numeric_value, array_numer
```

パラメータについて、次の表に説明します。

パラメータ	説明	値
array_char_valu	文字型の値で構成される 入力列	配列形式の値 (例：array[clientIP, sourceIP, path, logstore])

パラメータ	説明	値
array_char_name	文字型の値で構成される入力列に対応する名前	配列形式の値 (例：array[‘clientIP’，‘sourceIP’，‘path’，‘logstore’])
*array_numeric_*	数値で構成される入力列	配列形式の値 (例：array[Inflow, OutFlow])
array_numeric_n	数値で構成される入力列に対応する名前	配列形式の値 (例：array[‘Inflow’，‘OutFlow’])
condition	データフィルタリング条件。"True" は正の例を示し、"False" は負の例を示します。	例：レイテンシー ≤ 300
supportScore	パターンマイニングのための正例および負例のサポートの度合い	double 型の値。範囲：(0,1)
posSampleRatio	デフォルト値 "0.5" の正の例のサンプリング比は、正の例の半分だけが使用されていることを示しています。	double 型の値。範囲：(0, 1)
negSampleRatio	デフォルト値 "0.5" の負の例のサンプリング比は、負の例の半分だけが使用されていることを示しています。	double 型の値。範囲：(0, 1)

例：

・照会と分析の命令文：

```
* |  select   pattern_di  ff ( array [  Category ,  ClientIP ,
 ProjectNam  e ,  LogStore ,  Method ,  Source ,  UserAgent  ],
 array [ ' Category ', ' ClientIP ', ' ProjectNam  e ', ' LogStore
 ', ' Method ', ' Source ', ' UserAgent ' ],  array [  InFlow ,
```

```
OutFlow ], array [ ' InFlow ', ' OutFlow ' ], Latency > 300 ,
0 . 2 , 0 . 1 , 1 . 0 ) limit 1000
```

・結果：

possupport	posconfidence	negsupport	diffpattern
0.11304206594120514	1.0	0.0	Category = 'sls_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sls-admin' and LogStore = 'sls_operation_log' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	ProjectName = 'ali-cn-hangzhou-stg-sls-admin' and LogStore = 'sls_operation_log' and Method = 'PostLogStoreLogs' and Source = '10.206.8.163' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sls_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sls-admin' and Method = 'PostLogStoreLogs' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sls_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sls-admin' and Method = 'PostLogStoreLogs' and Source = '10.206.8.163' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	ProjectName = 'ali-cn-hangzhou-stg-sls-admin' and LogStore = 'sls_operation_log' and Source = '10.206.8.163' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sls_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sls-admin' and LogStore = 'sls_operation_log' and Source = '10.206.8.163' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0
0.11304206594120514	1.0	0.0	Category = 'sls_operation_log' and ProjectName = 'ali-cn-hangzhou-stg-sls-admin' and Source = '10.206.8.163' and UserAgent = 'ali-log-logtail' and OutFlow >= 4.9E-324 and OutFlow <= 0.0 and InFlow >= 8800.0 and InFlow <= 8850.0

表示項目について、次の表に説明します。

表示項目	説明
possupport	マイニングされたパターンの正の例のサポートレベル
posconfidence	マイニングされたパターンの正の例の信頼性
negsupport	マイニングされたパターンの負の例のサポートレベル
diffpattern	マイニングされたパターンのコンテンツ

6.9 高度な分析

6.9.1 ケーススタディ

ケースリスト

1. エラー 500% が急上昇したときにアラームをトリガーする
2. トラフィックが急激に減少したときにアラームをトリガーする
3. 各バケットセットの平均レイテンシをデータ間隔別に計算する
4. GROUP BY 結果のパーセントを返します。
5. クエリ条件を満たすログの数を数えます

エラー 500% が急上昇したときにアラームをトリガーする

毎分エラー 500 の割合を数えます。 過去 5 分間にパーセンテージが 40% を超えるとアラームが発生します。

```
status : 500 | select __topic__ , max_by ( error_coun t ,
window_tim e )/ 1 . 0 / sum ( error_coun t ) as error_rati o ,
sum ( error_coun t ) as total_erro r from (
select __topic__ , count (*) as error_coun t , __time__
- __time__ % 300 as window_tim e from log group by
__topic__ , window_tim e

group by __topic__ having max_by ( error_coun t ,
window_tim e )/ 1 . 0 / sum ( error_coun t ) > 0 . 4 and sum (
error_coun t ) > 500 order by total_erro r desc limit
100
```

トラフィックが急激に減少したときにアラームをトリガーする

1 分ごとにトラフィックをカウントします。 最近トラフィックが急激に減少するとアラームが発生します。 最後の 1 分間のデータは 1 分をカバーしません。 したがって、1 分あたりの平均トラフィックをカウントするために、正規化のために統計値を (max(time) - min(time)) で割ってください。

```
* | SELECT SUM ( inflow ) / ( max ( __time__ ) - min ( __time__
)) as inflow_per _minute , date_trunc (' minute ', __time__ )
as minute group by minute
```

各バケットセットの平均レイテンシをデータ間隔別に計算する

```
* | select avg ( latency ) as latency , case when
originSize < 5000 then ' s1 ' when originSize < 20000
then ' s2 ' when originSize < 500000 then ' s3 ' when
originSize < 100000000 then ' s4 ' else ' s5 ' end as os
group by os
```

GROUP BY 結果のパーセントを返します。

異なる部署の集計結果と関連する割合を列挙します。 この問合せは、サブクエリ関数とウィンドウ関数を結合します。 sum(c) over()は、すべての行の値の合計を計算することを示します。

```
* | select department , c * 1 . 0 / sum ( c ) over () from (
select count ( 1 ) as c , department from log groupby
department )
```

クエリ条件を満たすログの数を数えます

特性によって URL を数える必要です。この状況では、CASE WHEN 構文を使用します。 より簡単な count_if構文を使うこともできます。

```
* | select count_if ( uri like '% login ') as login_num
, count_if ( uri like '% register ') as register_n um ,
date_forma t ( date_trunc (' minute ', __time__ ), '% m -% d % H
```

```
:% i ')  as   time   group   by   time   order   by   time   limit
100
```

6.9.2 分析のための最適化クエリ

分析効率は、クエリごとに異なります。 参考に、最適化するための一般的な方法をご紹介します：

可能であれば文字列で Group By を実行しないようにする

Group By を文字列で実行すると、大量のハッシュ計算が行われ、通常は合計計算の 50 %以上を占めます。

例：

```
* |  select   count ( 1 )  as   pv  ,  date_trunc (' hour ', __time__
 )  as   time   group   by   time
* |  select   count ( 1 )  as   pv  ,  from_unixt  ime ( __time__ -
 __time__ % 3600 )  as   time   group   by   __time__ - __time__ %
 3600
```

クエリー 1 とクエリー 2 の両方とも、毎時間ログカウント値を計算します。 ただし、クエリ 1 では時刻が文字列（たとえば、`2017 - 12 - 12   00 : 00 : 00` ）に変換され、この文字列で Group By が実行されます。 クエリ 2 は、時間単位の時間値を計算し、 結果に対して Group By を実行し、その値を文字列に変換します。 クエリー 1 はクエリー 2 よりも効率的ではありません。なぜなら、前者は文字列をハッシュする必要があるからです。

Group By を複数の列で実行するときに、比較的大きな辞書値を持つフィールドを一番上に表示する

たとえば、13 の州には 1 億人のユーザーがいます。

```
Fast : * |  select   province , uid , count ( 1 )  group   by
province , uid
Slow : * |  select   province , uid , count ( 1 )  group   by   uid ,
province
```

推定関数の使用

関数の推定は、正確な計算よりもはるかに強力なパフォーマンスを提供します。 見積もりは、迅速な計算のために許容可能な精度を犠牲にします。

```
Fast : * |  select   approx_dis  tinct ( ip )
Slow : * |  select   count ( distinct ( ip ))
```

SQL で必要な列を取得し、可能であればすべての列を読み取らない

クエリログを使用してすべての列を取得します。 計算を高速化するには、可能な場合は SQL で必要な列のみを取得します。

```
Fast : * | select   a , b   c
```

```
Slow : * | select  *
```

可能な限り集計ではなく、列ごとにグループ化しない関数

例： userid 、 user name に対応するもの。グループのuseridを使います。

```
Fast :  * |  select   userid ,  arbitrary ( username ),  count ( 1 )
groupby   userid
Slow : * |  select   userid ,  username ,  count ( 1 ) groupby
userid , username
```

6.10 JDBC クエリ分析

Overview に加えて、JDBC とスタンダード SQL92 を使用してログをクエリ/分析することができます。

接続パラメータ

接続パラメータ	例	説明
host	regionid.example.com	Service endpoint アクセスポイント。現在は、クラシックネットワークのイントラネットアクセスと、仮想プライベートクラウド（VPC）アクセスのみがサポートされます。
port	10005	10005 がデフォルトのポート番号です。
user	bq2sjzesjmo86kq	AccessKey ID .
password	4fdO1fTDDuZP	AccessKey パスワード.
database	sample-project	ご自身のアカウント内のプロジェクト。
table	sample-logstore	プロジェクト内のLogstore。

以下は、MySQL コマンドによる接続の例です。

```
mysql  - hcn - shanghai - intranet . log . aliyuncs . com  -
ubq2sjzesj  mo86kq  - p4fdO1fTDD  uZP  - P10005
use   sample - project ; // Use   a   project .
```

前提条件

JDBC インタフェースにアクセスするには、プライマリアカウントのアクセスキーまたはサブアカウントを使用する必要があります。 サブアカウントはプロジェクトオーナーに属し、プロジェクトレベルの読み取り権限を持っている必要があります。

構文の説明

注意事項

where 条件にはクエリの時間範囲を制限するために `__date__` または `__time__` が含まれていなければなりません。 `__date__` の型は timestamp で、 `__time__` の型は bigint です。

例：

- `__date__ > ' 2017 - 08 - 07 00 : 00 : 00 ' and __date__ < ' 2017 - 08 - 08 00 : 00 : 00 '`
- `__time__ > 1502691923 and __time__ < 1502692923`

前述の条件の少なくとも1つが満たされていなければなりません。

フィルタ構文

where 条件のフィルタ構文は次のとおりです。

セマンティクス	例	説明
文字検索	`key = " value "`	単語セグメンテーション後の結果が照会されます。
あいまい一致検索	`key = " valu *"`	ワードセグメンテーション後のファジーマッチの結果を照会する。
数値比較	`num_field > 1`	`>`、 `>=`、 `=` 、 `<`、`<=` などの比較演算子がサポートされています。
論理演算	`and or not`	たとえば、 `a =" x "と b =" y "` または `a =” x “で b =” y “`。
全文検索	`__line__ =" abc "`	フルテキストインデックスの検索には特別なキー（ `__line__` ）が必要です。

計算構文

サポートされている計算演算子については、解析構文を参照してください。

SQL92 構文

SQL92 構文は、フィルタ構文と計算構文の組み合わせです。

以下のクエリが例として使用されています。

```
status > 200  | select   avg ( latency ), max ( latency ) , count ( 1
)  as   c   GROUP   BY   method   ORDER   BY   c   DESC   LIMIT   20
```

クエリのフィルタ部分と時間条件は、標準の SQL92 構文に基づいて新しいクエリ条件に結合できます。

```
select   avg ( latency ), max ( latency ) , count ( 1 )  as   c
from   sample - logstore   where   status > 200   and   __time__ >=
1500975424   and   __time__  <  1501035044   GROUP   BY   method
ORDER   BY   c   DESC   LIMIT   20
```

JDBC プロトコルによるアクセス

プログラム呼び出し

開発者は MySQL 構文を使用して、 MySQLコネクタをサポートするプログラムでログサービスに接続できます。 たとえば、JDBC または Python MySQLdb を使用できます。

例：

```
import   com . mysql . jdbc .*;
Import   java .  SQL  .*;
import   java . sql . Connection ;
import   java . sql . ResultSetM  etaData ;
import   java . sql . Statement ;
public   class   testjdbc  {
    public   static   void   main ( String   args []){
        Connection   conn  =  null ;
        Statement   stmt  =  null ;
        try  {
           // STEP   2 :  Register   JDBC   driver
            Class . forName (" com . mysql . jdbc . Driver ");
           // STEP   3 :  Open   a   connection
            System . out . println (" Connecting   to   a   selected
  database ...");
            conn  =  DriverMana  ger . getConnect  ion (" jdbc :
mysql :// cn - shanghai - intranet . log . aliyuncs . com : 10005 /
sample - project "," accessid "," accesskey ");
            System . out . println (" Connected   database
successful  ly ...")
           // STEP   4 :  Execute   a   query
            System .  Out .  println  (" creating   statement  ...");
            stmt  =  conn . createStat  ement ();
            String   sql  = " SELECT   method , min ( latency , 10
)  as   c , max ( latency , 10 )  from   sample - logstore   where
  __time__ >= 1500975424   and   __time__  <  1501035044   and
latency  >  0   and   latency  <  6142629   and   not  ( method ='
Postlogsto  relogs '  or   method =' GetLogtail  Config ')  group
by   method  " ;
            String   sql - example2  = " select   count ( 1 ) , max
( latency ), avg ( latency ),  histogram ( method ), histogram (
source ), histogram ( status ), histogram ( clientip ), histogram
( __source__ )  from   test10   where   __date__  >' 2017 - 07 - 20
  00 : 00 : 00 '  and   __date__  <' 2017 - 08 - 02   00 : 00 : 00
'  and   __line__ =' abc # def '  and   latency  <  100000   and  (
```

```
method  = ' getlogstor  elogS '  or   method =' Get **'  and   method
 <> ' GetCursorO  rData ' )";
            String   sql - example3  = " select   count ( 1 )  from
  sample - logstore   where   __date__  > ' 2017 - 08 - 07   00 : 00
: 00 '  and   __date__  < ' 2017 - 08 - 08   00 : 00 : 00 '  limit
100 ";
            ResultSet   rs  =  stmt . executeQue  ry ( sql );
           // STEP   5 :  Extract   data   from   result   set
            while ( rs . next ()){
               // Retrieve   by   column   name
                ResultSetM  etaData   data  =  rs . getMetaDat  a ();
                System . out . println ( data . getColumnC  ount ());
                for ( int   i  =  0 ; i  <  data . getColumnC  ount
();++ i ) {
                    String   name  =  data . getColumnN  ame ( i + 1
);
                    System . out . print ( name +":");
                    System . out . print ( rs . getObject ( name ));

                System . out . println ();

            Rs .  Close  ();
        }  catch  ( ClassNotFo  undExcepti  on   e ) {
            e . printStack  Trace ();
        }  catch  ( SQLExcepti  on   e ) {
            e . printStack  Trace ();
        }  catch  ( Exception   e ) {
            E .  printstack  trace  ();
        }  Finally  {
            if  ( stmt  ! =  null ) {
                try  {
                    Stmt .  Close  ();
               }  catch  ( SQLExcepti  on   e ) {
                    e . printStack  Trace ();


            if  ( conn  ! =  null ) {
                try  {
                    conn . close ();
               }  catch  ( SQLExcepti  on   e ) {
                    e . printStack  Trace ();
```

ツールクラスの呼び出し

ネットワークイントラネットまたは VPC 環境では、 接続に MySQL クライアントを使用します。

注 :

1. **①にプロジェクト名を入力します。**
2. **②にログストア名を入力します。**

7 可視化分析

7.1 分析グラフ

7.1.1 グラフについて

Log Service には、SQL に似た集計機能があります。また、Log Service の可視化グラフに SQL 集計結果を表示することができます。

> 注：
> 可視化グラフを使用する前に、「クエリ構文の詳細」をご参照ください。

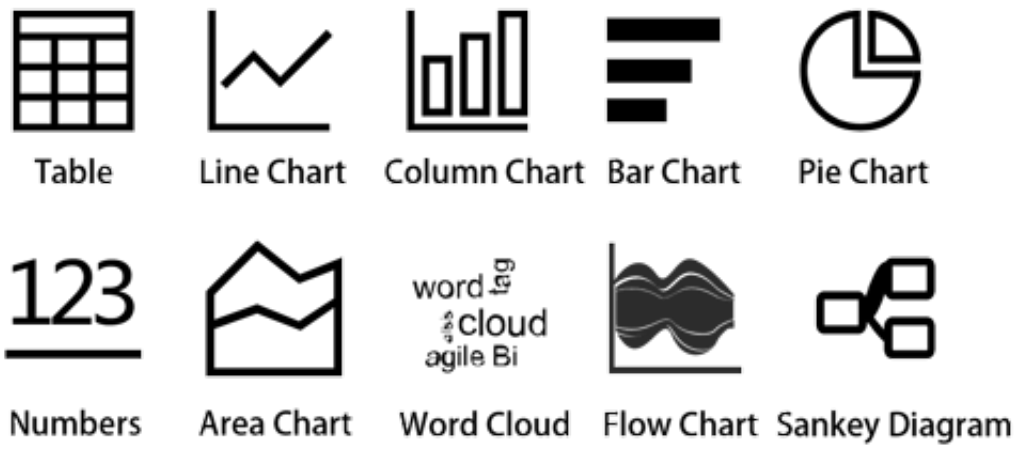
前提条件

1. インデックスを作成し、分析を有効にします。
2. グラフは、分析クエリ構文による統計結果のみを表示させることができます。

グラフの種類

現在、 Log Service には、次のグラフが用意されています。

図 7-1 : グラフの種類



各グラフの使用方法については、以下のドキュメントをご参照ください。

- 表
- 折れ線グラフ
- 棒グラフ
- 横棒グラフ

- 円グラフ
- 数値ダイアグラム
- 面グラフ
- 曲線グラフ
- サンキーダイアグラム
- ワードクラウド

7.1.2 表

表は、最も一般的なデータ表示の形式であり、データの整理に用いられる最も基本的な方法です。データを並べて整理することにより、一目でデータを照会/分析することができます。Log Service には、SQL と同様の集約機能があります。クエリ分析ステートメントの結果は、デフォルトでは表形式に表示されます。

基本コンポーネント

- ヘッダー
- 行
- 列

詳細:

- 列数は、`SELECT` の項目数
- 行数は、指定した期間のログ数により変動 (デフォルトは上限 `100`)

手順

1. クエリページで、クエリ欄にクエリステートメントを入力し、期間を選択して、クエリをクリックします。
2. グラフタブをクリックすると、デフォルトでクエリ結果が表形式  で表示されます。

例

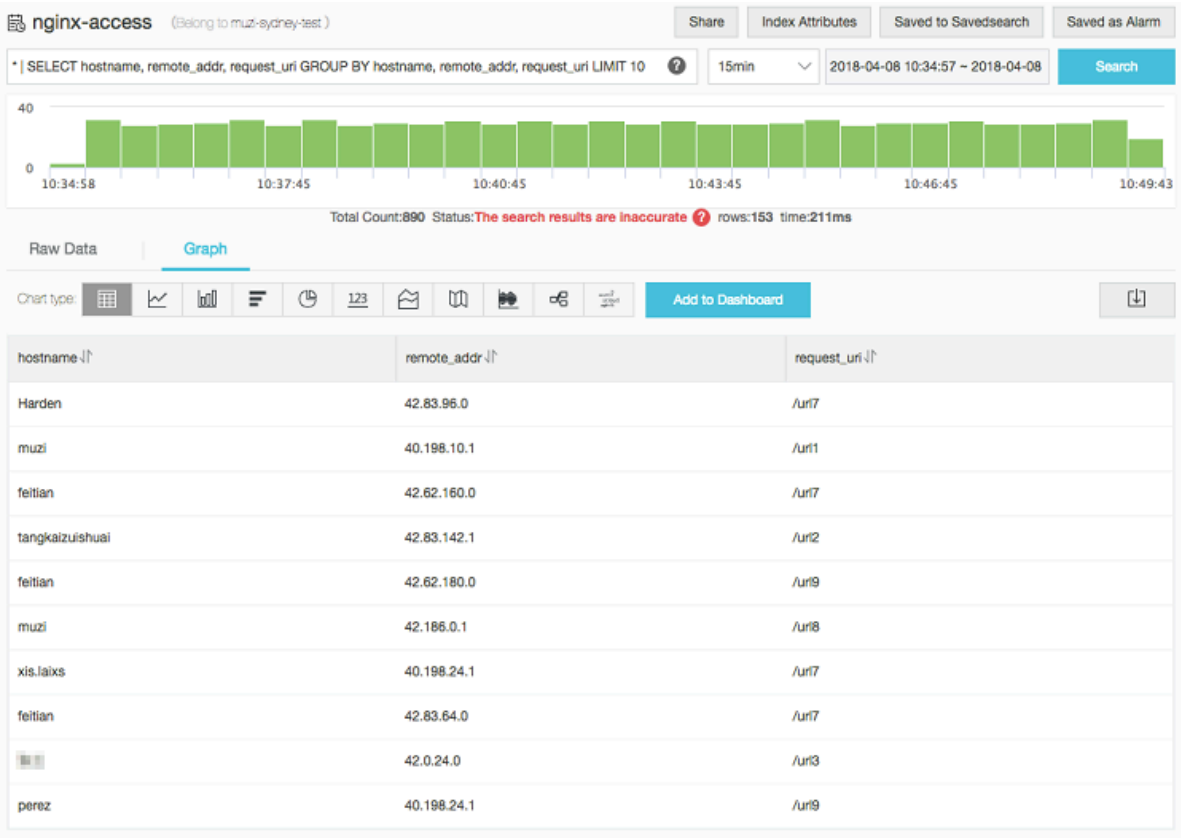
以下がオリジナルのログであった場合、

図 7-2 : オリジナルのログ

1. 直近 10 件のログの `hostname` 、 `remote_addr r` 、および `request_ur i` の各列の値を取得

```
| SELECT hostname , remote_addr r , request_ur i GROUP BY
  hostname , remote_addr r , request_ur i LIMIT 10
```

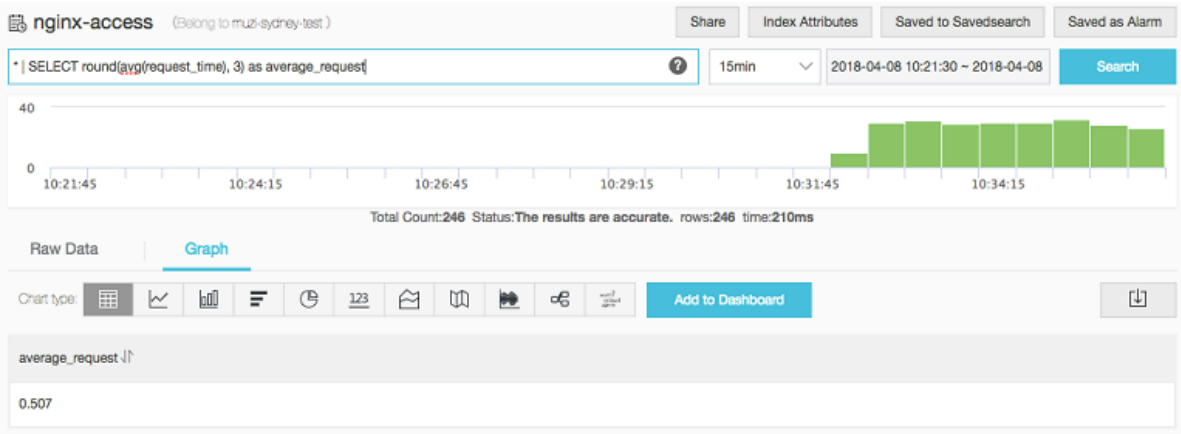
図 7-3 : ケース 1



2. 単一データ (例: 指定した期間における `request_ti me` の平均、平均リクエスト数) の小数点以下 3 桁を四捨五入して表示

```
* | SELECT round ( avg ( request_ti me ), 3 ) as average_re
quest
```

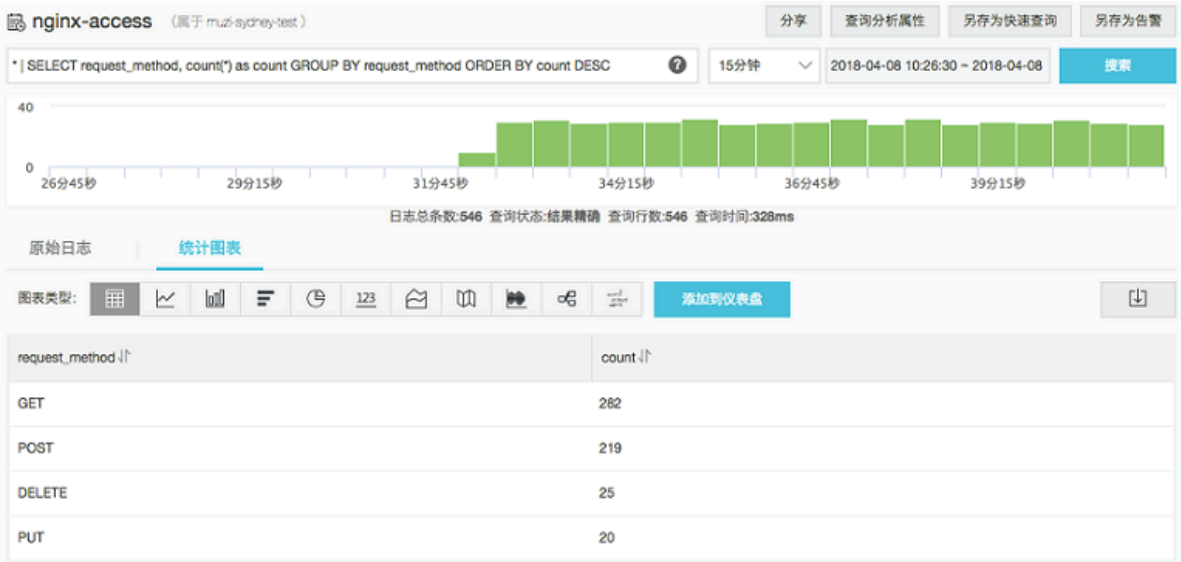
図 7-4 : ケース 2



3. グループ化データ (例: 指定した期間における `request_me thod` の時間分布) を降順に表示

```
* | SELECT request_me thod , count (*) as count GROUP BY
request_me thod ORDER BY count DESC
```

図 7-5 : ケース 3



7.1.3 折れ線グラフ

折れ線グラフは、傾向分析に用いられます。特定のカテゴリのデータを秩序づけて (通常は時系列)、直感的に傾向分析することができます。

折れ線グラフを使用することにより、ある一定期間におけるデータの変化が明確になります。次のような変化を表示する場合に用いられます。

- 漸増/漸減
- 増減率
- 増減の規則性 (周期的な変化など)
- ピークとボトム

したがって、折れ線グラフは時間の経過とともにデータがどのように変化しているか、傾向を分析するのに最適です。複数の折れ線が表示されることで、期間内における各データの傾向、および、関連を分析することができます (同時期に増加/減少している、反比例しているなど)。

基本コンポーネント

- X 軸
- 左 Y 軸
- 右 Y 軸 (オプション)
- データポイント
- 傾向変化線
- 凡例

設定項目

設定項目	説明
X 軸	通常、秩序づけられているカテゴリデータ (時系列)
左 Y 軸	左 Y 軸の値の範囲となる 1 つまたは複数のデータ列
右 Y 軸	右 Y 軸の値の範囲となる 1 つまたは複数のデータ列 (右 Y 軸が、左 Y 軸に重なる)
縦マーカー	左 Y 軸または右 Y 軸のいずれかを円柱表示
凡例	凡例の位置 (グラフの上部、下部、左または右)
パディング	座標軸とグラフ境界線の間隔

手順

1. クエリページで、検索ボックスにクエリステートメントを入力します。期間を選択して、検索をクリックします。
2. グラフタブの (折れ線グラフ) をクリックします。
3. グラフのプロパティを設定します。

> 注：
> データ傾向を分析するための折れ線グラフには、2 つ以上のデータレコードが必要になります。なお、折れ線グラフの線は、5 本以下にすることを推奨します。

例

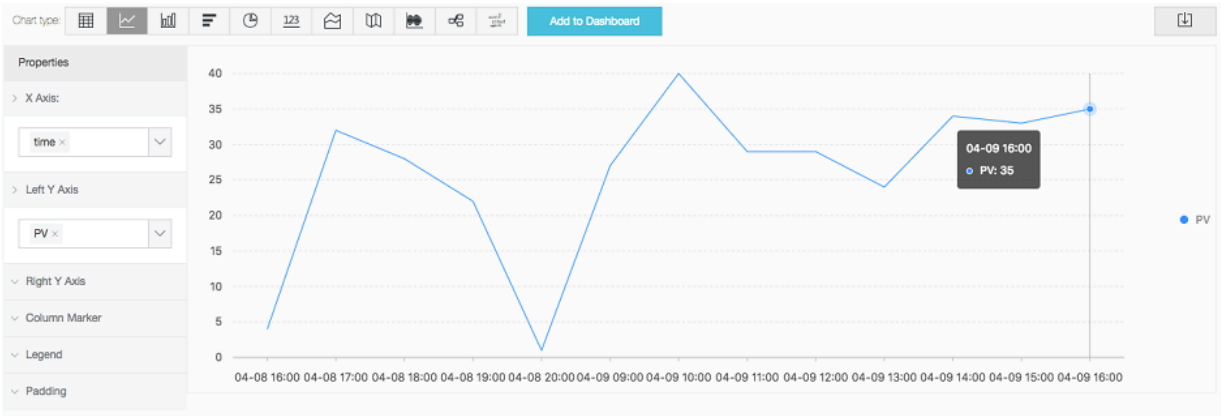
単純折れ線グラフ

1 日前の IP アドレス `42 . 0 . 192 . 0` のアクセス状況を照会する場合、ステートメントは次のようになります。

```
remote_add r : 42 . 0 . 192 . 0 | select date_forma t (
date_trunc (' hour ', __time__ ), '% m -% d % H :% i ')
as time , count ( 1 ) as PV group by time order by
time limit 1000
```

X 軸に `time` 、左 Y 軸に `PV` 、右 Y 軸に UV、列マーカーに PV を指定します。

図 7-6 : 単純折れ線グラフ



軸折れ線グラフ

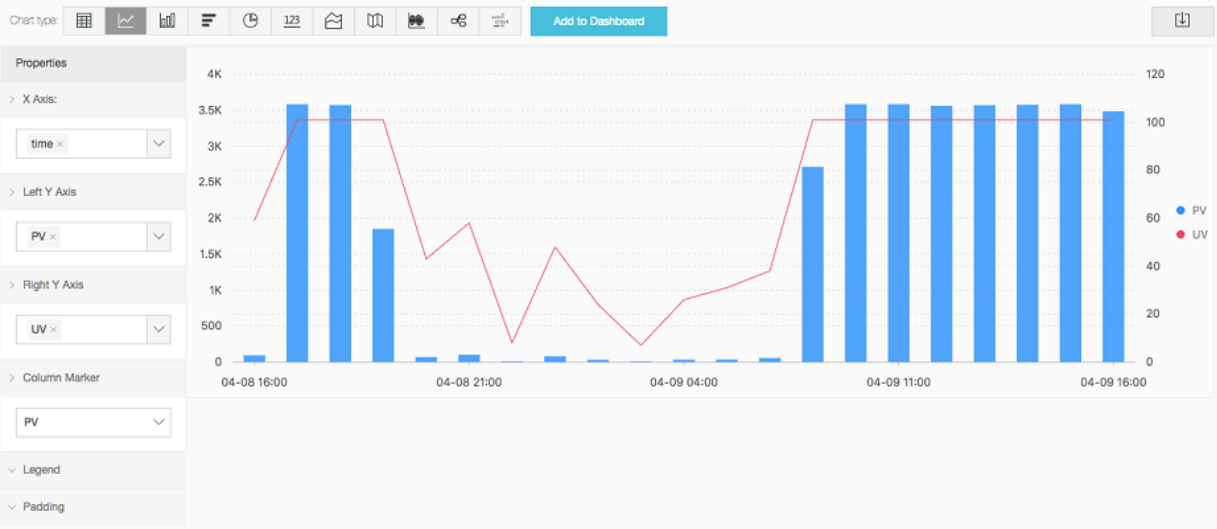
1 日前の PV および UV のアクセス状況を照会する場合、ステートメントは次のようになります。

```
* | select date_forma t ( date_trunc (' hour ', __time__ ), '% m
 -% d % H :% i ') as time , count ( 1 ) as PV , approx_dis
```

```
tinct ( remote_add r ) as UV group by time order by
time limit 1000
```

X 軸に `time` 、左 Y 軸に `PV` 、右 Y 軸に `UV` 、列マーカーに `PV` を指定します。

図 7-7 : 軸折れ線グラフ



7.1.4 棒グラフ

棒グラフは、カテゴリの数値データを縦棒または横棒で比較します。折れ線グラフは、順序付けられたデータを表示しますが、棒グラフは異なるカテゴリデータを表示して、各カテゴリデータの数を表示します。

また、1 つの属性に対して、複数の長方形の図形を使用して、集合棒グラフまたは積み上げ棒グラフで、異なる側面からカテゴリデータの違いを分析することもできます。

基本コンポーネント

- X 軸 (横軸)
- Y 軸 (垂直軸)
- 長方形の図形
- 凡例

Log Service の棒グラフは、デフォルトで縦棒を使用します。長方形の図形の横幅は固定されており、高さは数値を表示します。 複数のデータ列が Y 軸にマッピングされている場合、集合棒グラフを使用して、データを表示します。

設定項目

設定項目	説明
X 軸	通常、X 軸のカテゴリデータ

設定項目	説明
Y 軸	Y 軸の値の範囲となる 1 つまたは複数のデータ列
凡例	凡例の位置 (グラフの上部、下部、左または右)
パディング	座標軸とグラフ境界線の間の間隔

手順

1. クエリページで、検索ボックスにクエリステートメントを入力し、時間間隔を選択して、検索をクリックします。
2. グラフタブの (棒グラフ) をクリックします。
3. グラフのプロパティを設定します。

> 注 :
> データカテゴリの数が 20 以下の場合は、棒グラフの使用を推奨します。横幅が広すぎると、解析および比較結果を直感的に表示できないため、データの数を制御するために `LIMIT` でデータ数を制御するすることを推奨します。なお、Y 軸にマッピングするデータの列は 5 つ以下にします。

例

単純棒グラフ

次のステートメントで、指定した期間における各 `http_refer er` の訪問数を照会します。

```
* | select http_refer er , count ( 1 ) as count group by
 http_refer er
```

X 軸に `http_refer er` 、Y 軸に `count` を指定します。

図 7-8 : 単純棒グラフ

集合棒グラフ

次のステートメントで、指定した期間における各 `http_refer er` の訪問数および平均バイト数を照会します。

```
* | select http_refer er , count ( 1 ) as count , avg ( body_bytes _sent ) as avg group by http_refer er
```

X 軸に `http_refer er`、Y 軸に `count` および `avg` を指定します。

図 7-9 : 集合棒グラフ

7.1.5 横棒グラフ

横棒グラフは、棒が横に並んだ、棒グラフの一種です。通常、横棒グラフは、上位分析に使用します。設定方法は縦棒グラフと同様です。

基本コンポーネント

- X 軸 (縦軸)
- Y 軸 (横軸)
- 長方形の図形
- 凡例

横棒グラフの長方形の図形の高さは固定で、幅は数値を表します。 複数のデータ列が Y 軸にマッピングされている場合は、集合横棒グラフを使用してデータを表します。

設定項目

表 7-1 : 説明

説明	説明
X 軸	通常、X 軸はカテゴリデータ

説明	説明
Y 軸	Y 軸の値の範囲となる 1 つまたは複数のデータ列
凡例	凡例の位置 (グラフの上部、下部、左または右)
パディング	座標軸とグラフ境界線の間の間隔

手順

1. クエリページで、検索ボックスにクエリステートメントを入力し、時間間隔を選択して、検索をクリックします。
2. グラフタブの (棒グラフ) をクリックします。
3. グラフのプロパティを設定します。

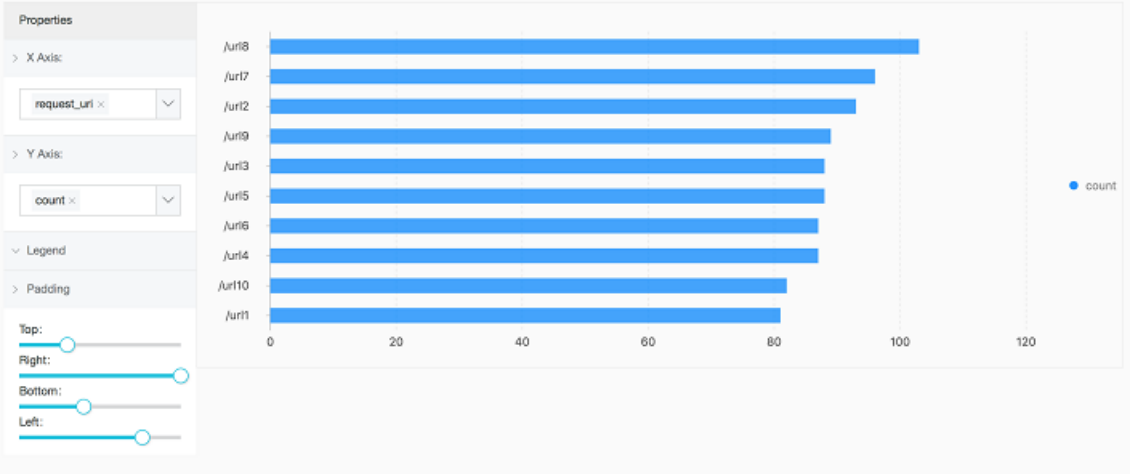
> 注 :
>
> - データカテゴリの数が 20 以下の場合、棒グラフを使用します。高さの値が大きすぎると、直感的な解析と比較結果を表示できないので、 `LIMIT` でデータの数を制御します。top ランキングを解析する際には `ORDER BY` 構文の使用を推奨します。また、Y 軸にマッピングするデータの列は 5 つ以下にします。
> - 集合横棒グラフを使用することはできますが、同時に増減するカテゴリのみに適しています。

単純横棒グラフ

アクセス数の多い上位 10 の request_uri を分析するクエリ構文は、次のとおりです。

```
* | select request_ur i , count ( 1 ) as count group by
 request_ur i order by count desc limit 10
```

図 7-10 : 単純横棒グラフ



7.1.6 円グラフ

円グラフは、各カテゴリの比率を表示する場合に使用します。カテゴリの比率は円弧の長さで表現されます。円は、構成カテゴリ数に合わせて複数に分割されます。円全体は全データの合計であり、各セクションは各カテゴリのデータ量と全データとの比率を示します。すべてのセクション (円弧) を合計すると 100% になります。

基本構成要素

- 扇形
- パーセンテージ表示テキスト
- 凡例

構成項目

設定項目	説明
カテゴリ	カテゴリのデータ
数値列	各カテゴリのデータ
凡例	凡例の位置 (グラフの上部、下部、左、または右)
パディング	座標軸とグラフ境界線との間隔

設定項目	説明
円グラフの種類	円グラフ (デフォルト)、ドーナツグラフ、または、ナイチンゲールローズダイアグラム

種類

Log Service には、円グラフ、ドーナツ グラフ、およびナイチンゲールローズダイアグラムの 3 種類の円グラフが用意されています (デフォルト: 円グラフ)。

ドーナツグラフ

円グラフの中心部を取り除き、ドーナツの形をした円形グラフです。円グラフと比較して、ドーナツグラフには次の利点があります。

- 基本的な円グラフに、合計数を表示することができ、情報量が多くなります。
- 2 つの円グラフで直感的に比較するのは難しいですが、ドーナッツグラフには円が 2 つあるため、各円の長さで直感的に比較することができます。

ナイチンゲールローズダイアグラム

ナイチンゲールローズダイアグラムは厳密には円グラフではなく、極座標系の棒グラフです。各カテゴリは同じ角度の扇形で分割され、扇形の半径はデータ値を示します。円グラフと比較して、ナイチンゲールローズダイアグラムには次の利点があります。

- 円グラフはカテゴリ数が 10 以下の場合に適しています。カテゴリ数が 11 ～ 30 の場合はナイチンゲールローズダイアグラムを使用します。
- 面積は、円の半径の 2 乗に比例するため、ナイチンゲールローズダイアグラムは、カテゴリ間のデータ値の差が拡大されます。したがって、データ値の近いものを比較するのに適しています。
- 円には周期性があるので、 鶏のとさか図は、週や月などの周期的な時間の概念にも適用されます。

手順

1. クエリページのクエリ欄にクエリステートメントを入力し、期間を選択して、照会 をクリックします。
2. グラフタブの (円グラフ) をクリックします。
3. 注意事項

> 注 :

- カテゴリ数が 10 以下の場合は、円グラフまたはドーナツグラフを使用します。色分けするセクション数が多すぎると、分析結果が見づらくなるため、LIMIT を指定してカテゴリ数に上限を設定されることを推奨します。
- カテゴリ数が 10 を超える場合は、ナイチンゲールローズダイアグラムまたは棒グラフを推奨します。

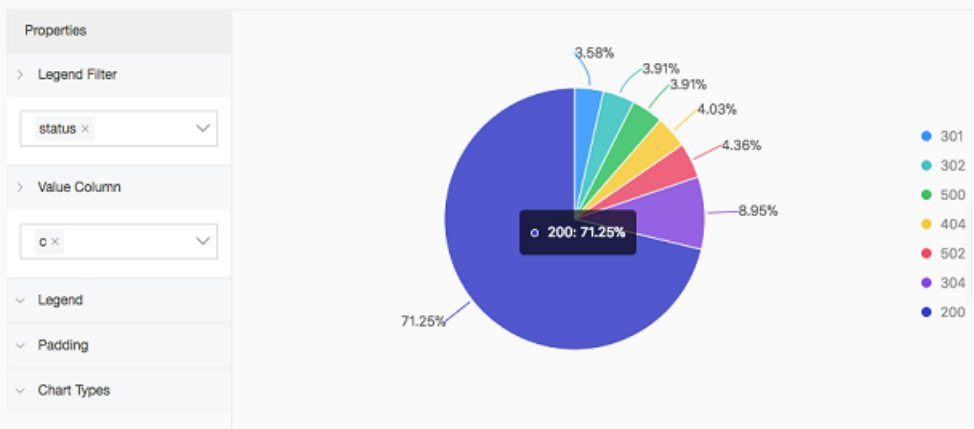
例

円グラフ

全アクセスにおける各 HTTP `status` の比率を分析:

```
* | select status , count ( 1 ) as c group by status
 order by c limit 10
```

図 7-11 : 円グラフ

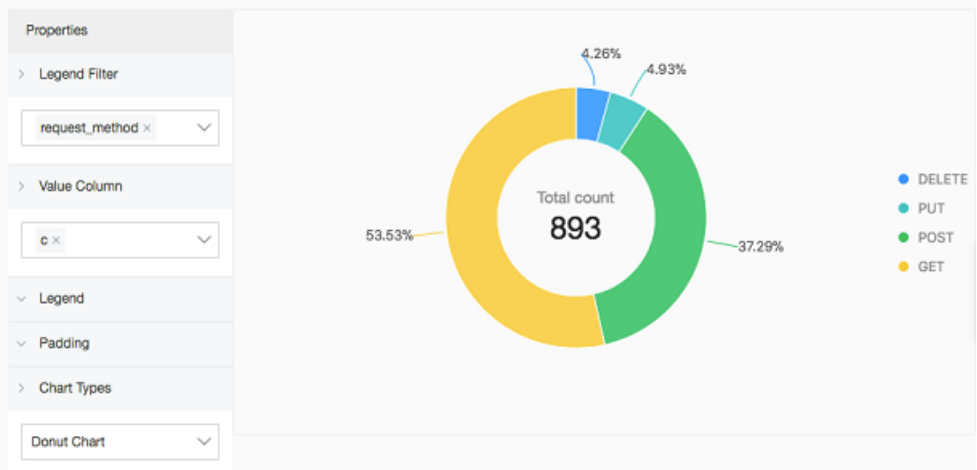


ドーナツグラフ

全アクセスにおける各 `request_me thod` の比率を分析:

```
* | select request_me thod , count ( 1 ) as c group by
 request_me thod order by c limit 10
```

図 7-12 : ドーナツグラフ

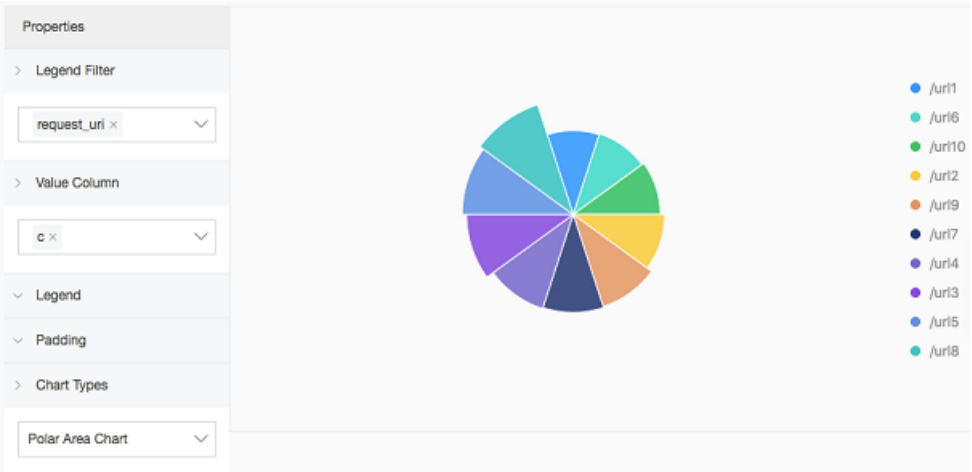


ナイチンゲールローズダイアグラム

全アクセスにおける各 `request_ur i` の比率を分析:

```
* | select request_ur i , count ( 1 ) as c group by
 request_ur i order by c
```

図 7-13 : ナイチンゲールローズダイアグラム



7.1.7 面グラフ

面グラフは折れ線グラフをベースに、折れ線グラフの線と座標軸との間が色で塗りつぶしたものです。塗りつぶされた部分が面 (エリア) であり、その傾向を色で強調表示させます。 折れ線グラフと同様に、面グラフは時間の経過におけるデータの変化が示され、総数の傾向を強調させることができます。折れ線グラフも面グラフも、特定の値ではなく、傾向や比較に用いられます。

基本コンポーネント

- X 軸 (横軸)
- Y 軸 (縦軸)
- エリアブロック

構成項目

設定項目	説明
X 軸	通常、順序付けられたカテゴリデータ (時系列)
Y 軸	Y 軸の値の範囲となる 1 つまたは複数のデータ列
凡例	凡例の位置 (グラフの上部、下部、左、または右)
パディング	座標軸とグラフ境界線との間隔

手順

1. クエリページで、クエリ欄にクエリステートメントを入力し、期間を選択して、 照会をクリックします。
2. グラフタブの  (面グラフ) をクリックします。
3. グラフのプロパティを設定します。

> 注 :
> 1 つのエリアブロックにつき、データレコードが 2 つ以上ない場合、データの傾向を分析することはできません。なお、面グラフのエリアブロックの数は 5 つ以下にされることを推奨します。

例

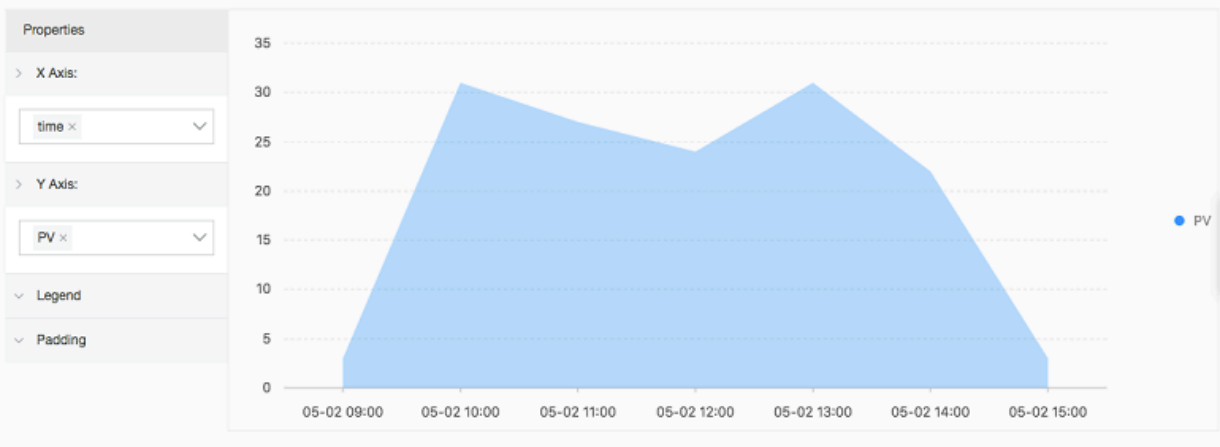
単純な面グラフ

前日の IP `42 . 0 . 192 . 0` の PV を表示:

```
remote_add r : 42 . 0 . 192 . 0 | select date_forma t (
date_trunc (' hour ', __time__ ), '% m -% d % H :% i ') as time
, count ( 1 ) as PV group by time order by time
limit 1000
```

X 軸に `time` 、Y 軸に `PV` を指定します。

図 7-14 : 単純な面グラフ

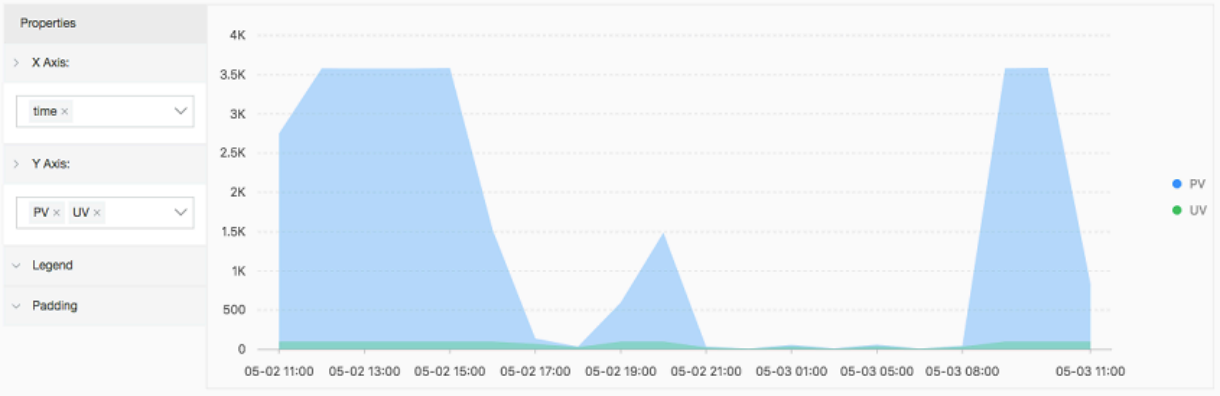


積み上げ面グラフ

```
* | select date_forma t ( date_trunc (' hour ', __time__ ), '% m
-% d % H :% i ') as time , count ( 1 ) as PV , approx_dis
tinct ( remote_add r ) as UV group by time order by
time limit 1000
```

X 軸に `time` 、Y 軸に `PV` および `UV` を指定します。

図 7-15 : 積み上げ面グラフ



7.1.8 数値ダイアグラム

数値ダイアグラムでは、単一の数値を強調表示します。 数値ダイアグラムの種類は次のとおりです。

- 長方形ボックス: 一般的な値を表示
- ダイアル: 値としきい値との差を表示
- 数値比較: 前年比関数および期間比関数の SQL 照会結果を表示 (クエリステートメントは、区間値比較および周期性値比較関数を参照)

デフォルトでは、長方形ボックスが表示されます。長方形ボックスは、最も単純で直接的なデータ表現です。ある時点におけるデータをわかりやすく可視化します。通常、ある時点における重要な情報を表示するときに使用します。比率を表示する場合は、ダイアルを使用します。

基本コンポーネント

- メインテキスト
- 単位 (オプション)
- 説明 (オプション)
- カテゴリー

設定項目

- 長方形ボックスを設定

長方形ボックスの設定	説明
グラフの種類	長方形ボックス
値列	デフォルト: 指定した列の最初の行のデータ
テキスト	テキストサイズの設定 - フォントサイズ (12 px～100 px) - 単位のフォントサイズ (12 px～100 px) - 説明のフォントサイズ(12 px～100 px)
色	ダイアグラムの色の設定 - フォントの色 - 背景色

- ダイアルを設定

ダイアルの設定項目	構成項目	説明
グラフの種類	ダイアル	クエリ結果をダイアル表示
値列	実際の値	デフォルト: 指定した列の最初の行のデータ

ダイアルの設定項目	構成項目	説明
	単位	ダイアルの値の単位
	ダイアルの最大値	ダイアルの最大値 (デフォルト: 100)
	エリア色数	ダイヤルはいくつかの数値エリアに分割される (各エリアは色で分けられる) 指定可能なエリア色数: 最大: 5、デフォルト: 3
	各エリアの最大値	ダイアルの各エリアの最大値。最後のエリアの最大値は、デフォルトでダイヤルの最大値であり、指定する必要がありません。 注 : ダイヤルは、デフォルトでは 3 色のエリアに均等に分割されます。エリアの色数を変更しても、色分けされた各エリアの値の範囲は変更されません。色分けされた各エリアの最大値は、適宜指定します。
	タイトルの表示	ダイアルのタイトルの表示/非表示 (デフォルト: 非表示) 有効化ボタンをクリックしても、現在のページにタイトルは表示されません。タイトルは、レポートを作成した後、または現在のレポートを変更した後、ダッシュボードページに表示されます。
テキスト	フォントサイズ	テキストのフォントサイズ (12 px～100 px の範囲)
	説明	数値の説明
	説明 のフォントサイズ	説明のフォントサイズ (12 px～100 px の範囲)
色	エリアの色	デフォルトでは、3 つのエリアに色分けされる (青、黄、赤) エリアの色数を 3 以上に変更する場合、デフォルトでは、追加エリアは青色になります。 各エリアの色は変更可

ダイアルの設定項目	構成項目	説明
	フォントの色	ダイアルに表示される数値の色

・ 数値比較ダイアグラムを設定

構成カテゴリ	構成項目	説明
グラフの種類	数値比較	照会結果を数値比較ダイアグラムで表示
値列	表示値	数値比較ダイアグラムの主要な値 (通常、比較関数の現時点における統計結果)
	比較値	しきい値と比較する値 (通常、数値比較ダイアグラムの現期間と前期間との比較結果)
	傾向の基準値	比較値の変動傾向を測定するために使用される値
テキスト	フォントサイズ	表示値のフォントサイズ (指定可能な値: 12px ～ 100px)
	単位	表示値の単位
	単位のフォントサイズ	表示値の単位のフォントサイズ (指定可能な値: 12px ～ 100px)
	比較単位	比較値の単位
	比較のフォントサイズ	比較値と傾向のフォントサイズ (指定可能な値: 12px ～ 100px)
	説明	表示値とその変動についての説明 (値の下部に表示)
	説明のフォントサイズ	説明のフォントサイズ (指定可能な値: 12px ～ 100px)
色	フォント色	表示値のフォント色
	増加時のフォント色	比較値がしきい値より大きい場合の比較値のフォント色
	増加時の背景色	比較値がしきい値より大きい場合の背景色
	減少時のフォント色	比較値がしきい値より小さい場合の比較値のフォント色
	減少時の背景色	比較値がしきい値より小さい場合の背景色
	均等時の背景色	比較値としきい値が等しい場合の背景色

手順

1. クエリページのクエリ欄にクエリステートメントを入力し、時間間隔を選択して、検索をクリックします。
2. グラフタブの123 (数値ダイアグラム) をクリックします。
3. 必要に応じてグラフの種類を選択し、グラフのプロパティを設定します。

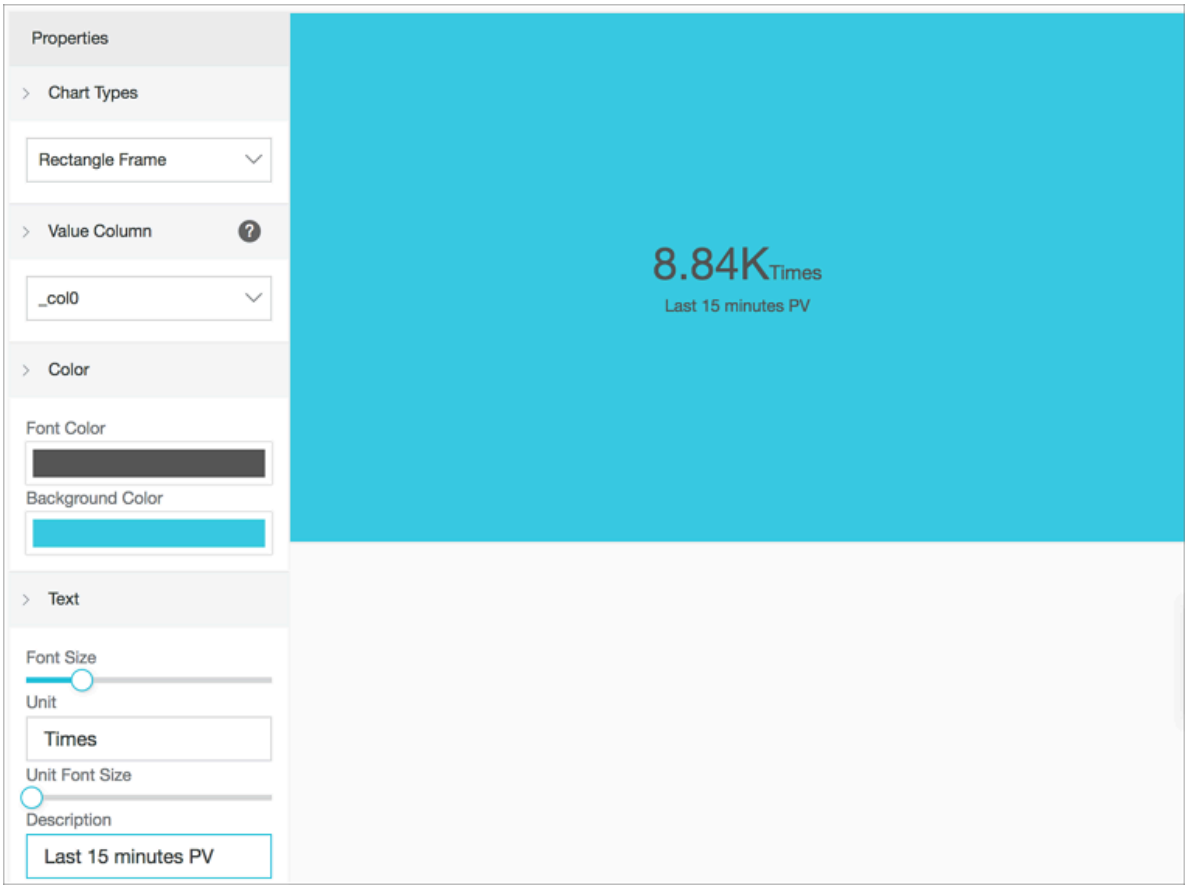
> 注：
> Log Service 数値ダイアグラムの数値は、値に応じて自動的に正規化されます。たとえば、`230000` は `230K` に処理されます。数値の書式を定義する場合は、 数学計算関数 でリアルタイム解析します。

例

訪問者数を各表示形式で表示するための各クエリ分析ステートメントは、以下のとおりです。

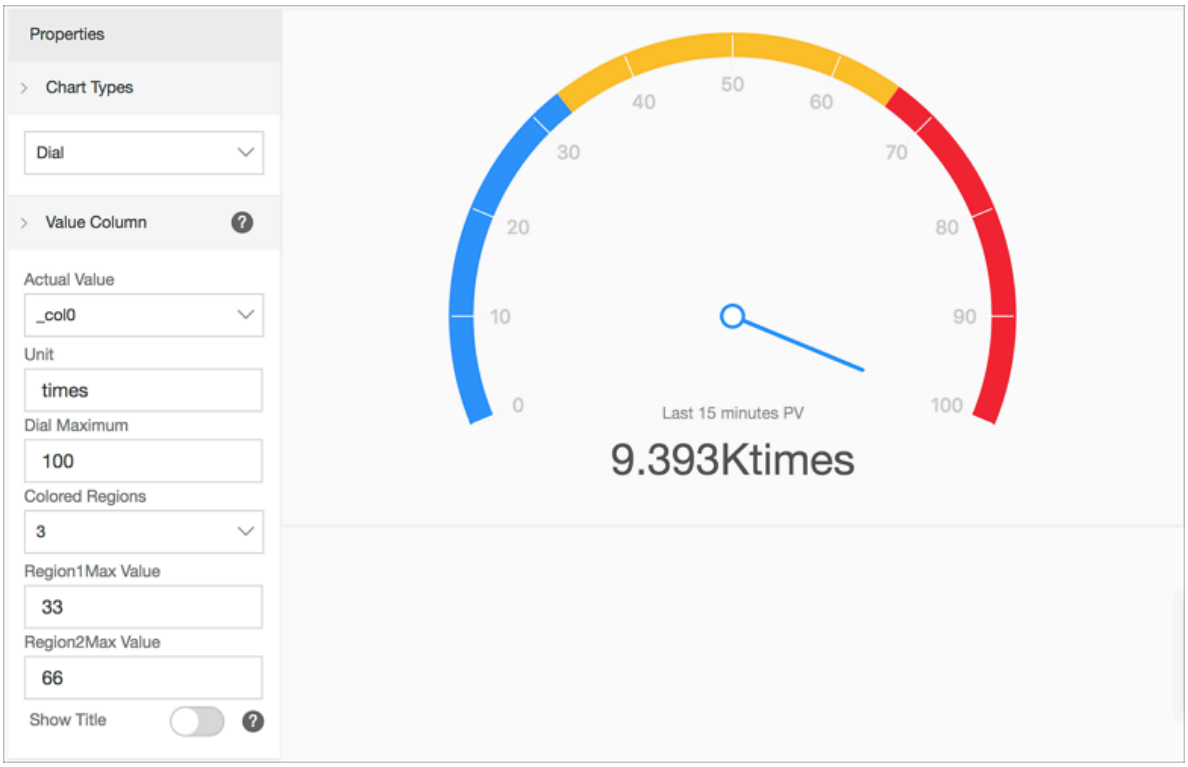
・ 長方形ボックス

```
* |  select   count ( 1 )  as   pv
```



- ダイアル

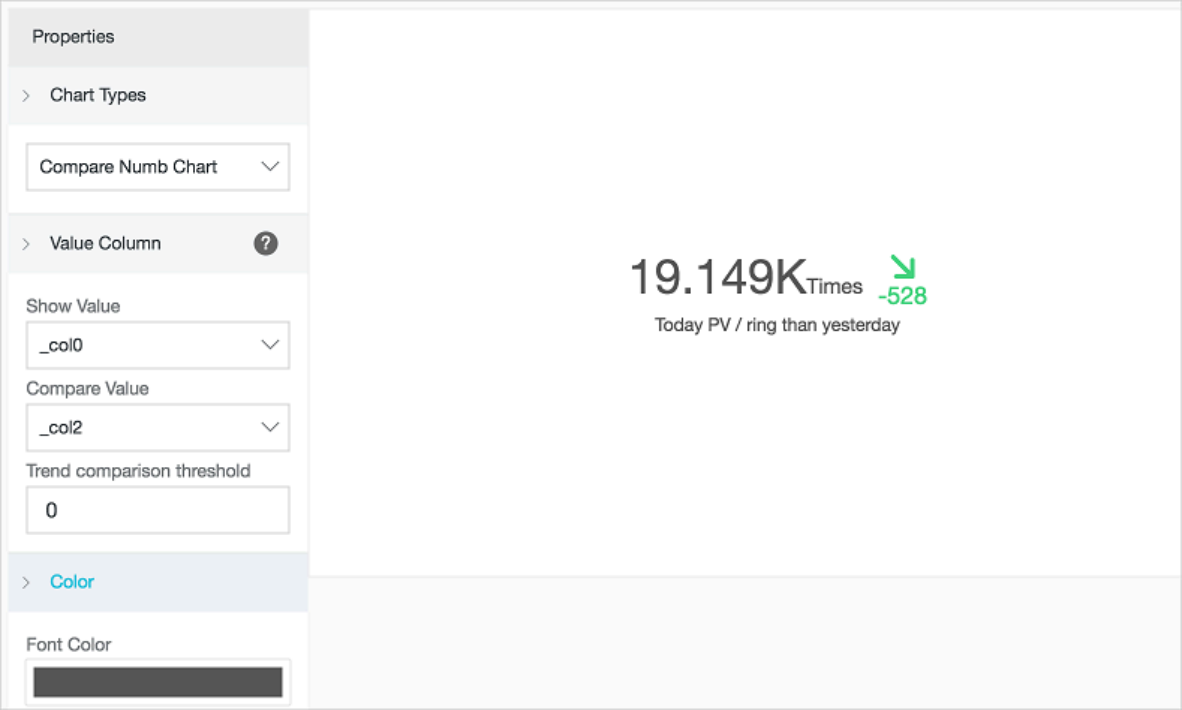
```
* | select count ( 1 ) as pv
```



- 前年比/前日比

  本日および昨日の訪問数を比較

```
* |  select   diff [ 1 ], diff [ 2 ],  diff [ 1 ]- diff [ 2 ]  from
  ( select   compare (  pv  ,  86400 )  as   diff   from  ( select
 count ( 1 )  as   pv   from   log ))
```



7.1.9 曲線グラフ

曲線グラフ (別名 ThemeRiver) は、中心軸を中心にした積み上げ面グラフです。各曲線は色でカテゴリ分けされ、曲線の幅はその数値を表します。なお、元のデータの時間属性は、X 軸にマッピングされ、3 次元で関係が表されます。

曲線グラフは、折れ線グラフまたは棒グラフに表示を切り替えることができます。なお、棒グラフは、デフォルトで積み重ねて表示され、各データカテゴリの始点は、その前の列の先頭になることにご注意ください。

基本コンポーネント

- X 軸 (横軸)
- Y 軸 (縦軸)
- 曲線

設定項目

設定項目	説明
X 軸	通常、順序付けられたカテゴリデータ (時系列)
Y 軸	Y 軸の値の範囲となる 1 つまたは複数のデータ列
集計列	3 次元の集計データ
凡例	凡例の位置 (グラフの上部、下部、左または右)
パディング	座標軸とグラフ境界線との間隔
グラフの種類	面グラフ (デフォルト)、折れ線グラフ、棒グラフ (積み上げ)

手順

1. クエリページで、検索欄にクエリステートメントを入力し、時間間隔を選択して、照会をクリックします。
2. グラフタブの  (ストリームグラフ) をクリックします。
3. グラフのプロパティを設定します。

例

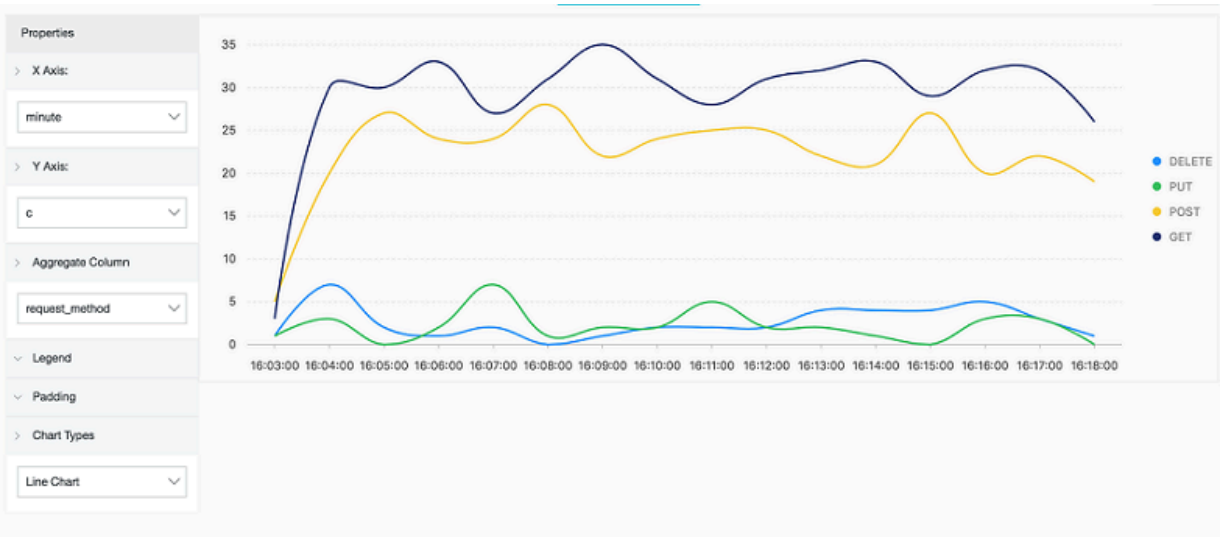
曲線グラフは、3 次元の関係を表現する場合に適用します。(例: time-type-value)

```
* |  select   date_forma  t ( from_unixt  ime ( __time__  -  __time__
 %  60 ), '% H :% i :% S ')  as   minute ,  count ( 1 )  as   c ,
```

```
request_me thod group by minute , request_me thod order
by minute asc limit 100000
```

X 軸に `minute` 、Y 軸に `c` 、集計列に `request_me thod` を指定します。

図 7-16 : 曲線グラフ



7.1.10 サンキーダイアグラム

サンキーダイアグラム (Sankey diagram) は、あるデータセットから、別のデータセットへのトラフィック量を表現することができる、特殊なフローチャートです。例えば、 `source` 、 `target` および `value` の 3 つの値で構成されるネットワークトラフィックの量を表すのに適しています。各ノードを `source` および `target` 、また、ノード間の関係 (`value`) を線で表現します。

特徴

サンキーダイアグラムの特徴は、以下のとおりです。

- 始点と終点のトラフィック量は等しく、また、主線の合計幅と副線の合計幅は等しく保たれます。
- 線で各トラフィックは表現され、線の太さで各トラフィックの量を示します。
- ノードの幅は、特定の状況下におけるトラフィックの量を示します。

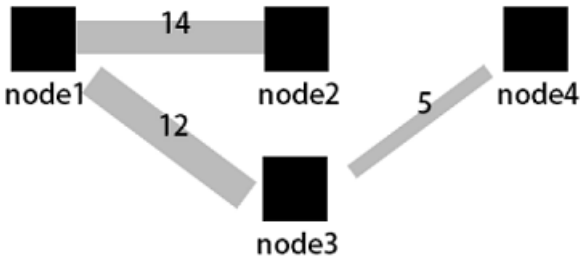
たとえば、下表のデータをサンキーダイアグラムに表示することができます。

Source	Target	Value
node1	node2	14
node1	node3	12

Source	Target	Value
node3	node4	5
…	..	…

上記のデータの関係は、下図のように表現されます。

図 7-17 : サンキーダイアグラムデータ関係



基本コンポーネント

- ノード
- 線

設定項目

設定項目	説明
開始列	開始ノード
終了列	終了ノード
数値列	開始ノードと終了ノードを結ぶ線の値
パディング	座標軸とグラフ境界線の間の間隔

手順

1. 照会/分析ページのクエリ欄にクエリステートメントを入力し、期間を選択して、照会/分析をクリックします。
2. グラフタブの (サンキーダイアグラム) をクリックします。
3. グラフのプロパティを設定します。

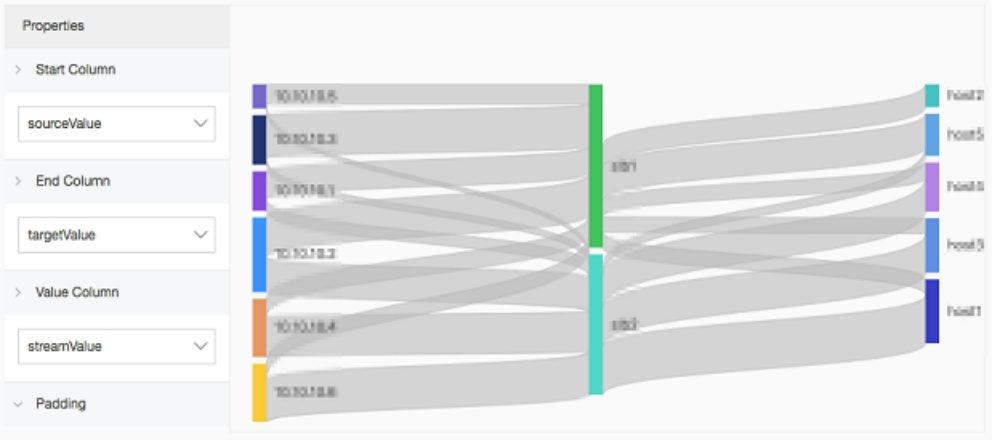
例

通常のサンキーダイアグラム

ログに `source` 、 `target` 、 `value` の各フィールドが含まれる場合、各ログはノードとエッジの関係を示します。したがって、入れ子サブクエリ を使用して `streamValu e` の合計値を取得することができます。

```
* | select sourceValu e , targetValu e , sum ( streamValu e ) as streamValu e from ( select sourceValu e , targetValu e , streamValu e , __time__ from log group by sourceValu e , targetValu e , streamValu e , __time__ order by __time__ desc ) group by sourceValu e , targetValu e
```

図 7-18 : 通常のサンキーダイアグラム



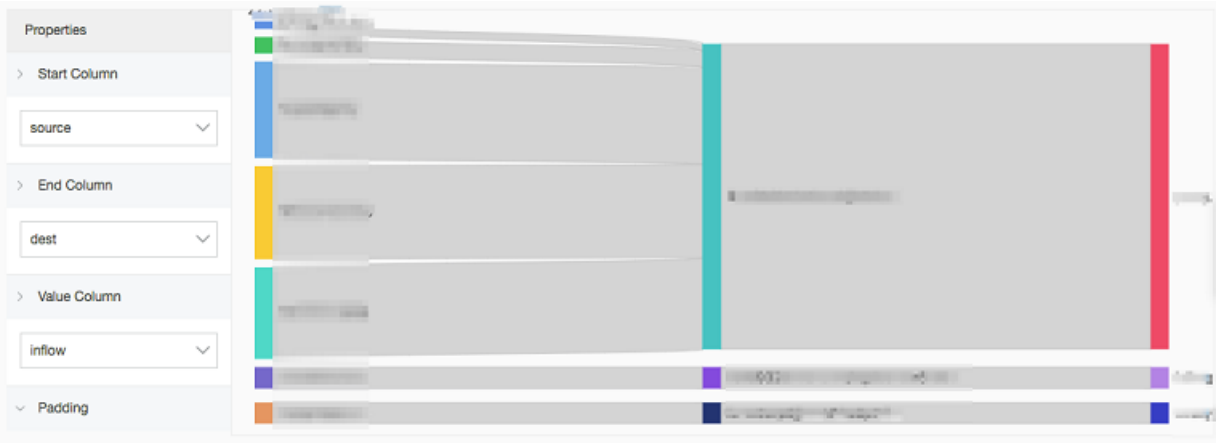
レイヤー 7 Server Load Balancer のアクセスログのシナリオ

Log Service では、レイヤー 7 Server Load Balancer のアクセスログを収集することができます。そのアクセスログのサンキーダイアグラムを作成します。

```
* | select COALESCE ( client_ip , slbid , host ) as source , COALESCE ( host , slbid , client_ip ) as dest , sum ( request_le
```

```
ngth ) as inflow group by grouping sets ( ( client_ip ,
slbid ), ( slbid , host ))
```

図 7-19 : 入れ子サブクエリ



7.1.11 Word Cloud

Word Cloud は、テキストデータを可視化します。さまざまなキーワードを雲状のカラフルなグラフに表示し、テキストデータを可視化できるようにします。フォントサイズまたは色でキーワードの重要度が分けられます。キーワードの重要度が直感的に判別できます。

基本コンポーネント

Word Cloud は、キーワードを計算、分類して表示します。

設定項目

設定項目	説明
ワードの列	表示する単語
値列	単語の値
フォントサイズ	キャンバスに適用するフォントサイズの範囲 ・ 最大フォントサイズ (50–80 px) ・ 最小フォントサイズ (12–24 px)
間隔	座標軸およびグラフ境界線の間隔

手順

1. 照会/分析ページのクエリ欄にクエリステートメントを入力し、期間を選択して、照会/分析をクリックします。
2. グラフタブの (未定義の Word Cloud) をクリックします。

3. グラフのプロパティを設定します。

例

Nginx のログ内の hostname 分布を分析

```
* | select hostname , count ( 1 ) as count group by
 hostname order by count desc limit 1000
```

ワード列に `hostname` 、値列に `count` を選択します。

7.1.12 ツリーマップ

ツリーマップは、長方形ブロックでツリー構造のデータを表示するグラフです。長方形ブロックの面積が大きいほど、比率が高いことを示します。

コンポーネント

長方形ブロック (データの集計結果)

設定項目

設定項目	説明
カテゴリー	データカテゴリーフィールド
データ列	データ値フィールド、データ型: 数値 (データ値が大きいほど、長方形ブロックの面積も大きい)
間隔	長方形ブロック間の間隔 (指定可能な値: 0 ～ 100 px)

手順

1. 照会/分析ページのクエリ欄にクエリステートメントを入力し、期間を選択して、照会/分析をクリックします。
2. グラフタブの  (数値グラフ) をクリックします。
3. グラフのプロパティを設定します。

例

Nginx ログ内のホスト名の比率を分析:

```
* | select hostname , count ( 1 ) as count group by
 hostname order by count desc limit 1000
```

分類 ドロップダウンリストより `hostname` を選択します。また、値列ドロップダウンリストより `count` を選択します。

7.2 ダッシュボード

7.2.1 ダッシュボード

Log Service には、リアルタイムにデータを分析できるダッシュボードがあります。頻繁に実行するクエリ/分析ステートメントのグラフをダッシュボードに保存することができます。ダッシュボードにより、複数の分析グラフを 1 ページに表示させることができます。ダッシュボードを開く際、また、更新する際、ダッシュボードの各グラフには、クエリ/分析ステートメントの実行結果が表示されます。

なお、Log Service にはコンソールに埋め込んで共有機能もあります。Log Service コンソールに表示されるダッシュボードを他の Web サイトのページに埋め込むことができます。データ分析および表示方法の選択肢が増えます。また、ダッシュボードに追加したグラフにドリルダウン分析を設定し、グラフをクリックして、データの詳細を掘り下げた分析結果を得ることができます。

制限

- 各プロジェクトに最大 50 のダッシュボードを作成可能
- 各ダッシュボードに最大 50 の分析グラフを作成可能

ダッシュボードを無料で試す

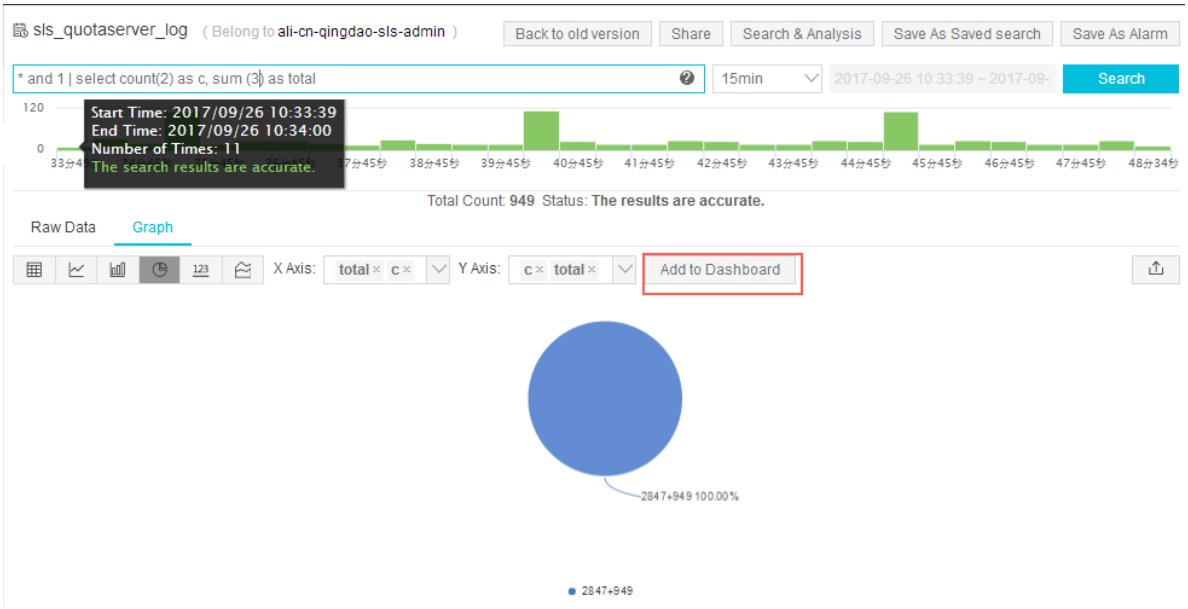
無料で試すには、こちらをクリックします。

アカウント: sls-reader1

パスワード: pnX-32m-MHH-xbm

ダッシュボードの作成

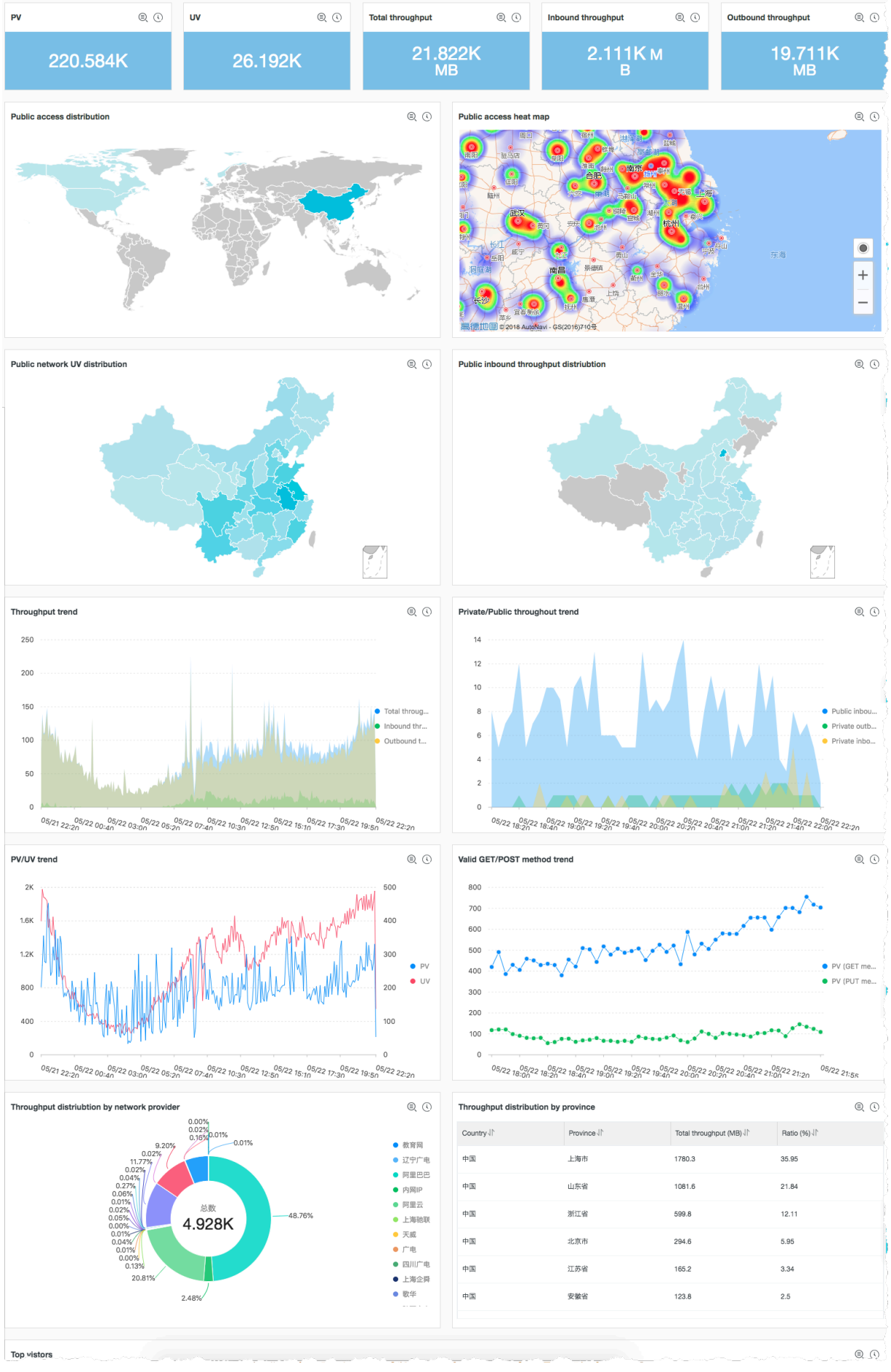
1. 登#日志服#控制台，##Project名称。
2. Logstore リストページで、照会/分析列の照会をクリックします。
3. クエリ欄にクエリ/分析ステートメントを入力し、照会をクリックします。
4. グラフタブページで、グラフを構成します。
5. ダッシュボードに追加をクリックします。
6. ダッシュボードのパラメータを設定します。



7. OK をクリックして構成を完了します。

   複数の分析グラフをダッシュボードに追加する場合は、上記手順を繰り返します。

下図は、ダッシュボードに複数の分析グラフを表示させている例です。

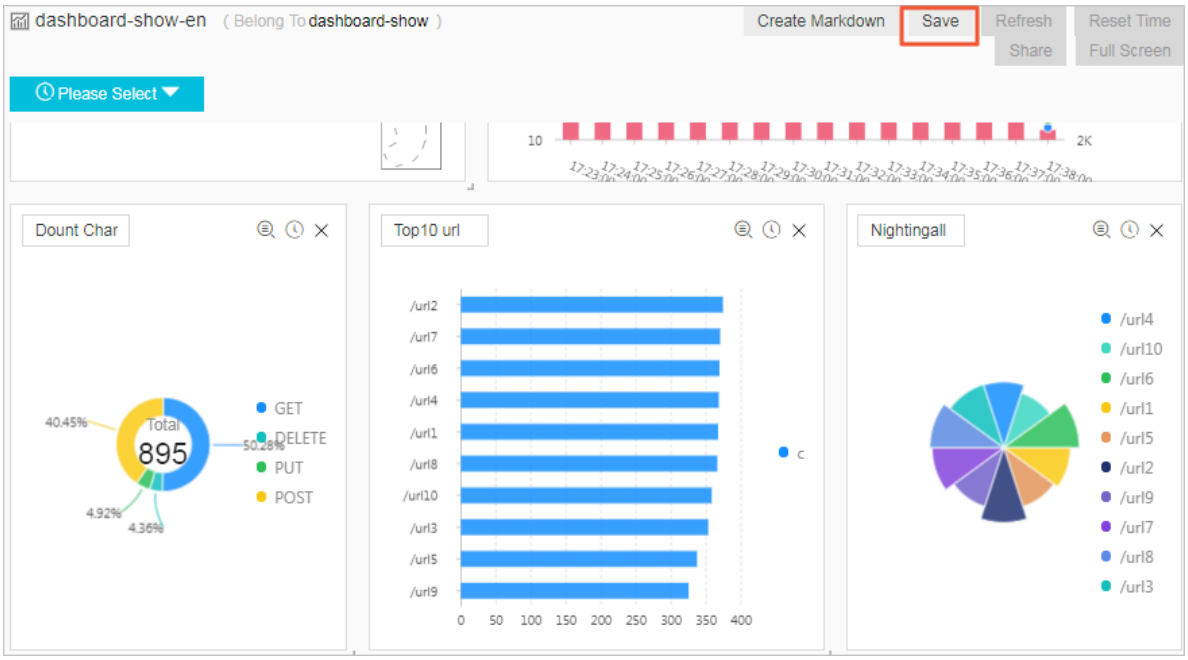

PV
220.584K
UV
26.192K
Total throughput
21.822K MB
Inbound throughput
2.111K MB
Outbound throughput
19.711K MB
Public access distribution
Public access heat map
Public network UV distribution
Public inbound throughput distriubtion
Throughput trend
Total throug...
Inbound thr...
Outbound t...
Private/Public throughout trend
Public inbou...
Private outb...
Private inbo...
PV/UV trend
PV
UV
Valid GET/POST method trend
PV (GET me...
PV (PUT me...
Throughput distriubtion by network provider
总数
4.928K
48.76%
20.81%
2.48%
教育网
辽宁广电
阿里巴巴
内网IP
阿里云
上海驰联
天威
广电
四川广电
上海企鹅
歌华
Throughput distribution by province
Country
Province
Total throughput (MB)
Ratio (%)
中国 上海市 1780.3 35.95
中国 山东省 1081.6 21.84
中国 浙江省 599.8 12.11
中国 北京市 294.6 5.95
中国 江苏省 165.2 3.34
中国 安徽省 123.8 2.5
Top visitors

ダッシュボードの編集

1. Logstore リスト ページで、照会/分析列の照会をクリックします。
2. 左側のメニューより、ダッシュボード名をクリックします。
3. 右上隅の編集をクリックします。

   ダッシュボードの編集ビューでは、以下を実行できます。

   - 期間、サイズ、各分析グラフのタイトルの変更
   - マークダウングラフの作成
   - 分析グラフの削除
4. 右上隅の保存をクリックします。



ダッシュボードの表示

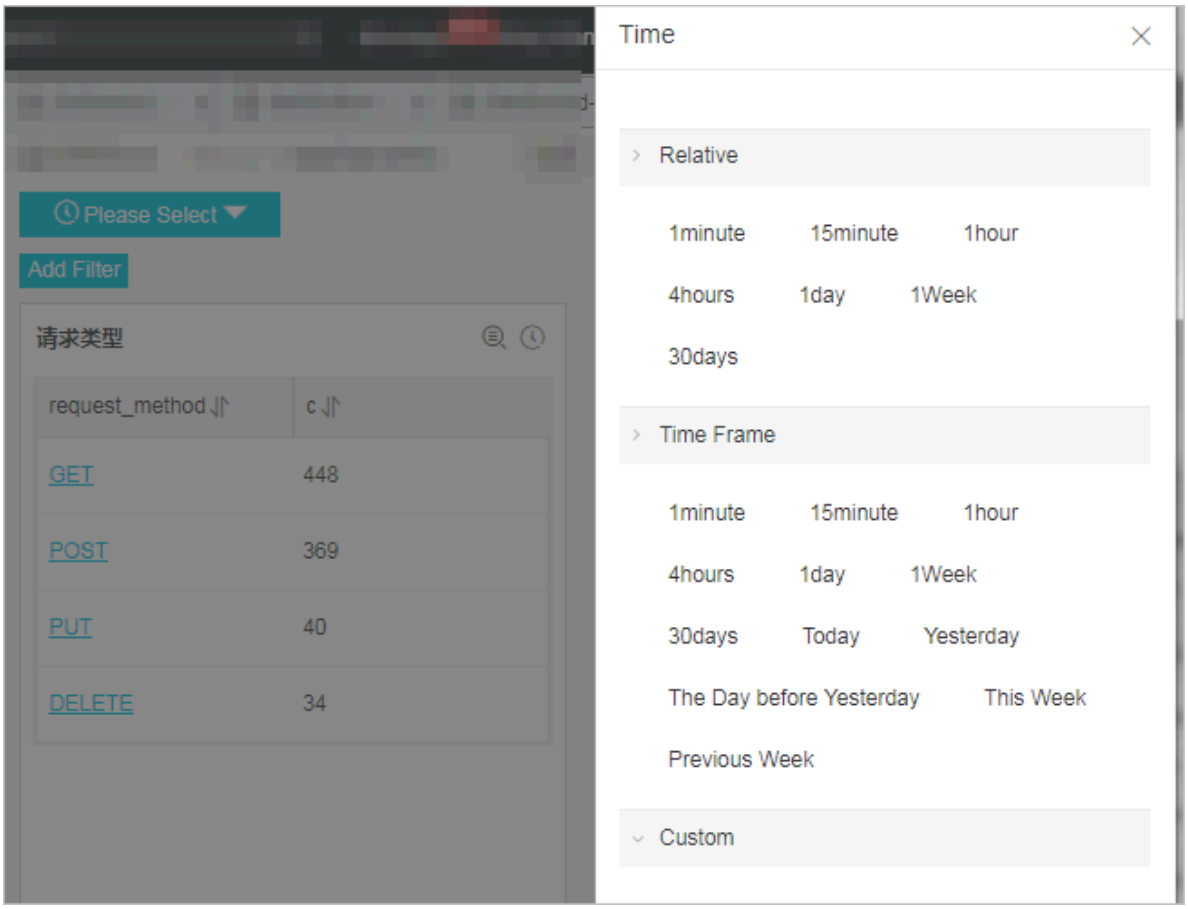
照会ページの左側のナビゲーションメニューより、ダッシュボード名をクリックしてダッシュボードを開きます。

- ダッシュボードのグローバル期間を設定

  1. ダッシュボードページの左上の選択してくださいをクリックします。
  2. 表示されるページで、期間を選択します。期間は、相対、時間枠、またはカスタムのいずれかを設定します。
  3. カスタム設定の場合は、期間を指定して OK をクリックします。

  > 注：
  >
  > - 期間を変更すると、ダッシュボード上のすべてのグラフの期間が変更されます。

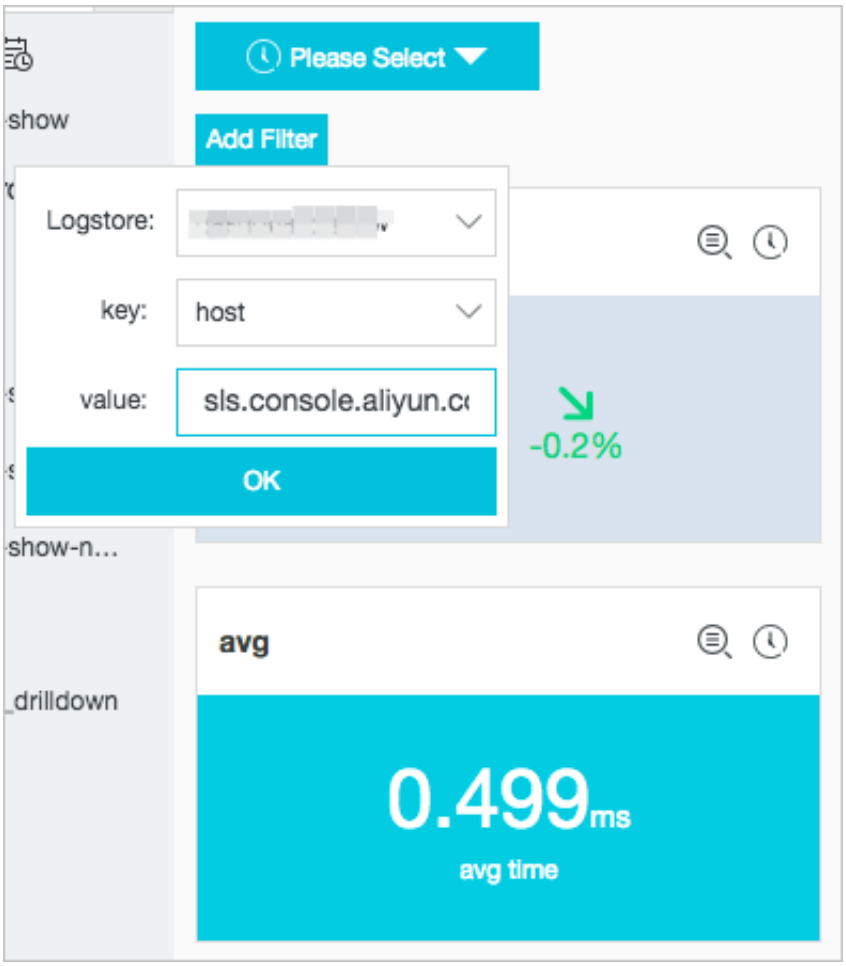
- 右上隅の期間をデフォルトにリセットをクリックすると、すべてのグラフの期間は、デフォルトの期間に戻ります。
- 左上隅の期間を指定するボタンより別の期間を選択して、ダッシュボードのグラフの期間を変えることもできます。次回データ分析結果をグラフ表示するときには、デフォルトの期間で表示されます。



- フィルタリング条件を追加して、ダッシュボードに表示する分析結果をフィルタリング

  1. ダッシュボードページの左上隅にあるフィルタの追加をクリックします。
  2. Logstore、キー、および値を設定します。

     たとえば、「oss」という名の Logstore にフィルタリング条件を追加して、「host」フィールドの値が「sls.console.aliyun.com」であるアクセスログを抽出することできます。



- ダッシュボードの自動更新を設定

  1. ダッシュボードページの右上隅にある自動更新をクリックします。
  2. ダッシュボードページの自動更新間隔を設定します。

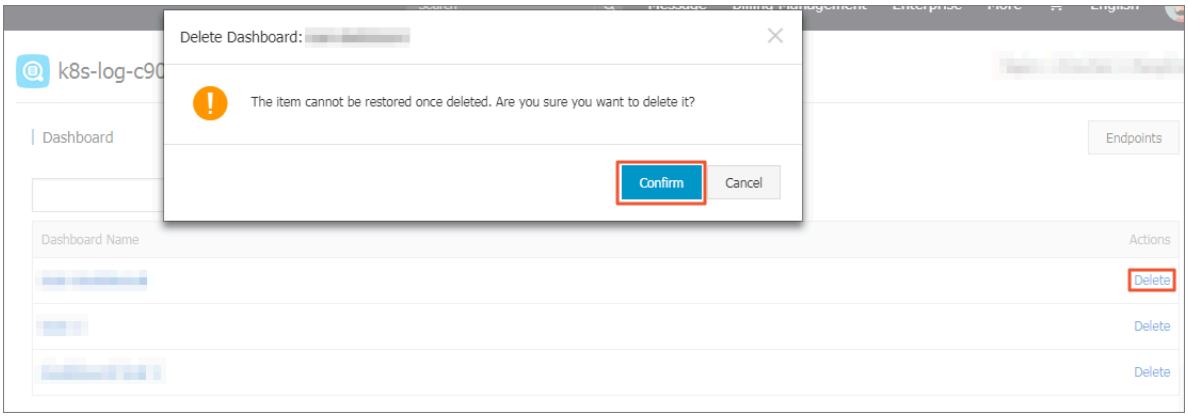
     ダッシュボードは、設定した間隔で自動更新されます。

ダッシュボードの削除

不要になったダッシュボードは、削除します。削除したダッシュボードは元に戻すことはできません。

1. Logstores ページの左側のナビゲーションペインより、ダッシュボードをクリックします。

2. 削除するダッシュボードの削除をクリックします。
3. 表示されるダイアログボックスの確認をクリックします。



7.2.2 ダッシュボードのディスプレイパラメータの設定

本ドキュメントでは、Log Service でダッシュボードのディスプレイパラメータを設定する方法について説明します。 ダッシュボードではデフォルトでディスプレイモードがオンになっています。

対象のダッシュボードを開くには、次の 2 つの操作のいずれかを実行します。

- ログストアページの左側のナビゲーションペインで、ダッシュボードをクリックしてから、対象ダッシュボードの名前をクリックします。
- 検索と分析ページ、保存済みの検索ページ、または Log Service コンソール内の他のページの左側にある折りたたみナビゲーションペインで、カーソルをペインの上に移動して項目を表示し、対象のダッシュボード名をクリックします。

利用可能な設定

- ダッシュボードのディスプレイの設定

  ダッシュボードのディスプレイモードが選択されている場合、ダッシュボードの右上隅（左から右）に次の使用可能な機能ボタンがあります：選択してください、編集、サブスクライブ、アラート、更新、共有、全画面表示、タイトル設定、及びデフォルト時間をリセット。

- チャートのディスプレイの設定

  ダッシュボードでディスプレイモードが選択されている場合、チャートの右上隅にある折りたたみ機能リストには、チャート分析結果に関連するパラメータを設定するための機能があります。

  > 注：
  > 異なるチャートには異なる機能リストがあります。

ダッシュボードのクエリ時間範囲の設定

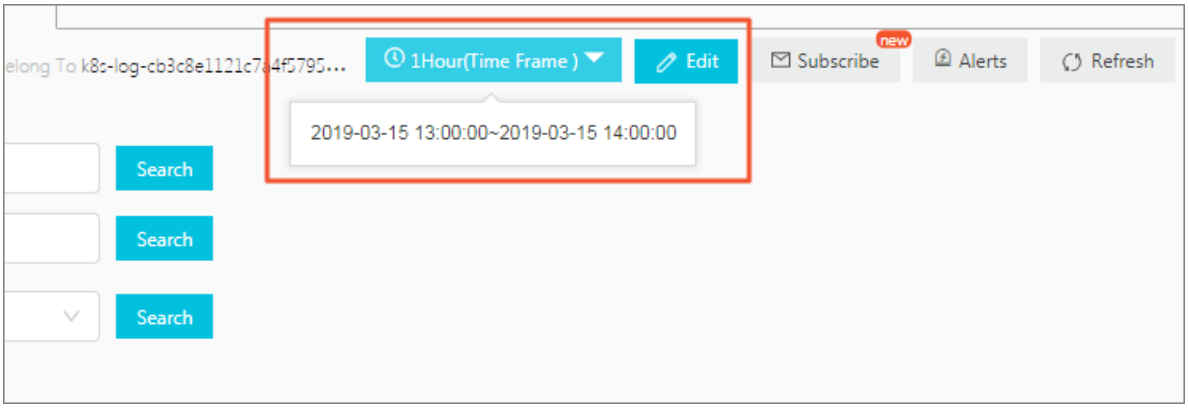
ダッシュボード内のすべてのチャートは、そのダッシュボードに設定されているクエリ時間範囲を使用します。 単一チャート専用のクエリ時間範囲を設定するには、チャートのクエリ時間範囲の設定 をご参照ください。

> 注：
> ダッシュボードのカスタムクエリ時間範囲は一時的な設定で、システムには保存されません。つまり、ダッシュボードを開きなおしてチャートを表示すると、デフォルトのクエリ時間範囲のクエリと分析結果が表示されます。

1. 選択してください をクリックします。
2. 時間範囲を選択します。

   ダッシュボードで利用可能なクエリ時間範囲のタイプは次のとおりです。

   - 相対：1 分、5 分、15 分、または現在の時点から始まるその他の時間長の正確な時間範囲（秒までの正確さ）でログをクエリすることができます。 たとえば、現在の時刻が 19:20:31 で、このパラメータを 1 時間に設定した場合、18:30:31 から 19:20:31 までのチャートが生成されます。
   - タイムフレーム：1 分、5 分、15 分、または現在の時点から始まるその他の期間で、ログを全時間範囲でクエリすることができます。 たとえば、現在の時刻が 19:20:31 で、このパラメータを 1 時間に設定した場合、18:00:00 から 19:00:00 までのチャートが生成されます。
   - カスタム：カスタマイズされた時間範囲でログをクエリすることができます。
3. カーソルを 選択してください に移動して、設定した時間範囲が有効になっていることを確認します。



編集モードへの切り替え

編集モードに入るには、編集をクリックします。 詳細は、ダッシュボードの編集 をご参照ください。

アラート通知の設定

アラート通知を作成または変更するには、右上隅のアラート > 作成を選択するか、アラート > 変更を選択します。 アラートは少なくとも 1 つのチャートに関連付ける必要があります。

詳細は、アラートの設定 をご参照ください。

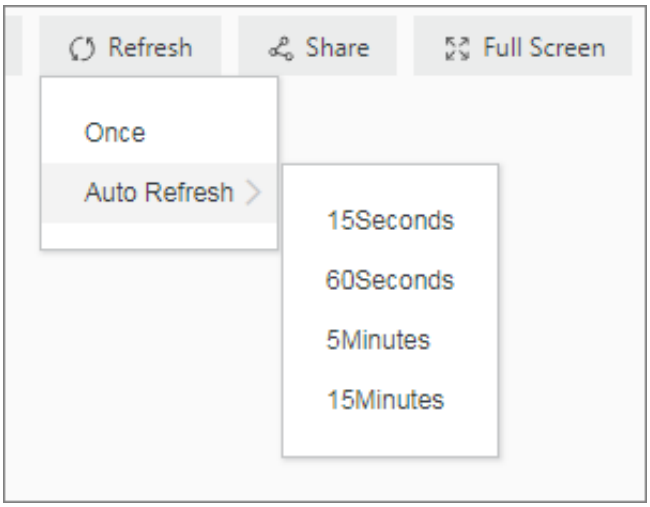
ページの更新頻度の設定

ダッシュボードを手動で更新することも、ダッシュボードが自動的に更新される間隔を設定することもできます。

- ダッシュボードを手動で更新するには、 更新 > 1 回のみを選択します。
- 指定した時間間隔でダッシュボードが自動的に更新されるように設定するには、更新 > 自動更新を選択します。 次に間隔を選択します。

  ダッシュボードは、15 秒間隔、60 秒間隔、5 分間隔、または 15 分間隔で自動的に更新されます。

> 注：
> ページがアクティブになっていない場合は、自動更新が機能されない可能性があります。



ダッシュボードの共有

ダッシュボードを他のユーザーと共有するには、共有をクリックすることでダッシュボードのリンクをコピーし、ダッシュボードを表示する権限を持つユーザーに送信します。 現行ダッシュ

ボードの設定は、共有ダッシュボードページに保存されます（クエリの時間範囲やチャートタイトルの表示スタイルなど）。

> 注：
> ダッシュボードを共有する前に、ダッシュボードを表示する権限を対象のユーザーに付与する必要があります。

ダッシュボードの全画面表示

全画面表示 をクリックします。 データの展示やプレゼンテーション場合は、この操作を実行することをお勧めします。

チャートのタイトル形式の設定

タイトル設定をクリックし、次のいずれかの形式を選択します：

- タイトルと時間の並び表示
- タイトルと時間のスクロール表示
- タイトルと時間の交代表示
- タイトルのみ
- 時間のみ

デフォルト時間をリセット

ダッシュボード内のすべてのチャートのデフォルトのクエリ時間範囲を復元するには、デフォルト時間をリセットをクリックします。

チャート表示方法の選択

- チャートの分析詳細の表示

  チャートの分析の詳細（チャートに関連付けられているクエリステートメントやチャートのプロパティなど）を表示するには、チャートの右上隅にあるオプション > 分析詳細の表示をクリックします。
- チャートの時間範囲の設定

  チャートのクエリ時間範囲を設定するには、チャートの右上隅にあるオプション > 時間範囲の選択を選択します。 設定は現行チャートにのみ有効です。
- チャートのアラート通知の設定

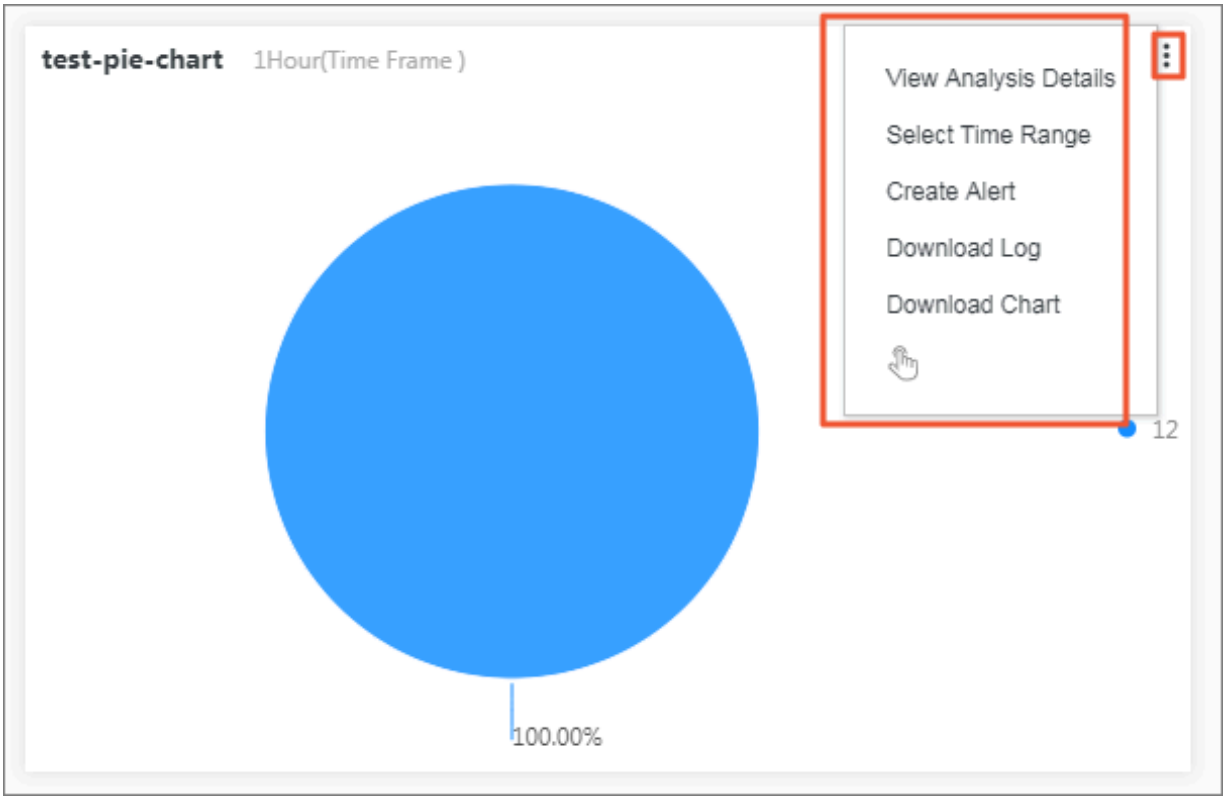
  チャートのアラート通知を設定するには、チャートの右上隅にある オプション > アラートの作成を選択します。 詳細は、アラートの設定 をご参照ください。

- ログのダウンロード

  ログをダウンロードするには、チャートの右上隅にあるオプション > ログのダウンロード を選択します。 生のログ分析結果は、csv ファイルとしてダウンロードされます。
- チャートのダウンロード

  チャートをダウンロードするには、チャートの右上隅にあるオプション > チャートのダウンロードを選択します。 チャートは、psd ファイルとしてダウンロードされます。
- チャートにドリルダウン分析が設定されているかどうかの確認

  チャートにドリルダウン分析が設定されているかどうかを確認するには、チャートの右上隅にあるオプションボタンにカーソルを移動し、折りたたみリストの下部にあるアイコンの色を確認します。 アイコンは赤色の場合、チャートにドリルダウン分析が設定されていることを示します。 アイコンは灰色の場合、チャートにドリルダウン分析が設定されていないことを示します。



7.2.3 ダッシュボードの編集

本ドキュメントでは、ダッシュボードを編集する方法について説明します。

ダッシュボードで次の編集操作を実行できます。

- ダッシュボードパラメータの設定。
  - 左上隅で、現行ダッシュボード名をクリックしてダッシュボード名を変更できます。
  - ダッシュボードにチャート要素を追加します。 たとえば、必要に応じて Markdown チャート、カスタマイズチャート、テキスト、アイコンなどのチャート要素を追加できます。
  - 2 つのチャートを接続線を追加します。 線はチャートの位置に従って自動的に調整されます。
  - フィルターを追加します。 ダッシュボードがディスプレイモードにある場合、フィルタは特定のチャートデータをフィルタリングできます。
  - グリッド線を表示するようにダッシュボードを設定します。
  - メニューバーのツールを使用してチャートのプロパティを設定します。
- チャートを設定します。

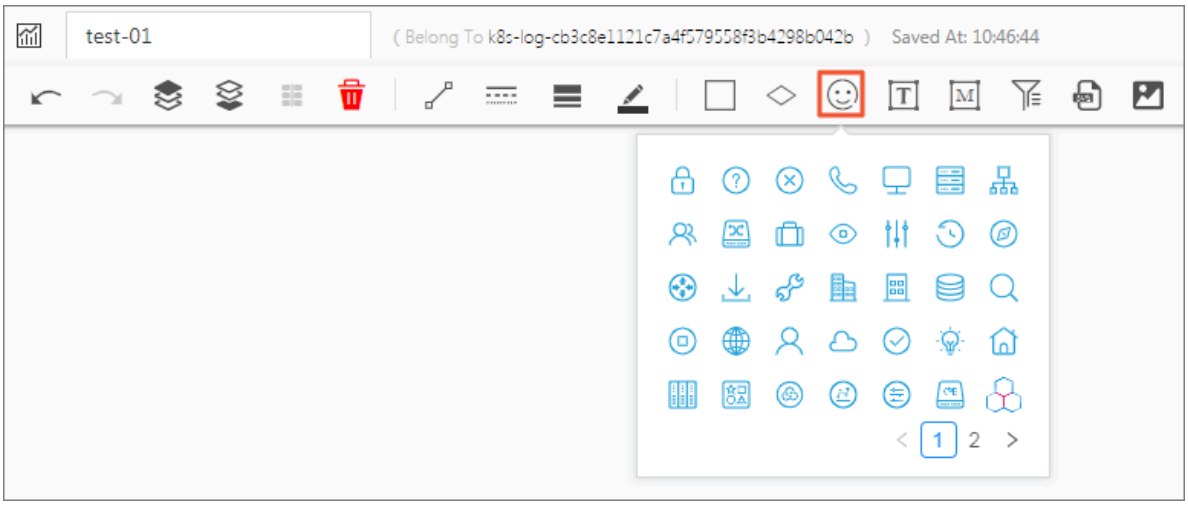
  チャートで表されているクエリステートメントとチャートのプロパティの変更や、ドリルダウン分析の設定ができます。

> 注：
> 編集モードでダッシュボードを変更したら、右上隅にある保存をクリックする必要があります。そうしないと、変更内容が反映されません。

ダッシュボードへのチャート要素の追加

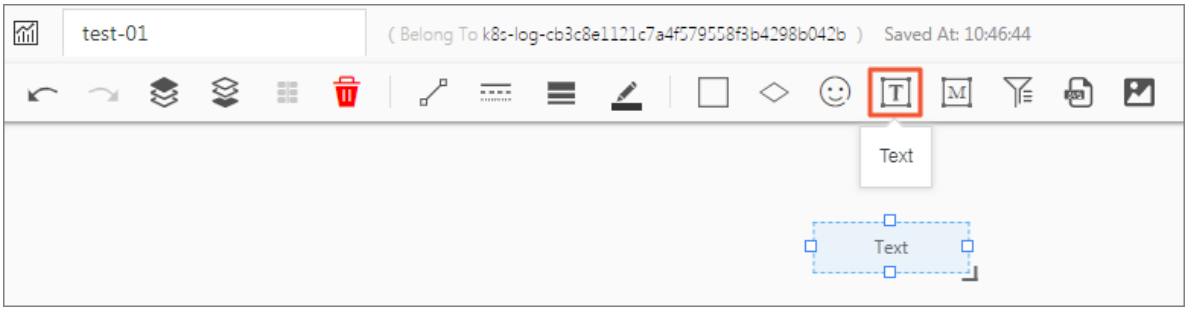
- アイコン

  ダッシュボードにアイコンを追加するには、アイコンツールをクリックし、対象のアイコンを選択して、それをダッシュボードの任意の場所にドラッグします。



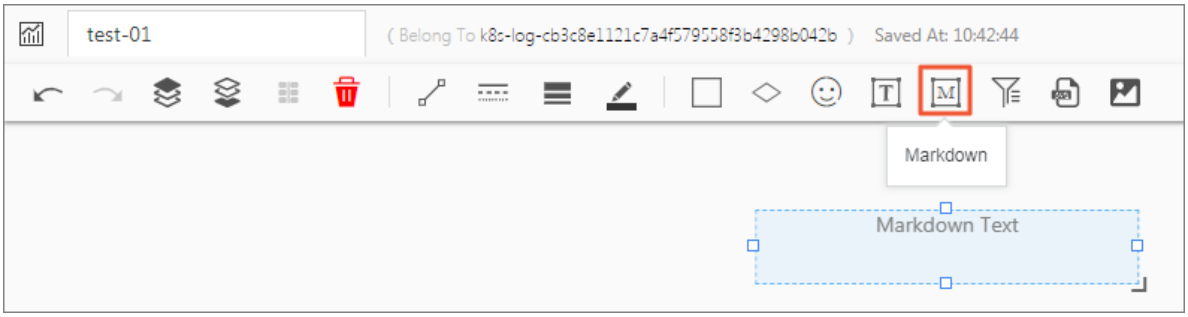
- テキスト

  ダッシュボードにテキストを追加するには、テキストアイコンをダッシュボードの目的の位置にドラッグします。 次に、テキストボックスをダブルクリックしてテキストを追加します。



- Markdown チャート

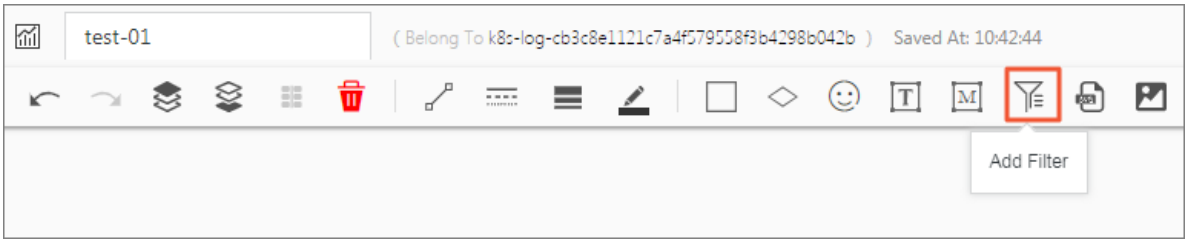
  ダッシュボードに Markdown チャートを追加するには、まずマークダウンツールアイコンをダッシュボードの目的の位置にドラッグします。 Markdown ボックスを編集するには、カーソルを Markdown ボックスの右上隅に移動し、 オプションアイコンをクリックして、編集をクリックします。



- フィルタ

  フィルタを使用して、クエリ範囲を絞り込んだり、ダッシュボードの変数を置き換えることができます。

  ダッシュボードにフィルタを追加するには、フィルタツールアイコンをクリックしてから、表示されたページでフィルタを設定します。 デフォルトでは、ダッシュボードの左上隅にフィルタが追加されています。 フィルタ設定を変更するには、カーソルをフィルタボックスの右上隅に移動し、オプションアイコンをクリックしてから編集をクリックします。



- SVG ファイル

  SVG ファイルをダッシュボードに追加するには、SVG ツールアイコンをクリックしてから、表示されたダイアログボックスの灰色のエリアをクリックしてローカルディレクトリから SVG ファイルを選択するか、表示されたダイアログボックスで SVG ファイルを灰色のエリアにドラッグアンドドロップします。

  > 注：
  > SVG ファイルの最大サイズは 10 kb です。

- Web からの画像

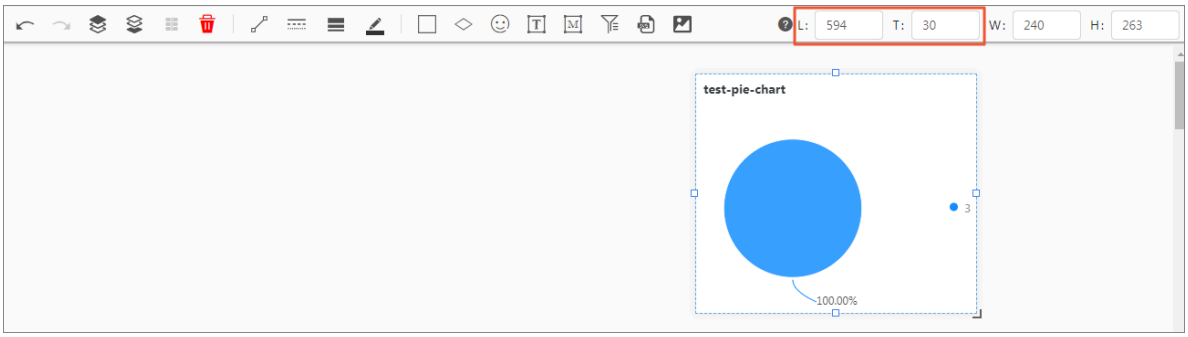
  Web サイトからダッシュボードに画像を追加するには、[画像]アイコンをクリックし、表示されたダイアログボックスに画像の URL を入力するか貼り付けて、OK をクリックします。

ダッシュボードのレイアウトの設定

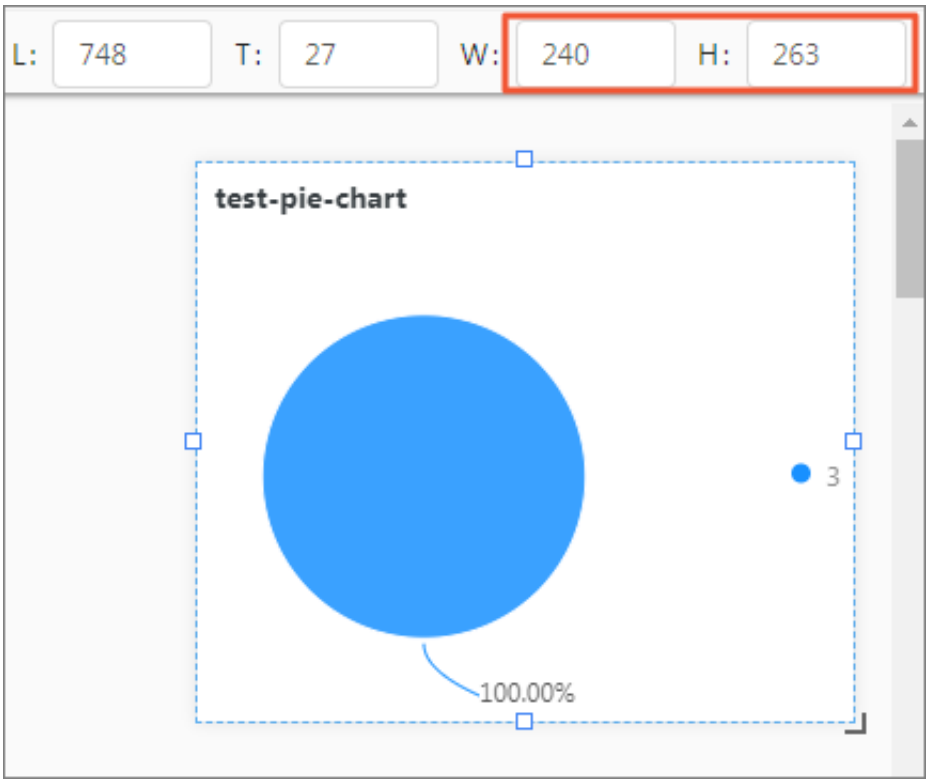
編集モードでは、ダッシュボード内のすべてのチャートと要素を任意の位置にドラッグしたり、縦横比を維持しながらサイズを変更したりできます（チャートと要素の接続に使用される線を除く）。 水平ディスプレイ上のダッシュボードの制限は 1000 ピクセルです（垂直ディスプレイにはピクセル制限はありません）。 チャートの位置とチャート間の間隔を正確に設定するには、ダッシュボードのレイアウトを設定する前に、右上隅にあるグリッド線の表示をクリックすることをお勧めします。

次の操作を実行することもできます:

- チャート位置の調整
  - 対象のチャートを直接にドラッグします。
  - 対象のチャートを選択してから、ツールバーに L 値と T 値を設定します。



- チャートの幅と高さの調整
  - 対象のチャートを選択し、グラフの右下隅をドラッグしてチャートのサイズを変更します。
  - 対象のチャートを選択してから、ツールバーに L 値と T 値を設定します。

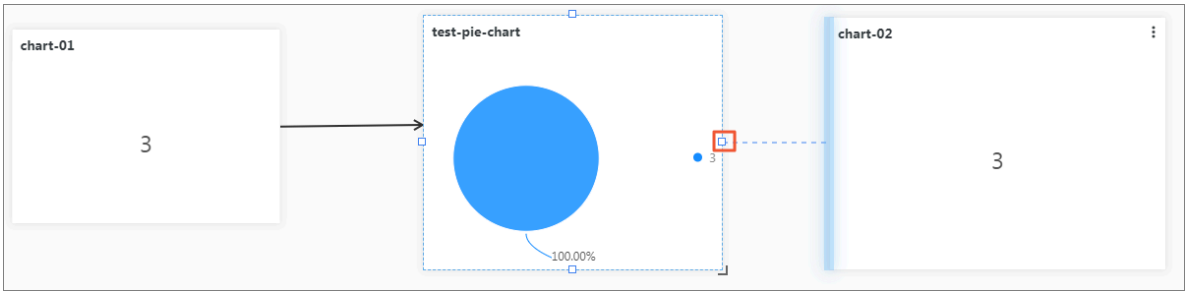


- 2 本のチャートを接続する線の追加

  線を使用してソースチャートと対象のチャートを接続するには、次の手順に従います。

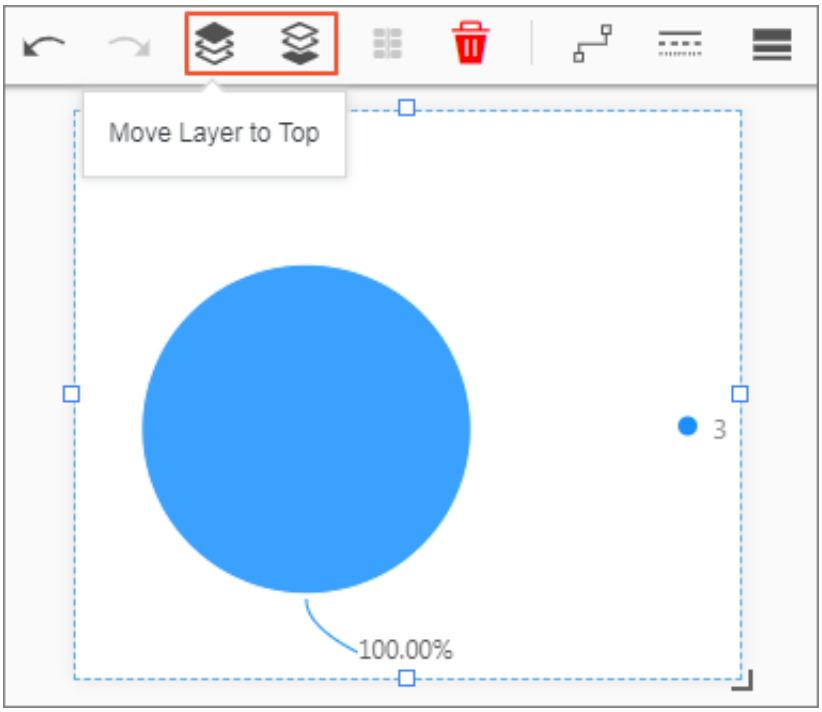
  1. ソースチャートを選択します。
  2. ソースチャートのアウトラインのサイズ変更ハンドルを対象のチャート側にドラッグします。
  3. 対象チャートのターゲット側が青になったら、カーソルを離します。

  その後、接続されているチャートの位置とサイズを調整すると、線はそれに合わせて自動的に調整します。



- チャートレイヤの設定

  チャートレイヤを設定するには、対象のチャートを選択してから、ツールバーのレイヤを上に移動アイコンまたはレイヤを下に移動アイコンをクリックします。



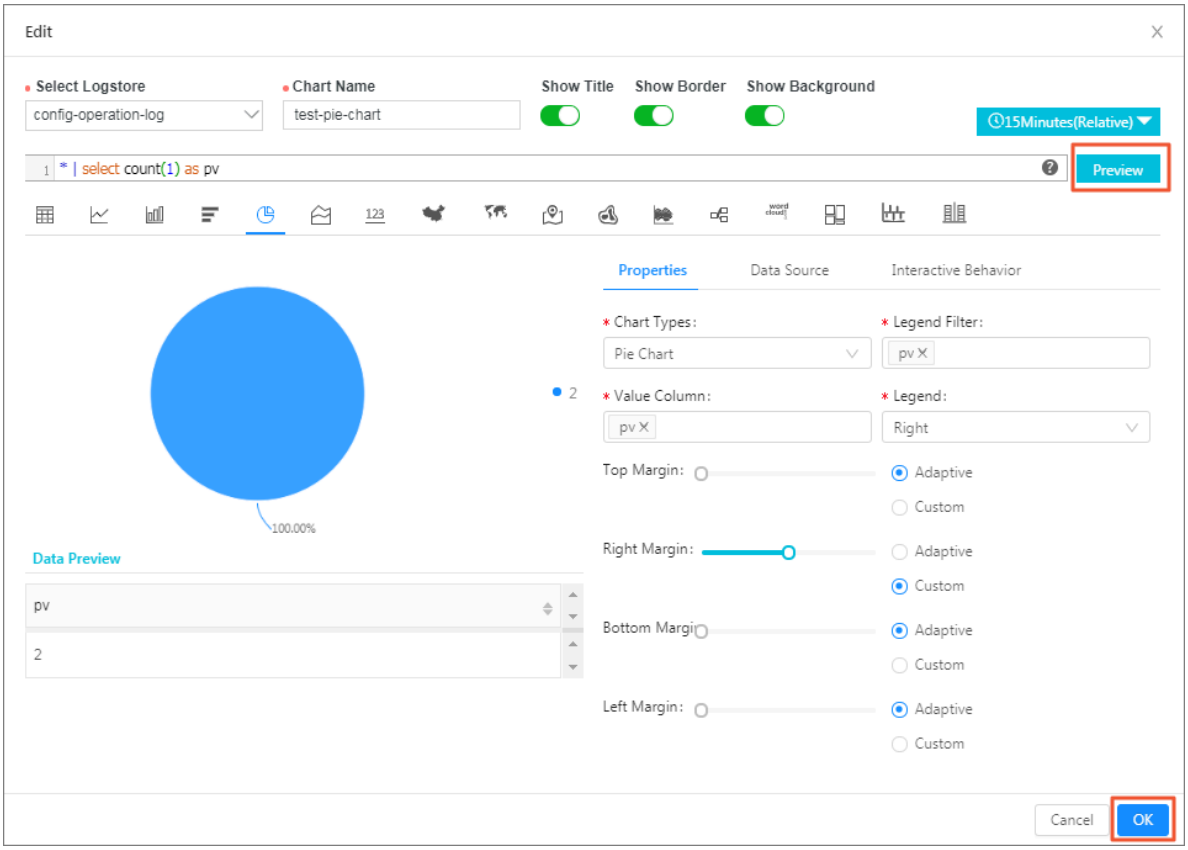
チャートの変更

編集モードでは、以下の操作を実行してダッシュボードのチャートを変更できます。

・ チャートを編集します。

チャートで表されるクエリステートメントとチャートのプロパティの変更、ドリルダウン分析の設定、及びその他の操作を実行できます。

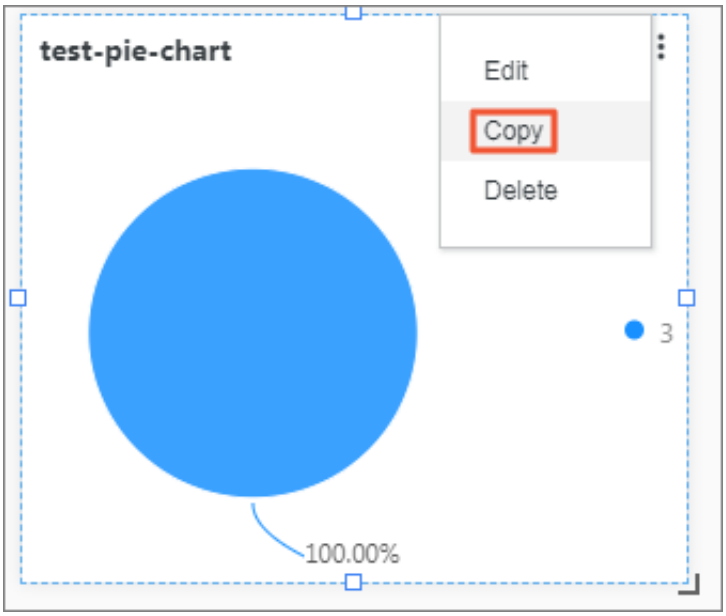
1. ダッシュボードページの右上隅にある編集をクリックします。
2. 対象チャートの右上隅にあるオプションアイコンをクリックして、編集をクリックします。
3. 表示されたダイアログボックスで、クエリステートメントを変更するか、プロパティ、データソース、または Interactive Behavior を設定します。
4. プレビュー をクリックして、OK をクリックします。
5. ダッシュボードページの右上隅にある保存をクリックします。



- チャートのコピー

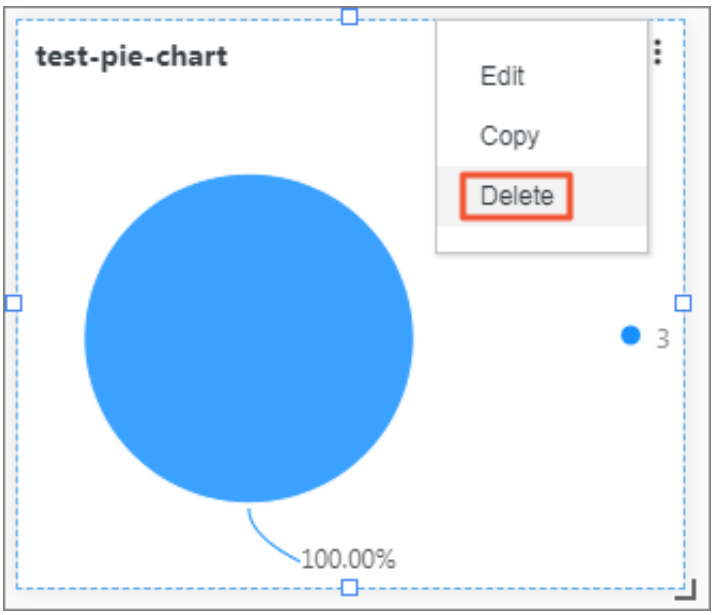
  チャートの現在の設定をすべて保持するためにチャートのコピーを作成できます。

  1. ダッシュボードページの右上隅にある編集をクリックします。
  2. 対象チャートの右上隅にあるオプションアイコンをクリックして、コピーをクリックします。
  3. レプリケートチャートを新しい位置にドラッグします。
  4. ダッシュボードページの右上隅にある保存をクリックします。



- チャートの削除

  1. ダッシュボードページの右上隅にある編集をクリックします。
  2. 対象チャートの右上隅にあるオプションアイコンをクリックして、削除をクリックします。
  3. ダッシュボードページの右上隅にある保存をクリックします。



7.2.4 ダッシュボードスナップショットサービスの購入

本ドキュメントでは、Log Service でダッシュボードスナップショットをサブスクリプション方式で購入する方法を説明します。 具体的には、ダッシュボードのスナップショットに関する通知を特定の E メールアドレスに送信したり、DingTalk チャットボットメッセージとして送信したりするようにシステムを設定できます。

制約

- ダッシュボードごとに最大 1 つのサブスクリプションを作成できます。
- 毎日、Log Service は各アカウントにつき最大 50 通の E メールを送信できます。
- Cron 式で指定された間隔は少なくとも 1 分である必要があります。 ただし、Cron 式を使用する場合は、1 時間以上の間隔を設定することを推奨します。
- Log Service プロジェクトに対して作成できるサブスクリプションとアラート通知の最大数は 100 です。 これらのクォータの一方または両方を増やす必要がある場合は、チケットを起票し、サポートセンターへお問い合わせください。
- 表のデータがページングされて表示されている場合、ダッシュボードを購入すると、表の最初のページのデータのスクリーンショットのみ送信することができます。

注：

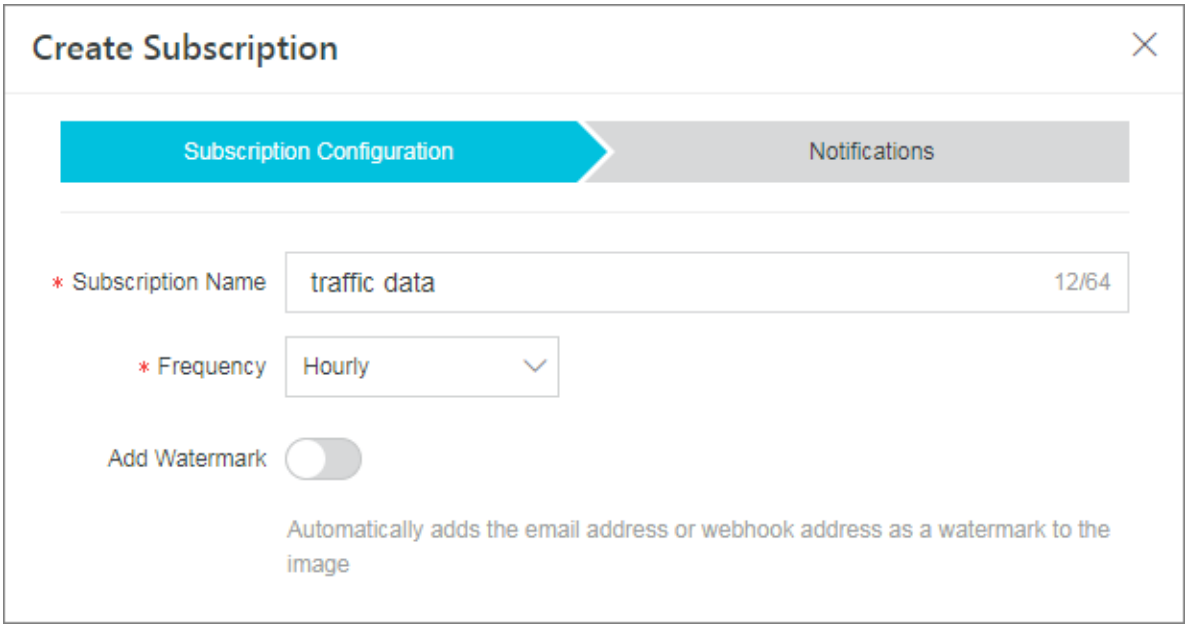
- 表示モードでの時間選択は、ユーザーが異なる期間のチャートデータを動的に確認する際に使用するものです。
- ダッシュボードチャートのデフォルトの時間は、編集モードに進み、チャートの編集ボタンをクリックすることで変更できます。
- 購入するダッシュボードのデータクエリ時間は、ダッシュボード内のチャートのクエリ時間です。

サブスクリプションの作成

1. Log on to the Log Service console, and then click the target project name.
2. 左側のナビゲーションウィンドウで、ダッシュボードをクリックします。
3. 対象のダッシュボードをクリックします。
4. ページ内の サブスクライブをクリックします。
5. サブスクリプションを設定し、次へをクリックします。

設定項目	説明	例:
サブスクリプション名	サブスクリプション名は 1 ~ 64 文字の長さである必要があります。	ダッシュボードサブスクリプション
頻度	ダッシュボードのスナップショット通知の送信頻度。 有効値： ・ 1 時間ごと ・ 毎日 ・ 毎週 ・ 固定間隔（日） ・ Cron Cron の最小間隔は 1 分です。 ただし、1 時間以上の間隔を設定することを推奨します。	`* 0 / 1 * *` `*` Cron 式は、ダッシュボードのスナップショット通知が 00:00 から 1 時間間隔で送信されることを示します。

設定項目	説明	例:
ウォーターマークの追加	ダッシュボードのスナップショットにウォーターマークを追加します。 ウォーターマークの内容は、 E メールアドレスまたは DingTalk チャットボットの WebHook アドレスの access_token です。	-

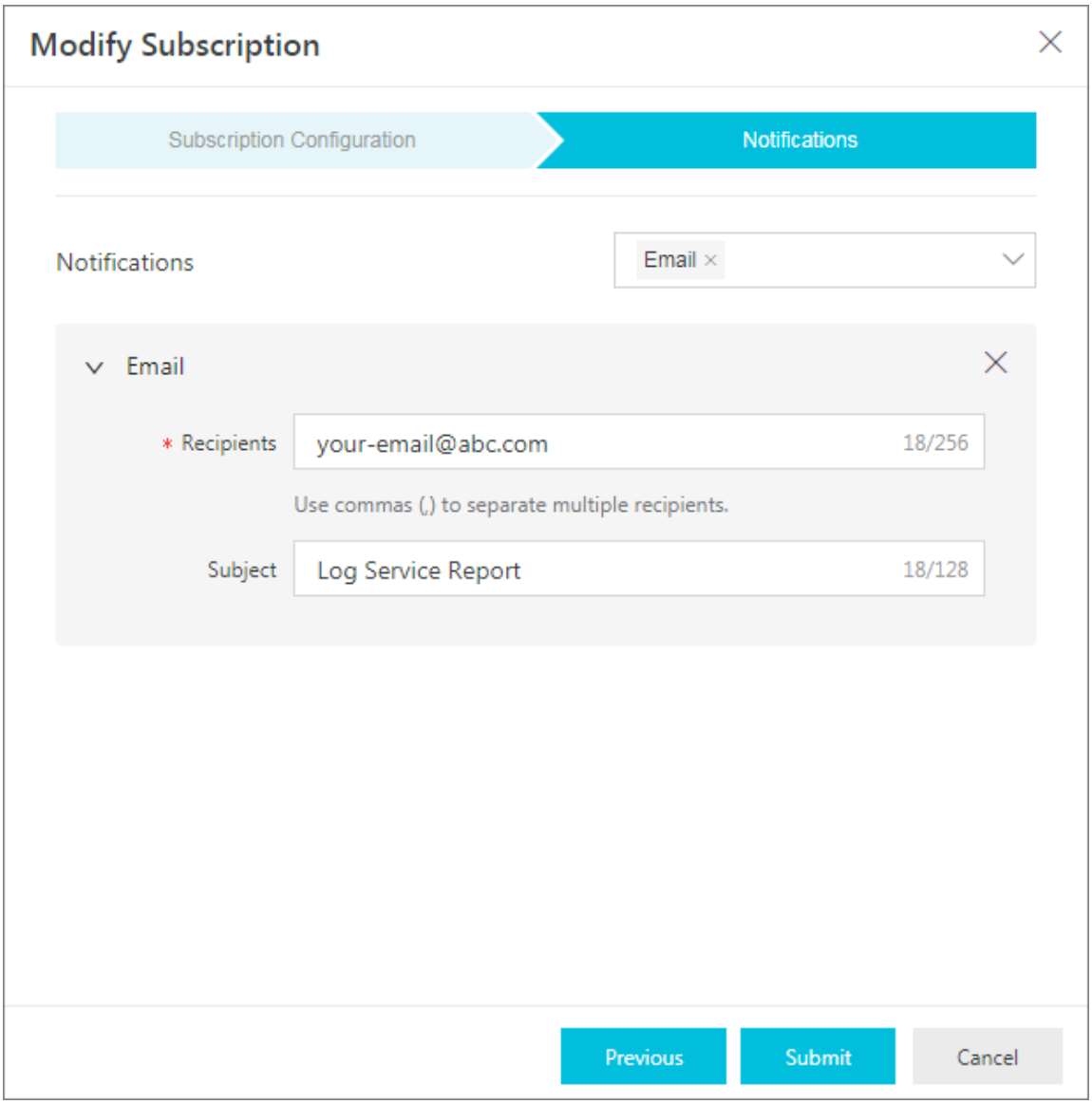


6. 通知を設定します。

ダッシュボードのスナップショット通知は、特定の E メールアドレスへの送信と、DingTalk チャットボットメッセージとしての送信を設定できます。

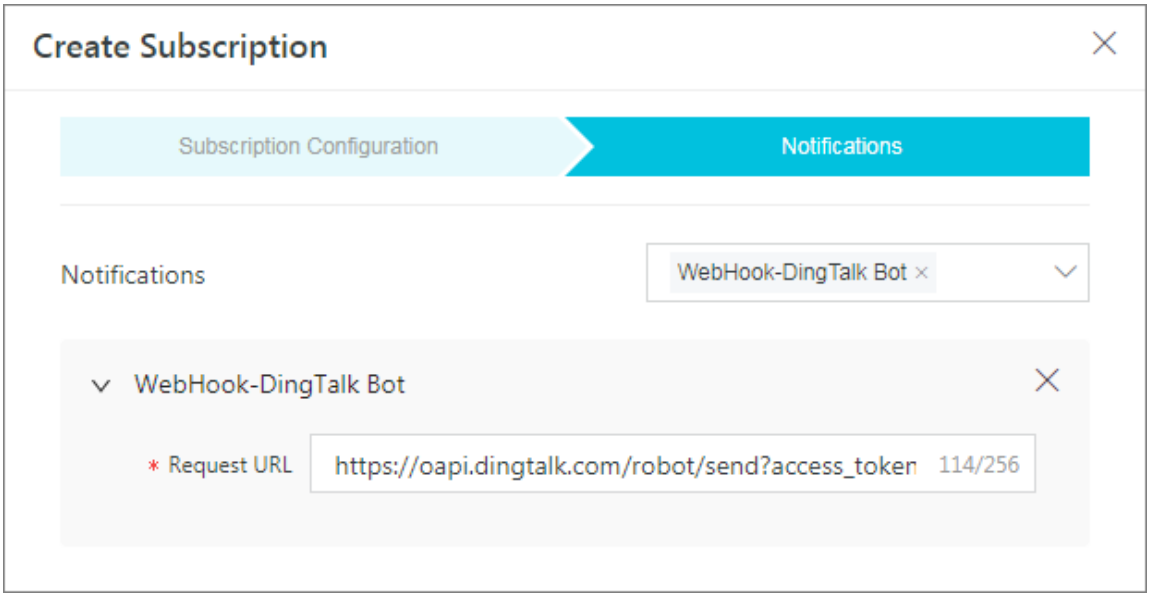
・ E メール

受信者ボックスで、メールアドレスを入力し、 件名を設定します。 メールの件名を設定しない場合、 `Log Service Report` と設定されます。



・ DingTalk チャットボット

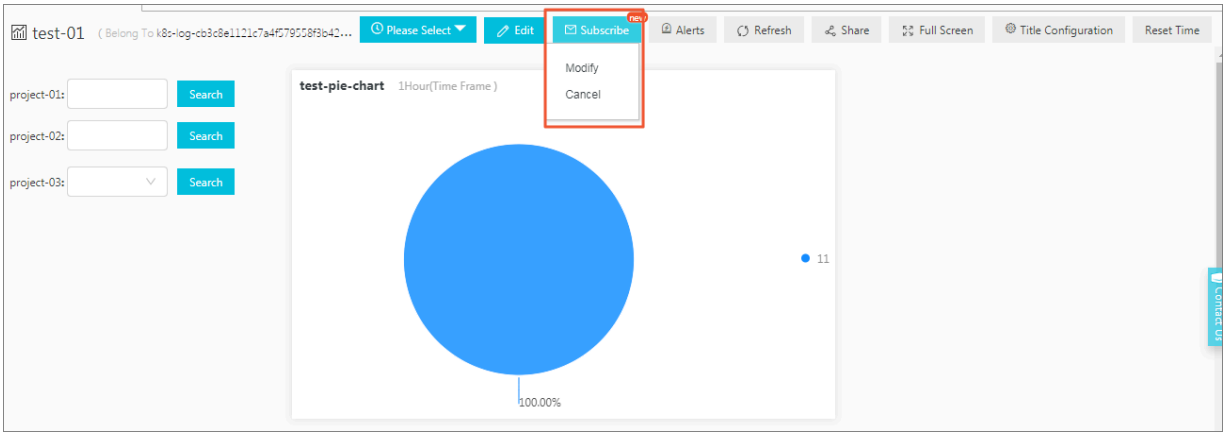
リクエスト URLボックスで、DingTalk チャットボットアドレスを入力します。 詳細は、DingTalk ロボットの設定 をご参照ください。



サブスクリプションの変更とキャンセル

必要に応じて、ページ内のサブスクライブ > 変更、またはサブスクライブ > キャンセルを選択します。

サブスクリプションをキャンセルすると、ダッシュボードスナップショット通知の送信が直ちに停止されます。



7.2.5 ドリルダウン分析

Log Service の分析グラフには、基本的なデータ可視化機能に加えてドリルダウン機能があります。グラフをダッシュボードに追加後、ドリルダウンリストの構成を編集して、より詳細なデータを表示させることができます。

データ分析にドリル機能は欠かせません。データの集計レベルを変更してより詳細な情報を表示することができます。ドリル機能にはロールアップとドリルダウンがあります。ロールアップにより、上位レベルのデータ、より要約された情報を表示させることができます。ドリルダウンにより、掘り下げてより詳細な情報を表示させることができます。データの集計レベルをドリルダ

ウンすることで、より的確なデータ、価値の高いデータに基づいて、より的確な判断を迅速に下すことができます。

また、ダッシュボードの分析グラフをドリルダウン分析できます。ドリルダウンのディメンションとレベルを構成し、ダッシュボードのデータポイントをクリックすると、より詳細なディメンションの分析ページに移動できます。ダッシュボードの分析グラフは、クエリステートメントの実行結果です。リクエストステータス表のドリルダウン分析を構成し、ダッシュボードに追加すると、ダッシュボードの「リクエストステータス」をクリックしたときに、リクエストステータスのログが表示されます。

制限

Log Service で、ドリルダウン分析に使用可能なグラフは次のとおりです。

- 表
- 折れ線グラフ
- 縦棒グラフ
- 横棒グラフ
- 円グラフ
- 単一値グラフ
- 面グラフ
- ツリーマップ

前提条件

1. インデックスを有効にし、設定が完了していること。
2. クイック照会、ダッシュボード、およびリンクの移動先を設定していること。
3. 変数を追加する必要がある場合、移動先のクイック照会のダッシュボード構成でクエリステートメントの変数プレースホルダを設定していること。 詳細は、クイック照会およびダッシュボードをご参照ください。

手順

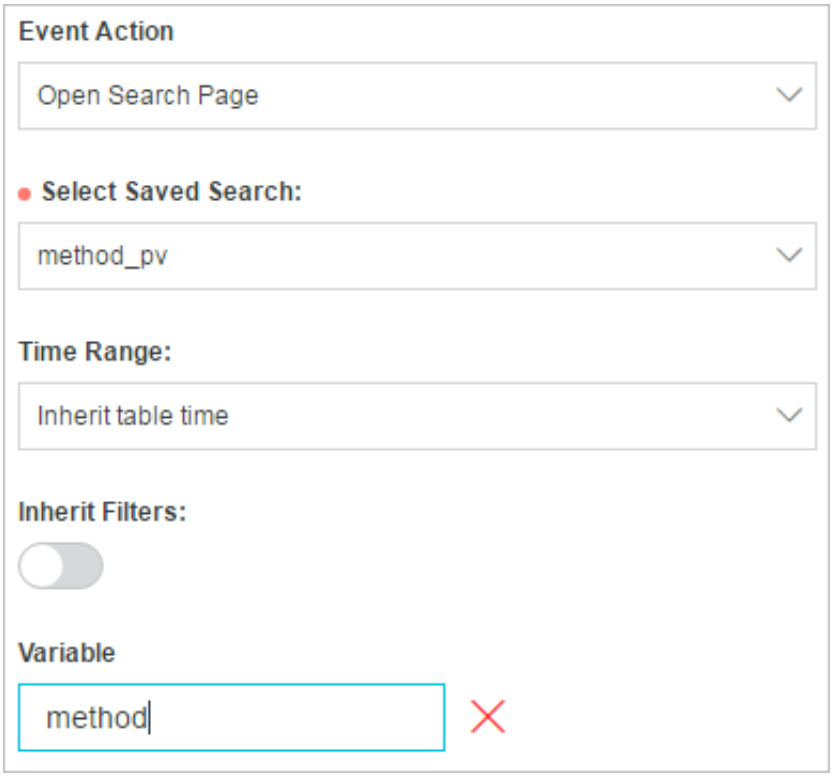
1. 登#日志服#控制台，##Project名称。
2. Logstore リストの照会/分析列で照会をクリックします。
3. クエリ分析ステートメントを入力し、期間を設定して、照会/分析をクリックします。
4. グラフタブでグラフの種類を選択し、グラフのプロパティを設定します。

5. プロパティ列の右側のドリルダウンをクリックし、ドリルダウンイベントを構成します。

   デフォルトでは、ドリルダウン構成は無効です。ドリルダウンイベントはクリック 1 つでトリガーされます。ドリルダウンイベントは、ダッシュボードページの分析グラフをクリックするとトリガーされるイベントです。ドリルダウンイベントを構成してダッシュボードのグラフデータをクリックすると、構成したドリルダウンイベントのページに移動します。次の 4 つの選択肢のいずれかを選択します。

   - 無効化: ドリルダウン機能を無効にします。
   - 照会ページを開く: ドリルダウンを有効にします。照会ページを開くドリルダウンイベントです。

     グラフ内の値をクリックすると、保存した照会ステートメントに設定されているプレースホルダがクリックしたグラフの値に置き換えられ、グラフの値に合わせてより掘り下げたクエリが実行されます。



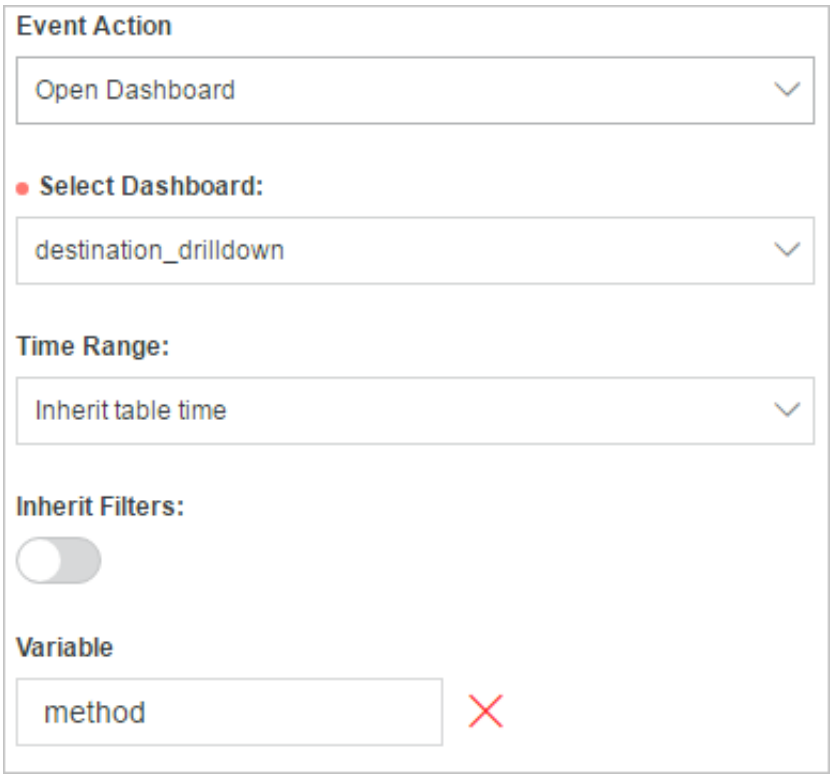
構成項目	説明
クイック照会	移動先のクイック照会の名前。クイック照会の構成に関しては、「クイック照会」をご参照ください。

構成項目	説明
期間	移動先のクイック照会の期間。デフォルトは、表の期間を適用です (ダッシュボードのグラフをクリックした際の、移動先のクイック照会のクエリステートメントの期間は、イベントのトリガーされたダッシュボードに設定されている期間)。
フィルターを継承	フィルターを継承 スイッチをオンにすると、ダッシュボードに追加されたフィルタリング条件がクイック照会と同期され、`AND` を使用してクエリステートメントの前にフィルタリング条件が追加されます。
変数	プレースホルダ変数名を入力するには、変数の追加をクリックします。 クイック照会の変数が追加された変数名と一致すると、クエリステートメントの変数はドリルダウンイベントをトリガーするグラフ値に置き換えられます。クイック照会ンションを柔軟に変更することができます。 注 : 変数を追加するには、まず移動先のクイック照会でクエリステートメントのプレースホルダ変数を構成する必要があります。

- ダッシュボードを開く: ドリルダウン機能を有効にします。ダッシュボードを開くドリルダウンイベントです。

  ダッシュボードのグラフは、クエリテートメントのグラフ形式の結果です。 移動先のダッシュボードグラフにクエリステートメントレースホルダを事前構成する必要があります。上位レベルのダッシュボードでグラフの値をクリックすると、事前に構成されたプレース

ホルダがグラフの値に置き換えられ、グラフの値に従ってより詳細なクエリが実行されます。

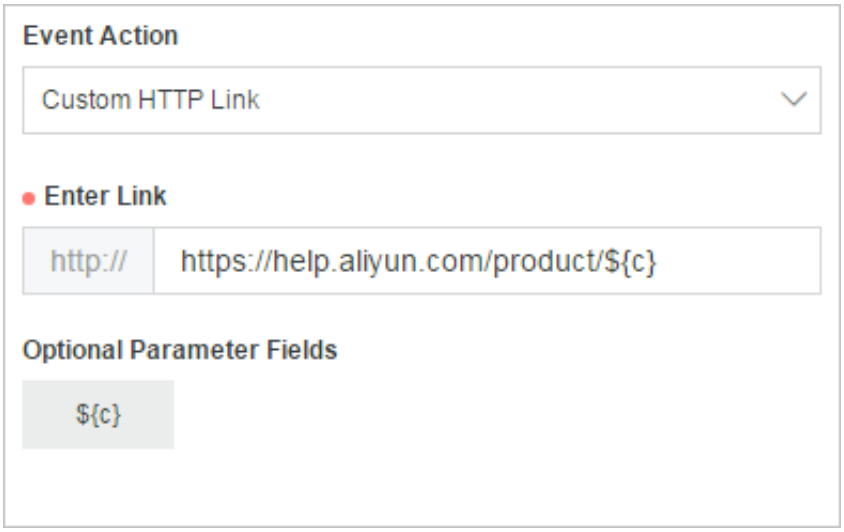


構成項目	説明
ダッシュボード	移動先のダッシュボードの名前。 ダッシュボードの構成については、「ダッシュボード」をご参照ください。
期間	移動先のダッシュボードの時間範囲を構成します。 デフォルトではInherit table timeとなります。 つまり、現行ダッシュボードのグラフをクリックして構成済みのダッシュボードに移動した後、構成済みのダッシュボードの時間範囲は、デフォルトで現行ダッシュボードグラフでドリルダウンイベントがトリガーされた時間に設定されます。移動先のダッシュボードの期間。デフォルトは、表の期間適用です (ダッシュボードのグラフをクリックした際の、移動先のクイック照会のクエリステートメントの期間は、イベントのトリガーされたダッシュボードに設定されている期間)。
フィルターを継承	フィルターを継承 スイッチをオンにすると、ダッシュボードに追加されたフィルタリング条件がクイック照会と同期され、 `AND` を使用してクエリステートメントの前にフィルタリング条件が追加されます。

構成項目	説明
変数	プレースホルダ変数名を入力するには、変数の追加をクリックします。 クイック照会数が追加された変数の名前と一致すると、クエリステートメントドリルダウンイベントをトリガーするグラフ値に置き換えられます。 これでクイック照会のデ照会ョンを柔軟にすることができます。 注： 変数を追加するには、まず、移動先のダッシュボードでクエリステートメントのプレースホルダ変数を構成する必要があります。

- カスタム HTTP リンク：ドリルダウン機能を有効にします。 ここのドリルダウンイベントは、カスタム HTTP リンクを開くことです。

  HTTP リンクのパス部分は、アクセス先ファイルの階層パスを示します。 カスタム HTTP リンクのパス部分にオプションのパラメータフィールドを追加してダッシュボードのグラフの内容をクリックすると、追加されたパラメータフィールドがグラフの値に置き換えられ、移転した HTTP リンクに移動します。



構成項目	説明
リンク	移動先の宛先アドレス。
オプションパラメータフィールド	オプションのパラメータ変数をクリックすると、HTTP リンクの一部をドリルダウンイベントをトリガするグラフ値に置き換えることができます。

6. 新しいダッシュボードに追加をクリックしてダッシュボードを構成し、OK をクリックします。

   そしてダッシュボードページで分析グラフを表示し、グラフをクリックしてより深い分析結果を表示できます。

例

収集した Nginx アクセスログを「accesslog」という名前の Logstore として保存し、「RequestMethod」という名前のダッシュボードに Nginx ログの一般的な分析シナリオを表示し、「destination_drilldown」という名前のダッシュボードに PV 分布の傾向を表示できます。 リクエスト方式のテーブルのドリルダウン分析を構成し、それを RequestMethod ダッシュボードに追加し、「destination_drilldown」ダッシュボードに移動するようにドリルダウンイベントを構成できます。 RequestMethod ダッシュボードで、各リクエスト方式をクリックして「destination_drilldown」ダッシュボードに移動し、対応する PV トレンドを表示します。

手順は次のとおりです。

1. 移動先のダッシュボードを作成します。

   a. ログをリクエストタイプごとにフィルタリングし、PV の経時変化を表示します。

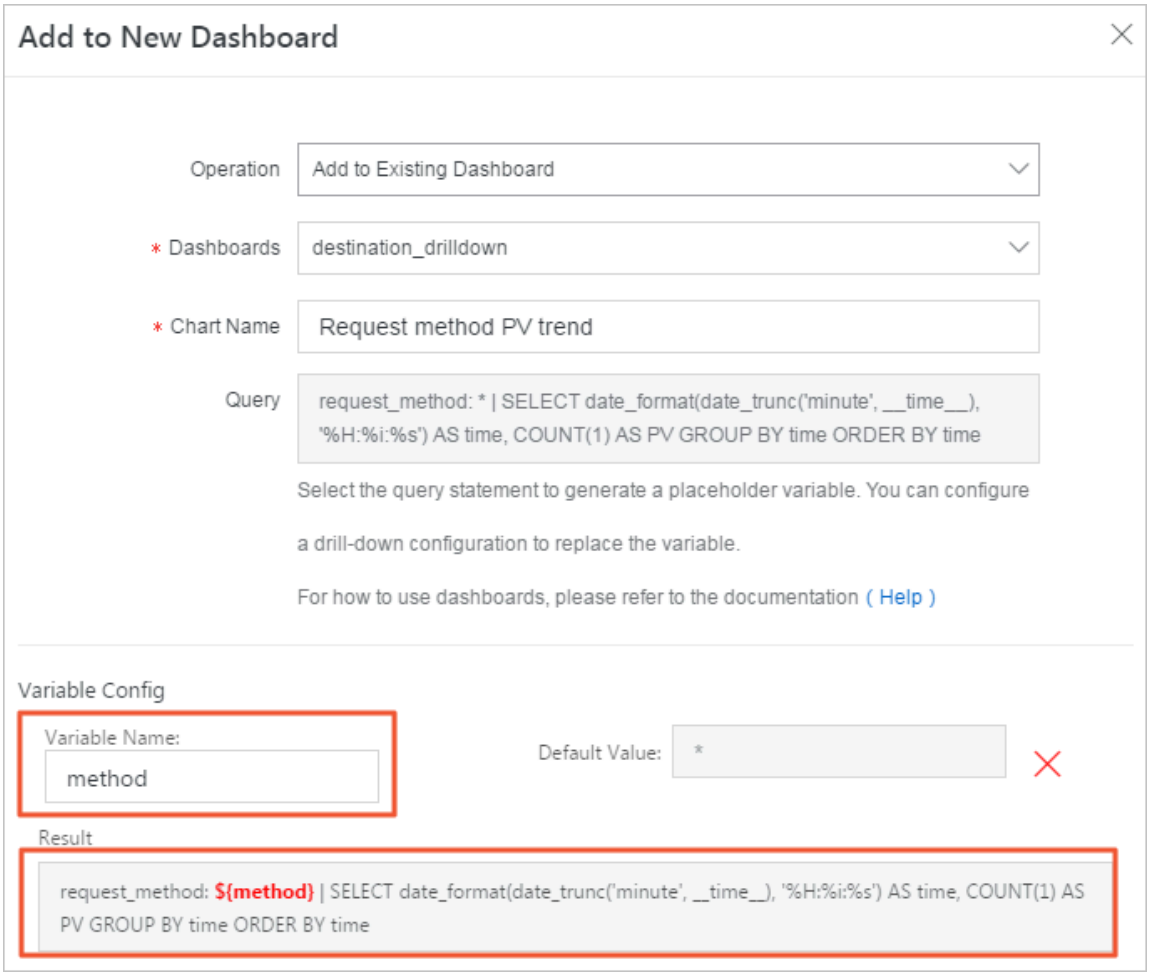
      クエリステートメント:

      ```
      request_me  thod : * |  SELECT   date_forma  t ( date_trunc ('
      minute ',  __time__ ), '% H :% i :% s ')  AS   time ,  COUNT ( 1
      )  AS   PV   GROUP   BY   time   ORDER   BY   time
      ```

   b. 折れ線グラフにクエリ結果を表示させ、その折れ線グラフをダッシュボードに保存します。

      グラフをダッシュボードに保存する際に、`*` をプレースホルダとして設定し、「method」という名前をつけます。クイック照会のドリルダウンイベントの変数名も「method」に

します。クリックしたグラフの値で `*` を置き換えて、検索と分析を再度実行することができます。

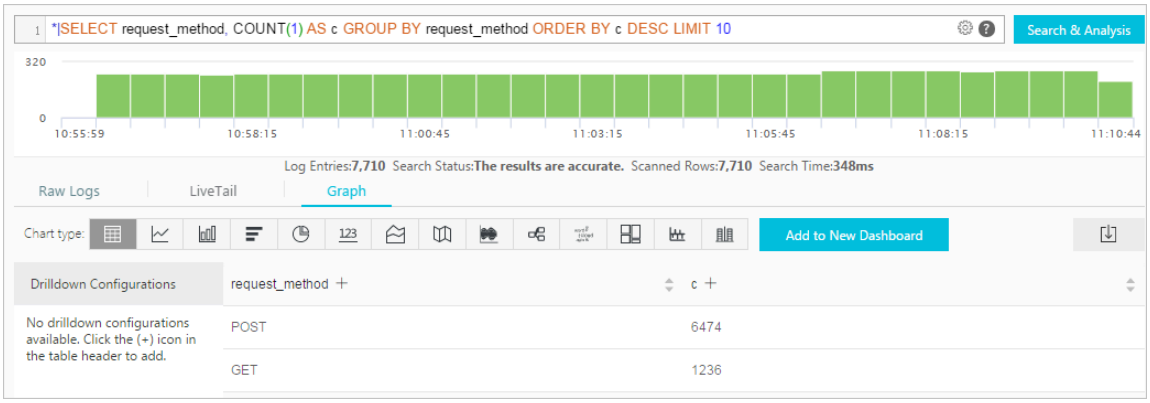


2. ドリルダウン分析するグラフを作成し、ダッシュボードに追加します。

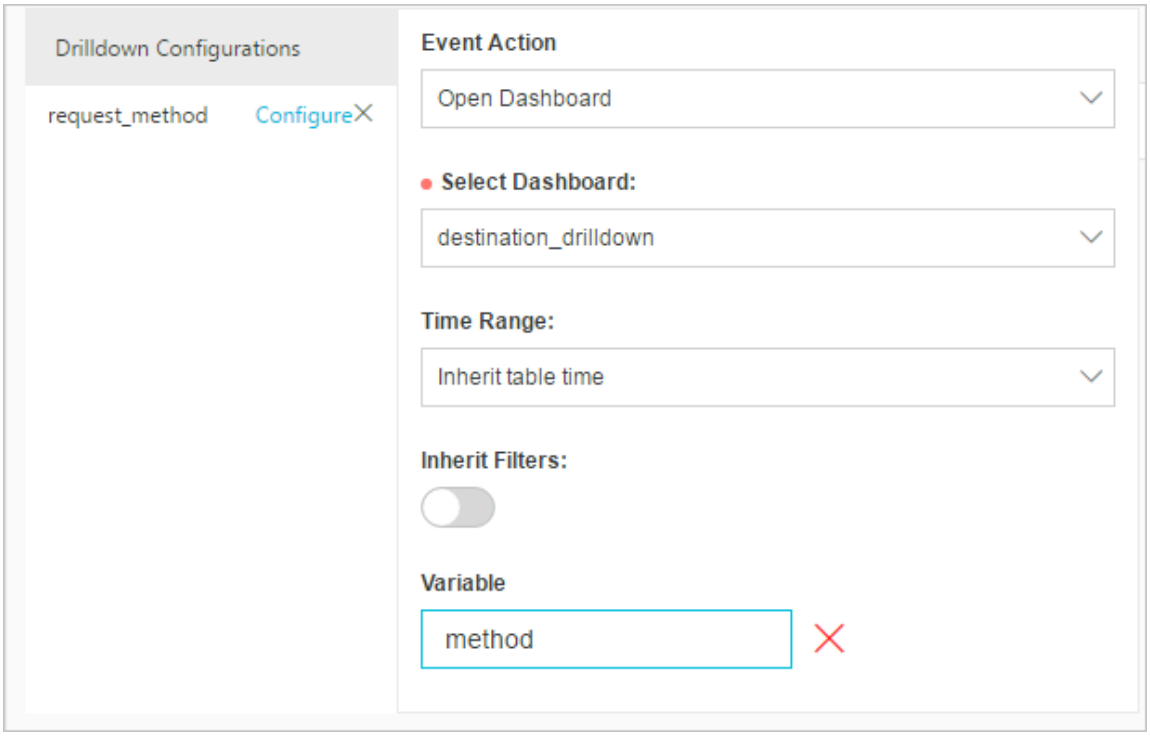
   a. 照会ページで、クエリステートメントを実行すると、Nginx アクセスログ内の各リクエスト方式のログ数が表に表示されます。

```
*| SELECT  request_me  thod , COUNT ( 1 ) AS  c  GROUP  BY
 request_me  thod  ORDER  BY  c  DESC  LIMIT  10
```

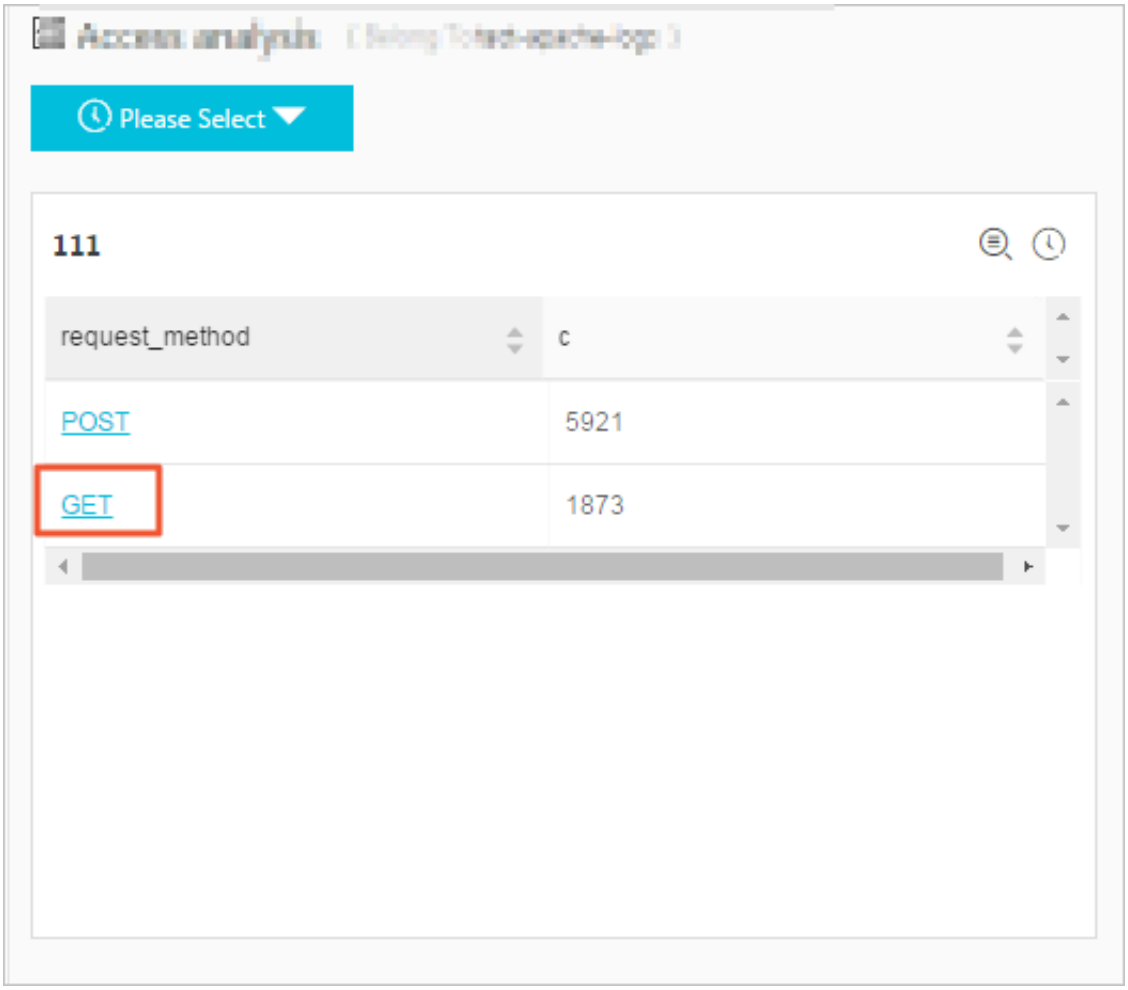
   クエリステートメントの実行結果:



   b. 表の `request_me  thod` 列にドリルダウン分析の設定をします。

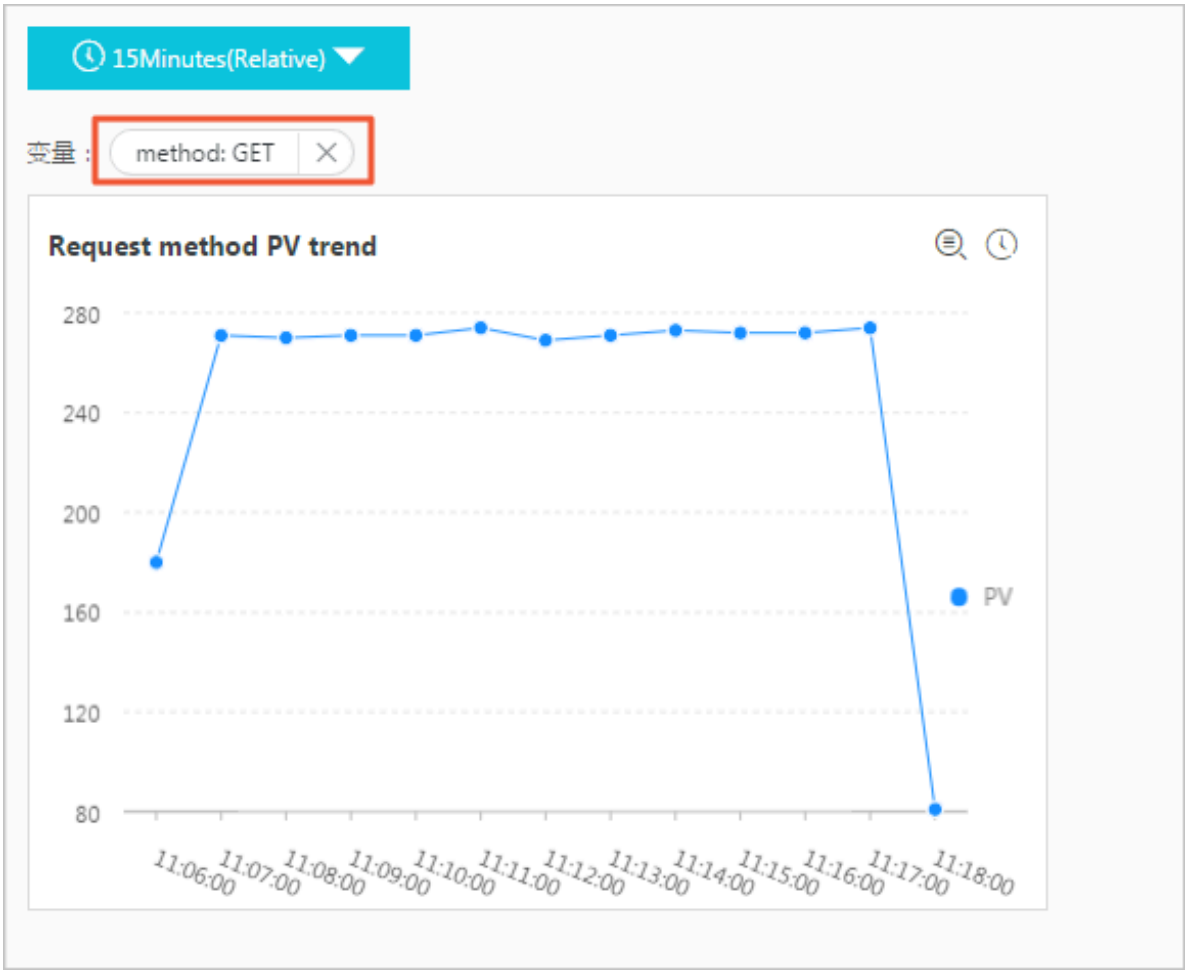


3. 「RequestMethod」ダッシュボードで GET リクエストをクリックします。



4. 「destination_drilldown」ダッシュボードに移動します。

   1で設定したダッシュボードページに移動します。クエリステートメントの `*` は、 `GET` に置き換えられます。ダッシュボードには、GET リクエストによる PV の経時変化が表示されます。



7.2.6 ダッシュボードのフィルター

Log Service のダッシュボードにフィルターを適用することで、ダッシュボード全体のクエリをより詳細にすることや、プレースホルダーを置き換えることができます。

Log Service の各グラフは、クエリ分析ステートメントの実行結果です。ダッシュボードにフィルター (条件) を適用すると、すべてのグラフにフィルター条件が適用され、すべてのグラフのプレースホルダーが置き換えられます。フィルターには次の 2 種類があり、いずれかに設定します。

- フィルタータイプ: `[ search query ]`の前に、フィルターの条件となるキーと値を追加します。クエリステートメントは `key : value AND [ search query ]`になります。もとのクエリステートメントの結果から`キー:値`を含むログのみが抽出されます。

- プレースホルダータイプ: ダッシュボードにプレースホルダーを設定しているグラフがある場合、グラフのクエリステートメントのプレースホルダーは選択した値に置き換えられます。

コンポーネント

各グラフには、1 つまたは複数のフィルターを設定することができます。通常、各フィルターは次の要素で構成されます。

- キー (フィルターするキー)
- リスト項目 (フィルターするキーの値)

制限

- 各ダッシュボードに追加できるフィルターは最大 5 つ
- フィルタータイプの場合、複数の値を選択するか、もしくは、入力してくださいテキストボックスに値を入力します。複数の値を選択した場合、各値は `OR` 条件としてフィルターされます。

前提条件

1. インデックスを有効にして、設定していること
2. ダッシュボードを作成し、プレースホルダーを設定していること

手順

1. 登#日志服#控制台，##Project名称。
2. 照会/分析列の照会をクリックします。
3. 左側のナビゲーションメニューより、作成したダッシュボードをクリックします。
4. ダッシュボードページの右上隅にあるフィルターの追加をクリックします。
5. ダッシュボードのフィルターの表示設定を行います。

表 7-2 : フィルターグラフ設定

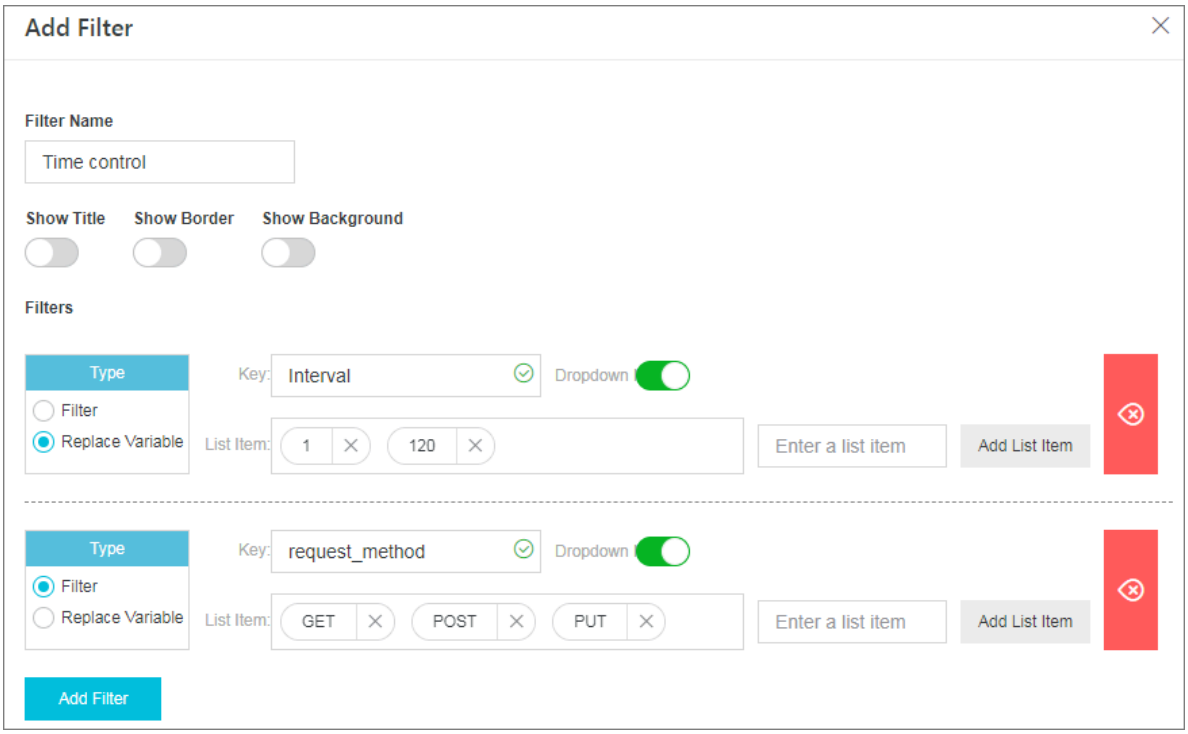
構成項目	説明
グラフ名	フィルターグラフの名前
境界線の表示	フィルターグラフの境界線を表示
タイトルの表示	ダッシュボードにグラフのタイトルを表示
背景の表示	フィルターグラフの背景色を白く表示

6. フィルターの追加をクリックしてフィルターを設定し、OK をクリックします。

表 7-3 : フィルターの構成

構成項目	説明
タイプ	フィルターは 2 種類 ・ フィルター ・ プレースホルダー
キー	・ フィルタータイプ: フィルター条件のキー ・ プレースホルダータイプ: 設定されているプレースホルダー 注 : 前提条件で設定したプレースホルダーを指定します。設定されていないプレースホルダーが指定されている場合、変換されません。
リスト項目	フィルターするキーの値のリスト ・ フィルタータイプ: フィルターキーの値 (複数の値を指定可)。フィルターを作成後、ダッシュボードを表示すると、値を選択できます。 ・ プレースホルダータイプ: プレースホルダーキーの置き換える値。プレースホルダー値を設定します。 フィルターを作成後、ダッシュボードを開いた際に、置換する値を選択できます。 注 : リスト項目の右側のテキストボックスにリスト項目の値を入力し、リスト項目の追加をクリックします。

構成項目	説明
ドロップダウンモード	リスト項目の表示方法 ・ オン: ドロップダウンリストにリスト項目が表示される ・ オフ: リスト項目がオプションとして次の方法で表示される - フィルタータイプ: チェックボックス - プレースホルダータイプ: ラジオボタン



ダッシュボードページは自動的に更新され、新規フィルターの設定ページが表示されます。選択してくださいドロップダウンリストより、プレースホルダーに入れる値を選択し、追加をクリックします。

フィルタータイプの場合、複数の値を選択するか、入力してくださいテキストボックスに値を入力します。複数の値を選択した場合、各値は `OR` 条件としてフィルターされます。

シナリオ

フィルターは、ダッシュボードのクエリ条件を動的に変える場合や、グラフのプレースホルダーの値を別の値に置き換える場合に適用します。各グラフは、照会および分析ステートメント`[ search query ] | [ sqlquery ]`の実行結果です。フィルターを適用することにより、そのステートメントを変えることができます。

- フィルタータイプ: `[ search query ]` の前に、フィルター条件を指定して `AND` で続けると、クエリステートメントは`キー:値 AND [ search query ]`に書き換えられます。
- プレースホルダータイプ: ダッシュボードでプレースホルダーの設定されているグラフのプレースホルダーが、選択した `値` に置き換えられます。

例

Nginx ログを収集し、収集したログをリアルタイムに照会/分析します。

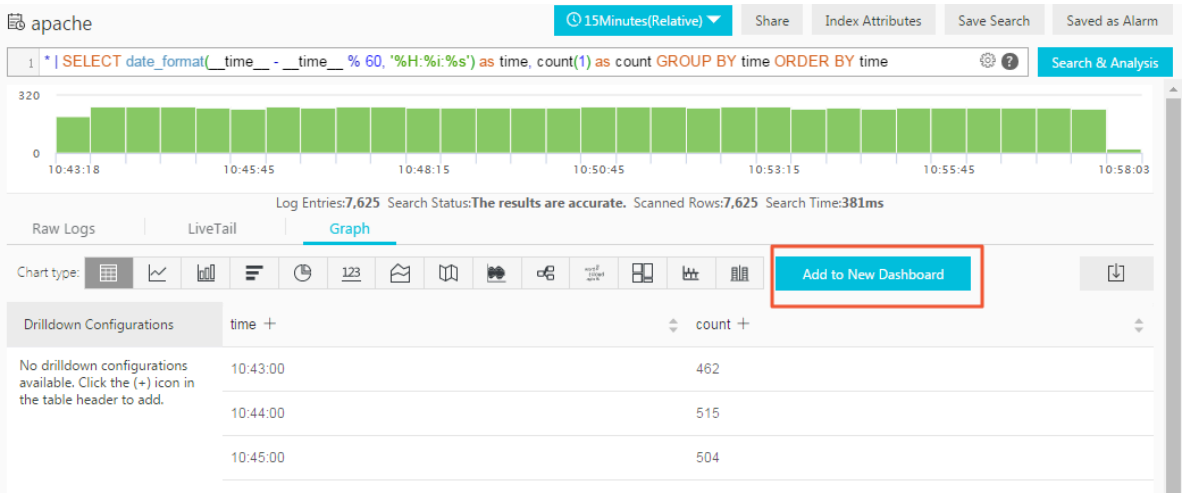
- シナリオ 1 : 時間の精度を変更

  クエリ分析ステートメントを使用して、分単位の PV を表示することができます。秒単位の測定データを表示するには、`__time__ - __time__ % 60` の値を変更する必要があ

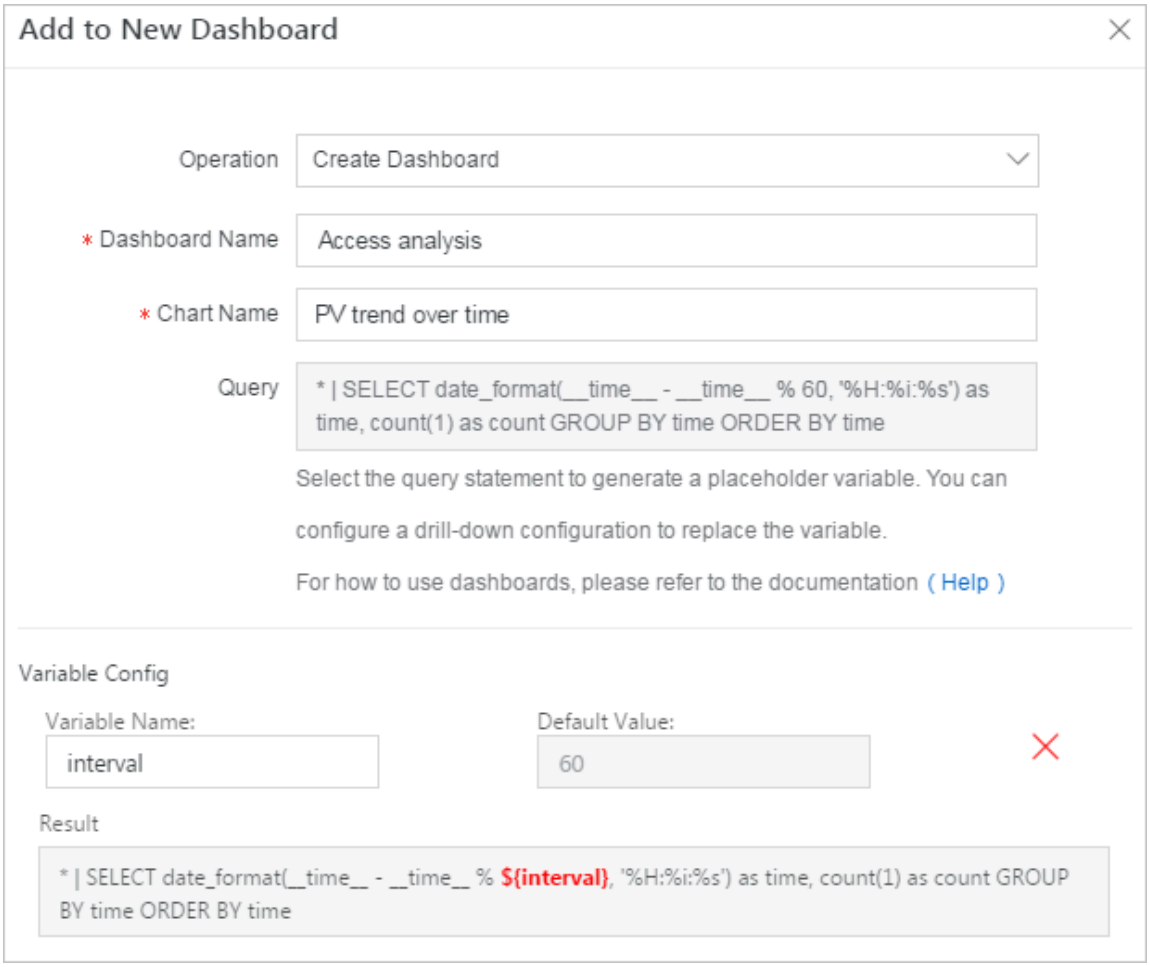
ります。従来の方法では、クエリ分析ステートメントを変更する必要があり、クエリの階層が深い場合には大変でしたこのような場合、にフィルターを使用して変数を置き換えます。

次のステートメントにより、毎分の PV 数を表示します。

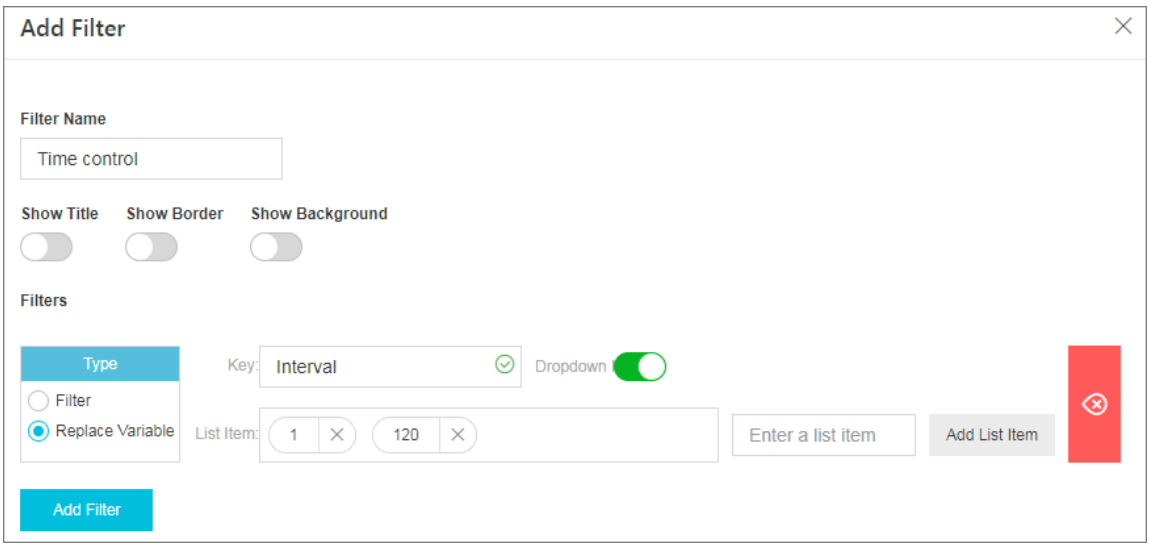
```
* | SELECT date_forma t ( __time__ - __time__ % 60 , '% H :% i :% s ') as time , count ( 1 ) as count GROUP BY time ORDER BY time
```



1. ダッシュボードにグラフを追加します。なお、プレースホルダーのデフォルト値を `60` 、変数名を `interval` にします。

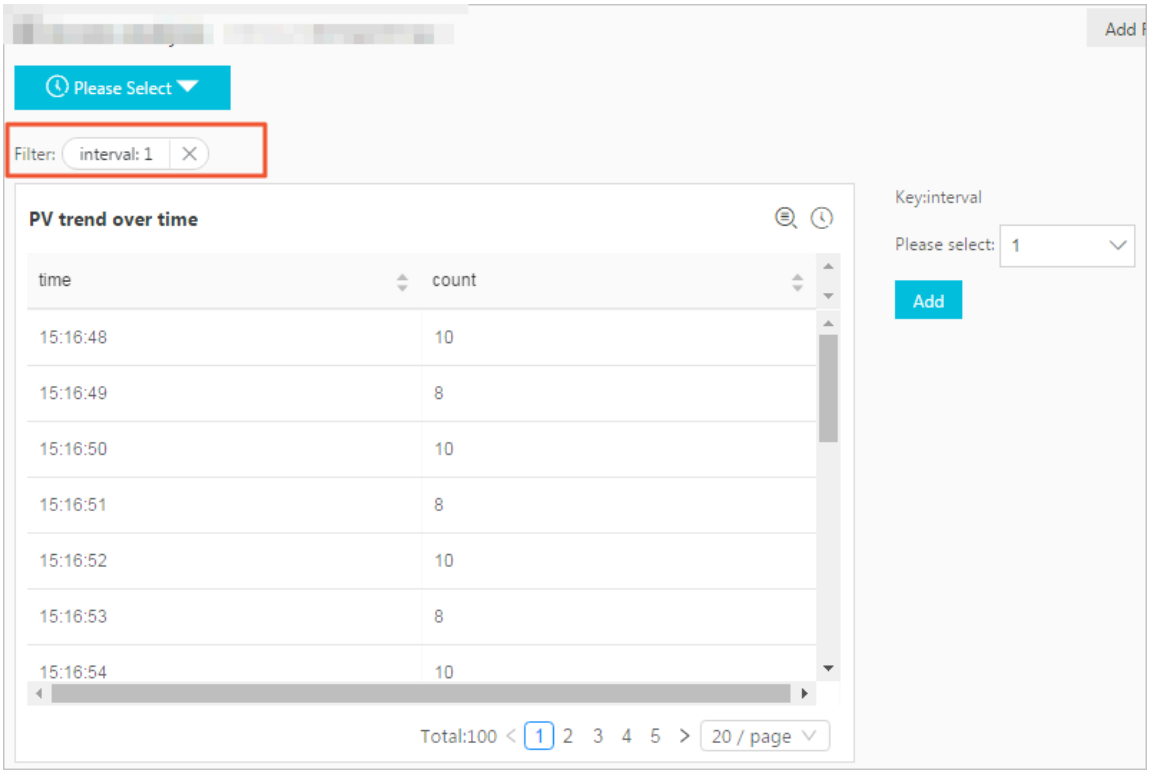


2. フィルターを追加し、プレースホルダータイプを選択します。
   - タイプ: プレースホルダー
   - キー: `interval`
   - リスト項目: `1` (1 秒ごと) および `120` (2 分ごと)



3. フィルターのリスト項目に `1` を選択すると、プレースホルダーのクエリステートメントは次の通りとなり、ダッシュボードには秒単位の測定データが表示されます。

```
* | SELECT date_forma t ( __time__ - __time__ % 1 , '% H :
% i :% s ') as time , count ( 1 ) as count GROUP BY
 time ORDER BY time
```

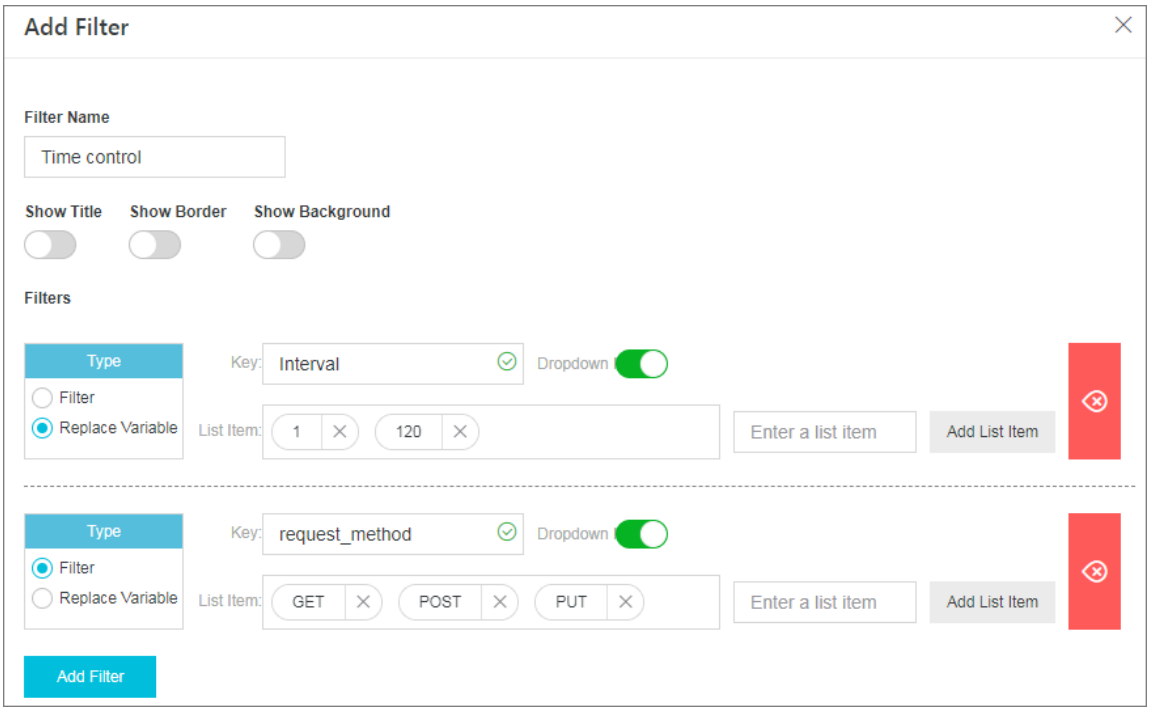


- シナリオ 2 : フィルター条件を動的に切り替える

  フィルターを使用して動的にリクエストメソッドを切り替えることができます。シナリオ 1 のクエリステートメントは `*` で始まるため、フィルター条件が指定されていないことになります

(すべてのログがクエリ対象)。フィルターを追加して各アクセスの `request_me thod` 統計を表示させることができます。

1. シナリオ 1 のダッシュボードに新しいフィルターを追加し、設定を次のとおりにします。
   - タイプ: `フィルター`
   - キー: `request_me thod`
   - リスト項目: `GET` 、 `POST` 、および `PUT`

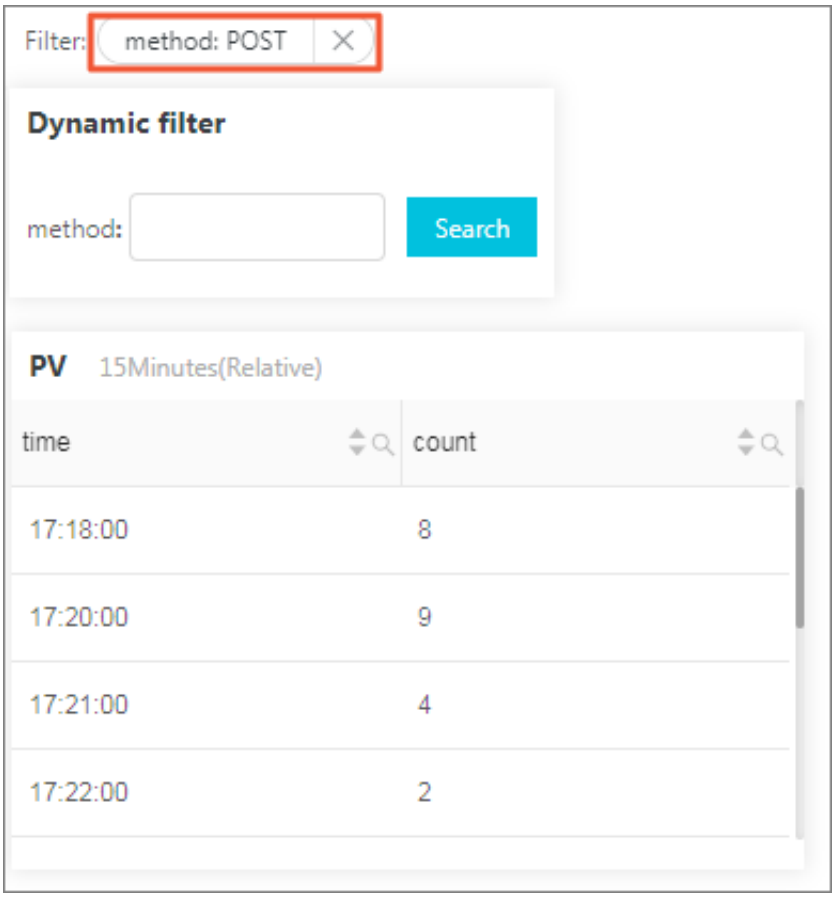


2. フィルターのドロップダウンリストより `GET` を選択し、また、リスト項目に `DELETE` を追加します。

   クエリ分析ステートメントは次のように変更され、 `request_me thod` が `GET` および `DELETE` のアクセス統計のみがグラフに表示されます。

```
(*) and ( request_me thod : GET OR request_me thod :
 DELETE ) | SELECT date_forma t ( __time__ - __time__ % 60
```

```
, '% H :% i :% s ')  as  time ,  count ( 1 )  as  count  GROUP
  BY  time  ORDER  BY  time
```



7.2.7 Markdown ダイアグラム

Log Service のダッシュボードには、Markdown ダイアグラムを追加できます。Markdown ダイアグラムには、画像、リンク、ビデオ、およびその他の要素を挿入することができます。より見やすいダッシュボードになります。

Log Service でのログデータ照会/分析では、ダッシュボードに複数のグラフを追加し、複数のサービスをリアルタイムにモニタリングすることができます。Log Service のダッシュボードには、Markdown ダイアグラムを追加することもできます。Markdown ダイアグラムは、Markdown 言語を記述して編集します。画像、リンク、ビデオ、およびその他の要素を Markdown ダイアグラムに挿入して、より見やすいダッシュボードページにすることができます。

Markdown ダイアグラムは、さまざまに利用できます。Markdown ダイアグラムには、背景説明、ダイアグラムの説明、ページメモ、詳細情報といったテキストを挿入できます。ダッシュボードを豊かに表現できます。あるいは、別のクエリページに簡単に移動できるように、Markdown ダイアグラムに、クイック照会、別プロジェクトのダッシュボードへのリンクを挿

入することができます。Markdown ダイアグラムに画像を挿入して、ダッシュボードの情報量をさまざまに充実させることもできます。

シナリオ

Markdown ダイアグラムには、プロジェクト内の別のダッシュボードにリダイレクトするリンクを作成することができます。また、各リンクに画像を挿入することもでき、識別しやすくできます。グラフの項目の説明として、Markdown ダイアグラムを挿入することもできます。

図 7-20 : シナリオ

前提条件

1. 収集済みのログがあること
2. ダッシュボードを作成していること

手順

1. ダッシュボードページの右上隅の編集をクリックします。
2. Markdown の作成をクリックします。
3. 表示されたページで、Markdown ダイアグラムのプロパティを設定します。

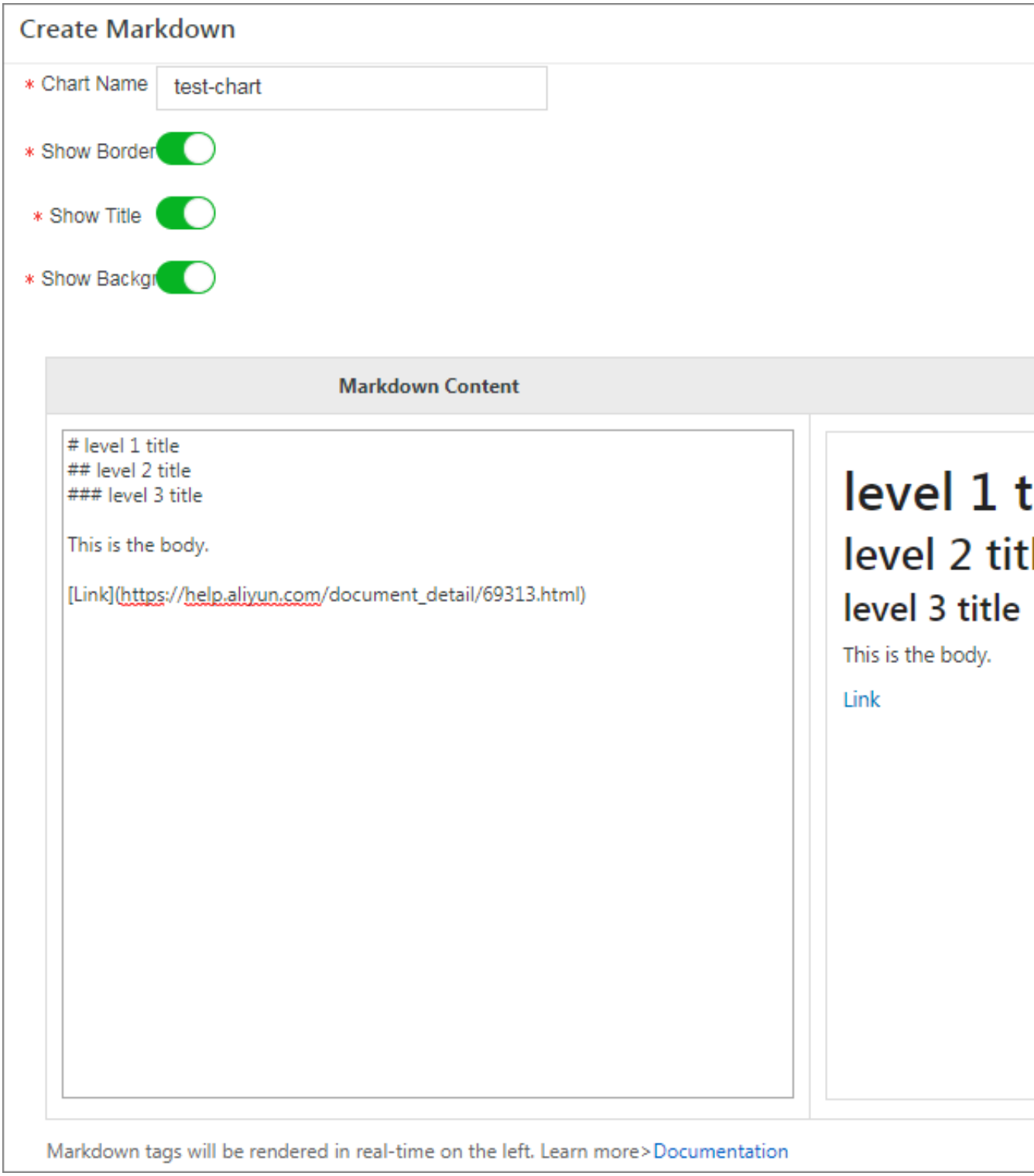
構成項目	説明
グラフ名	Markdown ダイアグラムの名前
枠線の表示	Markdown ダイアグラムの枠線を表示
タイトルの表示	ダッシュボードに Markdown ダイアグラムのタイトルを表示
背景の表示	Markdown ダイアグラムの背景色を白にする

4. Markdown コンテンツを設定します。

   Markdown コンテンツ欄に、Markdown を入力します。右側のグラフを表示欄に、プレビュー表示されます。プレビューを確認し、Markdown を変更します。

5. 設定を完了したら、OK をクリックします。

図 7-21 : Markdown ダイアグラムの作成



設定を完了すると、ダッシュボードの下部に作成した Markdown ダイアグラムが表示されます。

Markdown ダイアグラムの変更

- ダイアグラムの位置とサイズを変更
  1. ダッシュボードページの右上隅にある編集をクリックします。
  2. Markdown ダイアグラムをドラッグして位置を調整し、ダイアグラムの右下隅をドラッグしてサイズを調整します。
  3. 右上にある 作成をクリックします。
- ダイアグラムのタイトルを変更
  1. ダッシュボードページの右上隅にある編集をクリックします。
  2. ダイアグラムのタイトルボックスに新たなタイトルを入力します。
  3. ダッシュボードページの右上隅にある保存をクリックし、表示されるダイアログボックスの OK をクリックします。
- ダイアグラムのコンテンツを変更
  1. ダッシュボードページで、右上隅にある編集をクリックします。
  2. Markdown ダイアグラムの右上隅にある編集をクリックします。
  3. ダイアグラムの設定を変更して OK をクリックします。
- ダイアグラムを削除
  1. ダッシュボードページの右上隅の編集をクリックします。
  2. Markdown ダイアグラムの右上隅にある削除をクリックします。
  3. ダッシュボードページの右上隅にある保存をクリックし、表示されるダイアログボックスの OK をクリックします。

一般的な Markdown 記法

- タイトル

  Markdown 記法

```
#  Level   1   title
##  Level   2   title
```

```
###  Level   3   title
```

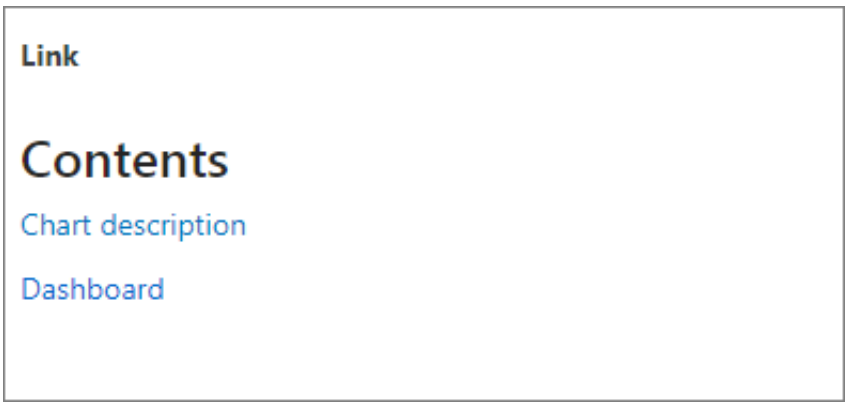
図 7-22 : タイトルプレビュー

- **リンク**

  Markdown 記法

```
###  Contents

[ Chart   descriptio  n ]( https :// help . aliyun . com /
 document_d  etail / 69313 . html )

[ Dashboard ]( https :// help . aliyun . com / document_d  etail /
 59324 . html )
```

図 7-23 : リンクのプレビュー



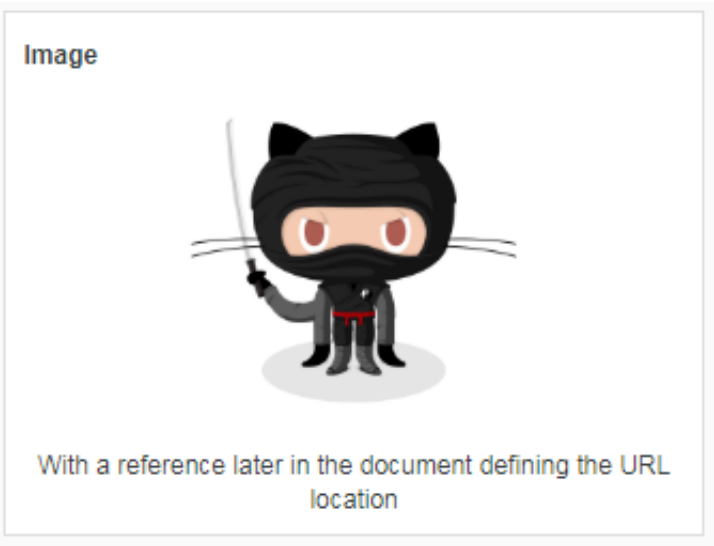
- **画像**

  Markdown 記法

```
< div   align = center >

![  Alt   txt ][ id ]
```

```
 With   a   reference   later   in   the   document   defining   the
   URL   location

[ id ]:  https :// octodex . github . com / images / dojocat . jpg
 " The   Dojocat "
```

図 7-24 : 画像のプレビュー



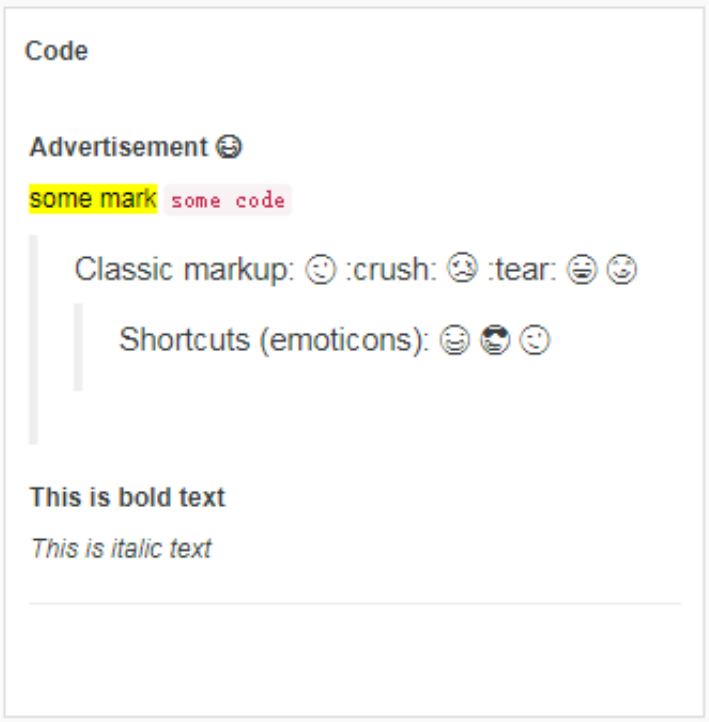
- **絵文字**

  Markdown 記法

```
---

 __Advertis  ement  :) __

== some   mark ==   ` some   code `
>  Classic   markup : : wink : : crush : : cry : : tear : : laughing
 : : yum :
>>  Shortcuts  ( emoticons ): :-)   8 -) ;)

 __This   is   bold   text__

* This   is   italic   text *
```

```
---
```

図 7-25 : 絵文字のプレビュー



Markdown 記法の詳細は、Markdown 記法を参照してください。

7.3 その他の可視化

7.3.1 コンソールの埋め込みで共有

Log Service の収集および照会/分析の設定を行うと、ログの照会/分析およびダッシュボード機能をそのまま使用できるとともに、他のユーザーと共有することもできます。ただし、RAM を使用して共有した場合、サブアカウントの管理にかかるコストが嵩む可能性があります。この状況を回避するために、Log Service では、照会/分析およびダッシュボードが統合された、埋め込みページにログインしてシングルポイントにすることができます。

利点

特定の Logstore 検索ページとダッシュボードページを構築 Web サイトに埋め込むことができます。これにより、Alibaba Cloud にログインしなくても Log Service の分析および可視化機能にアクセスできます。

- 専用の検索ページとダッシュボードページは、 Web サイトにも簡単に埋め込むことができます。

・ セキュリティトークンサービス (STS) を使用してログインリンクを生成し、RAM (Resource Access Management) を使用して読み取り権限といった操作権限を制御できます。

1. 構築 Web サイトにログインします。

   ログインすると、Web サーバー STS は一時的な ID を取得します。

   ・ STS の詳細は、「Overview」をご参照ください。
   ・ 指定した Logstore へのアクセスをユーザーに許可します。 詳細は、「RAM ユーザーに Log Service へのアクセスを許可」をご参照ください。

2. Alibaba Cloud ログインサービスにリクエストを送信して、ログイントークンを取得します。

   STS から一時アクセスキーペアとセキュリティトークンを取得した後、ログインサービスインターフェイスを呼び出してログイントークンを取得します。

   例:

```
http :// signin . aliyun . com / federation ? Action = GetSigninT
oken
                    & AccessKeyI  d =< STS  より返された一時アクセスキー
ペア>
                    & AccessKeyS  ecret =< STS  より返された一時
 Secret >
                    & SecurityTo  ken =< STS  より返されたセキュリティ
トークン>
```

3. ログイン不要のリンクを生成します。

   a) ログイントークンを取得すると、埋め込みページへのリンクと共にアクセスリンクを生成します。

   トークンは 3 時間有効のため、新しいログイントークンを生成し、各アクセスリクエストを 302 メッセージを使用して構築 Web サイトへの埋め込みリンクにリダイレクトすることをお勧めします。

   例:

```
 http :// signin . aliyun . com / federation ? Action = Login
                                & LoginUrl =<ログイン失敗時にログインリクエ
ストがリダイレクトされるアドレス。通常、 302 メッセージを通じて構築 Web サ
イトの URL に設定されます。>
                                & Destinatio  n =<アクセスする Log
 Service ページ。 検索とダッシュボードページをサポートします。>
```

```
                                & SigninToke  n =<取得されたログイントークン>
```

b) ページに埋め込み

- 検索と分析のための完全なページ (複数のタグが許可されています)

```
https :// sls4servic  e . console . aliyun . com / next /
project /<プロジェクト名>/ logsearch /<ログストア名>? hideTopbar =
true & hideSideba  r = true
```

- 検索ページ

```
https :// sls4servic  e . console . aliyun . com / next /
project /<プロジェクト名>/ logsearch /<ログストア名>? isShare =
true & hideTopbar = true & hideSideba  r = true
```

- ダッシュボードページ

```
https :// sls4servic  e . console . aliyun . com / next /
project /<プロジェクト名>/ dashboard /<ダッシュボード名>? isShare =
true & hideTopbar = true & hideSideba  r = true
```

Java、PHP、Python の例は、次のとおりです。

- Java：

```
< dependency >
                    < groupId > com . aliyun </ groupId >
                    < artifactId > aliyun - java - sdk -
 sts </ artifactId >
                    < version > 3 . 0 . 0 </ version >
                    </ dependency >
                    < dependency >
                    < groupId > com . aliyun </ groupId >
                    < artifactId > aliyun - java - sdk -
 core </ artifactId >
                    < version > 3 . 5 . 0 </ version >
                    </ dependency >
                    < dependency >
                    < groupId > org . apache . httpcompon
 ents </ groupId >
                    < artifactId > httpclient </
 artifactId >
                    < version > 4 . 5 . 5 </ version >
                    </ dependency >
                    < dependency >
                    < groupId > com . alibaba </ groupId >
                    < artifactId > fastjson </ artifactId
 >
                    < version > 1 . 2 . 47 </ version >
                    </ dependency >
```

- PHP
- Python

7.3.2 JDBC 経由のデータベース接続

MySQL は一般的なリレーショナルデータベースです。多くのソフトウェアは、MySQL 転送プロトコルと SQL 構文を使用して MySQL データを取得できます。SQL 構文が解れば、MySQL に接続できます。 Log Service は、ログを照会および分析するための MySQL プロトコルを提供します。 標準 MySQL クライアントを使用して Log Service に接続し、標準 SQL 構文を使用してログを処理および分析することができます。 MySQL 転送プロトコルをサポートするクライアントには、MySQL クライアント、 JDBC、および Python MySQLdb が含まれます。

本ドキュメントでは、自転車シェアリングのログを例として、 JDBC を使って Log Service に接続してログデータを読み取る方法、MySQL プロトコルと SQL 構文を使用してログを計算する方法、 DataV を使用してログデータや計算結果をダッシュボードで視覚化する方法について説明します。

JDBC の利用イメージ

- MySQL プロトコルを使用して Log Service に接続するために、DataV、Tableau、あるいは Kibana などの可視化ツールを使用します。
- Java の JDBC あるいは Python の MySQLdb などのライブラリを使用して、 Log Service にアクセスし、プログラム上でクエリ結果を処理します。

データの例

自転車シェアリングのログには、年齢、性別、バッテリー使用量、車両 ID、操作待ち時間、緯度、ロックタイプ、経度、操作タイプ、操作結果、ロック解除方法が含まれます。 データは `project : tripdemo` の `Logstore : ebike` に格納されます。プロジェクトのリージョンは中国 (上海) です。

ログのサンプルは次のとおりです。

```
Time  : 10 - 12   14 : 26 : 44
__source__ :  11 . 164 . 232 . 105
__topic__ :  v1
age :  55
battery :  118497 . 673842
bikeid :  36
gender :  male
latency :  17
latitude :  30 . 2931185245
lock_type :  smart_lock
longitude :  120 . 052840484
op :  unlock
op_result :  ok
open_lock :  bluetooth
```

```
userid :  292
```

前提条件

コンソールまたは API より、Logstore の各列のインデックスおよび分析機能が有効になっていること。

JDBC 統計

1. Maven プロジェクトを作成し、pom 依存関係に JDBC ライブラリを追加します。

```
< dependency >
 < groupId > MySQL </ groupId >
 < artifactId > mysql - connector - java </ artifactId >
 < version > 5 . 1 . 6 </ version >
</ dependency >
```

2. Java クラスを作成し、JDBC をクエリするコード。

```
/**
*  Created   by   mayunlei   on   2017 / 6 / 19 .
*/
 import   com . mysql . jdbc .*;
 import   java . sql .*;
 import   java . sql . Connection ;
 import   java . sql . Statement ;
/**
*  Created   by   mayunlei   on   2017 / 6 / 15 .
*/
 public   class   jdbc  {
  public   static   void   main ( String   args []){
      // Input   your   configurat  ion   here .
      final   String   endpoint  = " cn - hangzhou - intranet . sls
 . aliyuncs . com ";// Log   Service   intranet   or   VPC   domain
   name
      final   String   port  = " 10005 "; // The   MySQL   protocol
   port   of   Log   Service .
      final   String   project  = " trip - demo ";
      final   String   logstore  = " ebike ";
      final   String   accessKeyI  d  = "";
      final   String   accessKey  = "";
      Connection   conn  =  null ;
      Statement   stmt  =  null ;
      try  {
         // Step   1 :  Load   the   JDBC   driver .
          Class . forName (" com . mysql . jdbc . Driver ");
         // Step   2 :  Create   a   link .
          conn  =  DriverMana  ger . getConnect  ion (" jdbc : mysql
 ://"+ endpoint +":"+ port +"/"+ project , accessKeyI  d , accessKey
 );
         // Step   3 :  Create   a   statement .
          stmt  =  conn . createStat  ement ();
         // Step   4 :  Define   query   statements .  Query   the
   number   of   logs   that   are   generated   on   October   11
 ,  2017   and   meet   the   condition   op  = " unlock ",  and
 query   the   average   operation   latency .
          String   sql  = " select   count ( 1 )  as   pv , avg (
 latency )  as   avg_latenc  y   from  "+ logstore +" " +
                " where   __date__  >= ' 2017 - 10 - 11   00 : 00 :
 00 ' " +
```

```
                "  and   __date__  < ' 2017 - 10 - 12   00 : 00 :
00 '" +
                "  and   op  =' unlock '";
        // Step   5 :  Execute   query   conditions .
         ResultSet   rs  =  stmt . executeQue  ry ( sql );
        // Step   6 :  Extract   the   query   result .
         while ( rs . next ()){
            // Retrieve   by   column   name
             System . out . print (" pv :");
            // Obtain   pv   from   the   result .
             System . out . print ( rs . getLong (" pv "));
             System . out . print (" ;  avg_latenc  y :");
            // Obtain   avg_latenc  y   in   the   result .
             System . out . println ( rs . getDouble (" avg_latenc
y "));
             System . out . println ();
         }
          rs . close ();
    }  catch  ( ClassNotFo  undExcepti  on   e ) {
          e . printStack  Trace ();
    }  catch  ( SQLExcepti  on   e ) {
          e . printStack  Trace ();
    }  catch  ( Exception   e ) {
          e . printStack  Trace ();
    }  finally  {
          if  ( stmt  ! =  null ) {
              try  {
                  stmt . close ();
              }  catch  ( SQLExcepti  on   e ) {
                  e . printStack  Trace ();
              }
         }
          if  ( conn  ! =  null ) {
              try  {
                  conn . close ();
              }  catch  ( SQLExcepti  on   e ) {
                  e . printStack  Trace ();
              }
         }
    }
 }
}
```

DavaV でデータ表示

DataV ダッシュボードでデータを表示できます。Log Service に接続してログデータを読み取り、ログ処理結果を表示します。

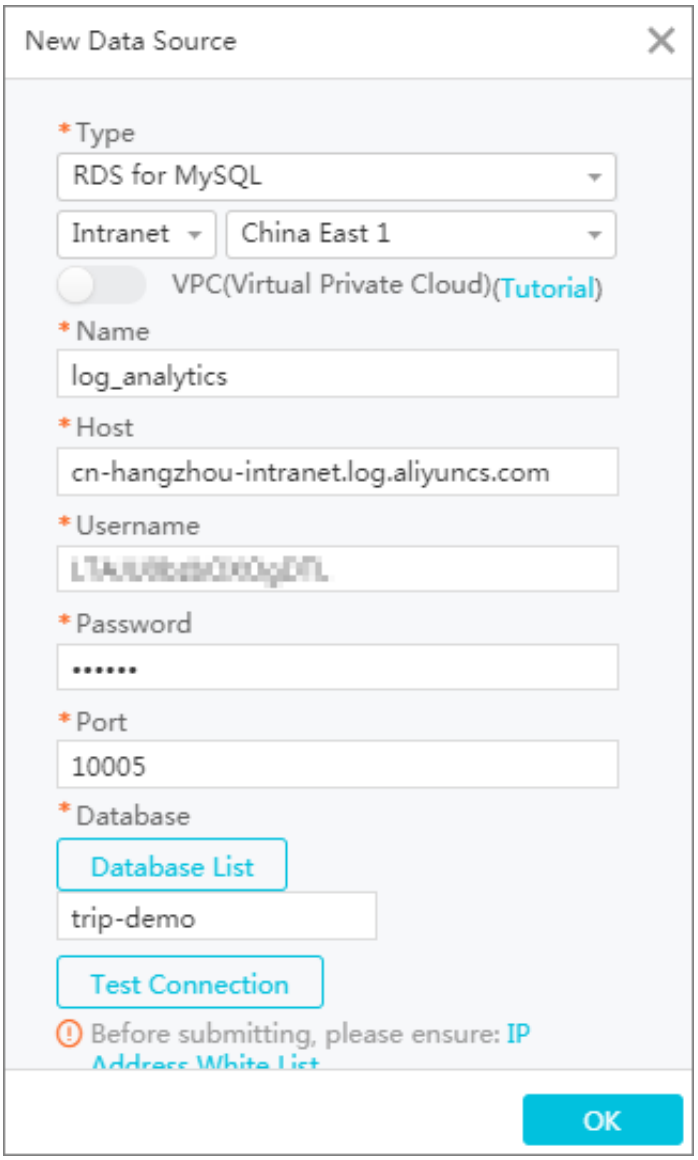
1. データソースの作成

   データソースに、MySQL for RDS または Log Service を選択できます。 次のセクションでは、MySQL を Log Service に接続する方法を説明します。

   図に示すように、リージョンとイントラネットを選択し、ユーザー名とパスワードの AccessKey を入力します。 AccessKey は、Log Service への読み取り権限があるメインア

カウントまたはサブアカウントとなることができます。ポートは10005を指定し、データベースにはプロジェクト名を指定します。

図 7-26 : データのエディタ

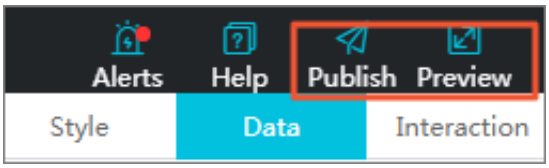


2. ビューの作成

ビューのサービステンプレートを選択し、ダッシュボードで任意のビューをクリックします。ビューを右クリックしてデータとデータソースを変更します。

図に示すように、前の手順で作成したデータベースをデータソースとして選択し、クエリの SQL 文を入力し、上記のフィールドマッピングでは、クエリ結果とビューフィールドの間のマッピング関係を入力します。

図 7-27 : データベースの選択



3. ビューのプレビューおよび公開

プレビューをクリックして表示します。

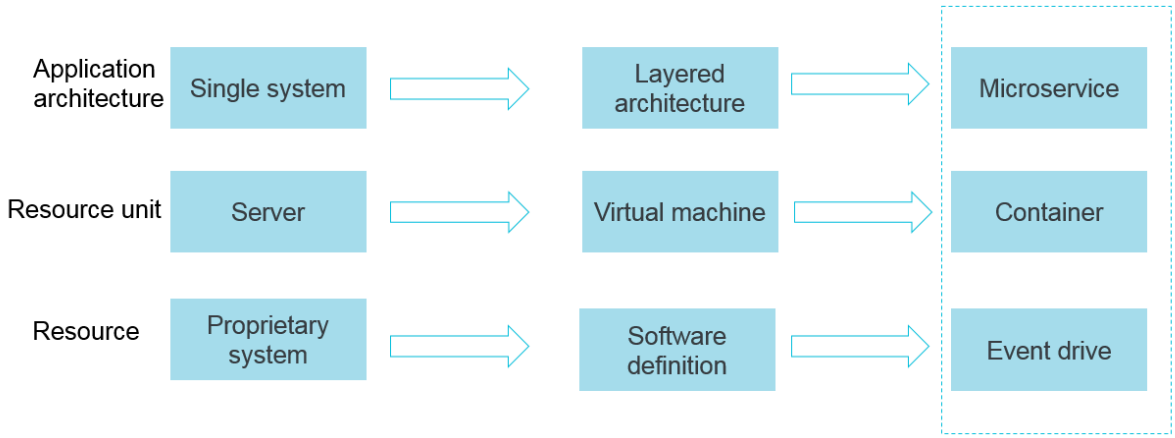
図 7-28 : プレビュー

7.3.3 Jeager の OpenTracing 実装

コンテナ、サーバーレスプログラミングの出現により、ソフトウェアの配信とデプロイの効率が大幅に向上しました。 アーキテクチャの進化で次の 2 点が変わりました。

- アプリケーションアーキテクチャは、単一のシステムからマイクロサービスに変わりつつあり、 ビジネスロジックはマイクロサービス間の呼び出しや、リクエストに変化しています。
- リソースに関しては、従来の物理サーバーは、次第に少なくなってきており、目に見えない仮想リソースに変化しています。

図 7-29 : アーキテクチャの進化



前述の 2 つの変化は、柔軟で標準化されたアーキテクチャが柔軟の一方、運用管理 (O&M) と診断の要件がますます複雑化していることを示しています。これらの変化に対応するために、集中ログシステム (ロギング)、集中測定システム (メトリック)、および分散トレースシステム (トレーシング) を含む一連の DevOps 向けの診断および分析システムが登場しました。

ロギング、メトリック、およびトレーシング

ロギング、メトリック、トレーシングの特徴は次のとおりです。

- ロギングにより、あらゆるイベントが記録されます 。

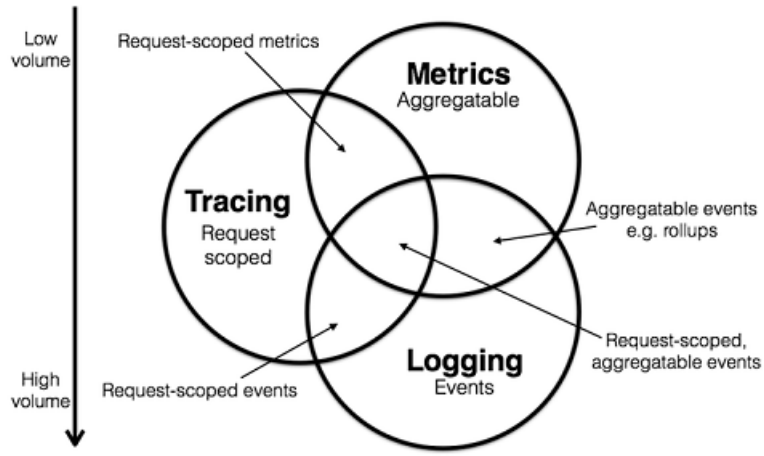
  たとえば、問題診断の基礎となるアプリケーションのデバッグ情報またはエラー情報です。
- メトリックにより、集約可能なデータを記録します。

  たとえば、キューの現在の深さをメトリックとして定義し、要素がキューに追加またはキューから削除される際に更新します。 HTTP リクエスト数は カウンターとして定義されることができ、新しいリクエストを受信するたびに累積されます。

- トレーシングにより、リクエスト範囲内で情報を記録するために使用されます 。

  たとえば、リモートメソッドを1回呼び出しするプロセスと消費時間。これは、システムパフォーマンスを調査するために使用されるツールです。ロギング、メトリック、および、トレーシングは、下図のように重なり合う部分があります。

図 7-30 : ロギング、メトリック、トレーシング



以上から、既存のシステムを分類することができます。たとえば、Zipkin は Tracing に特化しています。プロメテウスはメトリックに特化し始めており、より多くのトレーシング機能が実装されていく可能性は高いですが、ロギングにはあまり関心がないようです。ELK や Alibaba Cloud Log Service といったシステムはロギングに特化し始めていますが、他の機能も継続的に統合されており、上図の中心に向かっています。

詳細については、メトリック、トレーシング、ロギングをご参照ください。次は、トレーシングシステムの紹介です。

トレーシングの技術的背景

トレーシング技術は 1990 年代から存在していました。Google 記事「Dapper, a Large-Scale Distributed Systems Tracing Infrastructure」により、この分野は主流となりました。また、記事「Uncertainty in Aggregate Estimates from Sampled Distributed Traces」には、サンプリングに関して詳細に分析されています。左記記事の発表により、優れたトレーシングソフトウェアプログラムを開発するグループが多く現れました。

よく利用されている製品

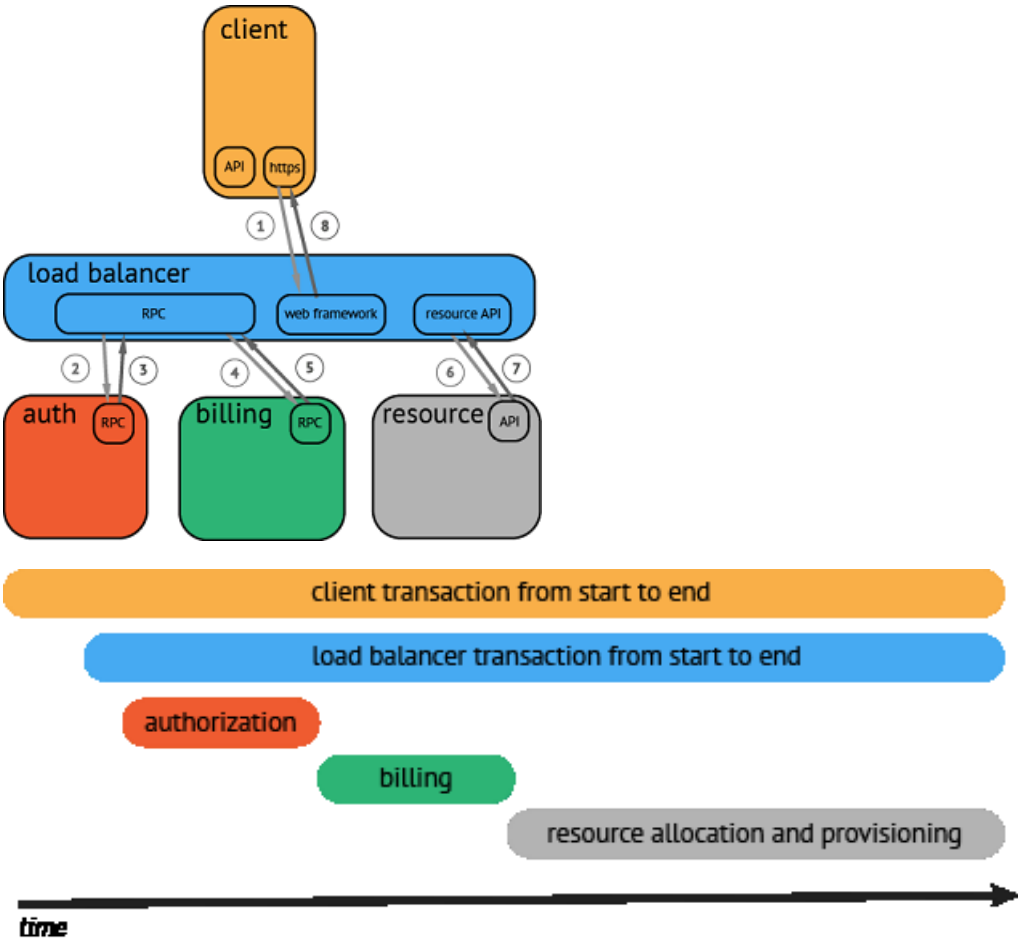
- Dapper (Google): あらゆる Tracer の基盤
- StackDriver Trace (Google)
- Zipkin (Twitter)
- Appdash (golang)

- EagleEye (Taobao)
- Ditecting (Pangu、Alibaba Cloud のプロダクトで使用される Tracing システム)
- Cloud Map (Ant Tracing システム)
- sTrace (Shenma)
- X-ray (AWS)

分散トレーシングシステムは急速に発展しており、種類も数多くあります。 ただし、いずれも共通してコードトラッキング、データストレージ、およびクエリの表示の 3 ステップがあります。

下図に、分散呼び出しの例を示します。 クライアントがリクエストを開始すると、まずロードバランサに送信され、次に認証サービス、請求サービス、そして最後にリクエストされたリソースの順に渡されます。 最後に、システムは結果を返します。

図 7-31 : 分散呼び出しの例

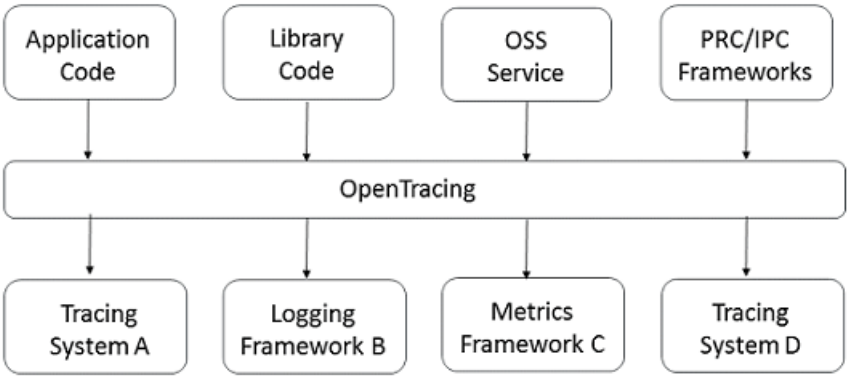


データが収集され保存されると、分散型トレーシングシステムは通常、タイムラインを含むタイミング図としてこの Trace を提示する。 ただし、データ収集プロセスでは、システムがユーザーコードに割り込まなければならない、 その上、異なるシステム間の API 互換性がないため、トレーシングシステムを切り替え流には、大きな変更を加える必要があります。

OpenTracing

分散型トレーシングシステム間の API 非互換性に対処するため、OpenTracing が誕生しました。OpenTracing は、軽量の標準化レイヤーで、アプリケーション/クラスライブラリと、トレーシングまたはログ解析プログラムの中間に位置します。

図 7-32 : OpenTracing



利点

- OpenTracing は CNCF に入られており、グローバル分散トレースシステムの統一コンセプトとデータ標準を提供しています。
- OpenTracing は、プラットフォームやベンダーに依存しない API を提供しているため、開発者はトレーシングシステムを容易に実装 (または変更) できます。

データモデル

OpenTracing のトレーシング (呼び出しチェーン) はこの呼び出しチェーン内の Span によって暗黙的に定義されます。具体的には、Trace (呼び出しチェーン) は 複数の Span で構成される有向非巡回グラフ (DAG) と見なすことができます。Span と Span との関係は References と呼ばれます。

たとえば、次の trace には 8 つの Span で構成されています。

```
単一の  trace  の  Span  間における因果関係
        [ Span   A ] ←←←( the   root   span )
             |
     +------+------+
     |             |
 [ Span   B ] [ Span   C ] ←←←( Span   C  は  Span   A  の子ノード、
 ChildOf )
     |             |
 [ Span   D ] +---+-------+
              |           |
          [ Span   E ] [ Span   F ] >>> [ Span   G ] >>> [ Span   H ]
                                        ↑
                                        ↑
                                        ↑
```

```
                              ( Span   G  は  Span   F  の後に呼び出されます、
 FollowsFro  m )
```

次の例に示すように、タイムラインに基づくタイミング図が、Trace (呼び出しチェーン) をより適切に表示できることがあります。

```
単一の  trace  の  Span  間における時間的関係。
--|-------|-------|-------|-------|-------|-------|-------|->  time
 [ Span   A ·····································································]
   [ Span   B ·····································································]
      [ Span   D ·····································]
    [ Span   C ·····································]
         [ Span   E ·····] [ Span   F ··] [ Span   G ··] [ Span   H
 ··]
```

各 Span には以下が示されます。

- 操作名
- 開始時間
- 終了時間
- Span タグ。キーと値のペアで構成された Span タグのコレクション。キーは文字列でなければなりません。値は文字列、ブール値、または数値にすることができます。
- Span log。Span ログのコレクション。各ログ操作には、キー、値のペア、及びタイムスタンプが 1 つずつ含まれます。キーは文字列でなければなりません。値は任意のデータ型にすることができます。 ただし、OpenTracing をサポートするすべての Tracer が すべてのデータ型をサポートする必要があるわけではないことにご注意ください。
- SpanContext (Span 内容)
- References (Span 間の関係)。 関連しているゼロまたは複数の Span (Span の間では、SpanContext を通じて関係を確立します)。

各 SpanContext には、次のステータスが含まれます。

- いかなる OpenTracing 実装も、一意の Span に基づいてプロセスの境界を越えて現行の呼び出しチェーンのステータス (Trace ID と Span ID など) を送信する必要があります。
- Baggage items (Trace に付属するデータ)。Trace に保存されているキーと値のペアのコレクションであり、プロセスの境界を越えて送信する必要があります。

OpenTracing データモデルの詳細については、OpenTracing のセマンティック標準をご参照ください。

関数の実装

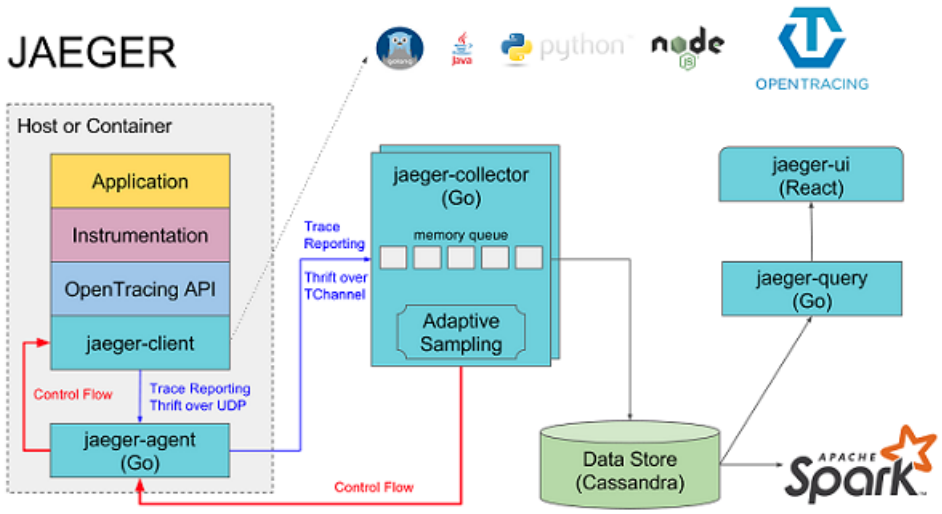
サポートされている Tracer の実装にはすべての OpenTracing の実装が一覧表示されています。Jaeger と Zipkin の実装が最も一般的です。

Jaeger

Jaeger は、Uber によりリリースされたオープンソースの分散トレースシステムです。OpenTracing API と互換性があります。

アーキテクチャ

図 7-33 : アーキテクチャ



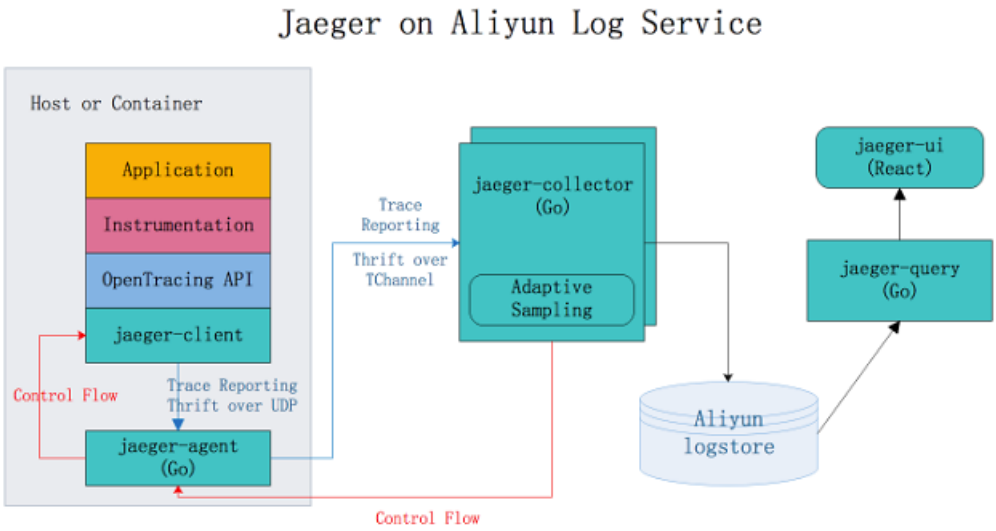
上図に示すように、Jaeger は次のコンポーネントで構成されています。

- Jaeger client: 異なる言語の OpenTracing 標準に準拠した SDK を実装します。 アプリケーションは API を使用してデータを書き込みます。 Client Library は、 アプリケーションで指定されたサンプリングポリシーに従って Trace 情報を jaeger-agent に送信します。
- Agent: UDP ポートが受信した Span データをモニタリングし、そのデータをまとめて collector に送信するネットワークデーモン。 Agent は基本コンポーネントとして設計されており、すべてのホストにデプロイされています。 Agent は Client Library と collector を分離し、 Client Library を ルータ、および Collector 詳細の検出から遮断します。
- Collector: jaeger-agent によって送信されたデータを受信してから、そのデータをバックエンドストレージに書き込みます。 Collector はステートレスコンポーネントとして設計されています。 そのため、任意の数の jaeger-collectors を同時に 実行できます。
- Data store: バックエンドストレージは、Cassandra と Elasticsearch へのデータ書き込みをサポートするプラグイン可能なコンポーネントとして設計されています。
- Query: クエリリクエストを受信し、バックエンドストレージシステムから Trace 情報を取得して UI に表示します。 Query はステートレスで、複数のインスタンスを起動できます。 インスタンスを Nginx などの ロードバランサの後にデプロイできます。

Jaeger on Alibaba Cloud Log Service

Jaeger に基づいています。これにより、収集したトレースデータを Log Service に永久保存し、ネイティブ Jaeger インターフェイスを使用してデータをクエリおよび表示できます。

図 7-34 : Jaeger



利点

- ネイティブ Jaeger は Cassandra と Elasticsearch へのデータの永久保存のみをサポートします。ユーザーはバックエンドストレージシステムの安定性を保守し、ストレージ容量を調整する必要があります。 Jaeger on Alibaba Cloud Log Service は、 Log Service の大量データ処理機能を利用しているため、ユーザーはバックエンドストレージシステムの問題を意識することなく、分散トレース機能の利便性を享受できます。
- Jaeger UI は、クエリおよび Trace 表示機能のみを提供し、問題分析およびトラブルシューティングを十分に行うことができません。Jaeger on Alibaba Cloud Log Service を使用することで、Log Service の強力なクエリおよび分析機能を利用して、システム内の問題を迅速に分析することができます。
- バックエンドストレージに Elasticsearch を使用する Jaeger と比較し、Log Service は従量課金に対応しているため、そのコストは Elasticsearch のコストのわずか 13% となります。詳細については、「自己構築 ELK vs Log Service (SLS)」をご参照ください。

手順

詳細については、 GitHub をご参照ください。

設定例

HotROD は、複数のマイクロサービスで構成されたアプリケーションで、OpenTracing API を使用して Trace 情報を記録します。

Alibaba Cloud Log Service で Jaeger を使用して HotROD の問題を診断するには、次の手順に従ってください。 ビデオには次の内容が含まれます。

1. Log Service の設定
2. Jaeger を起動 (docker-compose コマンドを実行)
3. HotROD の実行
4. Jaeger UI で Trace 情報の取得
5. Jaeger UI での詳細な Trace 情報表示
6. Jaeger UI でのアプリケーションパフォーマンスのボトルネック特定
7. Log Service コンソールでのアプリケーションパフォーマンスのボトルネック特定
8. アプリケーションの OpenTracing API 呼び出し

構成チュートリアル

http://cloud.video.taobao.com//play/u/2143829456/p/1/e/6/t/1/50081772711.mp4

例では、次のクエリステートメントを使用します。

- 1 分ごとにフロントエンドサービスの HTTP GET / dispatch 操作の平均待ち時間およびリクエスト数をカウント

```
process . serviceNam  e : " frontend "  and   operationN  ame : "
HTTP   GET  / dispatch " |
select   from_unixt  ime (  __time__  -  __time__  %  60 )  as
time ,
truncate ( avg ( duration )/ 1000 / 1000 )  as   avg_durati  on_ms
,
count ( 1 )  as   count
group   by   __time__  -  __time__  %  60   order   by   time
desc   limit   60
```

- 2 つの Trace 操作の所要時間を比較

```
traceID : " trace1 "  or   traceID : " trace2 " |
select   operationN  ame ,
( max ( duration )- min ( duration ))/ 1000 / 1000   as
duration_d  iff_ms
group   by   operationN  ame
order   by   duration_d  iff_ms   desc
```

- 待ち時間が 1.5秒 を超える Trace の IP アドレスをカウント

```
process . serviceNam  e : " frontend "  and   operationN  ame : "
HTTP   GET  / dispatch "  and   duration  >  1500000000  |
select  " process . tags . ip "  as   IP ,
```

```
truncate ( avg ( duration )/ 1000 / 1000 )  as   avg_durati  on_ms
,
count ( 1 )  as   count
group   by  " process . tags . ip "
```

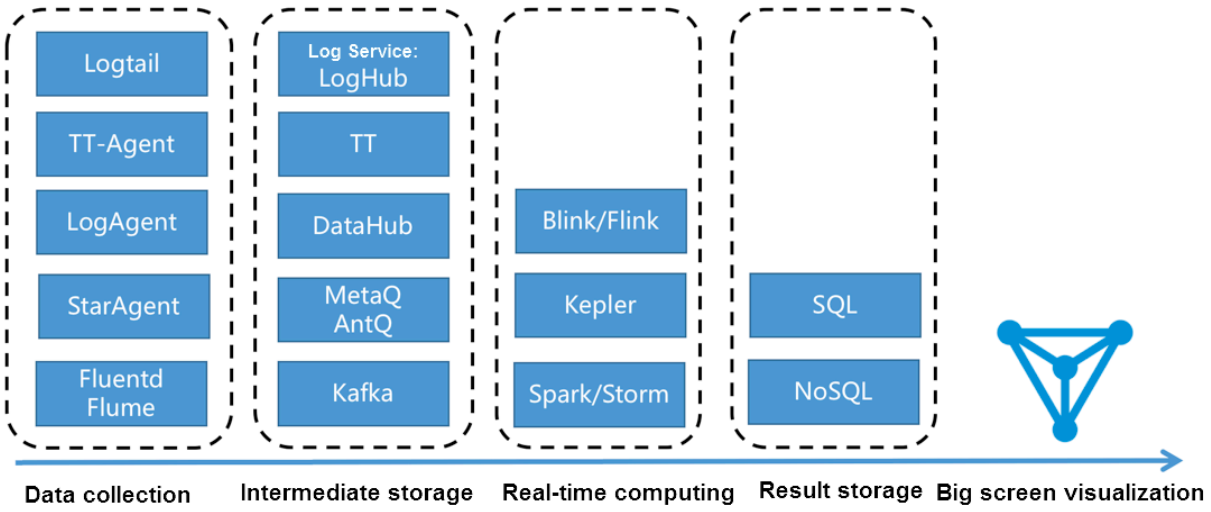
7.3.4 DataV と連携

中国の 11 月 11 日のショッピングキャンペーンと言えば、優れた Tmall リアルタイム大画面が想起されます。また、リアルタイムの大画面と言えば、最も典型的なストリームコンピューティングアーキテクチャが想起されます。

- データ収集: 各ソースからリアルタイムにデータ収集
- リアルタイムコンピューティング: コンピューティングルールを使用して、リアルタイムデータを取得し、期間内のデータを計算します。処理の要に当たります。
- 結果の保存: SQL および NoSQL データベースの処理結果を保存
- 可視化: API 呼び出しで結果を表示

Alibaba グループには、上記の処理を実行する高機能なプロダクトが多数用意されております。よく使用されているプロダクトは、下図のとおりです。

図 7-35 : 関連プロダクト



上記のソリューション以外に、Log Service の照会/分析 API を使用して DataV を連携させて、データを大画面表示させることもできます。

図 7-36 : Log Service + DataV

Log Service のリアルタイムログ分析機能 (照会/分析) は、2017 年 9月に強化されています。照会および SQL92 構文を使用してログをリアルタイムに分析できるようになっています。Log Service には埋め込みダッシュボードのほか、Grafana や Tableau (JDBC) と連携させて解析結果を可視化することもできます。

機能の特徴

一般的に、データ量、リアルタイム性、およびビジネス要件より、次の 2 つのコンピューティングモードに分けられます。

- リアルタイムコンピューティング (ストリームコンピューティング): 固定コンピューティング + 可変データ
- オフラインコンピューティング (データウェアハウス + オフラインコンピューティング): 可変コンピューティング + 固定データ

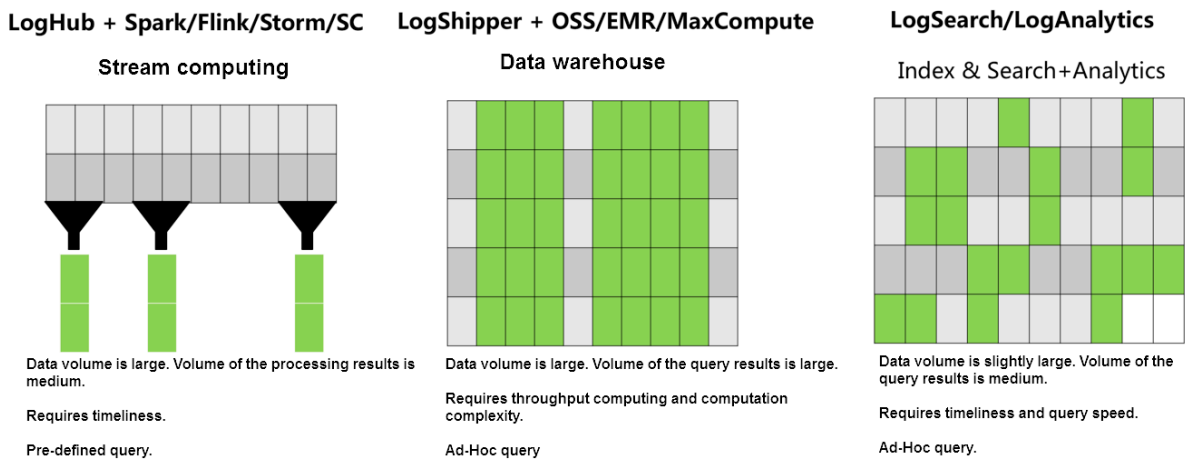
Log Service でのリアルタイムなデータ収集には、2 つの接続方法があります。また、ログ分析シナリオではリアルタイム性が問わますが、LogHub データはリアルタイムにインデックス作成されます。インデックスが作成されたら、さらに、「ログの照会/分析」よりデータを直接照会/分析することができます。この方法には次のような利点があります。

- 高速: API でクエリが引き渡されるとすぐに結果が得られます。待ち時間がなく、、または事前に計算することなく、結果を事前に計算する必要はありません。
- リアルタイム: 生成されたログを 1 秒以内に大画面表示 (99.9 %)
- 動的: 統計方法の変更やデータが補足された場合でも、亢進による再計算結果はリアルタイムに表示されます。

しかし、万能なコンピュータシステムはこの世にはありません。 この機能には、次の制限があります。

- データ量: 一度に処理可能なデータは最大 100 億 GB。データ量を超える場合は制限時間を設定する必要があります。
- コンピューティングの柔軟性: 現在、SQL92 構文のみ処理可能。カスタム化 UDF には非対応。

図 7-37 : Log Service の利点



構成プロセス

操作のデモンストレーション

Log Service データを DataV と連携させる手順は、次のとおりです。

1. データを収集します。 Log Service のデータソースを設定する方法は、5分でクイックスタートをご参照ください。
2. インデックスを設定するにはインデックスの設定及び可視化、またはベストプラクティスの Web サイトログ分析をご参照ください。
3. DataV プラグインと相互接続し、SQL ステートメントを使用して検索されたリアルタイム結果をビューに変換します。

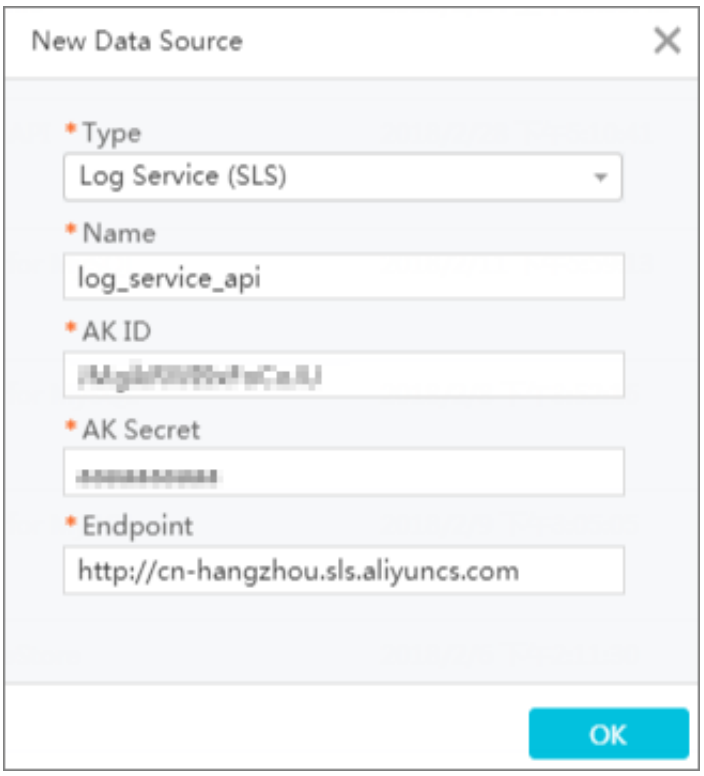
ステップ 1 と 2 を完了すると、検索ページで生ログを表示できます。 本ドキュメントでは、主にステップ 3 の実行方法について説明します。

手順

ステップ 1 DataV データソースの作成

左側のナビゲーションメニューで データソースをクリックします。 ソースを追加をクリックします。 新しいデータソースダイアログボックスが表示されます。データソースの基本情報を入力します。 設定項目は下表のとおりです。

図 7-38 : データの作成



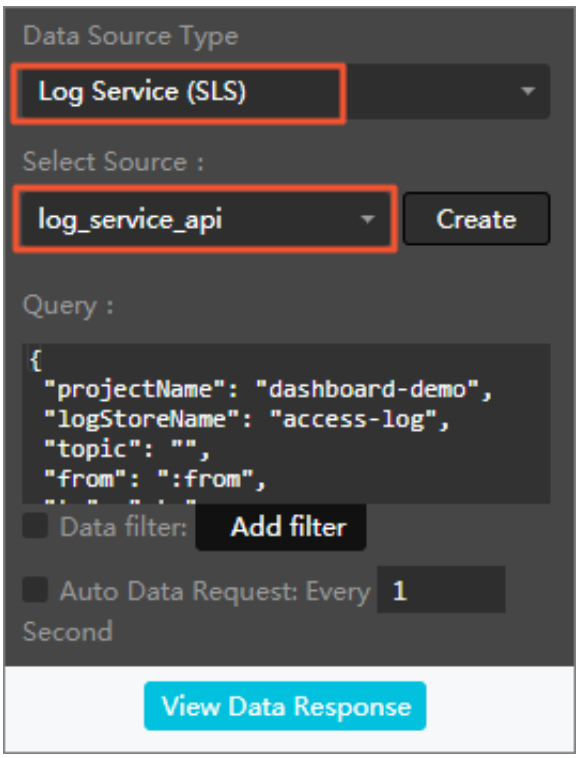
構成項目	説明
タイプ	Log Service (SLS) を選択します。
名前	データソースの名前を設定します。
AK ID	メインアカウントの AccessKey ID、または Log Service を読み込む権限を持つサブアカウントの AccessKey ID。
AK Secret	メインアカウントの AccessKey Secret、または Log Service を読み込む権限を持つサブアカウントの AccessKey Secret。
エンドポイント	Log Service プロジェクトが存在するリージョンのアドレス。 上図では、杭州リージョンのアドレスが入力されています。

ステップ 2 折れ線グラフの作成

1. 折れ線グラフを作成します。

   折れ線グラフのデータ構成で、データソースタイプを Log Service (SLS) に設定し、前のステップで作成したデータソース log_service_api を選択し、クエリテキストボックスにパラメーターを入力します。

   図 7-39 : データソース



   クエリパラメーターの例は次のとおりです。パラメーターを次の表に示します。

```
{
 " projectNam  e ": " dashboard - demo ",
 " logStoreNa  me ": " access - log ",
 " topic ": "",
 " from ": ": from ",
 " to ": ": to ",
 " query ": "*|  select   approx_dis  tinct ( remote_add  r )  as
   uv  , count ( 1 )  as   pv  ,  date_forma  t ( from_unixt  ime
 ( date_trunc (' hour ', __time__ ) ) ,'% Y /% m /% d  % H :% i :%
 s ')  as   time   group   by   time   order   by   time   limit
 1000 " ,
 " line ":  100 ,
 " offset ":  0
}
```

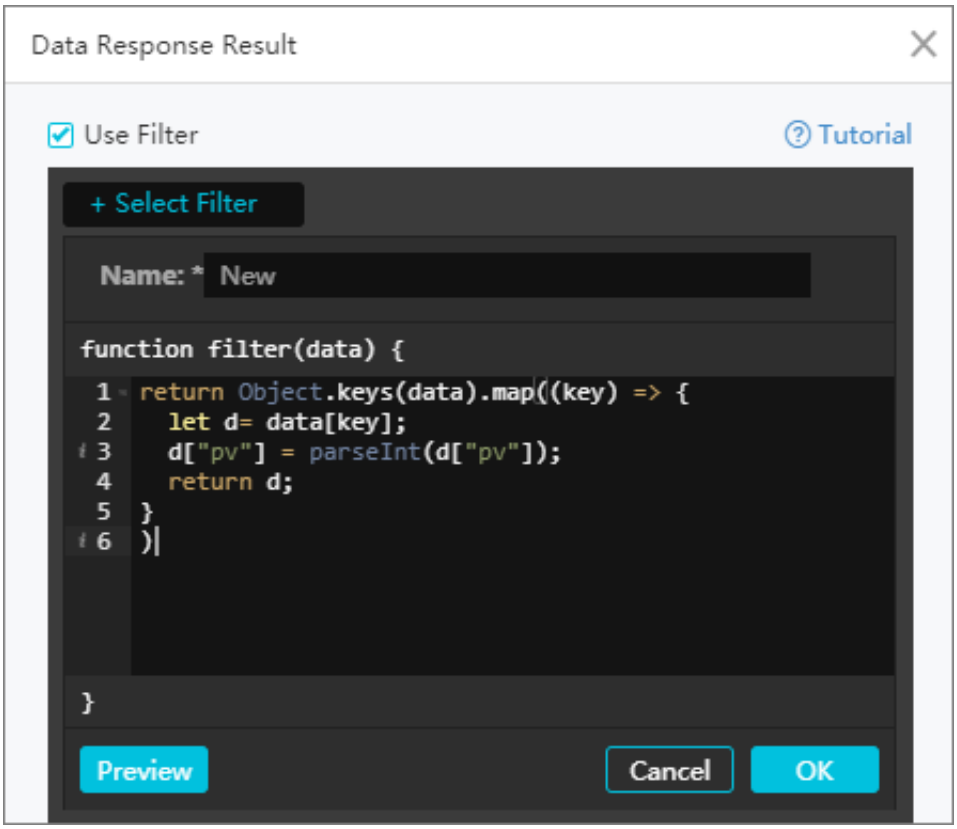
構成項目	説明
projectName	プロジェクト名

構成項目	説明
logstoreName	Logstore 名
topic	ログトピック。 トピックを設定しない場合は、パラメータ値を空のままにします。
from、to	ログの開始時刻と終了時刻をそれぞれ指定します。  注： 上記の例では、各パラメータは次のように設定されています。`: from` と `: to` 。 テスト中に、UNIX で時刻を入力できます 形式の例：1509897600。 リリース後に、時間を `: from` および `: to` に変換し、URL パラメータで特定の時間範囲を設定します。 たとえば、プレビューされた URL は `http :// datav . aliyun . com / screen / 86312` です。 `http :// datav . aliyun . com / screen / 86312 ? from = 1510796077 & to = 1510798877` を開くと、指定した時間に基づいて値が計算されます。
query	クエリ条件。 上記の例での検索条件は、1分当たりの PV です。クエリ構文の詳細は、分析文法をご参照ください。  注： クエリに記述する時間の書式は 2017/07/11 12:00:00 にします。時間の書式を変換する方法は、 `date_forma t ( from_unixt ime ( date_trunc (' hour ', __time__ ) ) ,'% Y /% m /% d % H :% i :% s ')`
line	デフォルト値の 100 のままにします。

構成項目	説明
offset	デフォルト値の 0 のままにします。

構成後、データレスポンスを表示をクリックします。

図 7-40 : データレスポンスを表示



2. フィルターを作成します。

   データレスポンスを表示をクリックすると、データレスポンス結果ダイアログボックスが表示されます。フィルターを使用を有効にし、フィルターの選択、新規フィルターの順にクリックしてフィルターを作成します。

   次の形式でフィルタの内容を入力します。

```
 return   Object . keys ( data ). map (( key ) => {
 let   d =  data [ key ];
 d [" pv "] =  parseInt ( d [" pv "]);
 return   d ;
}
)
```

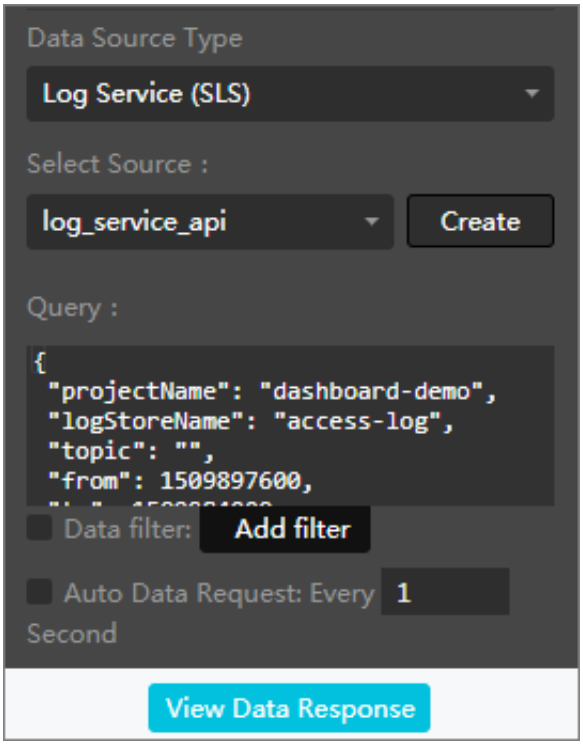
   フィルターで、y 軸で使用される結果を int 型に変換します。 例では、y 軸は pv を示します。 したがって、pv 列を変換する必要があります。

   結果には、t 列と pv 列の両方が含まれています。 x 軸を t に、y 軸を pv に設定できます。

ステップ 3 円グラフの構成

1. カルーセル円グラフを作成します。

   図 7-41 : 検索テキストボックス



   検索テキストボックスに次の内容を入力します。

```
{
 " projectNam  e ": " dashboard - demo ",
 " logStoreNa  me ": " access - log ",
 " topic ": "",
 " from ":  1509897600 ,
 " to ":  1509984000 ,
 " query ": "*|  select   count ( 1 )  as   pv  , method   group
 by   method " ,
 " line ":  100 ,
 " offset ":  0
}
```

   検索中に、さまざまなメソッドの比率を計算できます。

2. フィルタを追加して、フィルタに次のような内容を入力します：

```
 return   Object . keys ( data ). map (( key ) => {
 let   d =  data [ key ];
 d [" pv "] =  parseInt ( d [" pv "]);
 return   d ;
}
```

```
)
```

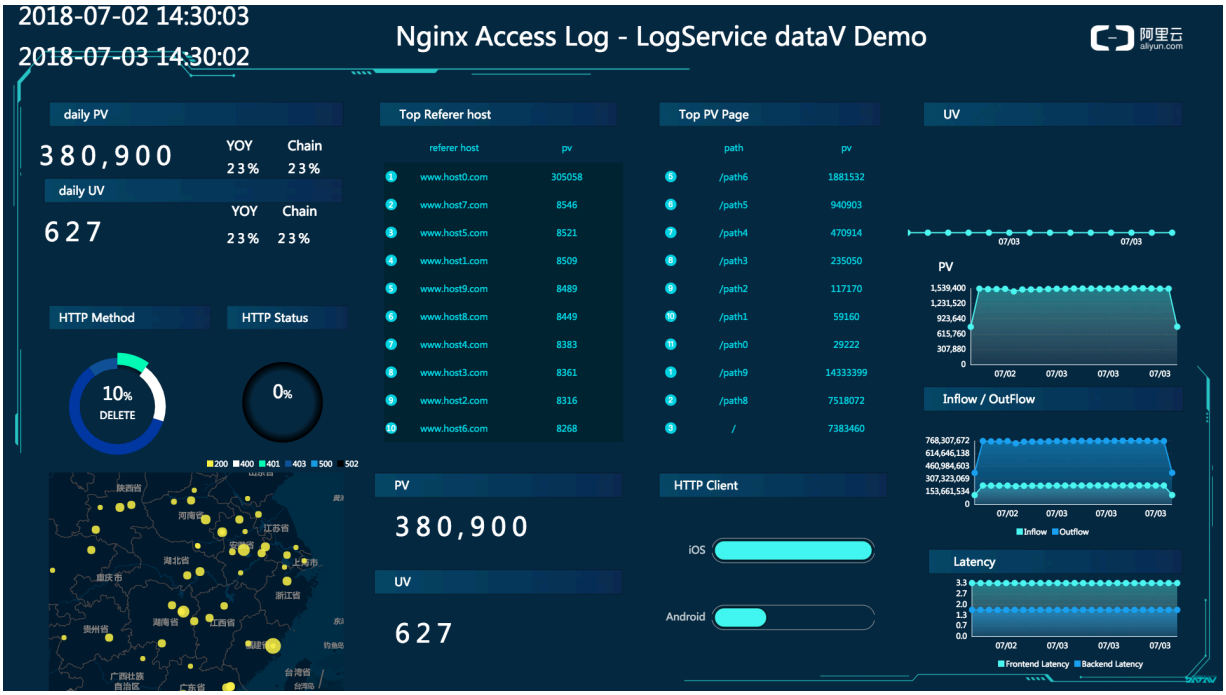
円グラフの type テキストボックスに method と入力し、value テキストボックスに pv と入力します。

ステップ 4 プレビューとリリース

大画面を作成するには、プレビューして公開をクリックします。 開発者とビジネス担当者は、11/11 ショッピングキャンペーンで、ビジネスアクセス状況をリアルタイムに確認できます。

お試しデモ。 URL の from および to パラメータの値はいつでも設定できます。

図 7-42 : リアルタイム画面



ユースケース: 統計基準の下でリアルタイムの大画面を継続的に調整する

たとえば、Computing Conference 中は、中国全体のオンライン (Web サイト) トラフィックをカウントするという一時的な要件が発生します。Log Service ですべてのログデータ収集が設定され、照会/分析が有効になります。 したがって、クエリ条件さえ入力すれば良いです。

1. たとえば、UV をカウントするには、10 月 11 日から現在までのすべてのアクセスログから、Nginx の下のforward フィールドの一意のカウントを取得する必要があります。

```
* |  select   approx_dis  tinct ( forward )  as   uv
```

2. システムが1日間オンラインになった後、要件が変更されます。 現在、ドメイン yunqi の下のデータのみをカウントすることになりました。リアルタイム検索にフィルタ条件（ホスト）を追加できます。

```
host : yunqi . aliyun . com  |  select   approx_dis  tinct (
forward )  as   uv
```

3. Nginx のアクセスログに複数の IP アドレスが含まれていることが検出されました。 デフォルトでは、最初の IP アドレスだけが必要となります。 したがって、検索で検索条件を処理します。

```
host : yunqi . aliyun . com  |  select   approx_dis  tinct (
split_part ( forward ,',', 1 ))  as   uv
```

4. 3日目の要件によると、アクセスコンピューティングからuc の中の広告アクセスを削除する必要があります。 この場合、フィルタ条件 not ... を追加することで、最新の結果をすぐに取得できます。

```
host : yunqi . aliyun . com   not   url : uc - iflow  |  select
approx_dis  tinct ( split_part ( forward ,',', 1 ))  as   uv
```

7.3.5 Grafana との相互接続

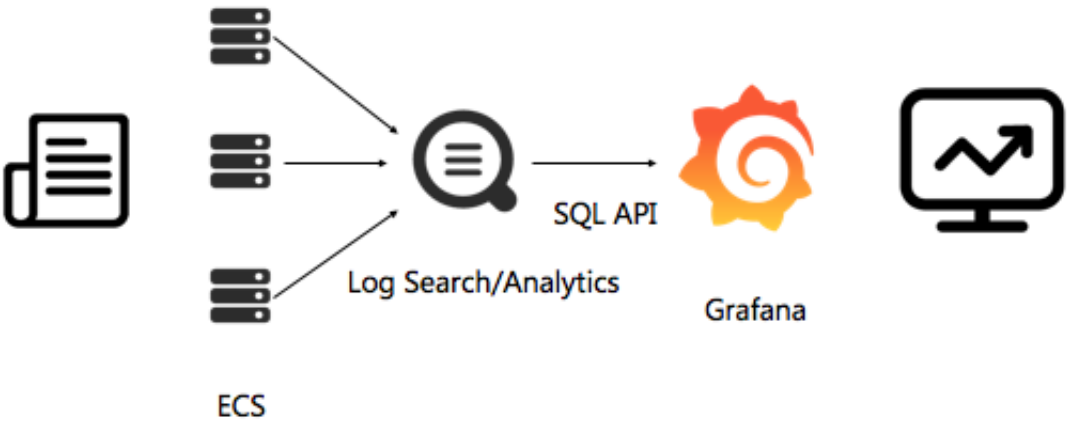
Log Service は、ログデータのオールインワンサービスです。ユーザーをデータ収集、ストレージの相互接続、インデックス作成といった仕事から解放し、Log Service を照会すれば分析に専念できるようになります。 2017 年 9 月に Log Service のインデックス作成機能がアップグレードされ、query + SQL-92 構文でリアルタイムにログ分析ができるようになりました。

ダッシュボードのほか、DataV、Grafana、Tableau、および Quick と連携して、解析結果を可視化することもできます。本ドキュメントでは、Grafana を例に Log Service で Nginx ログを解析/可視化する方法について説明します。

プロセス構造

ログ収集から分析までの流れは以下のとおりです。

図 7-43 : プロセス構造



手順

1. ログデータの収集。詳しい手順については、5 分でクイックスタートをご参照ください。
2. インデックス設定とコンソールで照会するための設定、 詳細な手順については、インデックスの作成、分析 - Nginx アクセスログ、または、ログ分析例をご参照ください。
3. Grafana プラグインをインストールすと、リアルタイムに SQL クエリ結果がビューに反映されます。

ステップ 1 と 2 を終えると、クエリページで生ログが表示されます。

本ドキュメントでは、手順 3 を詳述します。

手順

1. Grafana をインストール
2. Log Service プラグインをインストール
3. ログデータソースを指定

4. ダッシュボードを追加
   a. テンプレート変数を設定
   b. PV/UVを設定
   c. 送受信ネットワーク帯域幅を設定
   d. HTTP メソッドの割合
   e. HTTP ステータスコードの割合
   f. トップソースのページ
   g. 最大待ち時間のページ
   h. トップページ
   i. 応答ステータスコード 200 以外の多かったページ
   j. フロントエンドとバックエンドの平均待ち時間
   k. クライアント統計
   l. ダッシュボードの保存と公開
5. 結果を表示

1.Grafana をインストール

詳細なインストール手順については Grafana official document をご参照ください。

Ubuntu へのインストールコマンドは次のとおりです。

```
wget   https :// s3 - us - west - 2 . amazonaws . com / grafana -
releases / release / grafana_4 . 5 . 2_amd64 . deb
sudo   apt - get   install   - y   adduser   libfontcon   fig
sudo   dpkg   - i   grafana_4 . 5 . 2_amd64 . deb
```

円グラフを使用するには、円グラフのプラグインをインストールします。詳細なインストール手順については Grafana official document をご参照ください。

次のコマンドを実行します。

```
grafana - cli   plugins   install   grafana - piechart - panel
```

2.Log Service プラグインをインストール

Grafana プラグインのディレクトリを確認します。 Ubuntu プラグインは`/ var / lib / grafana / plugins /`にあります。 プラグインをインストール後に、Grafana サーバーを再起動します。

システムが Ubuntu の場合、プラグインをインストールし、Grafana サーバーを再起動するコマンドは次のとおりです。

```
cd   / var / lib / grafana / plugins /
```

```
git   clone   https :// github . com / aliyun / aliyun - log -
grafana - datasource - plugin
service   grafana - server   restart
```

3.ログデータソースを設定

ローカルマシンにインストールする場合、インストールに使用するポートは 3000 に初期設定されています。ブラウザでポート 3000 を開きます。

1. 左上隅の Grafana のロゴをクリックします。表示されるダイアログボックスでデータソースを選択します。
2. Add data sourceをクリックし、ログのグラフ分析に Grafana と Alibaba Cloud の Log Service を選択します。
3. 新しいデータソースの構成項目を指定します。

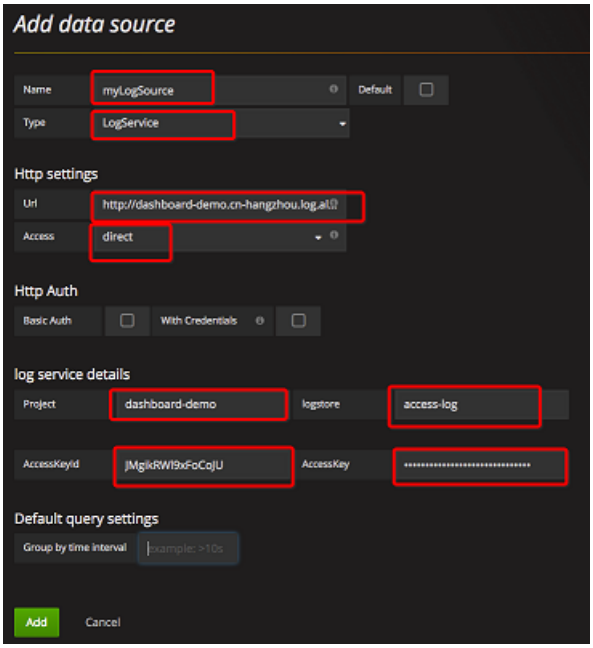
   構成には下表の各項目があります。

設定項目	設定内容
データソース	名前は指定します。タイプはLogService。
HTTP 設定	URL の例: `http :// dashboard - demo . cn - hangzhou . log . aliyuncs . com` 。 `dashboard - demo` (プロジェクト名) および `cn - hangzhou . log . aliyuncs . com` (プロジェクトのリージョン) は、環境に合わせてご変更ください。アクセス方法は、「直接」または「プロキシ」を選択します。
HTTP 認証	初期値をそのままご使用ください。

設定項目	設定内容
Log Service 詳細	読み取り権限のあるプロジェクト、Logstore、および AccessKey を入力して Log Service の詳細を設定します。プライマリアカウントおよびサブアカウントに AccessKey があります。

設定例

図 7-44 : 設定例



設定が完了したら、Addをクリックしてデータソースを追加します。次に、ダッシュボードを追加します。

4.ダッシュボードを追加

クリックして左上隅のメニューを開き、Dashboardsを選択し、 Newをクリックします。 左上のメニューに新しいダッシュボードが追加されます。

4.1 テンプレート変数を設定

Grafana にテンプレート変数を設定して、変数の値に応じてビューを表示できるようにします。本ドキュメントでは、時間間隔と各ドメインの設定方法について説明します。

1. ページ上部のsetアイコン、templateを順にクリックします。

2. 同ページに、設定されたテンプレート変数が表示されます。 新しいテンプレートを作成するには、newをクリックします。 まず、時間間隔を設定します。

   変数名は設定したものにします。例では「$myinterval」とします。クエリ条件には`$ myinterval`を記述します。設定については下表をご参照ください。

構成項目	構成内容
名前	変数名 (例: 「myinterval」)
タイプ	間隔を選択
ラベル	時間間隔を入力
選択肢	値に`1m , 10m , 30m , 1h , 6h , 12h , 1d , 7d , 14d , 30d`と入力

3. ドメイン名テンプレートを設定します。

   通常、1 つの VPS に複数のドメインがマウントされています。各ドメインの状況をひとつひとつ確認しなければなりません。 テンプレート値を、`*`、`www . host . com`、`www . host0 . com`、`www . host1 . com`と入力すればすべてのドメインが表示され、`www . host . com`, `www . host0 . com`または`www . host1 . com`とそれぞれ入力すればそれぞれのアクセス状況が表示されます。

   ドメイン名テンプレートの設定項目は、下表のとおりです。

構成項目	構成内容
名前	変数名 (例: 「hostname」)
タイプ	カスタムを選択
ラベル	ドメイン名を入力
選択肢	値に`*, www . host . com , www . host0 . com , www . host1 . com`を入力

   設定が完了すると、設定されたテンプレート変数がダッシュボードページの上部に表示されます。ドロップダウンボックスから任意の値を選択します。 例えば、時間間隔として以下の値を選択できます。

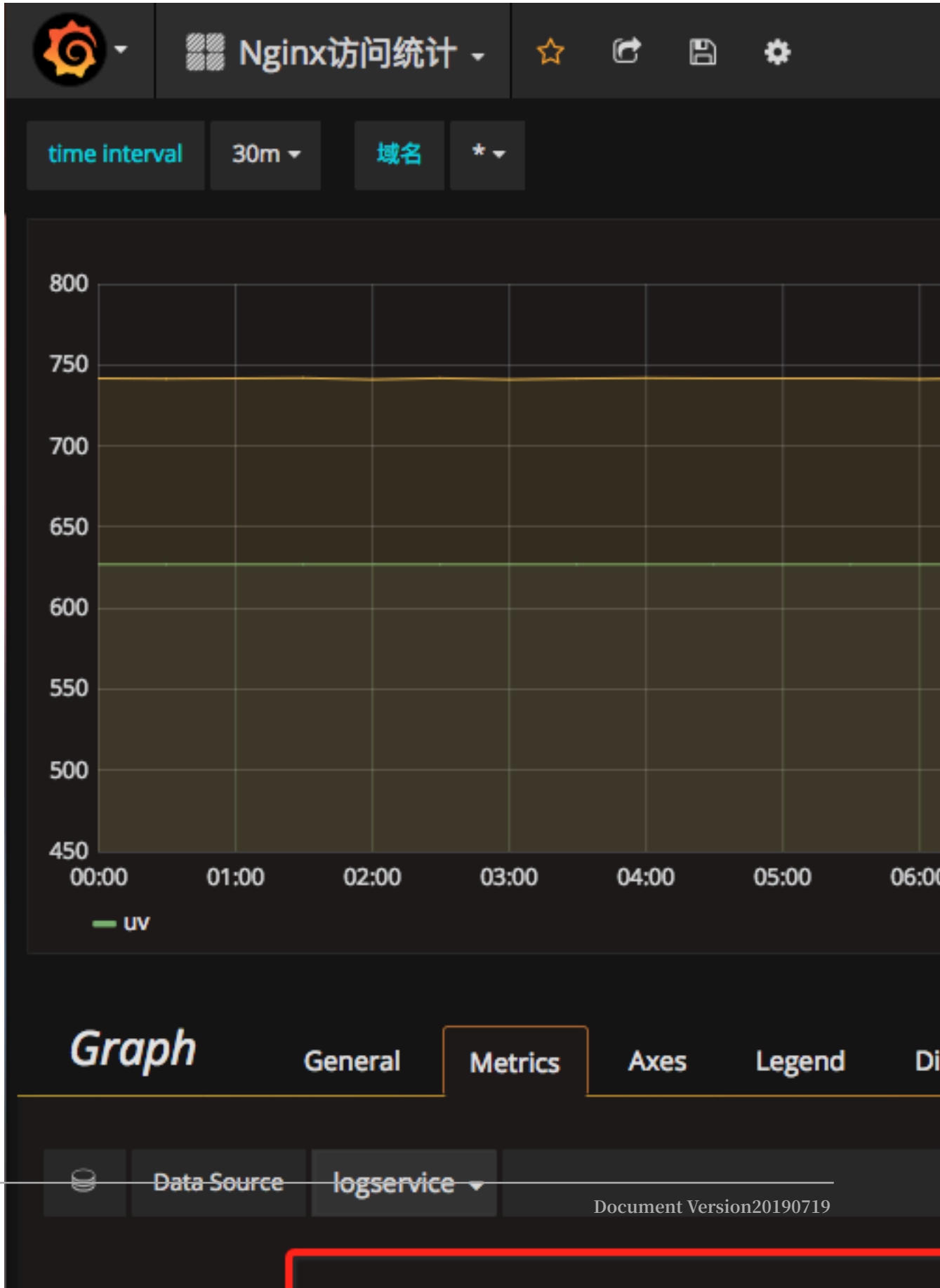
4.2 PV/UV を設定

1. 左側のAdd ROWをクリックして、新たな行を作成します。すでに行があれば、左側のメニューよりAdd Panelを選択します。

2. Grafana にはさまざまなビューが用意されています。PV および UV データの場合は、Graphビューを作成します。
3. Panel Titleをクリックして表示されるメニューよりEditをクリックします。

4. メトリック設定の、データソースには `logservice` を選択し、クエリ、Y 軸、X 軸を設定します。

図 7-45 : PV/UV の設定

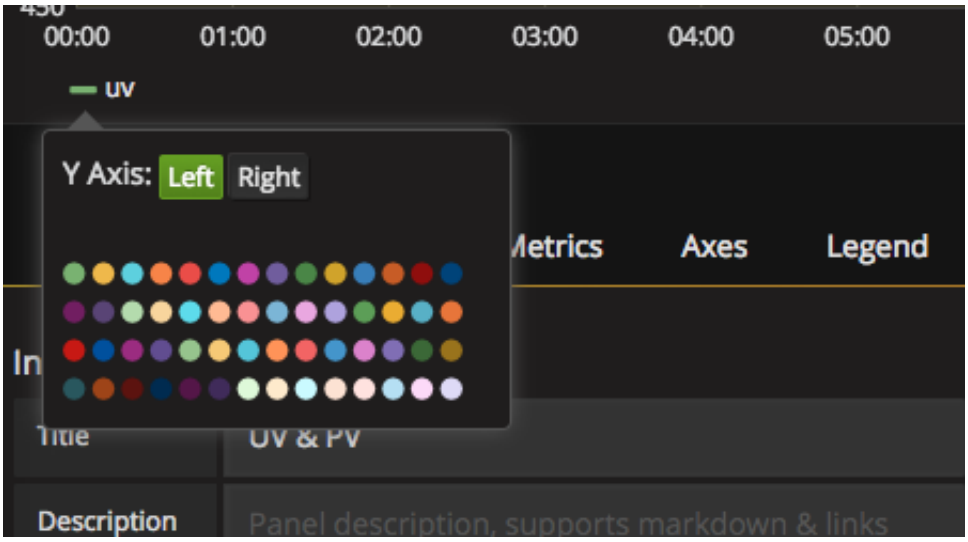


5. データソースのドロップダウンボックスより、`logService`を選択します。

構成項目	構成内容
Query	`$ hostname | select approx_distinct(remote_addr) as uv, count(1) as pv, __time__ - __time__ % $$myinterval as time group by time order by time limit 1,000` $hostnameは、任意のドメイン名に置き換えます。$$myintervalは時間間隔に置き換えられます。「myinterval」の前には「$」が2つあり、「hostname」の場合は1つしかないことにご注意ください。
X-Column	時間
Y-Column	UV、PV

UVとPVの値はかけ離れているため、Y軸を2つ表示します。UVの左側の色付きの線をクリックし、UVを左右のいずれのY軸にするかを指定します。

図 7-46 : Y軸の表示



タイトルは「Panel Title」に初期設定されています。変更するには、「基本」タブに移動し、タイトル設定に新しいタイトル(例:`PV & UV`)を入力します。

4.3 送受信ネットワーク帯域幅を設定

送受信のネットワーク帯域幅は4.2 PV/UV を設定と同様に追加します。

主な設定項目は次のとおりです。

構成項目	構成内容
Query	`$ hostname | select sum ( body_byte_ sent ) as net_out , sum ( request_le ngth ) as net_in , __time__ - __time__ % $$ myinterval as time group by __time__ - __time__ % $$ myinterval limit 10000`
X-Column	時間
Y-Column	net_in、net_out

4.4 HTTP メソッドの割合

4.2 PV/UV を設定同様に HTTPメソッドの割合を設定します。

新たに行を作成し、円グラフを選択し、Query、X 軸、Y 軸を設定します。

主な設定項目は次のとおりです：

構成項目	構成内容
Query	`$ hostname | select count ( 1 ) as pv , method group by method`
X-Column	pie
Y-Column	method、pv

4.5 HTTP ステータスコードの割合

4.2 PV/UV を設定と同様に HTTP ステータスコードの割合を設定します。

新しい作成された行に、ビューの円グラフビューを選択します。

主な設定項目は次のとおりです：

構成項目	構成内容
Query	`$ hostname | select count ( 1 ) as pv , status group by status`
X-Column	pie
Y-Column	status、pv

4.6 トップソースのページ

4.2 PV/UV を設定と同じ方法でトップソースのページを設定することができます。

新しい作成された Row の中に、ビューのPie Chartを選択します：

主な設定項目は次のとおりです：

構成項目	構成内容
Query	`$ hostname | select count ( 1 ) as pv , referer group by referer order by pv desc`
X-Column	pie
Y-Column	referer、pv

4.7 最大待ち時間のページ

4.2 PV/UV を設定と同様に最大待ち時間のページを設定します。

URL とテーブル内の待ち時間を表で表示するには、作成時にその表を表に指定します。

主な設定項目は次のとおりです。

構成項目	構成内容
Query	`$ hostname | select url as top_latenc y_url , request_ti me order by request_ti me desc limit 10`
X-Column	空欄のまま
Y-Column	top_latency_url、request_time

4.8 アクセス頻度の高いページ

4.2 PV/UV を設定と同様にアクセス頻度の高いページを追加します。

新しいテーブルビューを作成します。 データソースに Log Service を選択し、クエリ、X軸、およびY軸を指定します。下表をご参照ください。

構成項目	構成内容
Query	`$ hostname | select count ( 1 ) as pv , split_part ( url ,'?', 1 ) as path group by split_part ( url ,'?', 1 ) order by pv desc limit 20`
X-Column	空欄
Y-Column	path、pv

4.9 応答ステータスコード 200 以外の多かったページ

4.2 PV/UV を設定と同様に応答ステータスコード 200 以外の多かったページを追加します。

新しいテーブルビューを作成します。 データソースに Log Service を選択し、クエリ、X軸、およびY軸を指定します。下表をご参照ください。

構成項目	構成内容
Query	`$ hostname not status : 200 | select count ( 1 ) as pv , url group by url order by pv desc`
X-Column	空欄
Y-Column	url、pv

4.10 フロントエンドとバックエンドの平均待ち時間

4.2 PV/UV を設定と同様に平均フロントエンドを追加します。

新しいグラフビューを作成します。データソースに Log Service を選択し、クエリ、X軸、およびY軸を指定します。下表をご参照ください。

構成項目	構成内容
Query	`$ hostname | select avg ( request_ti me ) as response_t ime , avg ( upstream_r esponse_ti me ) as upstream_r esponse_ti me , __time__ - __time__ % $$ myinterval as time group by __time__ - __time__ % $$ myinterval limit 10000`
X-Column	time
Y-Column	upstream_response_time,response_time

4.11 クライアント統計

4.2 PV/UV を設定と同様にクライアント統計を追加します。

新しい円グラフを作成します。データソースに Log Service を選択し、クエリ、X軸、およびY軸を指定します。下表をご参照ください。

構成項目	構成内容
Query	`$ hostname | select count (1) as pv , case when regexp_lik e (http_user_ agent , ' okhttp ') then ' okhttp ' when regexp_lik e (http_user_ agent , ' iPhone ') then ' iPhone ' when regexp_lik e (http_user_ agent , ' Android ') then ' Android ' else ' unKnown ' end as http_user_ agent group by http_user_ agent order by pv desc limit 10`
X-Column	pie
Y-Column	http_user_agent,pv

4.12 ダッシュボードの保存と公開

ページ上部の保存ボタンをクリックして、ダッシュボードを公開します。

5.結果を表示

ダッシュボードのホームページを開き、結果を表示します。参照デモ: デモ

ページの上部で、統計の期間または間隔、または、別のドメイン名を選択します。

以上で Dashboard for Nginx のアクセス統計の設定は完了です。分析にお役立てください。

図 7-47 : 結果を表示

8 アラームと通知

8.1 アラーム機能の概要

本ドキュメントでは、アラームメカニズム、アラーム構成の制限、および一般的なシナリオでアラームで使用されるステートメントについて説明します。 Log Service が提供するアラーム機能を使用して、アラームを作成して、ダッシュボードのグラフに関連付けて、ログに記録されたサービスをリアルタイムでモニタリングできます。

概要

ダッシュボードの特定のグラフにあるデータに基づいてアラームが構成されます。 Log Service コンソールの検索またはダッシュボードページで、アラームを構成できます。 具体的に、アラームをトリガーするための条件とアラーム通知を設定できます。 アラームを構成後、Log Service はダッシュボードのグラフのクエリ結果を指定された間隔でチェックします。 クエリ結果がアラームルールで指定された条件を満たすと、Log Service はアラーム通知を送信します。 詳細は、アラームの構成 をご参照ください。


注：
アラーム機能は Log Service にてご利用できるようになりました。 旧バージョンのアラーム設定情報がまだ保持されていますが、 最新のバージョンへのアップグレードをお勧めします。 詳細は、 最新バージョンへのアラーム構成のアップグレード をご参照ください。

制約

構成項目	説明
アラームに関連付けられるグラフ	各アラームはグラフに関連付ける必要があり、最大 3 つのグラフに関連付けることができます。
条件式	条件式（コンソールではトリガー条件として表示される）の長さは 1 から 128 文字である必要があります。 ・ ステートメントのクエリアウトプットの内の最初の 100 のログ項目だけが分析されて、何らかのログ項目がアラームを起動するようにし設定た条件を満たしているかどうかが判別されます。 ・ 条件は最大 10,000回 の計算に使用できます。
ログエントリ文字	システムは最大 1024 文字のログ項目（ステートメントによるアウトプット）を計算に使用できます。

構成項目	説明
検索期間	各検索および分析ステートメントは、最大期間 24 時間のログデータを検索できます。

アラームに使用されるステートメント

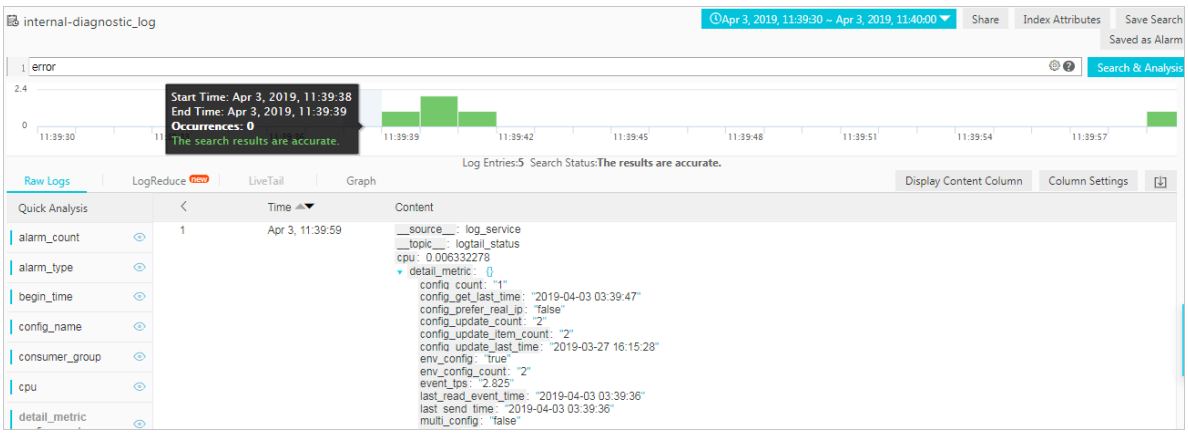
アラームはダッシュボードのグラフのデータに基づいています。 各グラフは、クエリステートメントまたは検索分析ステートメントの検索結果を示しています。

- クエリステートメントを使用すると、クエリステートメントの条件を満たすログエントリがアウトプットされます。
- 検索および分析ステートメントを使用すると、ステートメントの条件を満たすログ項目の統計が収集されてから、それらの統計がアウトプットされます。
- クエリステートメントのアウトプットを対象にしたアラームの構成

  この例では、過去 15 分以内に error という単語を含むログエントリをクエリするために error のクエリステートメントが使用され、システムは 144 個のログエントリをアウトプットします。 各ログエントリは Key と Value のペアで構成されています。 この例では、Key の値に対してアラームを設定できます。

> **注：**
> システムがクエリステートメントに対して 100 を超えるログエントリをアウトプットした場合、アラームは最初の 100 のログエントリのみを分析します。 これは、アラームを起動するための条件を満たし、最初の 100 件のログエントリの中にあるログエントリによってのみ、アラームを起動できることを意味します。

図 8-1 : クエリステートメント



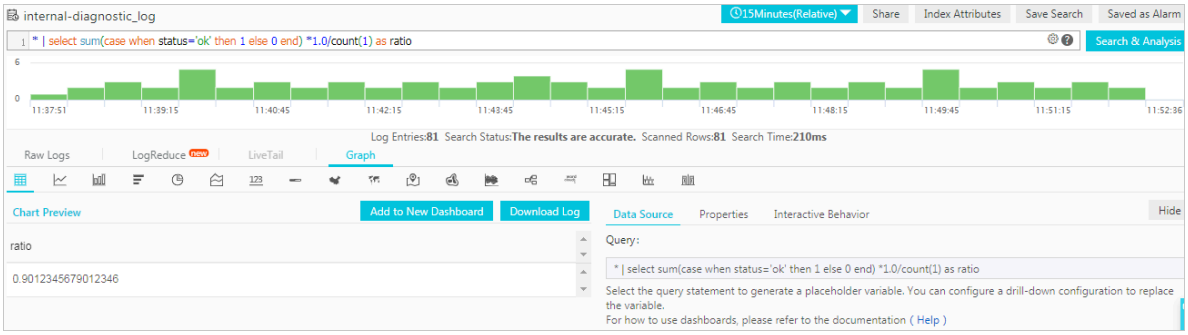
- 検索および分析ステートメントのアウトプットを対象にしたアラームの構成

  この例では、以下の検索および分析ステートメントを使用して、すべてのログ項目の中で OK というステータスコードを持つログ項目の比率を取得します。

```
* | select sum ( case when status =' ok ' then 1 else
 0 end ) * 1 . 0 / count ( 1 ) as ratio
```

> 注：
> 詳細は、クエリ構文 をご参照ください。

図 8-2 : 検索と分析のステートメント



この例では、アラームをトリガーする条件を `ratio < 0 . 9` として設定することでアラームを構成できます。 これは、すべてのログエントリの中で OK というステータスコードを持つログエントリの割合が 90% を下回ると、アラームがトリガーされることを意味します。

8.2 アラームの構成

8.2.1 アラームの設定

基本プロセス

1. クイック照会を設定
2. アラームルールを作成
3. アラーム方法を設定
4. アラームレコードを表示

1 クイック照会を設定

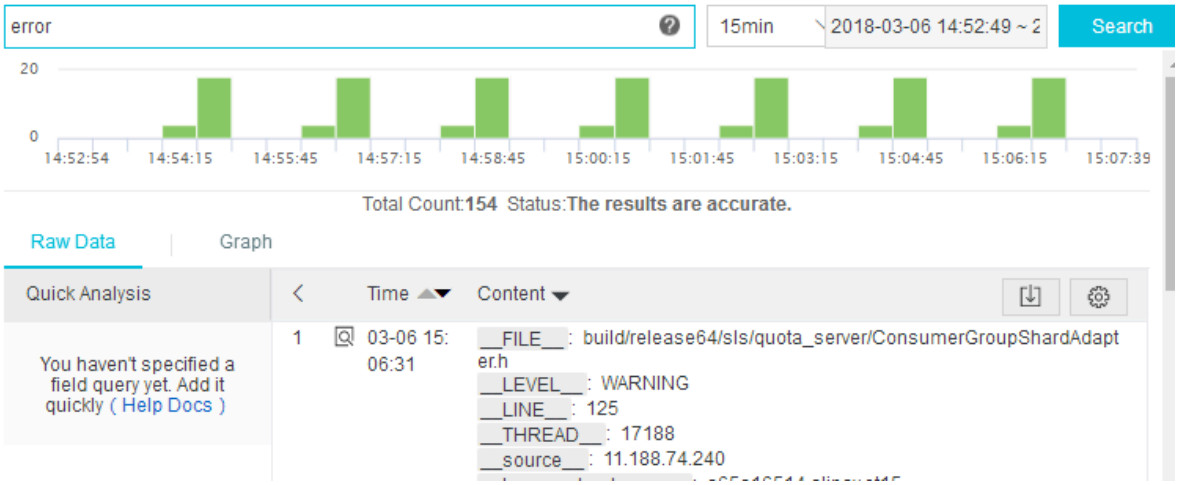
応答結果のモード

ログクエリの結果は、結果または結果の統計の 2 つのモードで表示することができます。結果モードの場合、クエリ条件を満たしたログ数が表示されます。結果の統計モードの場合、指定した時間における、クエリ条件を満たしたログ数の分布が表示されます。

・ 結果

たとえば、「error」の含まれる 15 分以内のデータをクエリするには、条件を「error」にします。合計 154 件のレコードがヒットしたとします。 分布は、下図のようになります。

図 8-3 : Raw ログ



各レコードは、キーと値のペアで構成されます。 キーの値をアラーム条件に指定することができます。

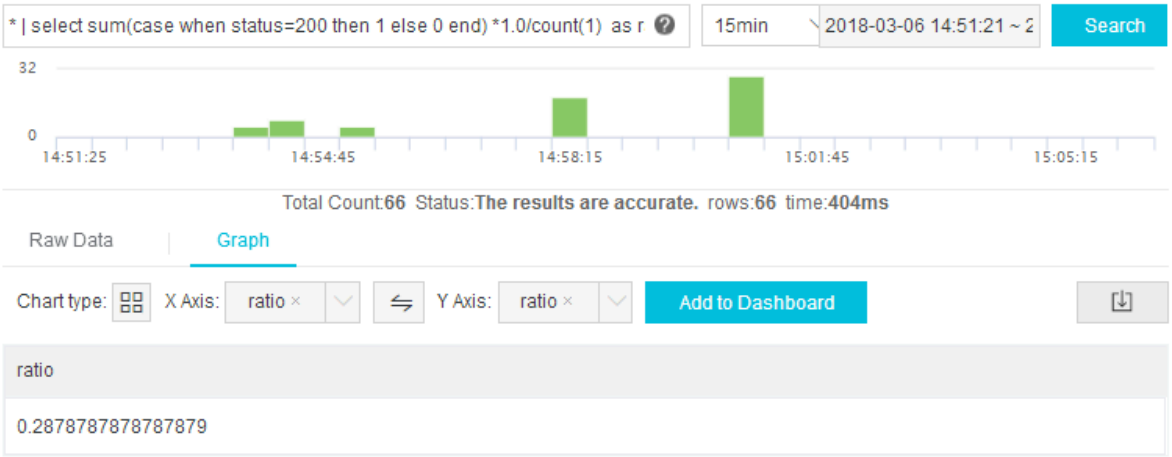
注 :

1 つのクエリの結果が 10 件を超える場合、最初の 10 件のみがアラーム条件の対象になります。 10 件の結果の内、条件を満たすものがあれば、アラームがトリガーされます。

- 結果の統計 (ヒストグラムクエリ)

  すべてのログに対する、ステータスコード 200 のログの比率 (ratio) をクエリする場合のクエリステートメントは次のとおりです (クエリ構文の詳細は、「クエリ構文」を参照)。

  図 8-4 : クエリ結果の統計




```
* | select sum ( case when status = 200 then 1 else 0
  end ) * 1 . 0 / count ( 1 ) as ratio
```

  アラーム条件を `ratio < 0 . 9` にしているため、すべてのログに対して、ステータスコード 200 のログが 90% 未満の場合にアラームは通知されます。

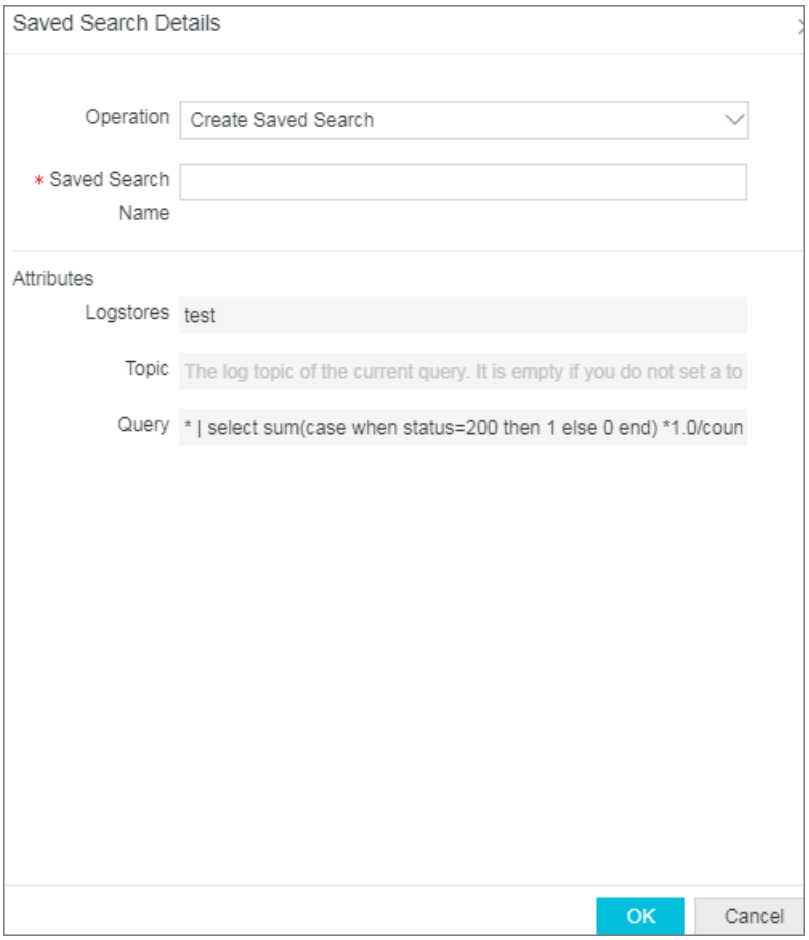
手順

1. Log Service コンソールにログインします。
2. プロジェクトリスト ページで、プロジェクト名をクリックします。
3. Logstore リストページで、Logstore の右側の照会をクリックします。
4. Logstore、トピック、およびクエリステートメントを指定します。 次に、指定したログをクエリします
5. クエリパラメータをクイック照会として保存するには、右上隅のクイック照会として保存 > をクリックします。 パラメータをクイック照会に保存します。

6. 保存したクイック照会を設定して確認をクリックします。

   - 操作: クイック照会を作成を選択します。
   - クイック照会名: クイック照会の名前。

図 8-5 : クイック照会の詳細



2 アラームルールを作成

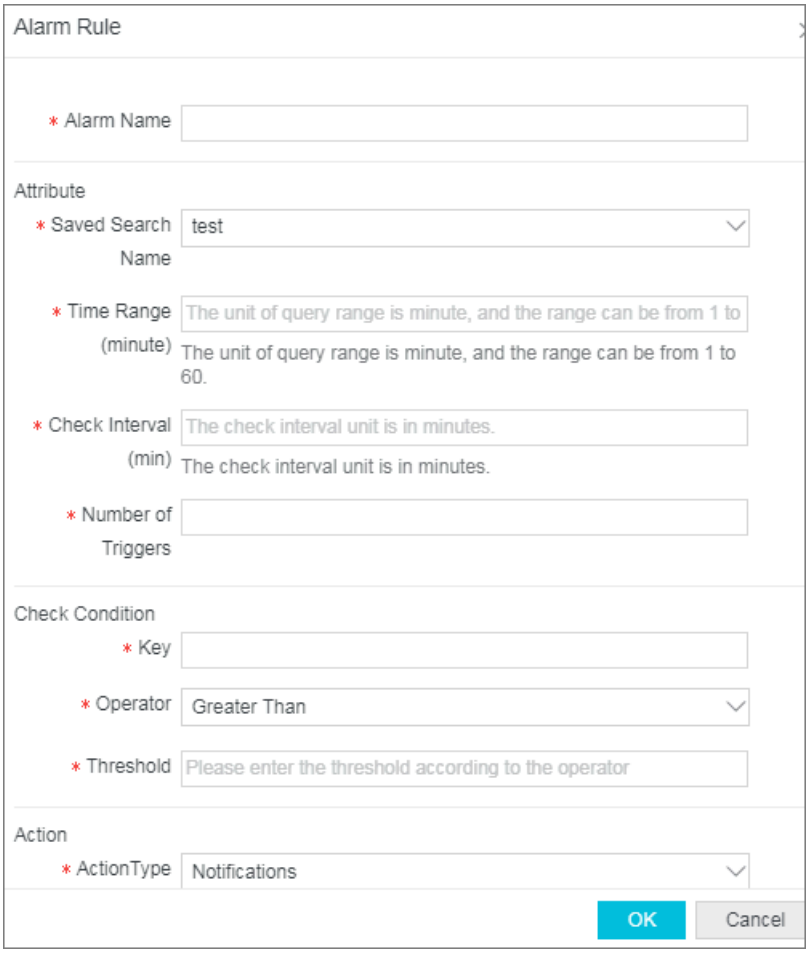
クエリパラメータをクイック照会として保存したら、アラームルールを作成します。

1. 右上隅の新規アラームに保存をクリックします。 アラームの作成ページが表示されます。

2. アラームルールを設定して、次へをクリックします。

   現時点では、サイト内通知または WebHook により、アラーム通知が送信されています。

   図 8-6 : ルールの説明



ルールの説明

- クイック照会名: 作成済みのクイック照会を選択します。
- 期間 (分): サーバーがアラームチェックを実行したときに読み取るデータの期間 (分単位)。 たとえば、値が「1」の場合、1 分以内にアラームチェックの対象となったデータがクエリされます。

  > 注 :
  > 現時点では、期間内の最初の 10 件のデータレコードのみがアラームチェックの対象になります。

- チェックの間隔 (分): サーバーがアラームチェックを実行する時間間隔 (分)。 現在、最小間隔は 5 分です。

- トリガー回数: アラームチェックの連続トリガー回数。 たとえば、チェック間隔を 5 分に指定し、本設定を「2」に指定した場合、 チェックで 2 回連続してアラーム条件を満たすとアラーム通知が送信されます (アラーム間隔は最低 10 分)。
- キー: アラームに使用される、ログコンテンツ内のキー。
- 演算子: 下表の算術演算子 (超える、以上、未満、以下) および文字列演算子 (Include、RegEx) を使用できます。

演算子	説明	例
>	列の値が値より大きい	$count > 0
<	列の値が値より小さい	$count<200
>=	列の値より大きいかまたは等しい	$count>=0
<=	列の値より小さいかまたは等しい	$count<=0
like	マッチした部分文字列	$project like “admin”
regex	正規表現にマッチする文字列	$project regex match “^/S+$”

3 アラーム方法の設定

Log Service には、現時点においては、次の通知方法があります。

- サイト内通知 (推奨)
- WebHook - DingTalk ボット
- カスタム WebHook

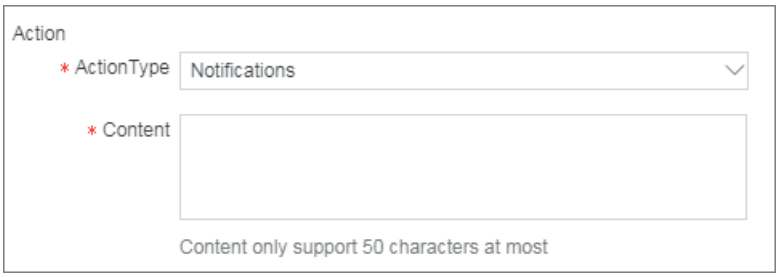
設定したアラームルールが起動されると、Log Service は指定された通知方法により、アラーム通知を送信します。

 注 :
Log Service アラームの SMS 通知機能は、間もなく実装される予定です。メッセージサービス (MNS) によるアラーム通知は送信できなくなります。

サイト内通知 (推奨)

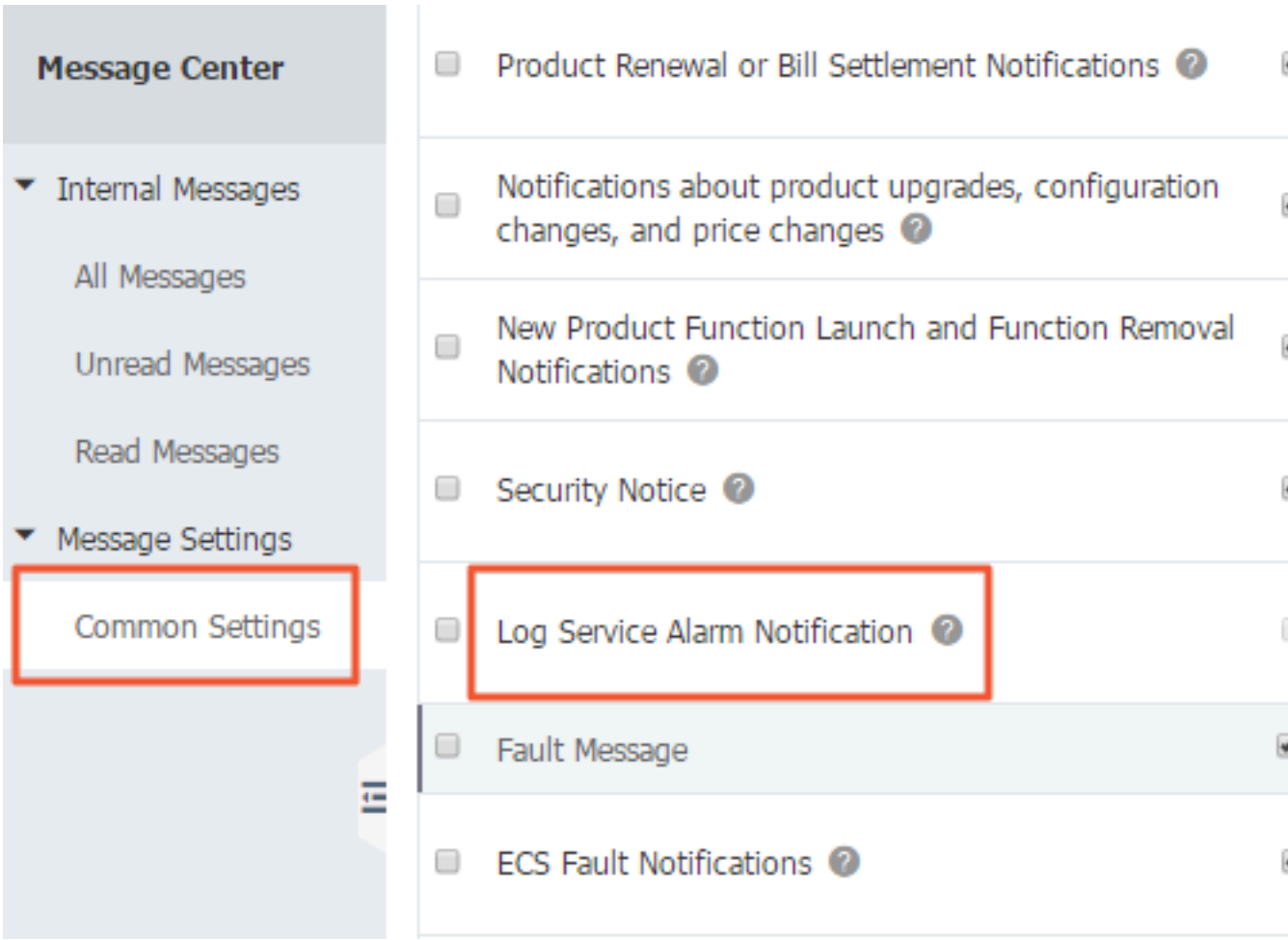
1. アラームルールページのアクション欄 > >アクション タイプドロップダウンリストより通知を選択し、通知内容を設定します。

   図 8-7 : Action



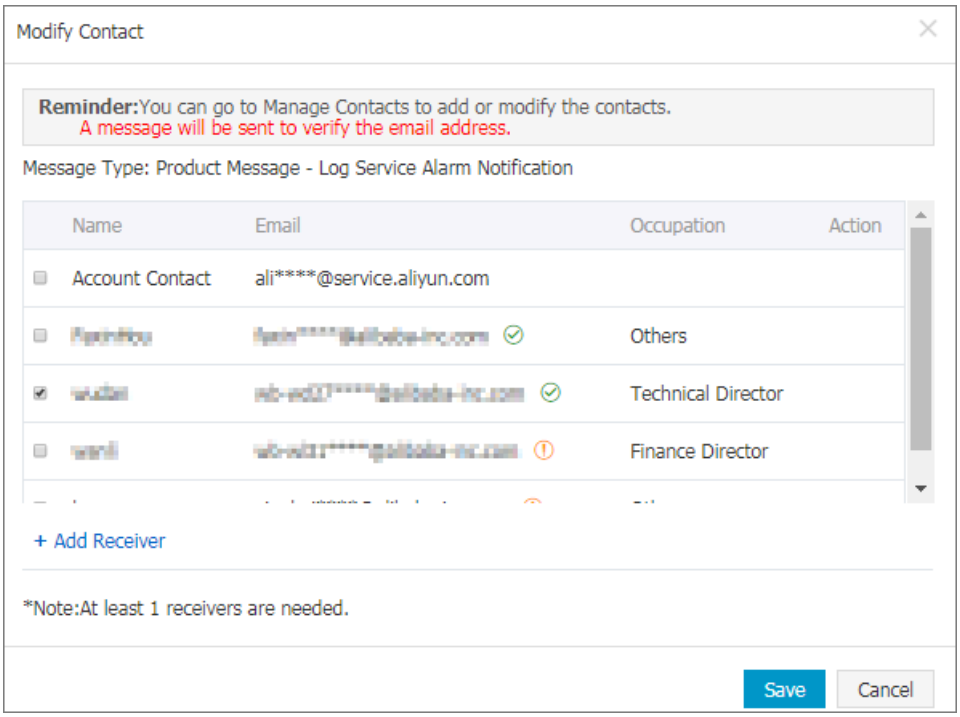
2. メッセージセンターのメッセージ設定 > 共通設定を順にクリックして、共通設定ページに移動します。

   図 8-8 : 共通設定



3. 該当する連絡先の通知タイプ > Log Service アラーム通知の列で変更をクリックします。 連絡先の変更ページを開きます。

図 8-9 : 連絡先の変更



4. 連絡先の変更ダイアログボックスで宛先を選択します。

宛先を追加するには、左下隅にある宛先の追加をクリックします。名前、メールアドレス、職業を指定します。アラーム通知が宛先に送信されます。

注 :

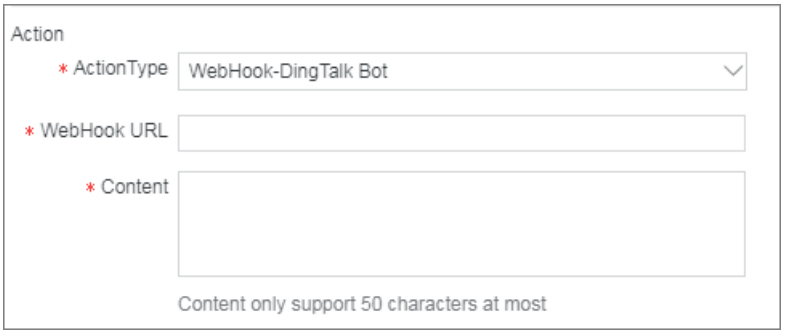
- 入力したメールアドレスに、確認情報が自動送信されます。 以上で、連絡先がアラーム通知を受信できることになります。
- 宛先を最低 1 つは指定する必要があります。
- アラーム操作は、メールに設定されており、変更することはできません。
- 各メールアドレスには、1 日に最大 50 件のアラーム通知が送信されます。

DingTalk ボット

1. DingTalk グループにボットを追加します。 WebHook を使用してカスタムサービスにアクセスするには、カスタムを選択します。
2. ボットの名前を入力し (オプション)、WebHook リンクをコピーします。

3. アラームルールページの Action 欄のアクションタイプドロップダウンリストよりWebHook-DingTalk ボットを選択し、WebHook URL欄に WebHook リンクを入力します。内容欄に通知内容を入力します。

   図 8-10 : 通知内容



カスタム WebHook

1. アラームルールページの Action 欄のAction タイプドロップダウンリストより、カスタム WebHookを選択し、WebHook URL欄に WebHook リンクを入力します。内容欄に通知の内容を入力します (半角英字 50 字以内)。

   図 8-11 : 通知の種類

2. 次の内容は、アラームがトリガーされると、Post モードで WebHook URL に送信されます。

   送信内容のサンプル

```
{" uid ":
            " 1341513451  3 "," project ":" ali - cn "," trigger ":"
 oplog_aler  t "," condition ":" 3413  >
             3000 ", " message ":" PV   count   down   30 %",
      " context ":" c : 3413 "}
```

4 アラームレコードを表示

アラームルールの作成後は、アラームレコードを確認することができます。

1. Log Service コンソールにログインします。
2. プロジェクトページでプロジェクト名をクリックします。
3. Logstore リストページページの左側のメニューよりLogSearch/Analytics > アラームの順にクリックします。
4. アラームルールの右側の表示をクリックして、アラームレコードを表示します。

アラームステータス:

- 正常: ルールが正常に実行され、アラームをトリガーする基準がトリガーの詳細に表示されます。
- 失敗: クエリ、アラームルール、または通知のいずれかに失敗しました。 詳細は、トリガー詳細をご確認ください。
  - ログのクエリに失敗 (一般に誤った構文が原因の失敗)
  - クエリステートメントの呼び出しに失敗 (チケットを起票して、サポートセンターにお問い合わせください)
  - ルールの呼び出しに失敗 (ルールパラメータの形式と応答のデータ形式が一致しているかどうかを確認)

8.2.2 アラート通知の設定

Log Service が提供するアラート機能を使用すると、E メール、DingTalk、WebHooks、およびメッセージセンターなどの複数の方法でアラート通知を設定できます。

通知方式：

- E メール
- DingTalk
- Webhook
- メッセージセンター

E メール

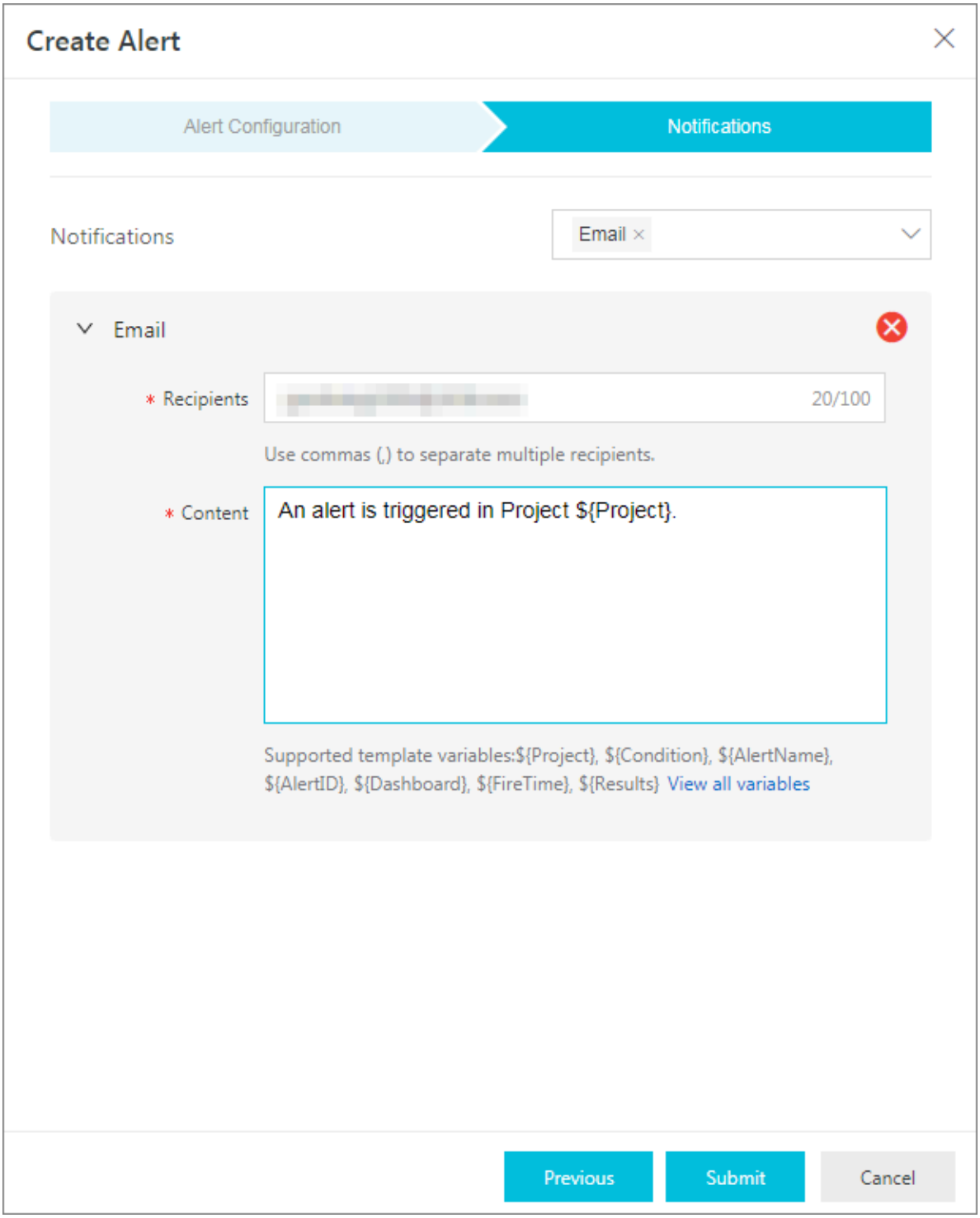
E メールでアラート通知を送信するように Log Service を設定できます。 アラートがトリガされると、Log Service は指定された E メールアドレスにアラート E メールを送信します。

手順

1. を参照して、Log Service コンソールでアラームを設定します。 通知ドロップダウンリストからE メールを選択します。

2. 受信者エリアにアラート通知を受信する電子メールアドレスを入力し、コンテンツエリアに電子メールの内容を入力します。

   複数の電子メールアドレスを区切るには、カンマ（、）を使用します。 E メールの内容は最大500文字です。 テンプレート変数がサポートされています。



3. 提出をクリックします。

DingTalk

DingTalk を通じてアラート通知を送信するように Log Service を設定します。 アラートがトリガされると、アラート通知が chatbot メッセージとして設定された DingTalk グループに送信されます。

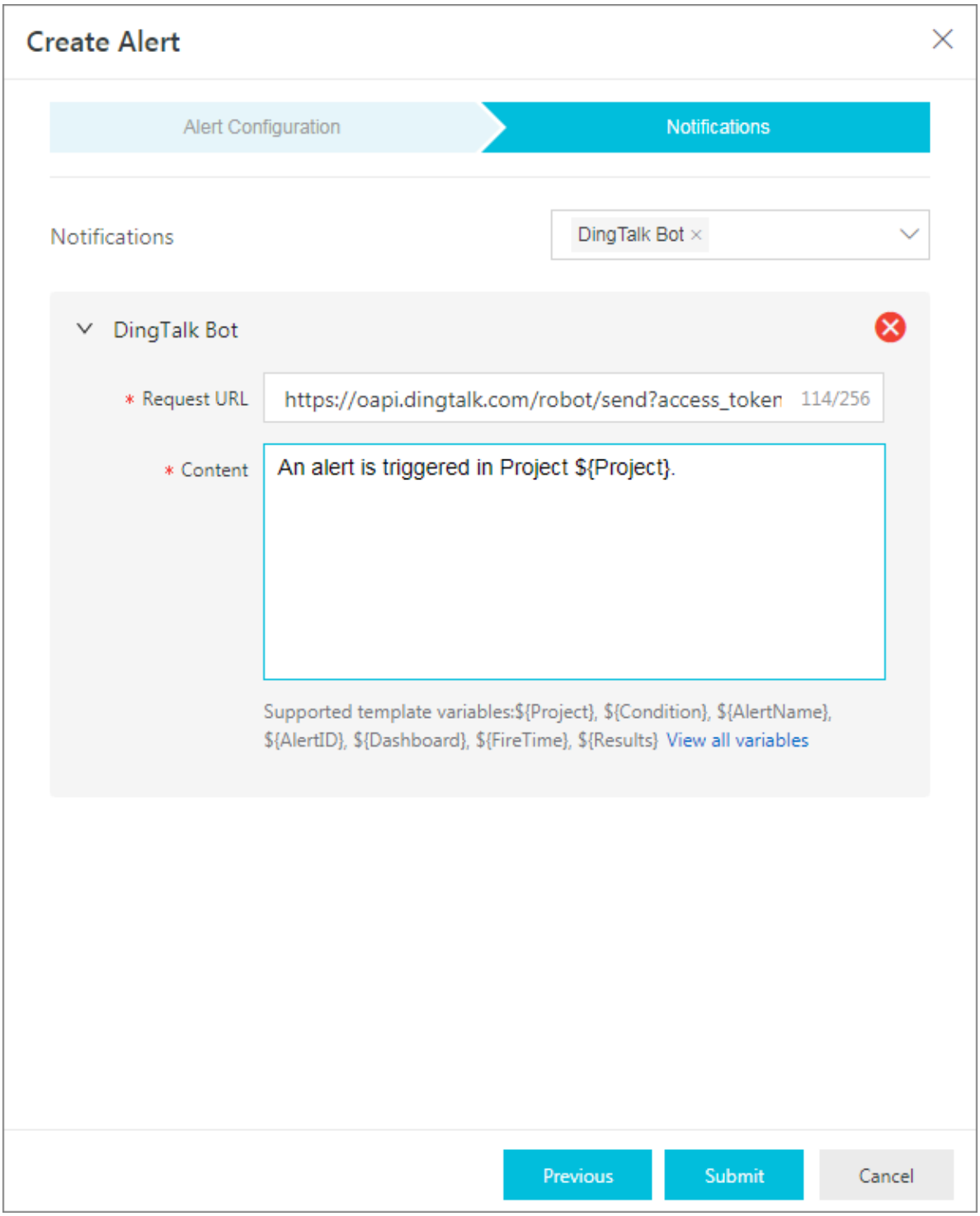
> 注 :
> 各 chatbot は、1分あたり最大 20 のアラートメッセージを送信できます。

手順

1. DingTalk を開き、対象の DingTalk グループを選択します。
2. 右上隅にあるグループ設定をクリックし、ChatBot をクリックします。
3. カスタム（Webhook によるカスタムメッセージサービス）を選択して追加をクリックします。
4. Chatbot 名を入力して、完了をクリックします。
5. コピーをクリックして webhook アドレスをコピーします。
6. Log Service コンソールで、[通知]ドロップダウンリストから DingTalk ボットを選択します。

7. ステップ 5 でコピーしたアドレスをリクエスト URL エリアに貼り付けます。 次に、内容エリアに通知内容を入力します。

図 8-12 : 通知内容



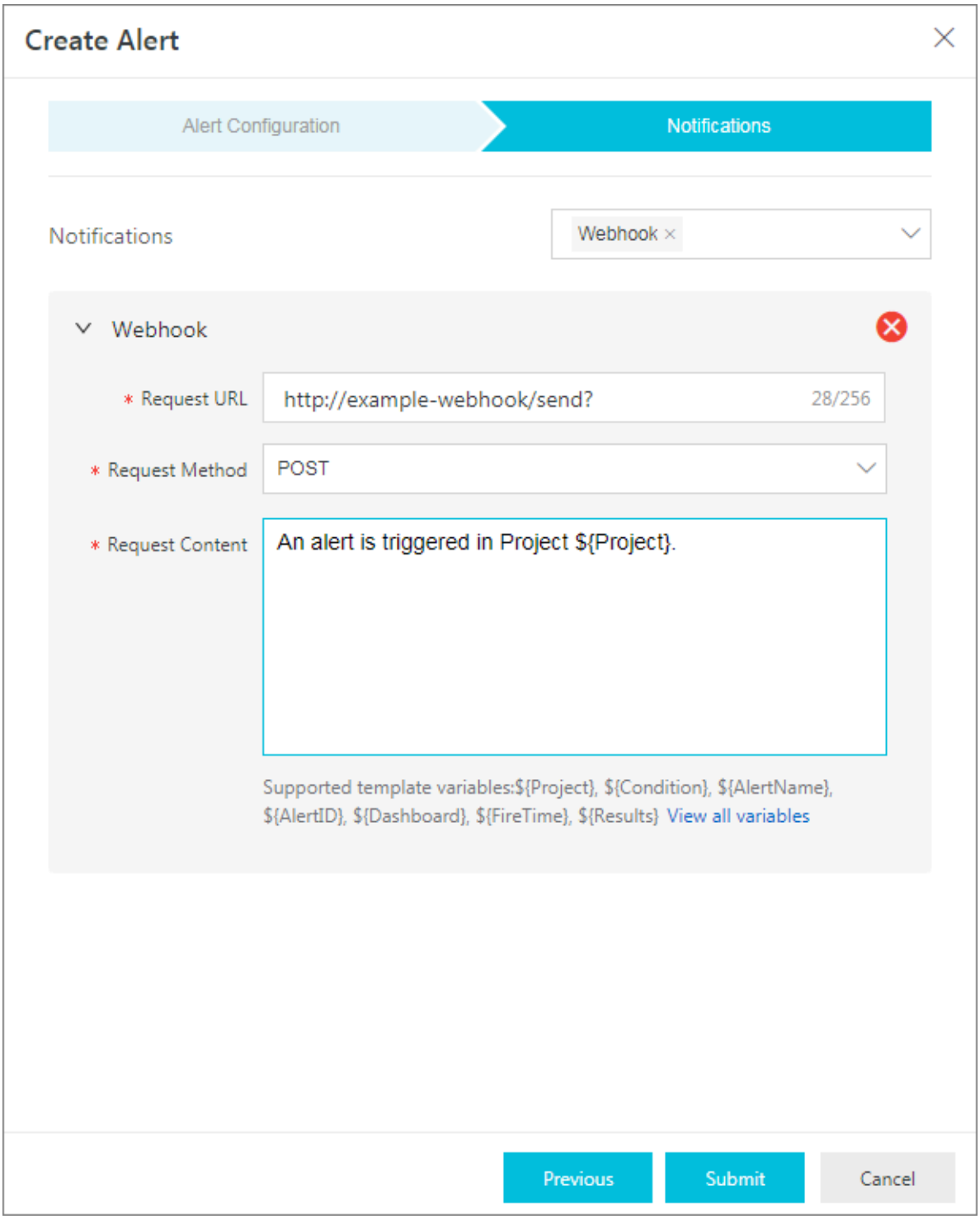
Webhook

Webhook を通じてアラート通知を送信するように Log Service を設定します。 アラートがトリガーされると、アラート通知が指定された方法でカスタム Webhook アドレスに送信されます。

手順

1. を参照して、Log Service コンソールでアラームを設定します。 「通知」ドロップダウンリストから WebHook を選択します。

2. リクエスト URL エリアで、カスタム WebHook アドレスを入力します。 リクエスト方式を選択します。 リクエスト内容を入力します。



アラートがトリガーされると、アラートの内容は指定された方法でカスタム WebHook アドレスに送信されます。

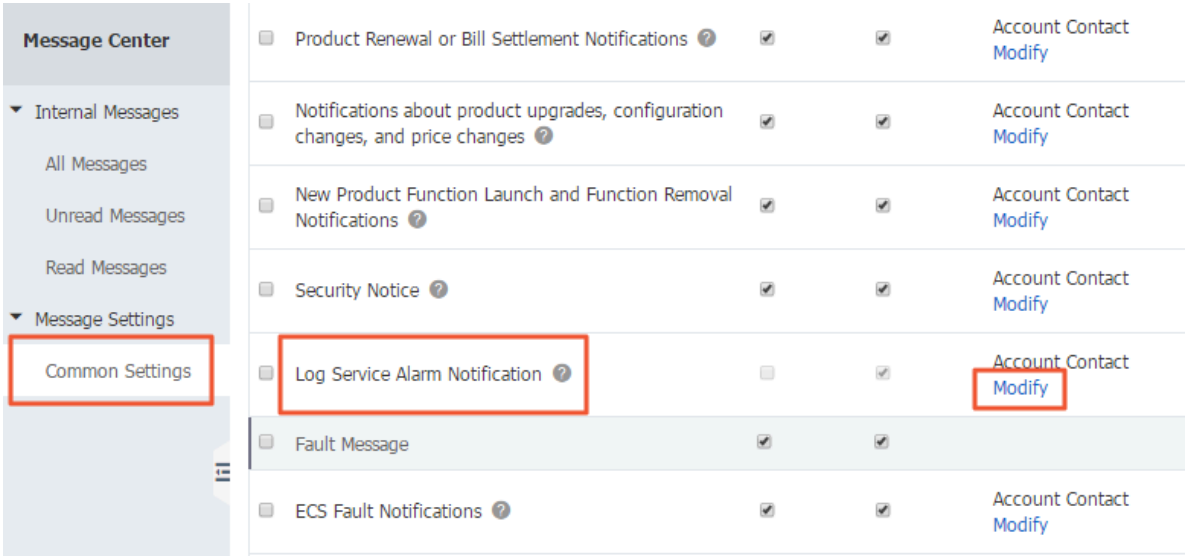
3. 提出をクリックします。

メッセージセンター（推奨）

メッセージセンターのコンソールでは、Log Service アラート通知の受信者を設定できます。アラートがトリガーされると、メッセージセンターは指定された方法でアラート通知を送信します。

手順

1. 「通知」ドロップダウンリストから通知を選択します。
2. メッセージセンターコンソールで、メッセージの設定 > 共通設定を選択します。



3. 通知のタイプ > Log Service アラーム通知のアカウント連絡先列で、変更をクリックします。
4. 連絡先の変更ダイアログボックスで、必要に応じてメッセージ受信者を選択します。

china site china site

> 注：
>
> - china site
> - 少なくとも一人のメッセージ受信者を設定する必要があります。
> - デフォルトでは、通知メッセージは E メールのみで送信され、他の方法は設定できません。
> - 1 日あたり最大 50 のアラート通知メッセージを指定の E メール受信箱に送信できます。

通知内容

各通知に通知内容を入力する必要があります。 アラートが発生したときに使用されるテンプレート変数は、${fieldName} の形式で参照できます。 Log Service は、アラート送信時に通知内

容のテンプレート変数を実際の値に置き換えます。 例：${Project} は、アラートルールが属するプロジェクトの名前に置き換えられます。


注：
有効な変数を参照する必要があります。 参照された変数が存在しないか、無効な変数を参照した場合、その変数は空の文字列として処理されます。 参照した変数の値が object 型の場合、その値を表示するために JSON 文字列に変換します。

次の表に、使用可能なすべての変数とそれらに対応する参照方法を示します。

変数	説明	例	参照の例
Aliuid	プロジェクトが属する Aliuid	1234567890	ユーザー ${Aliuid} のアラートルールがトリガーされています。
Project	アラートルールが属するプロジェクト	my-project	プロジェクト ${Project} のアラートがトリガーされています。
AlertID	実行されたアラートの一意の ID。	0fdd88063a611aa11493 8f9371daeeb6-1671a52eb23	実行されたアラートの ID は ${AlertID} となります。
AlertName	アラートルール名。プロジェクト内で一意である必要があります。	alert-1542111415-153472	アラートルール ${AlertName} がトリガーされています。
AlertDisplayName	アラートルールの表示名	マイアラートルール	アラート ${AlertDisplayName} がトリガーされています。
Condition	アラートをトリガーする条件式。 条件式の各変数はアラートをトリガーする値に置き換えられ、角括弧で囲まれます。	[5] > 1	アラート条件式は ${Condition} の形式です。
RawCondition	condition の変数が他の値に置き換えられない生条件式	count > 1	生条件式は ${RawCondition} の形式です。
Dashboard	アラートに関連付けられているダッシュボードの名前	mydashboard	アラートに関連付けられているダッシュボードは ${Dashboard} の形式です。

変数	説明	例	参照の例
DashboardUrl	アラートに関連付けられているダッシュボードのアドレス	https://sls.console.aliyun.com/next/project/myproject/dashboard/mydashboard	アラートに関連付けられているダッシュボードのアドレスは ${DashboardUrl} の形式です。
FireTime	アラートがトリガーされた時刻	2018-01-02 15:04:05	アラートの発生時間は ${FireTime} の形式です。

変数	説明	例	参照の例
FullResultUrl	トリガーされたアラートのレコードをクエリするために使用される URL	https://sls.console.aliyun.com/next/project/my-project/logsearch/internal-alert-history?endTime=1544083998&queryString=AlertID%3A9155ea1ec10167985519fccede4d5fc7-1678293caad&queryTimeType=99&startTime=1544083968	次をクリックして詳細を確認します： ${FullResultUrl}。
Results	クエリのパラメータと結果、および配列型。	[{ " EndTime ": 1542507580 , " FireResult ": { " __time__ ": " 1542453580 ", " count ": " 0 " }, " LogStore ": " test - logstore ", " Query ": "*	SELECT COUNT (*) as count ", " RawResultCount ": 1 , " RawResults ": [{ " __time__ ": " 1542453580 ", " count ": " 0 " }], " StartTime ": 1542453580 }]	最初のクエリは${Results [0]. StartTime}で始まり、${Results [0]. EndTime}で終わります。 count パラメータの値は ${Results[0]. FireResult.count}となります。

9 リアルタイムに読み込む (サブスクリプション)

9.1 概要

Log Service LogHub に収集されたら、次の 3 つの方法でログを活用できます。

メソッド	シナリオ	リアルタイム性	保存期間
リアルタイム読み込み (LogHub)	ストリームコンピューティングとリアルタイムコンピューティング	リアルタイム	指定します。
照会/分析 (LogSearch/ Analytics)	オンライン照会/分析	リアルタイム (99.99 % のケースでは 1 秒未満)	指定します。
送信/保管 (LogShipper)	オフライン解析用の完全なログストレージ	5〜30 分	ストレージシステムによって異なります。

リアルタイム読み込み

ログが書き込まれたら、読み込まれます。ログの読み取りおよび照会の両方にログを読み取る機能が必要です。シャードのログは次のように読み込まれます。

1. 時間、開始位置、終了位置といった条件に基づいてカーソルを取得します。
2. カーソルとステップを使用してログを読み取り、次のカーソルを返します。
3. カーソルを継続的に移動してログを読み込みます。

Log Service には、基本的な API に加えて、SDK、Storm の Spout、Spark Streaming クライアント、Flink コネクタ、コンシューマーライブラリ、Web コンソールといった、ログを読み込むさまざまな方法が用意されています。

- Spark Streamingでのログの読み込み
- Storm Spoutでのログの読み込み
- Flink Connector (Flink コンシューマーおよび Flink プロデューサを含む) でのログを読み込み
- LogHub コンシューマーライブラリでのログの読み込み (コンシューマーライブラリは、Loghub 読み込みの高度なモードです。軽量のコンピューティングフレームワークであり、複数のコンシューマーが同時に Logstore を読み込む場合、シャードの自動割り当ておよび読み込み順序保存の問題を解決します。)

- SDKでのログ読み込み (Log Service は、さまざまな言語 (Java および Python) の SDK でのログ読み込み API が用意されています。 SDK の詳細については、 Log service SDKをご参照ください。
- ログの読み込みに以下の Alibaba Cloud プロダクトをご利用いただけます。
  - CloudMonitor を用いた ログの消費を使用: 監視シナリオ。
  - E-MapReduce からのログの読み込み: Stormを使用した LogHub ログの読み込みとSpark + Log Serviceをご参照ください。
  - CLI (コマンドラインツール) よりログの読み込み

照会/分析

リアルタイムに照会/分析の概要

- Log Service コンソールでログをクエリするには、ログを照会をご参照ください。
- Log Service SDK/API を使用したクエリログ: Log Service には、HTTP プロトコルに基づいて実装された RESTful API を提供します。 Log Service API は、 フル機能のログクエリ API も提供します。 詳細については、Log Service APIをご参照ください。

送信と保管

- OSS へのログ転送: ログの長期保存、E-MapReduce を使用してログを分析します。
- Function Compute で Loghub のログを読み込み。

その他

セキュアな Log Service: Log Service は Alibaba Cloud のセキュリティプロダクトと連携し、ISV を使用して Alibaba Cloud プロダクトのログを読み込みます。

9.2 ログデータのプレビュー

ログプレビューは、通常のログ読み込み方法です。Log Service コンソールには、コンソールの Log Store の一部のログを直接プレビューするためのプレビューページが用意されています。

1. Log Serviceコンソールにログインします。
2. プロジェクト一覧ページで、プロジェクト名をクリックします。
3. Logstore リストページで、Logstore の右側のプレビューをクリックします。

4. 照会する Logstore シャードおよびログの時間を指定し、プレビューをクリックします。

   指定した) 初の 10 パケットが表示されます。

   図 9-1 : ログデータのプレビュー

9.3 コンシューマーグループで読み込む

9.3.1 コンシューマーグループ - 使用法

コンシューマーライブラリは、Log Service にログを読み込むための高度なモードです。コンシューマーグループ概念に基づき、コンシューマーをまとめて管理します。SDK を使用して直接データを読み取るときのように、Log Service 実装の知識は必要はありません。また、コンシューマー間の負荷分散、フェイルオーバーといったことも気にする必要がありません。コンシューマーライブラリを使用することで、サービスロジックに専念できます。

Spark Streaming、Storm および Flink Connector の実装にコンシューマーライブラリを使用します。

基本概念

コンシューマーライブラリを使用する前に、コンシューマーグループおよびコンシューマーの 2 つの概念を理解する必要があります。

- コンシューマーグループ

  コンシューマーグループは複数のコンシューマーで構成されます。コンシューマーグループ内のコンシューマーは、同じ Logstore のデータを読み込みます。コンシューマーごとに別々のデータが読み込まれます。
- コンシューマー

  コンシューマーは、データを読み込む、コンシューマーグループを構成する単位です。コンシューマー名は、コンシューマーグループ内で一意である必要があります。

LogService の Logstore は、複数のシャードで構成されます。コンシューマーグループのコンシューマーにシャードを割り当てるためにコンシューマーライブラリが使用されます。割り当てルールは次のとおりです。

・ 各シャードに割り当てることのできるコンシューマーは 1 つのみ

・ 各コンシューマーには、複数のシャードを紐づけることができる

コンシューマーグループに新たにコンシューマーを追加すると、読み込みが負荷分散されるようコンシューマーグループに紐づいているシャードは調整されます。 ただし、上記の割り当てルールは変わりません。割り当て処理はユーザーには見えません。

コンシューマーライブラリにもチェックポイントを保存することができます。したがって、コンシューマーはプログラムの例外が解決された後にブレークポイントから始まるデータを読み込み、データが一度だけ読み込まれるようにすることができます。

使用法

Maven ライブラリを追加

```
< dependency >
  < groupId > com . google . protobuf </ groupId >
  < artifactId > protobuf - java </ artifactId >
  < version > 2 . 5 . 0 </ version >
</ dependency >
< dependency >
  < groupId > com . aliyun . openservic  es </ groupId >
  < artifactId > loghub - client - lib </ artifactId >
  < version > 0 . 6 . 16 </ version >
</ dependency >
```

main .javaファイル

```
public   class   Main  {
 //  Log   Service  のドメイン名を指定
  private   static   String   sEndpoint  = " cn - hangzhou . log .
aliyuncs . com ";
 //  Log   Service  のプロジェクト名を指定
  private   static   String   sProject  = " ali - cn - hangzhou - sls
- admin ";
 //  Log   Service  の  Logstore  名を指定
  private   static   String   sLogstore  = " sls_operat  ion_log ";
 // コンシューマーグループ名を指定
  private   static   String   sConsumerG  roup  = " consumerGr  oupX
";
 // データ読み込みの  AccessKey  を指定
  private   static   String   sAccessKey  Id  = "";
  private   static   String   sAccessKey  = "";
  public   static   void   main ( String  [] args )  throws
LogHubClie  ntWorkerEx  ception ,  Interrupte  dException
 {
      // 2 つ目のパラメータはコンシューマー名です。コンシューマーグループ内のコンシューマー名はコンシューマーグループ内で一意である必要があります。ただし、コンシューマーグループ名は重複しても構いません。各コンシューマーは複数のマシンの複数の処理を負荷分散方式で読み込み開始します。 コンシューマー名はマシンの  IP  アドレス順に並べられます。 9  番目には、 maxFetchLo  gGroupSize  パラメータを指定します ( maxFetchLo  gGroupSize  は、 Log  Service  が  1  回に取得する Logstore  の数ですが、初期値をそのままご使用ください。変更する場合には、 1  以上 1000  以下にします)。
      LogHubConf  ig  config  =  new  LogHubConf  ig ( sConsumerG
roup , " consumer_1 ",  sEndpoint ,  sProject ,  sLogstore ,
```

```
 sAccessKey  Id ,  sAccessKey ,  LogHubConf  ig . ConsumePos  ition .
 BEGIN_CURS  OR );
       ClientWork  er   worker  =  new   ClientWork  er ( new
 SampleLogH  ubProcesso  rFactory (),  config );
         Thread   thread  =  new   Thread ( worker );
        //スレッドの実行が開始し、実行可能な  API  を  extend  すると
 ClientWork  er  が自動的に起動されます。
        thread . start ();
        Thread . sleep ( 60  *  60  *  1000 );
       // worker  の  Shutdown  関数を呼び出して読み込みインスタンスを 終了します。紐づいているスレッドは自動停止します。
        worker . shutdown ();
       // ClientWork  er  実行中に複数の非同期タスクが生成されます。シャットダウンしたら、実行タスクが終了するまで  30  秒待機することを推奨します。
        Thread . sleep ( 30  *  1000 );
  }
}
```

SampleLogHubProcessor.java ファイル

```
 public   class   SampleLogH  ubProcesso  r   implements   ILogHubPro
 cessor
{
   private   int   mShardId ;
  // 最後に残ったチェックポイント時間を記録します。
   private   long   mLastCheck  Time  =  0 ;
   public   void   initialize ( int   shardId )
  {
       mShardId  =  shardId ;
  }
  // データ読み込みの主要部。例外はすべてキャッチされますが、キャッチされた例外は返されません。
   public   String   process ( List < LogGroupDa  ta >  logGroups ,
          ILogHubChe  ckPointTra  cker   checkPoint  Tracker )
  {
      //  Write   checkpoint   to30  秒ごとに  Log   Service  にチェックポイントを書き込みます。 30  秒以内に  worker  がクラッシュすると、新たにクラッシュすると、新たに開始した  worker  が最終チェックポイントよりデータの読み込みを開始します。わずかに重複データが含まれる場合もあります。
        for ( LogGroupDa  ta   logGroup :  logGroups ){
          FastLogGro  up   flg  =  logGroup . GetFastLog  Group ();
          System . out . println ( String . format ("\ tcategory \ t
 :\ t % s \ n \ tsource \ t :\ t % s \ n \ ttopic \ t :\ t % s \ n \
 tmachineUU  ID \ t :\ t % s ",
                  flg . getCategor  y (),  flg . getSource (),  flg .
 getTopic (),  flg . getMachine  UUID ()));
          System . out . println (" Tags ");
          for  ( int   tagIdx  =  0 ;  tagIdx  <  flg . getLogTags
 Count (); ++ tagIdx ) {
              FastLogTag   logtag  =  flg . getLogTags ( tagIdx );
              System . out . println ( String . format ("\ t % s \ t
 :\ t % s ",  logtag . getKey (),  logtag . getValue ()));
          }
          for  ( int   lIdx  =  0 ;  lIdx  <  flg . getLogsCou  nt
 (); ++ lIdx ) {
              FastLog   log  =  flg . getLogs ( lIdx );
              System . out . println ("--------\ nLog : " +  lIdx  +
 ",  time : " +  log . getTime () + ",  GetContent  Count : " +  log .
 getContent  sCount ());
              for  ( int   cIdx  =  0 ;  cIdx  <  log . getContent
 sCount (); ++ cIdx ) {
                  FastLogCon  tent   content  =  log . getContent  s
 ( cIdx );
```

```
                    System . out . println ( content . getKey () + "\ t
 :\ t " +  content . getValue ());
                }
            }
        }
         long   curTime  =  System . currentTim  eMillis ();
        //  30  秒ごとに  Log   Service  にチェックポイントを書き込みます。 30
  秒以内に  worker  がクラッシュする場合、
        // 新たに起動した  worker  が最終チェックポイントから、データ読み込みを開始
します。わずかに重複データが含まれる可能性があります。
         if  ( curTime  -  mLastCheck  Time  >  30  *  1000 )
        {
             try
            {
                //パラメータが  true  の場合、 Log   Service  のチェックポイン
トが更新されます。パラメータが  false  の場合、チェックポイントはローカルマシンに
キャッシュされ、 Log   Service  にデフォルトで   60  秒ごとに  Log   Service
  のチェックポイントが更新されます。
                 checkPoint  Tracker . saveCheckP  oint ( true );
            }
             catch  ( LogHubChec  kPointExce  ption   e )
            {
                 e . printStack  Trace ();
            }
             mLastCheck  Time  =  curTime ;
        }
         return   null ;
    }
    // 終了する際、 worker  は本関数を呼び出します。ここできれいにします ( cleanup
 )。
     public   void   shutdown ( ILogHubChe  ckPointTra  cker
 checkPoint  Tracker )
    {
        // Log   Service  の読み込み中断位置の保存
         try  {
             checkPoint  Tracker . saveCheckP  oint ( true );
        }  catch  ( LogHubChec  kPointExce  ption   e ) {
             e . printStack  Trace ();
        }
    }
}
 class   SampleLogH  ubProcesso  rFactory   implements   ILogHubPro
 cessorFact  ory
{
     public   ILogHubPro  cessor   generatorP  rocessor ()
    {
        // 読み込みインスタンスの生成
         return   new   SampleLogH  ubProcesso  r ();
    }
}
```

上記のコードを実行して、すべてのデータを Logstore に書き込みます。1 つの Logstore を複数のコンシューマーで読み込むことのできるよう、コード内の説明に従ってプログラムを書き換え、同じコンシューマーグループ名と異なるコンシューマー名を使用して、読み込み処理を実行します。

制限と例外

各 Logstore には、コンシューマーグループを最大 10 個作成できます。上限を超えると、`ConsumerGroupQuotaExceed` エラーが報告されます。

コンシューマープログラムに Log4j を設定することをお勧めします。Log4j はコンシューマーグループで発生したエラーを返します。例外の特定に使用できます。resources ディレクトリに log4j.properties ファイルを 保存してプログラムを実行すると、次の例外が発生します。

```
[WARN ] 2018-03-14 12:01:52,747 method:com.aliyun.openservices.loghub.client.LogHubConsumer.sampleLogError(LogHubConsumer.java:159)
com.aliyun.openservices.log.exception.LogExceptionon: InvalidLoggroupCount, (0, 1000]
```

以下の log4j.properties 構成をご参照ください。

```
log4j.rootLogger=info, stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=[%-5p]%d{yyyy-MM-dd HH:mm:ss,SSS} method:%l%n%m%n
```

ステータスとアラーム

1. コンソールでコンシューマーグループのステータスを表示
2. CloudMonitor でコンシューマーグループの遅延を表示し、アラームを設定

高度な構成

通常は、上記のプログラムでデータを読み込みますが、高度な設定は以下のとおりです。

・ 特定の時間に始まるデータを読み込む

上記のコードの LogHubConfig には、コンストラクタが 2 つあります。

```
// consumerStartTimeInSeconds パラメータは、データが読み込まれてからの時間を 1970 以降の秒数を示します。
public LogHubConfig(String consumerGroupName,
                    String consumerName,
                    String loghubEndPoint,
                    String project, String logStore,
                    String accessId, String accessKey,
                    int consumerStartTimeInSeconds);
// Position is an enumeration variable, loghubconfig.glaseposition.begin_cursor は、最も古いデータから読み込みを開始することを示します。 loghubconfig.glaseposition.end_cursor は、最新のデータから読み込みが開始することを示します。
public LogHubConfig(String consumerGroupName,
                    String consumerName,
                    String loghubEndPoint,
                    String project, String logStore,
```

```
                        String   accessId ,  String   accessKey ,
                        ConsumePos  ition   position );
```

コンシューマーは必要に応じてさまざまに構築できますが、サーバーがチェックポイントで保存されている場合、読み込み開始位置はサーバーに保存されているチェックポイントに基づきます。

- RAM を利用した Log Service へのアクセス

  コンシューマーグループに関連する RAM 許可を設定し、RAM のドキュメントを参照するようにメソッドを設定する必要があります。設定する必要がある権限は以下の通りです。

操作	リソース
log:GetCursorOrData	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:CreateConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
log:ListConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
log:ConsumerGroupUpdateCheckPoint	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}
log:ConsumerGroupHeartBeat	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}
log:UpdateConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}

操作	リソース
log:GetConsumerGroupCheckPoint	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}

・ 読み込み位置をリセットする

いくつかのシナリオ (データの入力、繰り返し処理) では、ConsumerGroup ポイントを特定の時点に設定して、現在のコンシューマーグループが新しい時点から読み込まれるようにする必要があります。 2 つの方法があります。

1. コンシューマーグループを削除する
   - コンソールでコンシューマーグループを削除し、コンシューマーグループプログラムを再起動します。
   - コンシューマーグループ・プログラムは、デフォルトの開始点から読み込みを開始します（プログラムによって構成されます）。
2. SDK を使用して現在のコンシューマーグループを特定の時点にリセットする
   - Java のコード例は次のとおりです。
   - SDK を使用してサイトを変更してからコンシューマープログラムを再起動します。

```
Client client = new Client ( host , accessId , accessKey );
long time_stamp = Timestamp . valueOf (" 2017 - 11 - 15 00 :
00 : 00 "). getTime () / 1000 ;
ListShardR esponse shard_res = client . ListShard ( new
ListShardR equest ( project , logStore ));
ArrayList < Shard > all_shards = shard_res . GetShards ();
for ( Shard shard : all_shards )
{
  shardId = shard . GetShardId ();
  long cursor_tim e = time_stamp ;
  String cursor = client . GetCursor ( project , logStore ,
shardId , cursor_tim e ). GetCursor ();
  client . UpdateChec kPoint ( project , logStore , consumerGr
oup , shardId , cursor );
}
```

9.3.2 コンシューマーグループ - ステータス表示

コンシューマーライブラリ は、リアルタイムにデータを読み込む高度なモードで、複数のコンシューマーインスタンスによる Logstore 読み込みは自動的に負荷分散されます。Spark Streaming および Storm は、基本的にコンシューマーグループを使用します。

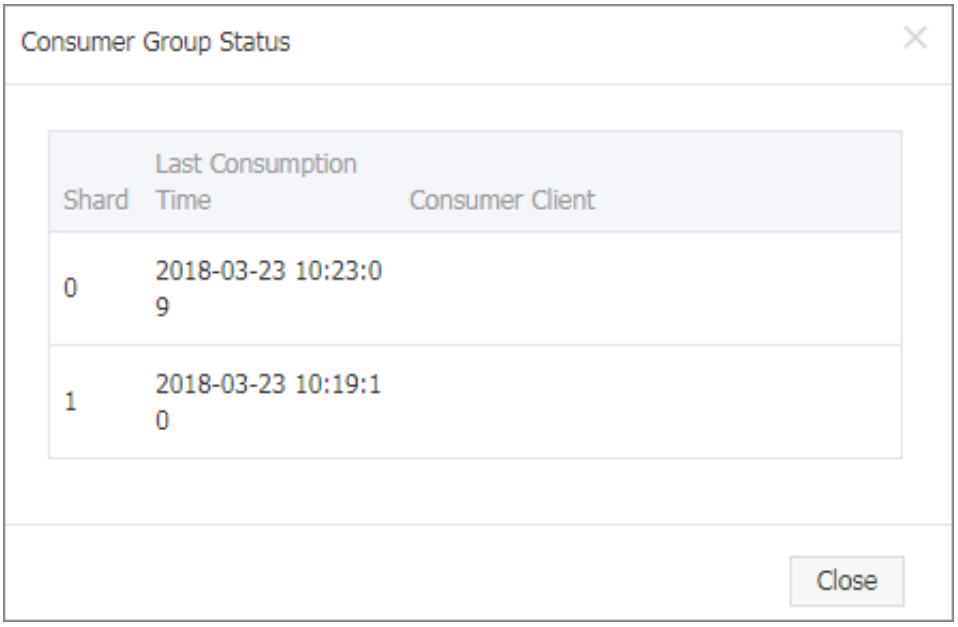
コンソールに読み込み状況を表示

1. Log Service コンソールにログインします。
2. プロジェクト一覧ページでプロジェクト名をクリックします。
3. 左側のナビゲーションメニューより、LogHub - コンシューマー> > コンシューマーグループをクリックします。
4. コンシューマーグループページで Logstore を選択し、コンシューマーグループ機能が有効かどうかを確認します。

   図 9-2 : コンシューマーグループ

5. コンシューマーグループの右にあるステータスをクリックすると、各シャードのデータ読み込み状況が表示されます。

   図 9-3 : 読み込み状況



   上図のとおり、Logstore には 6 つのシャードがあり、3 つのコンシューマーが紐づいています。各コンシューマーがデータを最後に読み込んだ時間は 2 列目に表示されます。データが読み込まれた時間から、データのデータ読み込みがデータ生成に追いついているかどうかがわかります。データ読み込みが大幅に遅れを取っている場合 (つまり、データの生成よりデータの読み込みが遅い場合)、コンシューマーの数を増やすことをお勧めします。

API/SDK より読み込みの進行状況を表示

以下のコマンドは Java SDK の使用例です。API より読み込み状況を取得しています。

```
package test ;
import java . util . ArrayList ;
import com . aliyun . openservic es . log . Client ;
import com . aliyun . openservic es . log . common . Consts .
CursorMode ;
import com . aliyun . openservic es . log . common . ConsumerGr
oup ;
import com . aliyun . openservic es . log . common . ConsumerGr
oupShardCh eckPoint ;
import com . aliyun . openservic es . log . exception .
LogExcepti on ;
public class ConsumerGr oupTest {
    static String endpoint = "";
    static String project = "";
    static String logstore = "";
    static String accesskeyI d = "";
    static String accesskey = "";
    public static void main ( String [] args ) throws
LogExcepti on {
        Client client = new Client ( endpoint , accesskeyI d
, accesskey );
       // Logstore 内のコンシューマーグループをすべて取得します。コンシューマーグループがない場合、 consumerGr oups の長さは 0 です。
        ArrayList < ConsumerGr oup > consumerGr oups ;
        try {
            consumerGr oups = client . ListConsum erGroup (
project , logstore ). GetConsume rGroups ();

        catch ( LogExcepti on e ){
            if ( e . GetErrorCo de () == " LogStoreNo tExist ")
                System . out . println (" this logstore does
not have any consumer group ");
            else {
               //内部サーバーエラー分岐

            return ;

        for ( ConsumerGr oup c : consumerGr oups ){
           // Print consumer group properties , including
  names , heartbeat timeout , and whether or not the
consumptio n is in order .
                      System . out . println ("名前:" + c .
getConsume rGroupName ());
            System . out . println ("ハートビートのタイムアウト:" + c .
getTimeout ());
            System . out . println ("読み込み順序" + c . isInOrder
());
            for ( ConsumerGr oupShardCh eckPoint cp : client .
GetCheckPo int ( project , logstore , c . getConsume rGroupName
()). GetCheckPo ints ()){
                System . out . println (" shard : " + cp . getShard
());
               //フォーマットします。正確な時間がミリ秒単位（ Long 型）で返されます。
                System . out . println ("最終読み込み時間:" + cp .
getUpdateT ime ());
                String consumerPr g = "";
                if ( cp . getCheckPo int (). isEmpty ())
                    consumerPr g = "読み込みはまだ開始されていません。";
```

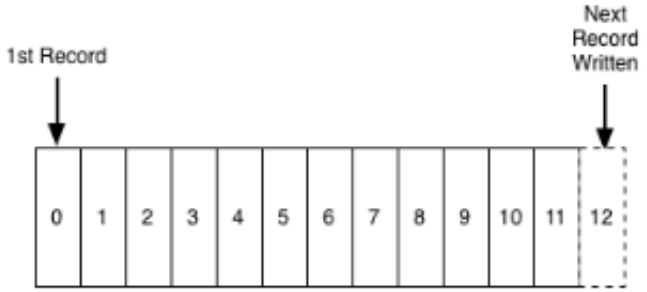
```
                else {
                   // UNIX  タイムスタンプ（単位: 秒）。出力前に値を初期化。
                    try {
                        int   prg  =  client . GetPrevCur  sorTime (
project ,  logstore ,  cp . getShard (),  cp . getCheckPo  int ()).
GetCursorT  ime ();
                        consumerPr  g  = "" +  prg ;

                    catch ( LogExcepti  on   e ){
                        if ( e . GetErrorCo  de () == " InvalidCur
sor ")
                            consumerPr  g  = "無効。最後の読み込み時間が、
Logstore  のデータライフサイクルを超えました。";
                        else {
                           //内部サーバーエラー
                            throw   e ;



                System . out . println ("読み込みの進行状況:" +
consumerPr  g );
                String   endCursor  =  client . GetCursor ( project
,  logstore ,  cp . getShard (),  CursorMode . END ).  GetCursor ();
                int   endPrg  =  0 ;
                try {
                    endPrg  =  client . GetPrevCur  sorTime ( project
,  logstore ,  cp . getShard (),  endCursor ).  GetCursorT  ime ();

                catch ( LogExcepti  on   e ){
                   // do   nothing

               // UNIX  タイムスタンプ（単位: 秒）。出力前に値を初期化。
                                System . out . println ("最終データの受
け取り時間:" +  endPrg );
```

9.3.3 コンシューマーグループ - アラームモニタリング

コンシューマーグループは、コンシューマーのグループです。各コンシューマーは、Logstore の一部のシャードを読み込みます。

シャードのデータモデルはキューに例えられます。新たな書き込みデータは、キューの末尾に追加されます。キュー内の各データは、データ書き込み時間に対応します。下図は、シャードのデータモデルです。

図 9-4 : シャードデータモデル



コンシューマーグループの待ち時間アラームの基本概念は次のとおりです。

- 読み込みプロセス: コンシューマーがキューの先頭から順にデータを読み取るプロセス
- 読み込みの進行状況: コンシューマーが現在読み込んだデータを書き込んでいる時間
- 読み込みの遅延: 現在の読み込み状況とキュー内の最新のデータ書き込み時間との差 (秒単位)

コンシューマーグループの読み込み遅延は、含まれるすべてのシャードの読み込み遅延のうち最大値をとります。値が指定されたしきい値を超えると、データの読み込みデータ生成よりもずっと遅れているとみなされ、アラームがトリガされる必要です。

手順

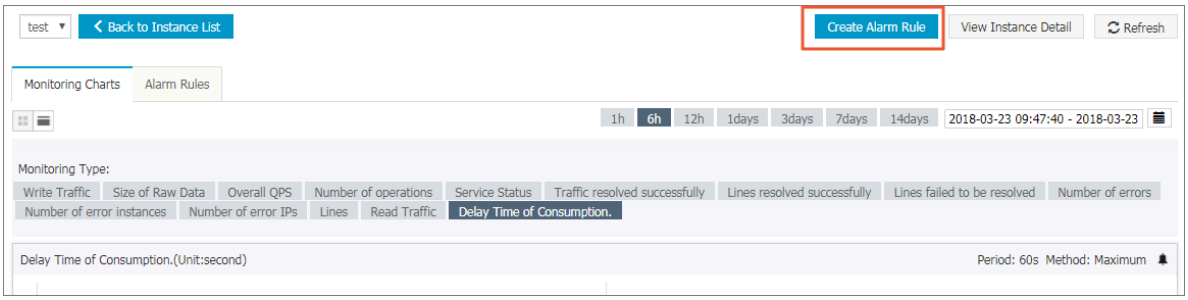
1. Log Service コンソールにログインします。プロジェクト一覧ページでプロジェクト名をクリックします。

2. Logstore リストページで、Logstore の右側のモニターアイコンをクリックします。

図 9-5 : グラフ名「読み込み遅延グラフ(秒)」をクリック

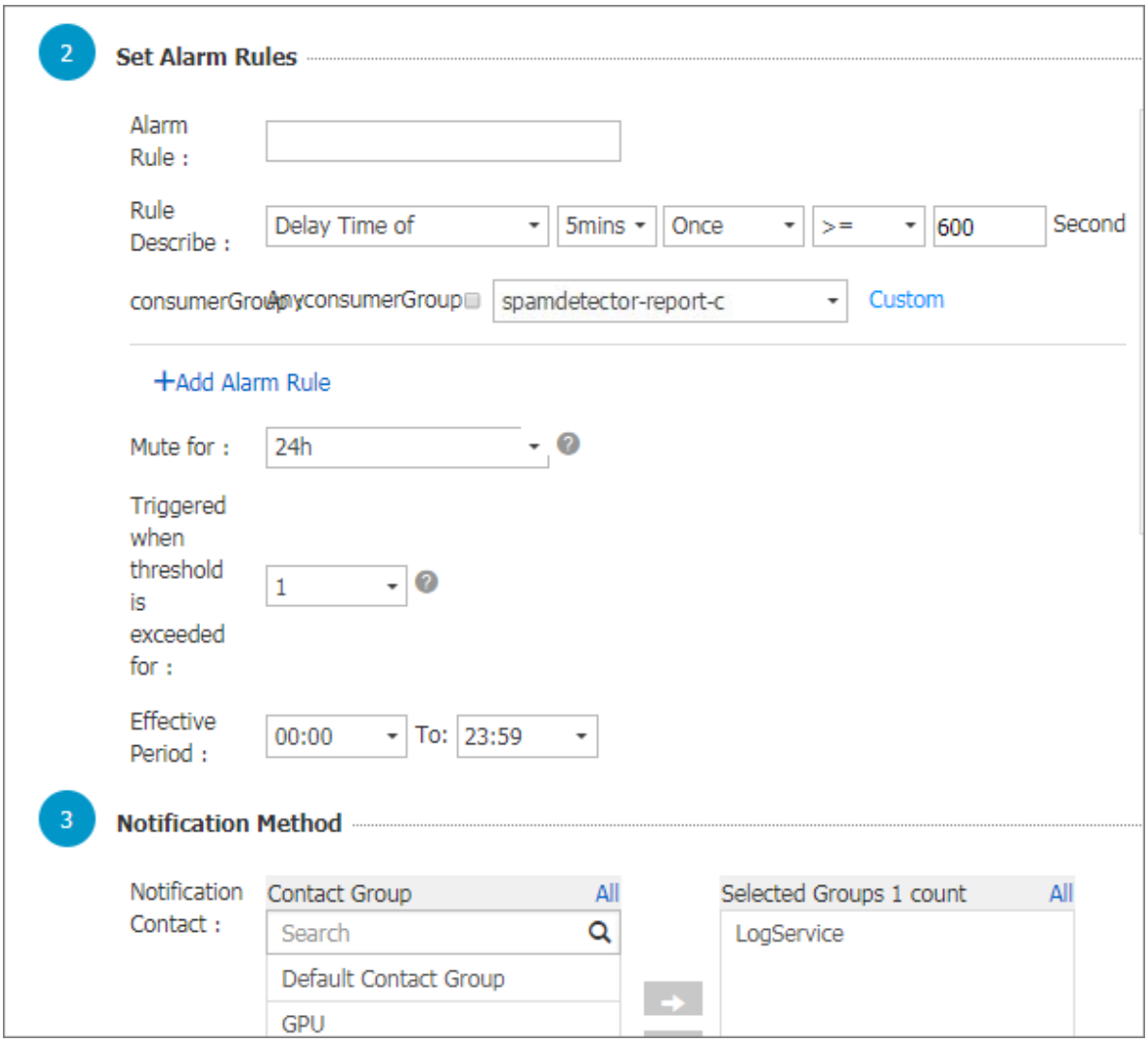
3. 下図は、Logstore の下にあるすべての Java グループの読み込み時間を秒単位で示しています。 秒単位で示したものです。 右上のアラームルールの作成をクリックして、アラームルールの作成ページに移動します。

図 9-6 : コンシューマーグループ spamdetector-report-c のアラームルールを作成



4. 5 分以内の遅延が 600 秒以上の場合にアラームがトリガーされます。有効期間および通知連絡先を設定し、ルールを保存します。

図 9-7 : アラームルールを設定



アラームルールが生成されます。アラームルールの設定に関して不明な点がありましたら、チケットを起票して、サポートセンターにお問い合わせください。

9.4 Function Compute で LogHub ログの読み込み

9.4.1 開発ガイド

Log Service カスタム ETL 関数 のデータ読み取り端末は、Alibaba Cloud の Function Compute 上で稼働しています。 さまざまな ETL の目的に応じて、Log Service の提供する関数テンプレートまたはユーザー定義関数を使用できます。

本ドキュメントでは、ユーザー定義の Log Service ETL 関数を実装する方法を紹介します。

関数イベント

関数イベント は、関数を実行するために使用される入力パラメーターのコレクションであり、 シリアル化された JSON オブジェクト文字列の形式です。

フィールドの説明

- jobName フィールド

  Log Service ETL ジョブの名前。Function Compute サービスの Log Service トリガーは、Log Service の ETL ジョブです。
- taskId フィールド

  ETL ジョブで、taskId は関数呼び出しに固有の識別子です。
- cursorTime フィールド

  関数呼び出しにより、Log Service の取得した最終ログの unix_timestamp です。
- source フィールド

  本フィールドは、Log Service によって生成されます。Log Service は、ETL ジョブで定義されたタスク間隔に基づいて関数の実行を定期的にトリガーします。source フィールドは、関数イベントの要です。本フィールドには、関数呼び出しによって読み取られるデータが定義されます。

  データソース範囲は、以下のフィールドで構成されます (関連するフィールド定義の詳細については、Log Service 用語集をご参照ください)。

フィールド	フィールドの説明
endpoint	Log Service プロジェクトと同じリージョンのサービスエンドポイント
projectNa	プロジェクト名
logstoreName	Logstore 名
Shardid	指定の Logstore のシャード
beginCursor	データの読み取り開始のシャード位置
endCursor	データの読み取り終了のシャード位置

注 :

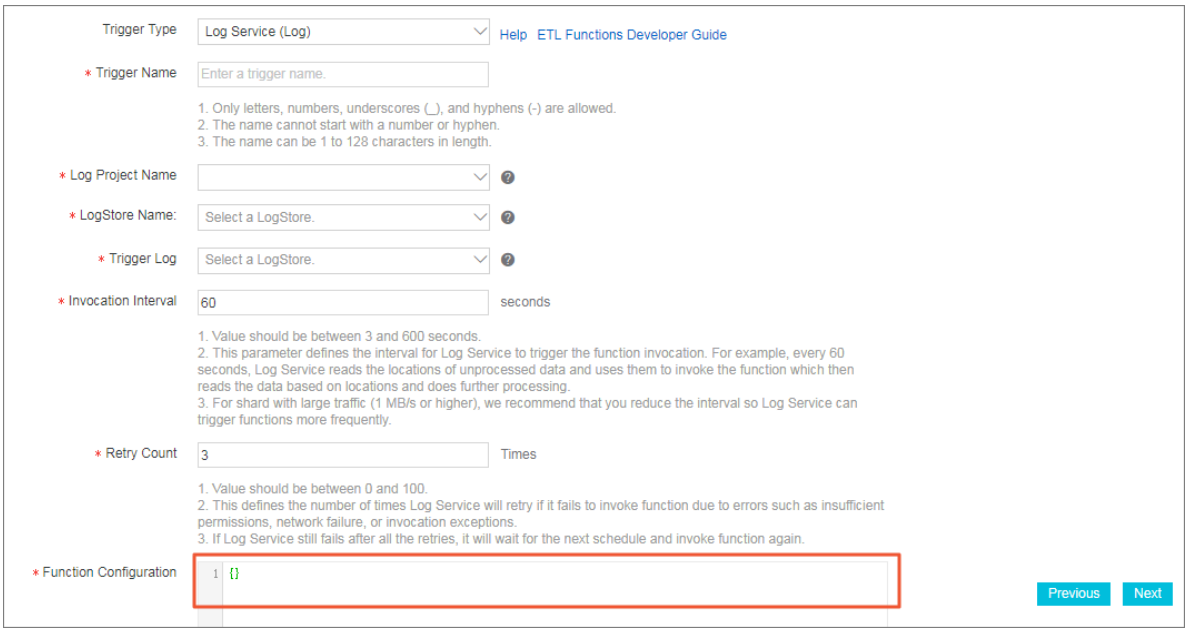
シャードの [beginCursor、endCursor) は、左閉じ、右開きの間隔です。

- parameter フィールド

  JSON オブジェクトフィールドは、ETL ジョブの作成時に設定されます (Function Compute の Log Service トリガー)。ユーザー定義関数の実行時には、本フィールドが解析され、関数に必要な操作パラメーターを取得します。

  Function Compute コンソールで Log Service トリガーを作成するときは、関数設定フィールドでこのフィールドを設定します。

  図 9-8 : 関数設定



関数イベントの例

```
{
    " source ": {
        " endpoint ": " http :// cn - shanghai - intranet . log .
 aliyuncs . com ",
        " projectNam  e ": " fc - 158429 **********",
        " logstoreNa  me ": " demo ",
        " shardId ":  0 ,
        " beginCurso  r ": " MTUwNTM5MD  I3NTY1ODcw  NzU2Ng ==",
        " endCursor ": " MTUwNTM5MD  I3NTY1ODcw  NzU2OA =="
    },
    " parameter ": {
        ...
    },
    " jobName ": " fedad35f51  a2a97b466d  a57fd71f31  5f539d2234 ",
    " taskId ": " 9bc06c96 - e364 - 4f41 - 85eb - b6e579214a  e4 ",
    " cursorTime ":  1511429883
}
```

関数をデバッグするときは、GetCursor API を使用してカーソルを取得し、上記の形式に沿ってテストする関数イベントを作成します。

関数の開発

Java、Python、Node.js といったさまざまな言語の関数を実装できます。Log Service は、関数の実装を容易にするために対応するランタイム、さまざまな言語の SDK を提供します。

この節では、Java 8 ランタイムを例として使用し、 Log Service ETL 関数を開発する方法を紹介します。 これには Java 8 関数のプログラミングの詳細が含まれているため、最初に Function Compute の Javaプログラミングガイド をお読みください。

Java 関数テンプレート

現在、 Log Service は Java 8 実行環境に基づいて ユーザー定義の ETL 関数テンプレートを提供しています。 これらのテンプレートを使用して、カスタム要件を実装することができます。

テンプレートには次の機能が実装されています。

- 関数イベント内の source、taskId、および jobName フィールドの解析
- Log Service Java SDKで source に定義されたデータソースからデータを読み込み、processData API 呼び出しにより、データのまとまりを処理

テンプレートには、以下も含まれている必要があります。

- 関数イベントのパラメーターフィールド解析に `UserDefine dFunctionP arameter . ja - JPva`
- 関数内のデータのサービスロジックをカスタマイズするに `UserDefine dFunction . java` の ProcessData API
- `UserDefine dFunction` を関数を的確に説明する名前に変更

processData メソッドの実装

processData では、必要に応じてデータのまとまりを読み込み、処理、送信する必要があります。

Logstore からデータを読み取り、別の Logstore に書き込む方法は、LogstoreReplication (英文)をご参照ください。

注意

注 :

1. processData を使用してデータが正常に処理された場合は、true が返されます。データ処理で例外が発生し、再試行しても例外が発生する場合は、false が返されます。ただし、例外が発生しても、関数は引き続き実行され、Log Service は正常に処理されなかったデータを無視して、成功した ETL タスクと判断します。

2. 致命的なエラーが発生した場合、またはサービスロジックにより関数の実行を早期に終了する必要があると判断する場合は、Throw Exception で関数の実行を終了してください。Log Service は、ETL ジョブルールに基づいて、関数処理の例外を検出し、 関数の実行を再度呼び出します。

注意事項

- シャードのトラフィックが多い場合は、関数の OOM による異常終了を防ぐため、その関数の実行に十分なメモリを設定してください。
- 関数の実行に時間のかかる場合、または、シャードのトラフィックが多い場合は、関数のトリガー間隔を小さく、また、関数のタイムアウトしきい値を高く設定してください。
- Function Compute サービスに必要な権限を付与してください。たとえば、Object Storage Service (OSS) データを関数に書き込むには、Function Compute サービスに OSS への書き込み権限を付与する必要があります。

ETL ログ

- ETL スケジューリングログ

  スケジューリングログには、ETL タスクの開始時刻と終了時刻、ETL タスクが成功したかどうか、および ETL タスクの情報が正常に返されたかどうかのみが記録されます。ETL タスクでエラーが発生すると、Function Compute サービスは ETL エラーログを生成し、システム管理者にアラーム E-メールまたは SMS を送信します。トリガーを作成するときは、トリガーログ Logstore を設定し、この Logstore のクエリおよびインデックス関数を有効にします。

  関数実行統計は、Java 8 関数 outputStream などの関数によって書き出され、返されます。Log Service に用意されているデフォルトのテンプレートは、シリアル化された JSON オブジェクト文字列を書き込みます。この文字列は ETL タスクスケジューリングログに記録され、統計とクエリが容易になります。

- ETL プロセスログ

  ETL プロセスログには、ステップの開始時刻、終了時刻、初期化完了、モジュールエラー情報といった、 ETL 実行プロセスにおける各ステップのキーポイントやエラーが記録されます。ETL プロセスログにより、ETL 操作状況を常に最新の状態に保つことができます。エラーが発生した場合は、直ちにプロセスログで原因を特定することができます。

  context.getLogger() を使用して、プロセスログを特定のプロジェクトや Log Service の Logstore に記録することができます。Logstore のインデックス機能およびクエリ機能を有効にすることをお勧めします。

9.4.2 Function Compute の設定

Log Service は、Function Compute を利用して、ストリーミングデータを処理する完全ホスト型のサービスを提供します。

ETL ジョブを設定すると、Log Service は定期的に更新データを取得し、関数の実行をトリガーします。つまり、 Log Service の Logstore データの増分を読み込み、カスタム処理関数のタスクを完了します。Log Service に用意されているテンプレート関数、およびユーザー定義関数をデータ処理に利用できます。

利用イメージ

データのクリーニングと処理

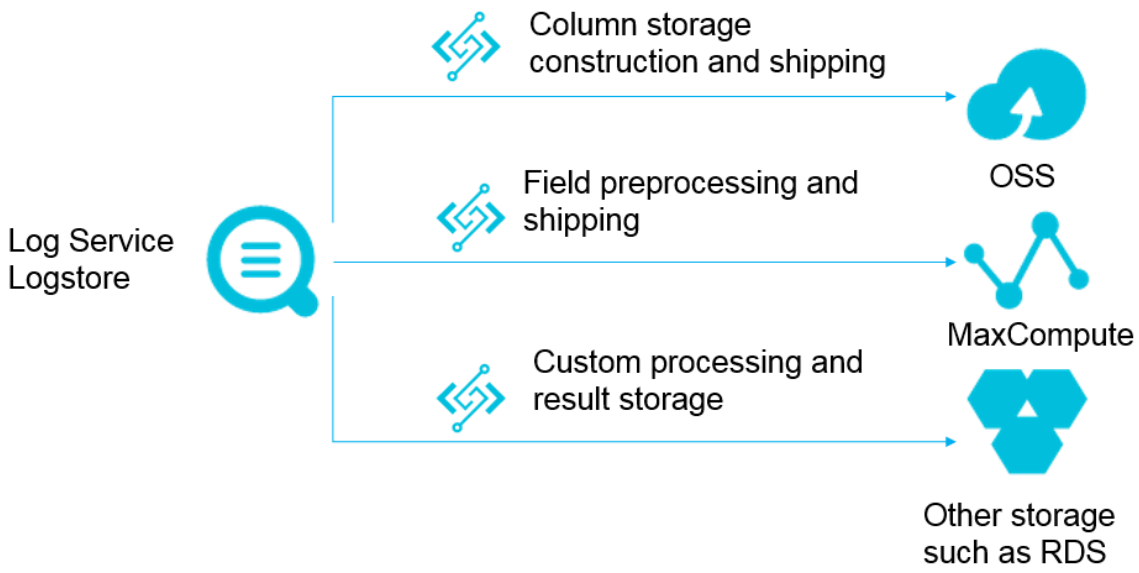
Log Service を使用して、素早くログを収集、処理、クエリ、および分析できます。

図 9-9 : データのクリーニングと処理

データ配信

Log Service はビッグデータを扱うクラウドプロダクトとのデータパイプラインを構築し、データを送信します。

図 9-10 : データ配信



操作の原則

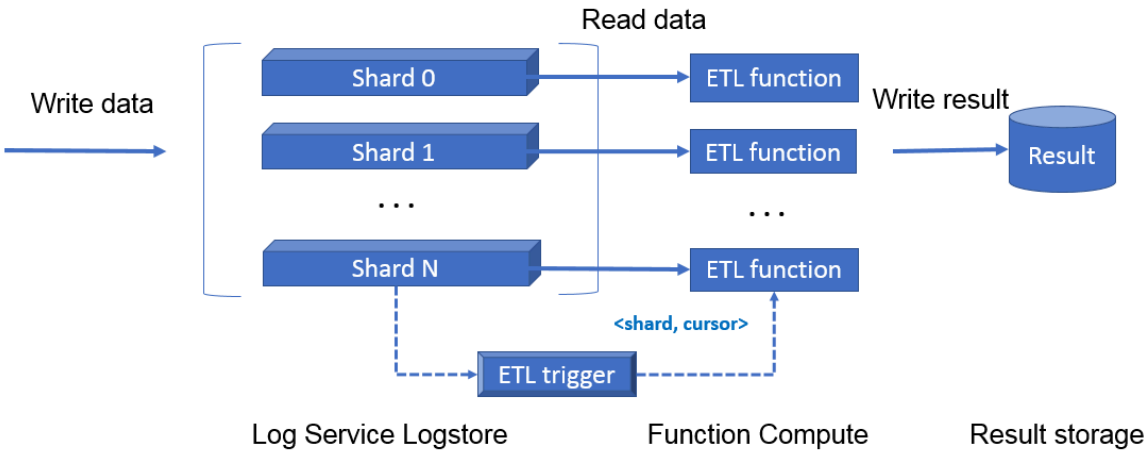
トリガー

Log Service ETL ジョブは、 Function Compute のトリガーに対応します。ETL ジョブを作成すると、Log Service のジョブ設定に基づいてタイマーが始動します。タイマーは、Logstore シャード情報をポーリングします。新しいログが書き込まれ、<shardid、 begin_cursor、end_cursor> の 3 つの情報を含む関数イベントが生成され、関数の実行がトリガーされます。

Log Service ETL ジョブは、時間を基にトリガーされます。たとえば、ETL ジョブのトリガー間隔が 60 秒で、Logstore のシャード 0 に定期的にデータが書き込まれている場合、シャード 0 に対して毎分、関数の実行がトリガーされます。シャード 0 に新しいデータが書き込まれなかった

場合、関数の実行はトリガーされません。関数実行の入力は、最後の 60 秒間のカーソル間隔です。関数では、シャード 0 の データのカーソルに基づいて読み取り、処理されます。

図 9-11 : トリガー



ETL 関数

関数テンプレートまたはユーザー定義関数を使用できます。ご使用の前に、Function Compute サービスの基本概念を習得されることが推奨されます。

- Log Service に用意されている関数テンプレート

  関数テンプレートは GitHub より入手できます。 aliyun-log-fc-functionsをクリックして、GitHub にアクセスします。
- ユーザー定義関数

  独自の関数を実装できます。関数の設定方法は、各関数で異なります。詳細は、ETL 関数の開発ガイドをご参照ください。

ユーザーガイド

手順 1 . Log Service を許可し、リソースを準備

1. クイック認可ページで、権限付与をクリックして、Log Service に関数トリガー権限を付与します。
2. 関数の生成するログ用の Log Service プロジェクトおよび Logstore を作成します。

   初めてプロジェクトまたは Logstore を作成する場合は、準備の手順に従って作成します。

   注 :

Log Service プロジェクトと Function Compute サービスは、同じリージョンでなければなりません。

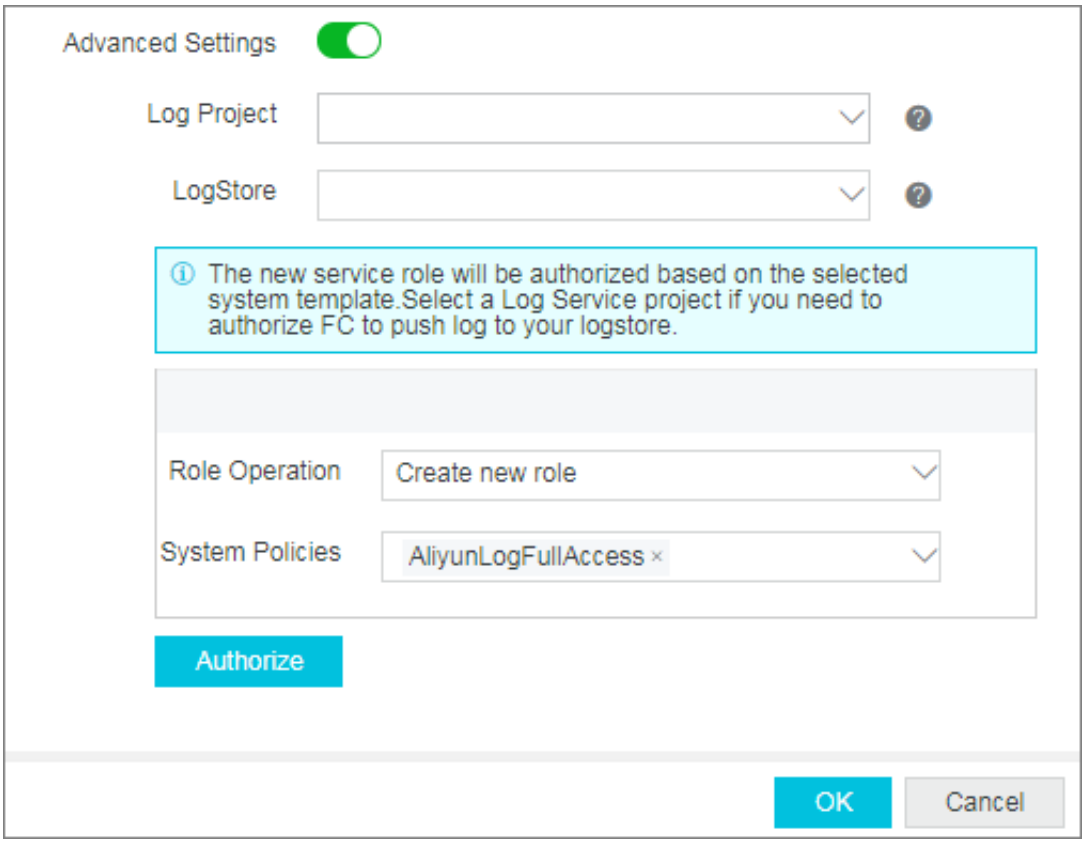
手順 2. サービスを作成

1. Function Compute コンソールで、サービスの作成をクリックします 。
2. サービス名と 説明を入力します 。 詳細設定 を有効にします。

設定項目	説明
サービス名	作成する Function Compute サービスの名前。 命名規則： ・ 名前には、大文字、小文字、数字、ハイフン (-)、およびアンダースコア () を使用できます。 ・ 大文字、小文字、またはアンダースコア () で始まる必要があります。 ・ 大文字と小文字が区別され、1～128 文字を含める必要があります。
説明	新しいサービスの説明。
ログプロジェクト	Log Service プロジェクトの名前。 LogStore は、新しい Function Compute サービスと同じリージョンでなければなりません。
Logstore	Log Service Logstore の名前。 LogStore は、新しい Function Compute サービスと同じリージョンでなければなりません。
ロール設定	サービスロールを作成し、選択したシステムテンプレートに基づいて対応する許可を作成します。 指定された Logstore にログをプッシュするために Function Compute を承認します。 新しいロールを作成したり、既存のロールを選択することができます。 既存のロールを使用するには、既存のロールを選択します。

設定項目	説明
システムポリシー	システム権限付与ポリシーを選択します。`AliyunLogF ullAccess` または `AliyunLogR eadOnlyAcc ess` のいずれかのポリシーを適用できます。

図 9-12 : サービスの作成



システム権限付与ポリシーを選択したら、許可をクリックします。ロールテンプレートページが表示されます。ポリシー名 、 ポリシーの説明 、 ポリシーの詳細といったロール情報および許可情報を確認します。新しいロールを作成する場合は、ロール名とロールの説明を確認します。 ポリシーの詳細では、権限付与ポリシーを、このロールに適した権限付与ポリシーをカスタマイズできます。

権限付与できたら、OK をクリックしてサービスの概要ページに移動します。

手順 3 関数とトリガーを作成

1. サービスの [概要] ページで 、関数の作成 をクリックします。

   関数テンプレートを選択します。

   お客様のビジネスモデルに似ているビジネステンプレートを選択できます。それを修正して関数を作成するか、空の関数テンプレートを選択して関数をカスタマイズすることができます。

   - Log Service テンプレート： Log Service は、ビジネステンプレートの `logstore_replication` と `oss - shipper - csv` を提供します。 これらのテンプレートに基づいて、関数とトリガーを作成できます。
   - 空のテンプレート：空の関数テンプレートを使用して空の関数を作成できます。 次に、ガイドページで、トリガー、関数パラメーターを設定し、関連するコードを記述して関数を作成します。
2. トリガーを設定し、次へ をクリックします。

   Log Service で提供されるテンプレートを選択すると、トリガーを直接設定できます。 空のテンプレートを選択した場合は、最初にトリガータイプを選択してからトリガーを設定する必要があります。

   トリガー名、プロジェクト名、Logstore 名などのトリガーを設定するために、必要な項目を入力します。 Function Compute の Log Service タイプトリガーは、Log Service の ETL ジョブに 対応します。

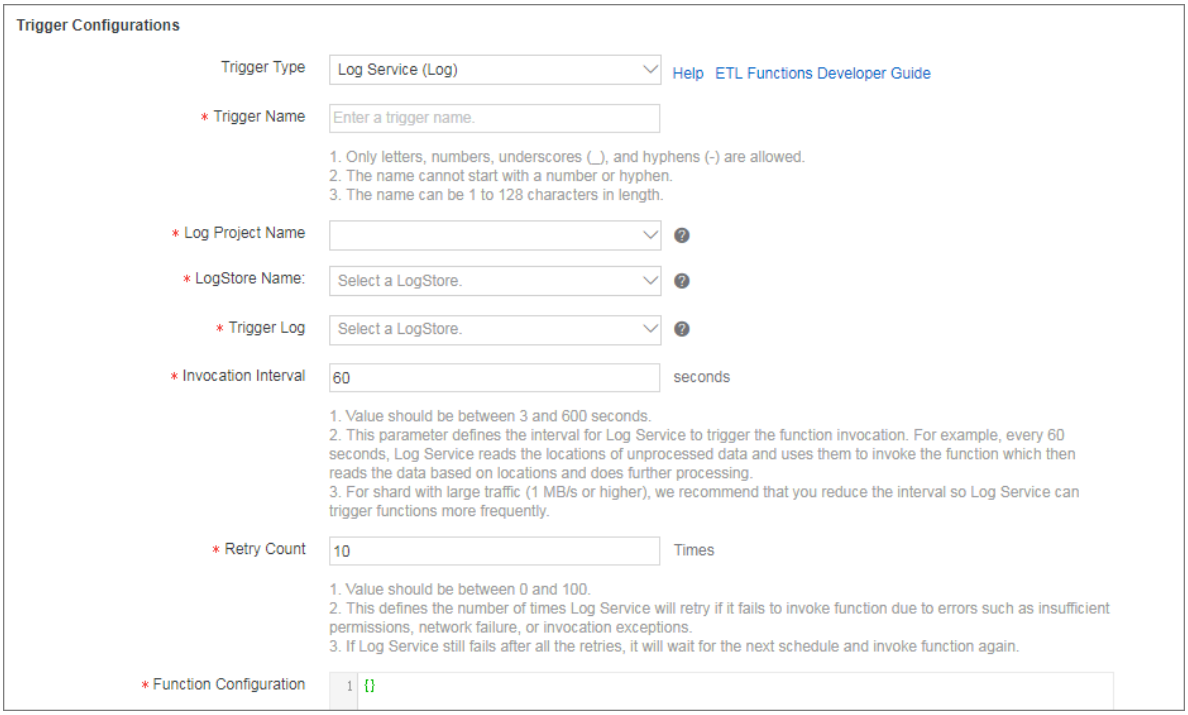
設定項目	意味	値
Trigger Name	新しいトリガーの名前	トリガー名の長さは、1〜256 バイトで、英数字、アンダースコア(_) 、ハイフン(-) を含めることができます。 数字やハイフン (-) で始めることはできません。
Project name	Log Service プロジェクトの名前	既存のプロジェクトの名前でなければなりません。 プロジェクトはサービスと同じリージョンでなければなりません。

設定項目	意味	値
Logstore name	Log Service プロジェクトの名前。 このトリガーは LogStore のサブスクライブされたデータを、カスタム処理のために Function Compute に定期的に転送します。 ETL ジョブの作成後は、このパラメーターを変更することはできません。	既存の Logstore を選択します。 Logstore は、ログプロジェクト名で選択したプロジェクトに属している必要があります。
Trigger log	Log Service は定期的に Function Compute の関数実行をトリガーします。 この Logstore には、トリガー処理中の例外と関数実行統計が記録されます。 Logstore のインデックスを作成して、後で表示することができます。	既存の Logstore の名前でなければなりません。Logstore はログプロジェクト名 で選択したプロジェクトに属している必要があります。
Invocation Interval	Log Service が関数の実行をトリガーする間隔。 たとえば、60 秒に設定すると、Log Service は各 Logstore シャードの最後の 60 秒間のデータ位置を読み取り、これを関数イベントとして使用して関数の実行を呼び出します。 この関数では、ユーザーロジックがシャードデータを読み取り、計算を実行します。 Logstore シャードのトラフィック量が多い (1 MB/秒以上) 場合は、各関数操作で処理されるデータ量が適切なサイズになるようにトリガー間隔を短く設定することを推奨します。	値の範囲は3 ～ 600 秒です。

設定項目	意味	値
Retries	Log Service が、設定されたトリガー間隔 に従って関数の実行をトリガーするときにエラーが発生した場合 (アクセス権限が不十分、ネットワーク障害、関数実行の例外など)、このパラメーターが関数を再トリガーできる最大回数を設定します。 関数が最大回数再トリガーされても、操作が引き続き失敗する場合は、 Log Service が関数の実行を再試行しようとするまでに、そのトリガー間隔が経過する必要があります。 ビジネス上のリトライの影響は、関数コード実装ロジックの内容に応じて変わります。	値の範囲は 0〜100 回です。

設定項目	意味	値
Function configuration	Log Service は、この設定コンテンツを関数イベントの一部として使用し、関数に渡します。 この関数の使用方法は、関数のカスタムロジックによって決まります。 異なる種類の関数では、関数設定の要件が異なります。 提供されている関数テンプレートの大部分については、パラメーターを入力する際に説明書をお読みください。 パラメーターが渡されない場合は、 デフォルトで {} と入力します。	設定内容は、JSON オブジェクト形式の文字列でなければなりません。

図 9-13 : トリガー設定



注 :

既に Log Service に関数呼び出し、Logstore データの読み書きする権限があること。

3. 関数名や関数ハンドラなどの基本設定を完了します。

   次へをクリックします。

4. 関数の許可を完了します。

   テンプレート許可を確認し、 ロール許可をトリガーします。 次に、 次へ をクリックします。

5. 関数情報とトリガー情報を確認します。 次に、 作成 をクリックします。

トリガーログを表示

Log Service コンソールにログインし、ジョブに設定されているトリガーログ Logstore のインデックスを作成します。タスク実行の統計が表示されます。

関数操作ログを表示

Log Service コンソールにログインして、関数実行プロセスの詳細を表示します。 詳細については、ロギングをご参照ください。

よくある質問

トリガーを作成しましたが、関数実行がトリガーされません

1. Log Service が関数の実行をトリガーできるよう、クイック認可していることを確認します。
2. シャードデータが変更されたときに、関数実行がトリガーされるように、ジョブの Logstore 内のデータが変更されていることを確認します。
3. Log Service コンソールにログインし、トリガーログおよび関数操作ログに例外がないことを確認します。

9.5 Flink で LogHub ログの読み込み

Flink log connector は、Alibaba Cloud Log Service が提供するツールで、 Flink に接続するために使用され、 コンシューマとプロデューサで構成されています。

コンシューマは、 Log Service からデータを読み込みます。 これは、一度の構文とシャードベースのロードバランシングをサポートしています。

プロデューサは Log Service にデータを書き込みます。 コネクタを使用するときは、 Maven 依存関係をプロジェクトに追加する必要があります。

```
< dependency >
            < groupId > org . apache . flink </ groupId >
            < artifactId > flink - streaming - java_2 . 11 </
 artifactId >
            < version > 1 . 3 . 2 </ version >
</ dependency >
< dependency >
            < groupId > com . aliyun . openservic  es </ groupId >
            < artifactId > flink - log - connector </ artifactId >
            < version > 0 . 1 . 7 </ version >
</ dependency >
< dependency >
            < groupId > com . google . protobuf </ groupId >
```

```
            < artifactId > protobuf - java </ artifactId >
            < version > 2 . 5 . 0 </ version >
</ dependency >
 < dependency >
            < groupId > com . aliyun . openservic  es </ groupId >
            < artifactId > aliyun - log </ artifactId >
            < version > 0 . 6 . 19 </ version >
 </ dependency >
< dependency >
            < groupId > com . aliyun . openservic  es </ groupId >
            < artifactId > log - loghub - producer </ artifactId >
            < version > 0 . 1 . 8 </ version >
</ dependency >
```

前提条件

1. アクセスキーが有効になり、プロジェクトとログストアが作成されました。 詳しい手順については、準備を参照してください。
2. サブアカウントを使用してログサービスにアクセスするには、ログストアのリソースアクセス管理（RAM）ポリシーを正しく設定していることを確実にしてください。 詳細は、RAM ユーザーに Log Service へのアクセスを許可を参照してください。

ログコンシューマ

コネクタでは、Flink ログコンシューマは、 Log Service 特定の LogStore に購読する機能を提供し、正確に 1 回の構文を実現します。 使用中に、LogStore のシャード数の変更について心配する必要はありません。

Flink の各サブタスクは、 LogStore でいくつかの断片を消費します。 LogStore 内のシャードが分割またはマージされている場合、サブタスクが消費するシャードはそれに応じて変更されます。

関連API

Flinkログコンシューマは次のAlibaba Cloud Log Service APIを使用します：

- Getcursorordata

  このAPIは、シャードからデータを引き出すために使用されます。 このAPIが頻繁に呼び出されると、データがLog Serviceのシャードクォータを超える可能性があります。 ConfigConstants.LOG_FETCH_DATA_INTERVAL_MILLISおよびConfigConstants.LOG_MAX_NUMBER_PER_FETCHを使用して、API呼び出しの時間間隔および各呼び出しによってプルされるログの数を制御できます。 シャードクォータの詳細については、シャードを参照してください。

  ```
  configProp  s . put ( ConfigCons  tants . LOG_FETCH_  DATA_INTER
  VAL_MILLIS ,  " 100 ");
  ```

```
configProp  s . put ( ConfigCons  tants . LOG_MAX_NU  MBER_PER_F
ETCH ,  " 100 ");
```

- ListShards

  このAPIは、ログストア内のすべてのシャード、及びシャードステータスのリストを取得するために使用されます。 シャードが常に分割されてマージされている場合は、APIの呼び出し期間を調整して、シャードの時間的な変化を検出できます。

```
// 30 分ごとに  ListShard  を呼び出します
configProp  s . put ( ConfigCons  tants . LOG_SHARDS  _DISCOVERY
_INTERVAL_  MILLIS ,  " 30000 ")
```

- CreateConsumerGroup

  このAPIは、消費進捗モニタリングが有効になっている場合にのみ呼び出されます。 チェックポイントを同期するためのコンシューマグループを作成するために使用されます。

- ConsumerGroupUpdateCheckPoint

  このAPIは、Flink のスナップショットを Log ServiceのConsumerGroup に同期させるために使用されます。

ユーザー権限

次の表は、サブユーザーがFlinkログコンシューマを使用するために必要な RAM 権限付与ポリシーを示しています。

API	リソース
log:GetCursorOrData	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:ListShards	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}
log:CreateConsumerGroup	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/*
log:ConsumerGroupUpdateCheckPoint	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/logstore/${logstoreName}/consumergroup/${consumerGroupName}

設定手順

1.スタートアップパラメータの設定

```
 Properties   configProp  s  =  new   Properties ();
//  Set   the   domain   to   access   Log   Service
 configProp  s . put ( ConfigCons  tants . LOG_ENDPOI  NT ,  " cn -
 hangzhou . log . aliyuncs . com ");
//  Set   the   AccessKey
 configProp  s . put ( ConfigCons  tants . LOG_ACCESS  SKEYID ,  "");
 configProp  s . put ( ConfigCons  tants . LOG_ACCESS  KEY ,  "");
//  Set   the   Log   Service   project
 configProp  s . put ( ConfigCons  tants . LOG_PROJEC  T ,  " ali - cn
 - hangzhou - sls - admin ");
//  Set   the   Log   Service   LogStore
 configProp  s . put ( ConfigCons  tants . LOG_LOGSTO  RE ,  "
 sls_consum  ergroup_lo  g ");
//  Set   the   start   position   to   consume   Log   Service
 configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
 OSITION ,   Consts . LOG_END_CU  RSOR );
//  Set   the   message   deserializ  ation   method   for   Log
 Service
 RawLogGrou  pListDeser  ializer   deserializ  er  =  new
 RawLogGrou  pListDeser  ializer ();
 final   StreamExec  utionEnvir  onment   env  =  StreamExec
 utionEnvir  onment . getExecuti  onEnvironm  ent ();
 DataStream < RawLogGrou  pList >  logTestStr  eam  =  env . addSource
 (
         new   FlinkLogCo  nsumer < RawLogGrou  pList >( deserializ
 er ,   configProp  s ));
```

上記は簡単な使用例です。 java.util.Propertiesが設定ツールとして使用されているので、すべてのコンシューマの設定はConfigConstantsにあります。

> 注：
> Flink ストリーム内のサブタスクの数は、Log Service LogStore 内のシャードの数とは無関係です。 シャードの数がサブタスクの数よりも多い場合、各サブタスクは複数のシャードを1回だけ消費します。 シャードの数がサブタスクの数より少ない場合、一部のサブタスクは新しいシャードが生成されるまでアイドルです。

2. 消費開始位置の設定

Flink ログコンシューマでシャードを消費するための開始位置を設定できます。 ConfigConstants.LOG_CONSUMER_BEGIN_POSITION を設定することで、シャードをヘッダーまたは末尾から消費するか、特定の時点で消費するかを設定できます。 具体的な値は次のとおりです：

- Consts.LOG_BEGIN_CURSOR: シャードがそのヘッダー、つまりシャードの最も早いデータから消費されることを示します。
- Consts.LOG_END_CURSOR: シャードがその末尾、つまりシャードの最新データから消費されることを示します。

- Constellation S. MAID: 特定のJavaグループから保存されたチェックポイントがconfigconstantsを通じて消費され始めることを示します。 特定のlocergroupを指定してください。
- UnixTimestamp: 1970-01-01からの秒数で表される整数値の String。 この時点からシャードが消費されたことを示します。

上記の3つの値の例は次のとおりです：

```
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION ,  Consts . LOG_BEGIN_  CURSOR );
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION ,  Consts . LOG_END_CU  RSOR );
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION , " 1512439000 ");
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ER_BEGIN_P
OSITION ,  Consts . LOG_FROM_C  HECKPOINT );
```

注：
flinkタスクを開始したときに flink 自体のステートバックエンドからのリカバリーをセットアップした場合、connector は上記の構成を無視し、statebackend に保存されたチェックポイントを使用します。

3. 消費進捗モニタリングの設定（オプション）

Flinkログコンシューマは消費の進捗モニタリングをサポートします。 消費の進行状況は、各シャードのリアルタイムの消費位置を取得することであり、これはタイムスタンプで表されます。 詳細については、コンシューマーグループ - ステータス表示とコンシューマーグループ - アラームモニタリングを参照してください。

```
configProp  s . put ( ConfigCons  tants . LOG_CONSUM  ERGROUP , "
your   consumer   group   name ");
```

注：
上記のコードはオプションです。 設定すると、コンシューマはまずコンシューマグループを作成します。コンシューマグループがすでに存在する場合、それ以上の操作は必要ありません。 コンシューマ内のスナップショットは自動的に Log Service のコンシューマグループに同期されます。 Log Service コンソールでコンシューマの消費状況を確認できます。

4. 耐災害性と 1 回限りの構文をサポート

Flink のチェックポイント機能が有効になっている場合、Flink ログコンシューマは各シャードの消費の進行状況を定期的に保存します。 ジョブが失敗すると、Flink はログコンシューマを再開し、保存されている最新のチェックポイントから消費を開始します。

チェックポイントの書き込み期間は、障害が発生した場合にロールバック（つまり再消費）されるデータの最大量を定義します。 コードは次の通りです。

```
 final   StreamExec  utionEnvir  onment   env  =  StreamExec
 utionEnvir  onment . getExecuti  onEnvironm  ent ();
//  Enable   the   exactly - once   syntax   on   Flink
 env . getCheckpo  intConfig (). setCheckpo  intingMode ( Checkpoint
 ingMode . EXACTLY_ON  CE );
//  Store   the   checkpoint   every   5s
 env . enableChec  kpointing ( 5000 );
```

Flink チェックポイントについて詳しくは、Flink の公式文書 Checkpoints を参照してください。

Log Producer

Flink ログプロデューサは、 Alibaba Cloud Log Service データを書き込みます。

> 注 :
> プロデューサは少なくとも一度の点で Flink をサポートしています。 つまり、ジョブ障害が発生した場合、 Log Service に書き込まれたデータは複製されますが、失われることはありません。

ユーザー権限

プロデューサは、ログサービスの次のAPIを使用してデータを書き込みます。

- Log: postlogstorelogs
- log:ListShards

RAM サブユーザーがプロデューサーを使用する場合は、上記の2つのAPIを許可する必要があります。

API	リソース
Log: postlogstorelogs	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/alert/${alarmName}
log:ListShards	acs:log:${regionName}:${projectOwnerAliUid}:project/${projectName}/alert/${alarmName}

手順

1. プロデューサの初期化。

   a. プロデューサの設定パラメータPropertiesを初期化します。

   これは、コンシューマの場合と似ています。 プロデューサにはいくつかのカスタムパラメータがあります。 通常は、これらのパラメータをデフォルト値に設定します。 特別なシナリオで値をカスタマイズできます。

```
//データの送信に使用された I / O スレッドの数 デフォルト値は 8 です。
 ConfigCons tants . LOG_SENDER _IO_THREAD _COUNT
//ログデータがキャッシュされた時刻。 デフォルト値は 3000 です。
 ConfigCons tants . LOG_PACKAG E_TIMEOUT_ MILLIS
//キャッシュされたパッケージのログ数 デフォルト値は 4096 です。
 ConfigCons tants . LOG_LOGS_C OUNT_PER_P ACKAGE
//キャッシュされたパッケージのサイズ デフォルト値は 3Mb です。
 ConfigCons tants . LOG_LOGS_B YTES_PER_P ACKAGE
//ジョブが使用できる合計メモリサイズ。 デフォルト値は 100Mb です。
 ConfigCons tants . LOG_MEM_PO OL_BYTES
```

   上記のパラメータは必須ではありません。 デフォルト値のままでかまいません。

   b. LogSerializationSchema をリロードして、データを RawLogGroup にシリアル化するためのメソッドを定義します。

   RawLogGroup はログのコレクションです。 各フィールドの意味についての詳細は、Data modelを参照してください。

   ログサービスのshardHashKey関数を使用するには、データが書き込まれるシャードを指定します。 Log Partitionerを次のように使用してデータのハッシュキーを生成できます：

   例：

```
FlinkLogPr oducer < String > logProduce r = new
FlinkLogPr oducer < String >( new SimpleLogS erializer (),
configProp s );
logProduce r . setCustomP artitioner ( new LogPartiti oner <
String >() {
     // Generate a 32 - bit hash value
      public String getHashKey ( String element ) {
          try {
              MessageDig est md = MessageDig est .
getInstanc e (" MD5 ");
              md . update ( element . getBytes ());
              String hash = new BigInteger ( 1 , md .
digest ()). toString ( 16 );
              while ( hash . length () < 32 ) hash = " 0 " +
hash ;
              return hash ;
          } catch ( NoSuchAlgo rithmExcep tion e ) {
          }
           return " 0000000000 0000000000 0000000000
0000000000 0000000000 0000000000 0000 ";
```

```
        }
    });
```

注：

LogPartitioner はオプションです。 このパラメーターが設定されていない場合、データはランダムにシャードに書き込まれます。

2. **次の使用例では、シミュレーションによって生成された文字列をLog Serviceに書き込みます。**

```
//  Log   Service のデータ形式にデータをシリアル化する
 class   SimpleLogS  erializer   implements   LogSeriali
 zationSche  ma < String > {
     public   RawLogGrou  p   serialize ( String   element ) {
         RawLogGrou  p   rlg  =  new   RawLogGrou  p ();
         RawLog   rl  =  new   RawLog ();
         rl . setTime (( int )( System . currentTim  eMillis () /
 1000 ));
         rl . addContent (" message ",   element );
         rlg . addLog ( rl );
         return   rlg ;
    }
}
 public   class   ProducerSa  mple  {
     public   static   String   sEndpoint  = " cn - hangzhou . log .
 aliyuncs . com ";
     public   static   String   sAccessKey  Id  = "";
     public   static   String   sAccessKey  = "";
     public   static   String   sProject  = " ali - cn - hangzhou -
 sls - admin ";
     public   static   String   sLogstore  = " test - flink -
 producer ";
     private   static   final   Logger   LOG  =  LoggerFact  ory .
 getLogger ( ConsumerSa  mple . class );
     public   static   void   main ( String []  args )  throws
 Exception  {
         final   ParameterT  ool   params  =  ParameterT  ool .
 fromArgs ( args );
         final   StreamExec  utionEnvir  onment   env  =
 StreamExec  utionEnvir  onment . getExecuti  onEnvironm  ent ();
         env . getConfig (). setGlobalJ  obParamete  rs ( params );
         env . setParalle  lism ( 3 );
         DataStream < String >  simpleStri  ngStream  =  env .
 addSource ( new   EventsGene  rator ());
         Properties   configProp  s  =  new   Properties ();
        //ログサービスへのアクセスに使用されるドメインの名前を設定します。
         configProp  s . put ( ConfigCons  tants . LOG_ENDPOI  NT ,
  sEndpoint );
        //ログサービスにアクセスするように AccessKey を設定します
         configProp  s . put ( ConfigCons  tants . LOG_ACCESS
 SKEYID ,   sAccessKey  Id );
         configProp  s . put ( ConfigCons  tants . LOG_ACCESS  KEY
 ,   sAccessKey );
        //ログが書き込まれる Log   Service プロジェクトを設定します
         configProp  s . put ( ConfigCons  tants . LOG_PROJEC  T ,
 sProject );
        //ログが書き込まれるログサービス LogStore を設定します
         configProp  s . put ( ConfigCons  tants . LOG_LOGSTO  RE ,
  sLogstore );
```

```
        FlinkLogPr  oducer < String >  logProduce  r  =  new
FlinkLogPr  oducer < String >( new   SimpleLogS  erializer (),
configProp  s );
        simpleStri  ngStream . addSink ( logProduce  r );
        env . execute (" flink   log   producer ");
    }
    //ログ生成をシミュレートする
     public   static   class   EventsGene  rator   implements
SourceFunc  tion < String > {
        private   boolean   running  =  true ;
       @ Override
        public   void   run ( SourceCont  ext < String >  ctx )
throws   Exception  {
           long   seq  =  0 ;
           while  ( running ) {
               Thread . sleep ( 1000 );
               ctx . collect (( seq ++) + "-" +  RandomStri
ngUtils . randomAlph  abetic ( 12 ));
           }
       }
       @ Override
        public   void   cancel () {
           running  =  false ;
       }
    }
}
```

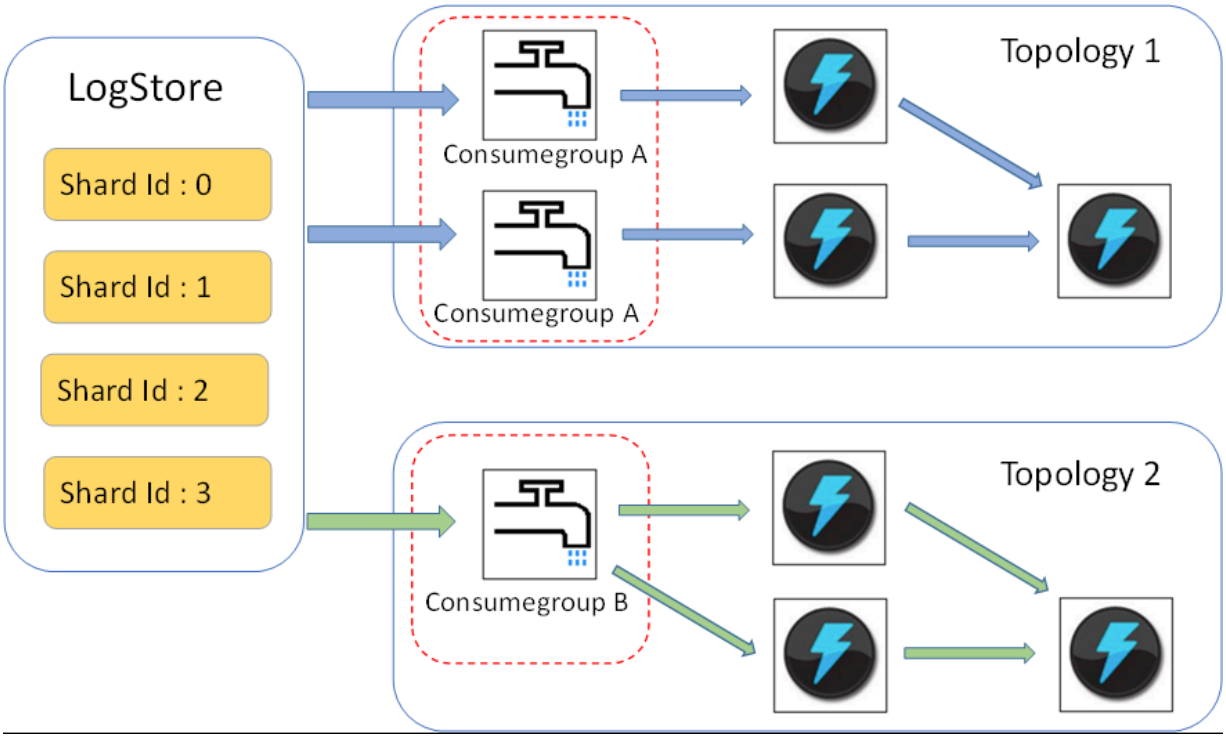
9.6 Storm で LogHub ログの読み込み

Log Service の LogHub は、Logtail と SDK を使用して、リアルタイムにログデータを収集するための効率的で信頼性の高いログチャネルを提供します。 ログを収集した後、Spark Stream や Storm といったリアルタイムシステムを使用して LogHub に書き込まれたデータを使用することができます。

Log Service は、LogHub Storm spout を使用して LogHub からリアルタイムでデータを読み取り、Storm ユーザーの LogHub 消費コストを削減します。

基本的なアーキテクチャとプロセス

図 9-14 : 基本的なアーキテクチャとプロセス



- 上の図では、LogHub Storm 吐き出し口が赤い点線で囲まれています。 各 Storm トポロジには、Logstore からすべてのデータを読み取るスパウトのグループがあります。 異なるトポロジーのスパウトはお互いに影響しません。
- 各トポロジは、一意の LogHub コンシューマグループ名によって識別されます。 同じトポロジのスパウトは、コンシューマライブラリを使用して負荷分散と自動フェールオーバーを実現します。
- Spouts はリアルタイムで LogHub からデータを読み取り、トポロジーのボルトノードにデータを送信し、Checkpoint として定期的に消費端点を LogHub に保存します。

制限

- 誤使用を防ぐため、各ログストアは最大 5 つのコンシューマグループをサポートしています。 Java SDK の DeleteConsumerGroup インタフェースを使用して、未使用のコンシューマグループを削除できます。 不要なコンシューマグループを削除できます。
- スパウトの数はシャードの数と同じであることを推奨します。 それ以外の場合、単一のスパウトが大量のデータを処理しないことがあります。
- シャードに単一スパウトの処理能力を超える大量のデータが含まれている場合は、シャード分割インターフェースを使用してシャードを分割し、各シャードのデータ量を減らすことができます。

- Storm への依存 LogHub スパウトでスパウトがメッセージを正しくボルトに送信することを確認するには、ACK が必要です。 したがって、ボルトは確認のために ACK を呼び出す必要です。

使用例

- Spout (トポロジの構築に使用) topology)

```
     public   static   void   main (  String []  args  )
   {
        String   mode  = " Local "; //  Use   the   local   test
mode .
          String   conumser_g  roup_name  = ""; //  Specify   a
unique   consumer   group   name   for   each   topology .  The
value   cannot   be   empty .  The   value   can   be   3 - 63
characters   long ,  contain   lowercase   letters ,  numbers ,
hyphens  (-),  and   underscore  s  ( _ ),  and   must   begin
and   end   with   a   lowercase   letter   or   number .
        String   project  = ""; //  The   Log   Service   project
.
        String   logstore  = ""; //  The   Log   Service   Logstore
.
        String   endpoint  = "";   //  Domain   of   the   Log
Service
        String   access_id  = "";  //  User ' s   access   key
        String   access_key  = "";
       //  Configurat  ions   required   for   building   a
LogHub   Storm   spout .
        Loghubspou  tconfig   Config  =  new   loghubspou  tconfig
( conumser_g  roup_name ,
                endpoint ,  project ,  logstore ,  access_id ,
                access_key ,  LogHubCurs  orPosition . END_CURSOR
);
        Topologybu  ilder   builder  =  new   topologybu  ilder
();
       // 构建  loghub   storm   spout
        Loghubspou  t   spin  =  new  ( config  );
       //  The   number   of   spouts   can   be   the   same   as
  that   of   Logstore   shards   in   actual   scenarios .
        builder . setSpout (" spout ",  spout ,  1 );
        builder . setBolt (" exclaim ",  new   SampleBolt ()).
shuffleGro  uping (" spout ");
        Config   conf  =  new   Config ();
        conf . setDebug ( false );
        conf . setMaxSpou  tPending ( 1 );
       //  The   serializat  ion   method   LogGroupDa  taSerializ
Serializer   of   LogGroupDa  ta   must   be   configured
explicitly   when   Kryo   is   used   for   data   serializat  ion
  and   deserializ  ation .
        Config . registerSe  rializatio  n ( conf ,  LogGroupDa  ta
. class ,  LogGroupDa  taSerializ  Serializer . class );
        if  ( mode . equals (" Local ")) {
            logger . info (" Local   mode ...") ;
            LocalClust  er   cluster  =  new   LocalClust  er ();
            cluster . submitTopo  logy (" test - jstorm - spout ",
conf ,  builder . createTopo  logy ());
            try  {
                Thread . sleep ( 6000  *  1000 ); // waiting   for
  several   minutes
            }  catch  ( Interrupte  dException   e ) {
                //  TODO   Auto - generated   catch   block
```

```
                e . printStack  Trace ();
            }
             cluster . killTopolo  gy (" test - jstorm - spout ");
             cluster . shutdown ();
        }  else   if  ( mode . equals (" Remote ")) {
             logger . info (" Remote   mode ...");
             conf . setNumWork  ers ( 2 );
             try  {
                 StormSubmi  tter . submitTopo  logy (" stt - jstorm
 - spout - 4 ",  conf ,  builder . createTopo  logy ());
             }  catch  ( AlreadyAli  veExceptio  n   e ) {
                 //  TODO   Auto - generated   catch   block
                 e . printStack  Trace ();
             }  catch  ( InvalidTop  ologyExcep  tion   e ) {
                 //  TODO   Auto - generated   catch   block
                 e . printStack  Trace ();
             }
        }  else  {
             logger . error (" invalid   mode : " +  mode );
        }
    }
}
```

- 次のボルトコードの例では、データを消費し、各ログの内容のみを出力します。

```
public   class   SampleBolt   extends   BaseRichBo  lt  {
    private   static   final   long   serialVers  ionUID  =
4752656887  774402264L ;
    private   static   final   Logger   logger  =  Logger .
getLogger ( BaseBasicB  olt . class );
    private   OutputColl  ector   mCollector ;
   @ Override
    public   void   prepare (@ SuppressWa  rnings (" rawtypes ")
Map   stormConf ,  TopologyCo  ntext   context ,
            OutputColl  ector   collector ) {
        mCollector  =  collector ;
   }
   @ Override
    public   void   execute ( Tuple   tuple ) {
        String   shardId  = ( String )  tuple
                . getValueBy  Field ( LogHubSpou  t . FIELD_SHAR
D_ID );
       @ SuppressWa  rnings (" unchecked ")
        List < LogGroupDa  ta >  logGroupDa  tas  = ( ArrayList
< LogGroupDa  ta >)  tuple . getValueBy  Field ( LogHubSpou  t .
FIELD_LOGG  ROUPS );
        for  ( LogGroupDa  ta   groupData  :  logGroupDa  tas ) {
           //  Each   LogGroup   consists   of   one   or   more
logs .
            LogGroup   logGroup  =  groupData . GetLogGrou  p ();
            for  ( Log   log  :  logGroup . getLogsLis  t ()) {
                StringBuil  der   sb  =  new   StringBuil  der ();
               //  Each   log   has   a   time   field   and
multiple   key :  value   pairs ,
                int   log_time  =  log . getTime ();
                sb . append (" LogTime :"). append ( log_time );
                for  ( Content   content  :  log . getContent
sList ()) {
                    sb . append ("\ t "). append ( content . getKey
()). append (":")
                            . append ( content . getValue ());
                }
                logger . info ( sb . toString ());
```

```
            }
        }
        // The dependency on the Storm ACK mechanism
   is mandatory in LogHub spouts to confirm that
 spouts send messages correctly
        // to bolts . Therefore , bolts must call ACK
 for such confirmati on .
         mCollector . ack ( tuple );
    }
    @ Override
     public void declareOut putFields ( OutputFiel dsDeclarer
   declarer ) {
        // do nothing
    }
}
```

Maven

Storm 1.0 より前のバージョン（たとえば、0.9.6）では、次のコードを使用します。

```
< dependency >
  < groupId > com . aliyun . openservic es </ groupId >
  < artifactId > loghub - storm - spout </ artifactId >
  < version > 0 . 6 . 5 </ version >
</ dependency >
```

Use the following code for Storm 1.0 and later versions:

```
< dependency >
  < groupId > com . aliyun . openservic es </ groupId >
  < artifactId > loghub - storm - 1 . 0 - spout </ artifactId >
  < version > 0 . 1 . 2 </ version >
</ dependency >
```

9.7 Spark Streaming で LogHub ログの読み込み

E-MapReduce は、Spark Streaming を使用してリアルタイムで LogHub ログを使用するユニバーサルインターフェイスを提供します。 詳細については、GitHubをご参照ください。

9.8 CloudMonitor で LogHub ログの読み込み

CloudMonitorは LogHub の Logstore データを直接使用して、以下をモニタリングできます。

- ログ内のキーワードでアラーム
- QPS と RT の統計 (時間単位)
- PV と UV の統計 (時間単位)

10 データ転送

10.1 概要

Log Service がログソースに接続されると、Log Service によってリアルタイムにログは収集され、コンソールや SDK/API を介してログを読み込み、ログを転送することができるようになります。LogHub に収集されたログは、リアルタイムに OSS (Object Storage Service) や Table Store といった Alibaba Cloud のストレージプロダクトに転送されます。ログ転送の設定は、コンソールより行います。なお、LogShipper には、ステータスを確認することのできる API が用意されており、API を自動的に再試行させることもできます。

シナリオ

データウェアハウスとの相互接続

ログソース

Log Service の LogShipper は、LogHub に収集されたログを転送します。ログが生成されると、Log Service は生成されたログをリアルタイムに収集し、他のクラウドプロダクトに転送し、格納され、分析に利用できるようになります。

目標

- OSS (大規模オブジェクトストレージ)
  - OSS へのログ送信
  - OSS のフォーマットは Hive を使って処理できます。E-MapReduce をお勧めします。
- Table Store (NoSQL データストレージサービス)
- Maxcompute (大規模データコンピューティングサービス):
  - DataWorks のデータ統合機能でデータ転送 DataWorks を使用して MaxCompute にデータを転送

10.2 OSS へのログ転送

10.2.1 OSS へのログ転送

Log Service によって Logstore データを自動的に Object Storage Service (OSS) にアーカイブすることにより、ログをより活用できるようになります。

- OSS に格納されたログのライフサイクルを長期に設定することができます。

- OSS に格納されているデータは、作成したプログラムや E-MapReduce といった他のシステムから読み込むことができます。

機能の利点

Log Service より OSS にログを転送する場合には、次の利点があります。

- 使いやすい: コンソールより、Log Service の Logstore データを OSS に同期させる設定を行うことができます。
- 効率化: Log Service でログ収集することにより、複数マシンのログを一元管理することができます。各マシンのログをそれぞれ OSS にインポートする必要がなくなります。
- 管理が容易: Log Service でグループ化したログをそのまま OSS にログ転送することができます。さまざまなプロジェクトや Logstore のログを、それぞれ別々の OSS バケットディレクトリに自動転送することができるため、OSS データの管理が容易になります。

前提条件

1. Log Service を有効にし、プロジェクトおよび Logstore を作成しており、ログデータが収集されていること。
2. OSS を有効にして、 Log Service プロジェクトと同じリージョンにバケットを作成していること。
3. RAM (Resource Access Management) が有効になっていること。
4. Log Service プロジェクトと OSS バケットが、同じリージョンにあること。なお、リージョン間のデータ転送には対応していません。

手順

ステップ 1. RAM (Resource Access Management) での権限付与

タスクを実行する前に、Log Service に OSS への書き込み権限を付与します。

RAM クイック許可ページに移動し、表示されたページで同意するをクリックします。Log Service に、OSS への書き込み権限が付与されます。

注 :

- ポリシーを変更してアカウント間の転送タスクを設定する方法については、「OSS へのログ転送 - RAM で権限付与」をご参照ください。
- 転送タスクを実行できるようサブアカウント (RAM ユーザー) に権限を付与する方法については、RAM ユーザーに Log Service へのアクセス権を付与をご参照ください。

ステップ 2 Log Service で OSS 転送ルールを設定

1. Log Service コンソールにログインします。
2. プロジェクト一覧ページでプロジェクト名をクリックします。
3. Logstore を選択し、左側のナビゲーションメニューより LogShipper - 転送 の OSSをクリックします。
4. 有効化をクリックします。OSS LogShipper を設定し、確認をクリックします。

   OSS 転送を設定する際は、下表をご参照ください。

設定項目	説明	指定可能な値
OSS 転送名	OSS 転送の名前	名前は 3～63 字で、英小文字、数字、ハイフン(-) 、アンダースコア (_) を含めることができ、先頭および末尾は英小文字または数字にする必要があります。
OSS Bucket	OSS バケット名	既存のバケット名 (Log Service プロジェクトと同じリージョンにある OSS バケットを指定)
OSS 接頭辞	OSS の接頭辞 (Log Service から OSS に同期転送されたデータを格納するバケットディレクトリ)	既存の OSS 接頭辞
パーティション形式	LogShipper タスクによって生成されるパーティション名の形式は、%Y、%m、%d、%H、%M を用いて指定します。OSS に書き込まれるオブジェクトファイルのディレクトリ階層の定義です。スラッシュ (/) がルートディレクトリです。OSS 接頭辞およびパーティション形式を用いて OSS に格納されるパーティションファイルのファイルパスを定義する方法は下表のとおりです。	書式の詳細は、「Strptime API」を参照

設定項目	説明	指定可能な値
RAM ロール	ロールの Arn およびロール名 (RAM ロールはアクセス制御に使用されます。OSS バケット所有者がロールの ID を作成します。RAM ロールの ARN は、該当 RAM ロールの基本情報で確認できます。)	例: `acs : ram :: 45643 : role / aliyunlogd efaultrole`
転送サイズ	圧縮されていない状態での OSS オブジェクトの最大サイズを設定 (LogShipper タスクの生成間隔は自動調整されます。)	5 ～ 256 (単位: MB)
ストレージ形式	OSS に転送するログデータの保存形式	JSON 形式、Parquet 形式、CSV 形式の 3 つのいずれか
圧縮	OSS データストレージの圧縮の有効化/無効化	・ 圧縮しない: raw データは圧縮されません。 ・ 圧縮 (snappy): snappy（https://google.github.io/snappy/）アルゴリズムでデータを圧縮し、OSS バケットストレージスペースの使用を減らします。

設定項目	説明	指定可能な値
転送時間	LogShipper タスクの間隔	300 ～ 900 (単位: 秒、初期値: 300)

図 10-1 : ログの転送

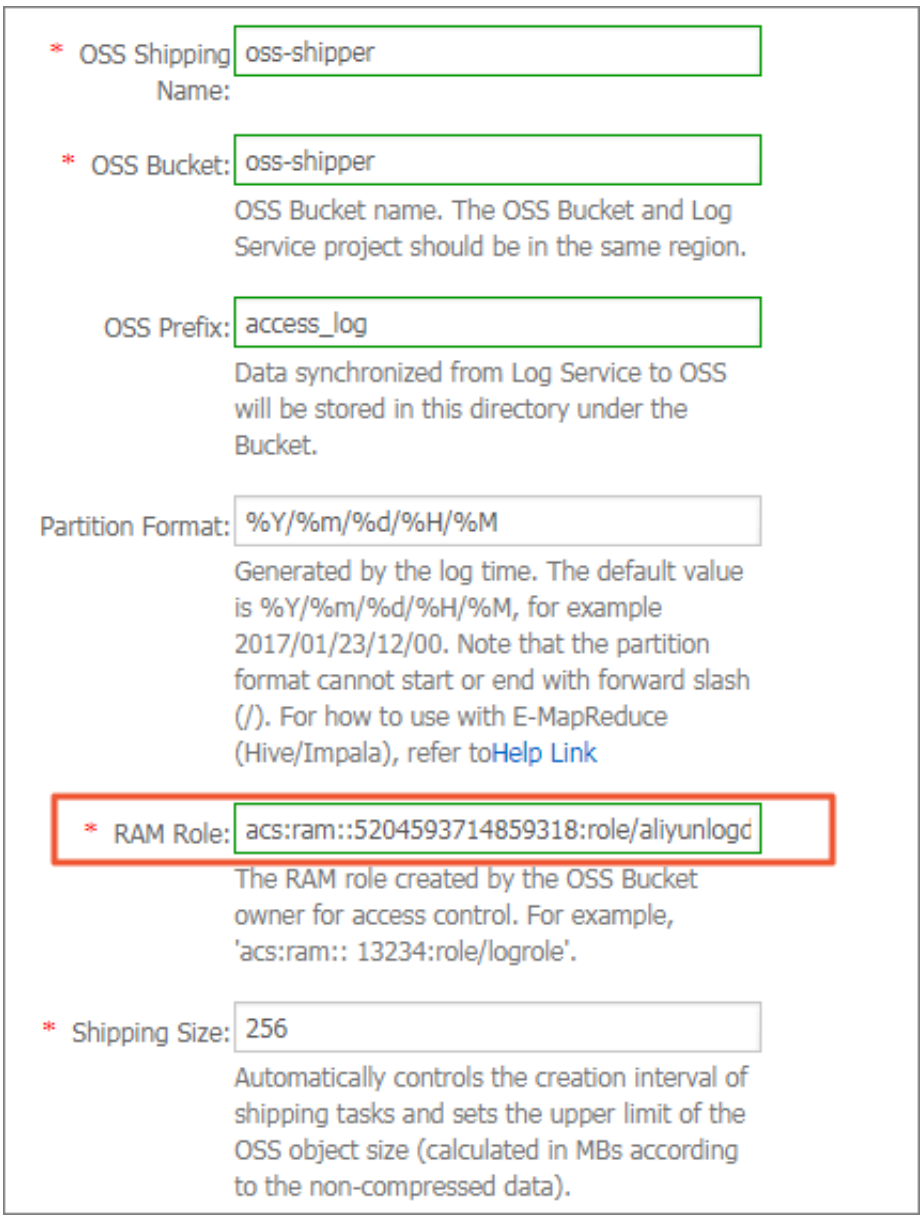
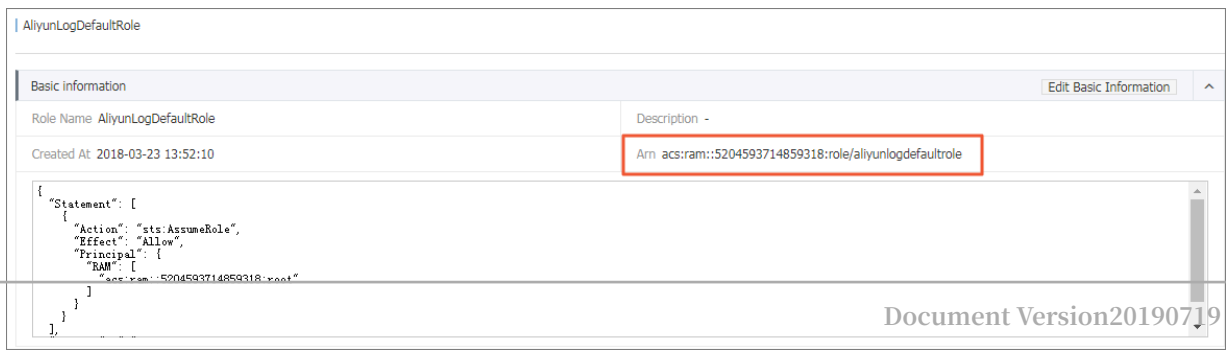


図 10-2 : ロールの arn



 注：

Log Service はバックエンドで、データを転送して各シャードに並行して書き込みます。転送するデータのサイズと時間を基に、データ転送タスクの生成される頻度は決定されます。設定されている条件に合致した場合に、転送タスクは生成されます。

パーティション形式

各 LogShipper タスクは、`oss://OSS-BUCKET/OSS-PREFIX/PARTITION-FORMAT_RANDOM-ID` のパス形式で OSS ファイルに書き込まれます。2017-01-20 19:50:43 に生成された LogShipper タスクを例に、パーティション形式の使用方法を説明します。

OSS バケット	OSS 接頭辞	パーティション形式	OSS ファイルパス
test-bucket	test-table	%Y/%m/%d/%H/%M	`oss://test-bucket/test-table/2017/01/20/19/50/43_1484913433515253_2850008`
test-bucket	log_ship_oss_example	year=%Y/mon=%m/day=%d/log_%H%M%s	`oss://test-bucket/log_ship_oss_example/year=2017/mon=01/day=20/log_195043_1484913043351525351_2850008.parquet`

OSS バケット	OSS 接頭辞	パーティション形式	OSS ファイルパス
test-bucket	log_ship_oss_example	ds=%Y%m%d/%H	oss :// test - bucket / log_ship_oss_example / ds = 20170120 / 19_1484913043351525351_2850008 . snappy
test-bucket	log_ship_oss_example	%Y%m%d/	oss :// test - bucket / log_ship_oss_example / 20170120 / _1484913043351525351_2850008
test-bucket	log_ship_oss_example	%Y%m%d%H	oss :// test - bucket / log_ship_oss_example / 2017012019 _1484913043351525351_2850008

OSS データは、Hive や MaxCompute といったビッグデータプラットフォームで分析します。パーティションデータを使用するには、ディレクトリの各階層を「キー=値」の形式にして指定します (Hive 形式の パーティション)。

たとえば、「oss://test-bucket/log_ship_oss_example/year=2017/mon=01/day=20/log_195043_1484913043351525351_2850008」。

Parquet は、年、月、日の 3 つのレベルのパーティションに設定できます。

LogShipper タスク管理

LogShipper 機能を有効にすると、Log Service は定期的にバックエンドの LogShipper タスクを開始します。コンソールに LogShipper タスクのステータスはコンソールに表示されます。LogShipper のタスク管理では、以下が行えます。

- 過去 2 日間の LogShipper タスクをすべて表示し、ステータスを確認します。 LogShipper タスクのステータスは、 Success 、 Failed 、 Running です。 ステータス Failed は、LogShipper タスクが外部の理由でエラーが発生したため、再試行できないことを示します。この場合、手動で問題を解決する必要があります。
- 2 日以内に作成された LogShipper タスクで失敗したものについては、障害の外部要因をタスクリストに表示できます。外部エラーを修正したら、その失敗タスクをそれぞれ、またはバッチで再試行できます。

手順

1. Log Service コンソールにログインします。
2. プロジェクト一覧ページでプロジェクト名をクリックします。
3. Logstore を選択し、左側のナビゲーションメニューより LogShipper - 転送、OSS を順にクリックします。

   タスクの開始時間、タスクの終了時間、ログの受信時間、データ行、タスクのステータスといった情報を表示します。

   LogShipper タスクが失敗すると、そのエラーメッセージがコンソールに表示されます。デフォルトでは、ポリシーに基づいてタスクが再試行されます。 タスクを手動で再試行することもできます。

タスクを再試行

一般に、ログデータは Logstore に書き込まれてから 30 分以内に OSS に同期されます。

Log Service はアニーリングポリシーに基づき、デフォルトでは、過去 2 日間のタスクを再試行します。再試行の最小間隔は 15 分です。1 度失敗したタスクは 15 分後に再試行され、2 度失敗したタスクは 30 分後 (2 x 15 分) に再試行され、 3 度失敗したタスクは 60 分後に再試行されます (2 x 30 分)。

失敗したタスクをただちに再試行するには、コンソールよりすべての失敗したタスクを再試行をクリックするか、タスクを指定して API/SDK で再試行します。

失敗したタスクのエラー

タスクの失敗は引き起こす次の一般的なエラーを参照してください。

エラーメッセージ	エラー原因	対処法
UnAuthorized	権限がない	以下を確認します。 ・ OSS ユーザーがロールを作成していること ・ ロールの説明のアカウント ID が正しいこと ・ ロールに OSS バケットへの書き込み権限があること ・ role-arn が正しく設定されていること
ConfigNotExist	設定情報がない	本エラーは、通常、転送ルールが削除されている場合に発生します。転送ルールを設定し直し、再度タスクを実行します。
InvalidOssBucket	存在しない OSS バケット	以下を確認します。 ・ OSS バケットと Log Service プロジェクトが同じリージョンにあること ・ バケット名が正しく設定されていること
InternalServerError	Log Service の内部エラー	タスクを再試行します。

OSS データストレージ

コンソールまたは API/SDK を使用して、OSS データにアクセスできます。

コンソールより OSS データにアクセスするには、OSS コンソールにログインし、左側のナビゲーションメニューよりバケット名をクリックします。

OSS の詳細については、OSS のドキュメントをご参照ください。

オブジェクトのアドレス

```
oss ://  OSS - BUCKET / OSS - PREFIX / PARTITION - FROMAT_RAN  DOM -
ID
```

・ パスの説明

- OSS-BUCKET には OSS バケット、OSS-PREFIX はそれぞれ、指定した OSS バケットおよびディレクトリ接頭辞です。INCREMENTID は、 システムによって追加される乱数です。
- PARTITION-FORMAT は、%Y/%m/%d/%H/%%M と定義されています。%Y、%m、%d、%H、および %M は、それぞれ年、月、日、時、および分を示します。Log Service の LogShipper タスクが生成される時間は、strptime API を用いて算出されます。
- RANDOM-ID は LogShipper タスクの一意の識別子です。

・ ディレクトリ時間

OSS データディレクトリは、LogShipper タスクが生成された時間に設定されます。5 分ごとにデータが OSS に転送されると仮定します。「2016-06-23 00:00:00」に作成された LogShipper タスクは、「2016-06-22 23:55」以降に Log Service に書き込まれたデータを転送します。「2016-06-22」の終日のログを分析するには、`2016 / 06 / 23 / 00` に含まれるすべてのオブジェクトに加え、`2016 / 06 / 23 / 00` ディレクトリの最初の 10 分間に「2016-06-22」のログが含まれているかどうかを確認する必要があります。

オブジェクトのストレージフォーマット

・ JSON

詳細については、JSON ストレージをご参照ください。

・ Parquet

詳細については、Parquet ストレージをご参照ください。

・ CSV

詳細については、CSV ストレージをご参照ください。

10.2.2 JSON ストレージ

本ドキュメントでは、Log Service ログを JSON 形式で OSS (Object Storage Service) に格納する方法について説明します。OSS へのログ転送の詳細については、OSS へのログ転送をご参照ください。

OSS ファイルの圧縮タイプとファイルアドレスは次のとおりです。

圧縮タイプ	ファイル拡張子	OSS ファイルのアドレス例
非圧縮	なし	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937
圧縮 (snappy)	.snappy	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy

非圧縮

オブジェクトは複数のログからなります。ファイルの各行は JSON 形式のログです。次の例をご参照ください。

```
{" __time__ ": 1453809242 ," __topic__ ":"," __source__ ":" 10 . 170
 . ***.***"," ip ":" 10 . 200 . **.***"," time ":" 26 / Jan / 2016 :
 19 : 54 : 02  + 0800 "," url ":" POST
              / PutData ?  Category = YunOsAccou  ntOpLog &
 AccessKeyI  d =< yourAccess  KeyId >& Date = Fri % 2C % 2028 % 20Jun
 % 202013 % 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature =<
 yourSignat  ure >
               HTTP / 1 . 1 "," status ":" 200 "," user - agent ":"
 aliyun - sdk - java "}
```

圧縮 (Snappy)

Snappy C ++ (Snappy.Compress メソッド) で非圧縮ファイルを圧縮します。`. snappy` ファイルを解凍すると、非圧縮ファイルを入手できます。

C ++ Lib で解凍

Snappy 公式 Web サイトより Lib をダウンロードし、Snappy.Uncompress メソッドを使用して .snappy ファイルを解凍します。

Java Lib で解凍

xerial snappy-java の Snappy.Uncompress または Snappy.SnappyInputStream で解凍します (SnappyFramedInputStream には非対応)。

```
< dependency >
< groupId > org . xerial . snappy </ groupId >
< artifactId > snappy - java </ artifactId >
< version > 1 . 0 . 4 . 1 </ version >
< type > jar </ type >
< scope > compile </ scope >
</ dependency >
```

 注 :

バージョン 1.1.2.1 はバグにより、圧縮ファイルの一部は解凍されません。バージョン 1.1.2.6 で修正されています。

Snappy.Uncompress

```
String fileName = " C :\\ My download \\ 36_1474212 9631886006
84_4451886 . snappy ";
RandomAcce ssFile randomFile = new RandomAcce ssFile (
fileName , " r ");
int fileLength = ( int ) randomFile . length ();
randomFile . seek ( 0 );
byte [] bytes = new byte [ fileLength ];
int byteread = randomFile . read ( bytes );
System . out . println ( fileLength );
System . out . println ( byteread );
byte [] uncompress ed = Snappy . uncompress ( bytes );
String result = new String ( uncompress ed , " UTF - 8 ");
System . out . println ( result );
```

Snappy.SnappyInputStream

```
String fileName = " C :\\ My download \\ 36_1474212 9631886006
84_4451886 . snappy ";
SnappyInpu tStream sis = new SnappyInpu tStream ( new
FileInputS tream ( fileName ));
byte [] buffer = new byte [ 4096 ];
int len = 0 ;
while (( len = sis . read ( buffer )) ! = - 1 ) {
    System . out . println ( new String ( buffer , 0 , len ));
}
```

Linux 環境での解凍ツール

Linux 環境には、.snappy ファイルの解凍ツールが用意されています。 こちらをクリックして snappy_tool をダウンロードします。

```
./ snappy_too l 03_1453457 0065480787 22_44148 . snappy
 03_1453457 0065480787 22_44148
 compressed . size : 2217186
 snappy :: Uncompress return : 1
 uncompress ed . size : 25223660
```

10.2.3 CSV ストレージ

本ドキュメントでは、Log Service ログを CSV 形式で OSS (Object Storage Service) に格納する方法について説明します。OSS にログを送信する方法については、「OSS へのログ送信」をご参照ください。

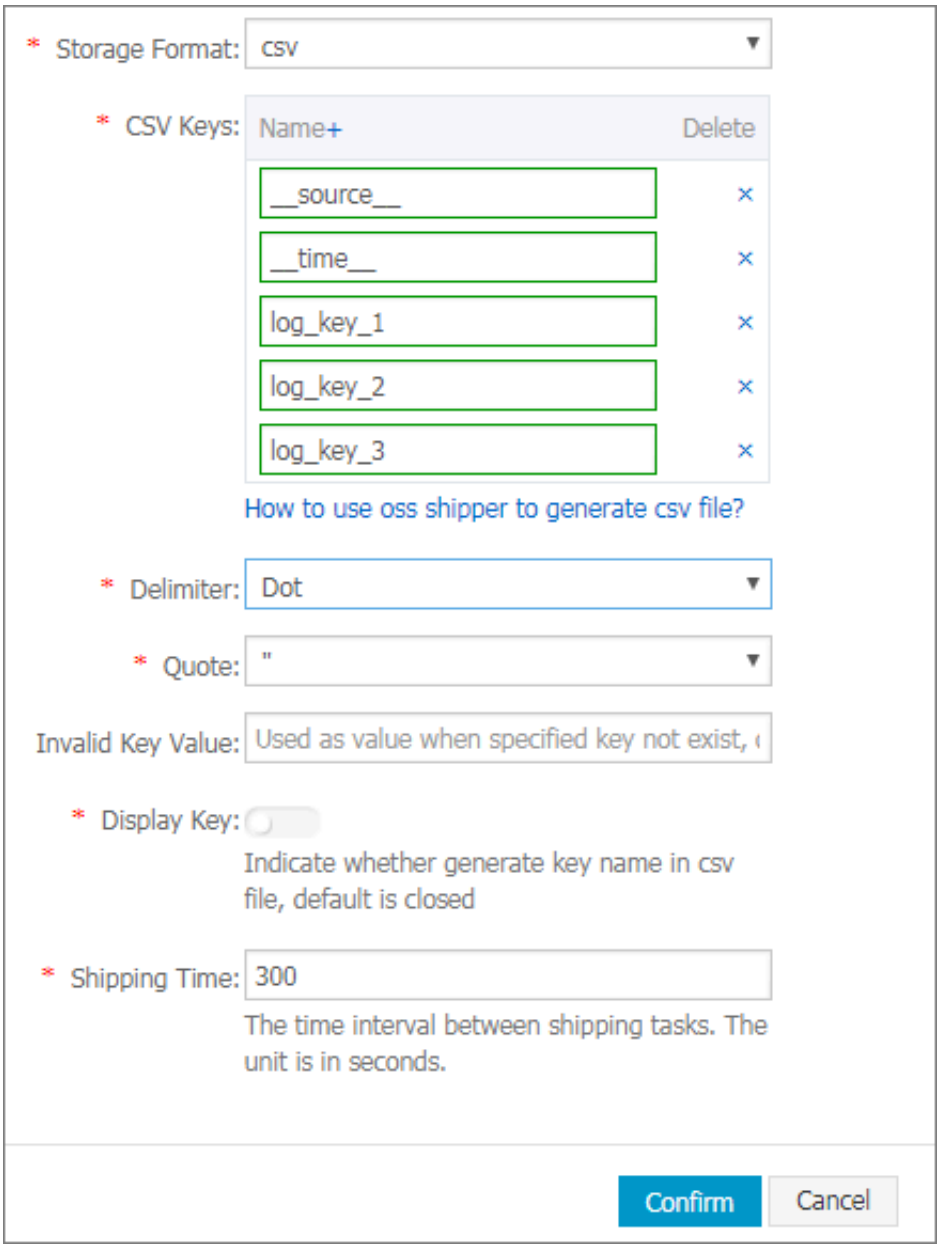
CSV 形式のストレージフィールドを設定

設定ページ

Log Service データプレビューページまたはインデックスクエリページには、ログの複数のキーと値のペアを表示されます。OSS に送信するフィールド名 (キー) を順番に入力します。

ログに指定したキーがない場合、そのカラムは NULL になります。

図 10-3 : 設定項目



設定項目

設定項目	値	備考
区切り文字	文字	各フィールドを区切るための 1 文字 (String 型)。

設定項目	値	備考
引用符	文字	1 文字 (String 型)。フィールド値に区切り文字または改行が含まれる場合、フィールド値を引用符で囲み、データ読み取り時にフィールド値が不適切に分割されないようにします。
エスケープ	文字	1 文字 (String 型)。初期値は引用符に指定した値です。現時点では、設定不可。フィールド値に引用符 (エスケープ文字としてではなく、通常の文字として使用) を含む場合、その引用符の前にエスケープ文字を追加する必要があります。
無効なキー値	文字列	指定されたキーに値がない場合、フィールドには本項目で指定した文字列が入り、フィールド値は NULL とみなされます。
キーヘッダーの表示	Boolean	CSV ファイルの最初の行にフィールド名を追加するかどうかを指定します。

詳細については、CSV 標準 (英文) およびPostgreSQL CSV の説明 (英文) をご参照ください。

設定可能な予約フィールド

ログのキーと値のペアのほか、Log Service は OSS にログを転送する際に下表の予約フィールドをオプションで設定できます。

予約フィールド	説明
`__time__`	ログの UNIX タイムスタンプ (1970-01-01 からの秒数)。 ログの時間フィールドを基に算出されます。
`__topic__`	ログトピック
`__source__`	ソースログクライアントの IP アドレス

上記フィールドは、JSON ストレージにはデフォルトで含まれます。

必要に応じて、CSV ストレージに含めるフィールドを選択します。たとえば、ログトピックが必要な場合は、フィールド名を `__topic__` と指定します。

OSS ストレージアドレス

圧縮タイプ	ファイル拡張子	OSS ファイルのアドレス例
非圧縮	.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.csv
snappy	.snappy.csv	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy.csv

データを活用

HybridDB

次の設定にすることをお勧めします。

- デリミタ: コンマ (,)
- 引用: 二重引用符 (")
- 無効なキー値: 空
- 表示キー: 選択されていません（HybridDB の場合、CSV ファイルの先頭行にフィールド名はありません）

詳細は、HybridDB のドキュメントをご参照ください。

CSV は読むことの可能な形式です。CSV 形式のファイルは、OSS から直接ダウンロードしてテキスト形式で表示させることができます。

snappy 方式で圧縮されている場合は、「JSON ストレージ」に記載されている snappy の解凍方法をご参照ください。

10.2.4 Parquet ストレージを使用した OSS 転送

本ドキュメントでは、 Object Storage Service (OSS) に転送される Log Service ログを格納する Parquet ストレージに関する設定を紹介します。 OSS へのログの転送の詳細については、OSS へのログの転送をご参照ください。

Parquet ストレージを構成

データ型

Parquet には、 String 、boolean、int32、int64、float、double の 6 つのデータ型を使用できます。

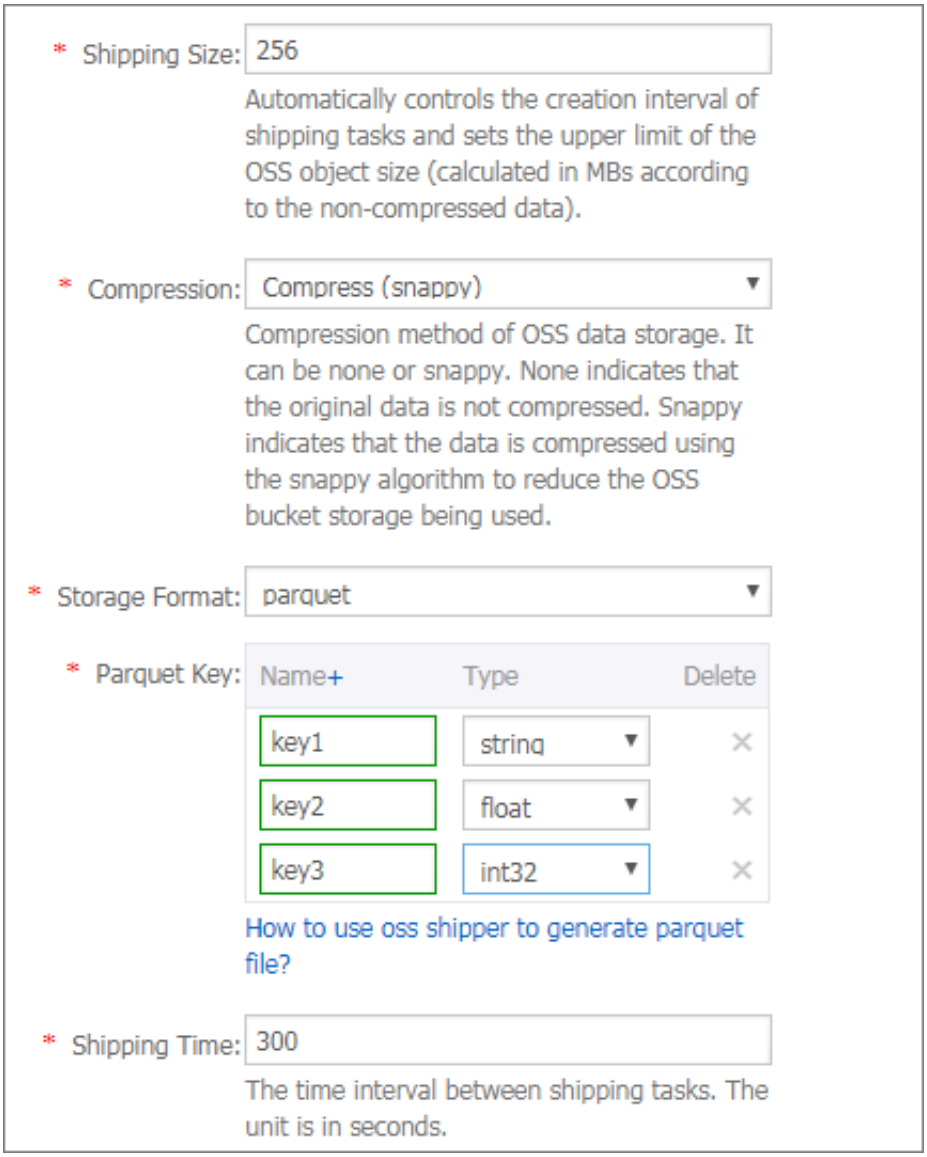
Log Service のデータは、ログ転送の際に文字列から任意の Parquet 型に変換されます。String 型以外のデータ型への変換に失敗した列は、null に設定されます。

列を構成

Log Service のフィールド名およびターゲットの Parquet のデータ型を指定します。Parquet データはここで設定したフィールド順に転送されます。Log Service のフィールド名は、Parquet のデータのかカラム名になります。以下の場合、データカラムは null に設定されます。

- Log Service データにないフィールド名
- String 型から非 String 型 (double や int64 など) への変換に失敗するフィールド

図 10-4 : フィールド設定ページ



設定可能な予約フィールド

ログのキーと値のペアのほか、Log Service は OSS にログを転送する際に下表の予約フィールドをオプションで設定できます。

予約フィールド	説明
`__time__`	ログの UNIX タイムスタンプ (1970-01-01 からの秒数)。ログの時間フィールドに従って 計算されます。
`__topic__`	ログトピック
`__source__`	ソースログクライアントの IP アドレス

上記フィールドは、JSON ストレージにはデフォルトで含まれます。

必要に応じて、parquet ストレージまたは CSV ストレージに含めるフィールドを選択します。たとえば、ログトピックが必要な場合は、フィールド名 `__topic__` を入力し、データ型には String を選択します。

OSS ストレージアドレス

圧縮タイプ	ファイル拡張子	OSS ファイルのアドレス例
圧縮しない	.parquet	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.parquet
snappy	.snappy.parquet	oss://oss-shipper-shenzhen/ecs_test/2016/01/26/20/54_1453812893059571256_937.snappy.parquet

データを活用

E-MapReduce/Spark/Hive

コミュニティ文書をご参照ください。

スタンドアロン検証ツール

オープンソースコミュニティの提供するparquet-toolsで、parquet のフォーマットを検証、スキーマを表示、ファイルごとデータを読み取ります。

ご自身でツールコンパイルするか、こちらをクリックして Log Service の提供するバージョンをダウンロードします。

- Parquet ファイルのスキーマの表示

```
$ java -jar parquet-tools-1.6.0rc3-SNAPSHOT.jar
schema -d 00_1490803_5321364704_39_124353.snappy.parquet
| head -n 30
```

```
 message   schema  {
   optional   int32   __time__ ;
   optional   binary   ip ;
   optional   binary   __source__ ;
   optional   binary   method ;
   optional   binary   __topic__ ;
   optional   double   seq ;
   optional   int64   status ;
   optional   binary   time ;
   optional   binary   url ;
   optional   boolean   ua ;
}
 creator :  parquet - cpp   version   1 . 0 . 0
 file   schema :  schema
--------------------------------------------------------------------------------
 __time__ :  OPTIONAL   INT32   R : 0   D : 1
 ip :  OPTIONAL   BINARY   R : 0   D : 1
.......
```

- Parquet ファイルのすべてのコンテンツの表示

```
$  java  - jar   parquet - tools - 1 . 6 . 0rc3 - SNAPSHOT . jar
   head  - n   2   00_1490803  5321364704  39_124353 . snappy .
 parquet
 __time__  =  1490803230
 ip  =  10 . 200 . 98 . 220
 __source__  = *. *. *.*
 method  =  POST
 __topic__  =
 seq  =  1667821 . 0
 status  =  200
 time  =  30 / Mar / 2017 : 00 : 00 : 30  + 0800
 url  = / PutData ?  Category = YunOsAccou  ntOpLog & AccessKeyI  d
 =< yourAccess  KeyId >& Date = Fri % 2C % 2028 % 20Jun % 202013 %
 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature =< yourSignat
 ure >  HTTP / 1 . 1
 __time__  =  1490803230
 ip  =  10 . 200 . 98 . 220
 __source__  = *. *. *.*
 method  =  POST
 __topic__
 seq  =  1667822 . 0
 status  =  200
 time  =  30 / Mar / 2017 : 00 : 00 : 30  + 0800
 url  = / PutData ?  Category = YunOsAccou  ntOpLog & AccessKeyI  d
 =< yourAccess  KeyId >& Date = Fri % 2C % 2028 % 20Jun % 202013 %
 2006 % 3A53 % 3A30 % 20GMT & Topic = raw & Signature =< yourSignat
 ure >  HTTP / 1 . 1
```

詳細な操作手順については、`java - jar parquet - tools - 1 . 6 . 0rc3 - SNAPSHOT . jar - h` コマンドを実行します。

10.2.5 RAM で権限付与

OSS 転送タスクを実行する前に、OSS バケットの所有者は クイック許可 を行う必要があります。権限を付与されると Log Service より OSS に書き込むことができるようになります。

本ドキュメントでは、さまざまなシナリオにおいて、RAM で権限を付与して OSS 転送タスクが実行されるようにする方法について説明します。

- OSS バケットに対するアクセス制御をより詳細に指定する場合は、「ポリシーの変更」をご参照ください。
- Log Service プロジェクトおよび OSS バケットが別々の Alibaba Cloud アカウントで作成されている場合は、「アカウント間の転送」をご参照ください 。
- サブアカウント (RAM ユーザー) が、別の Alibaba Cloud アカウントの OSS バケットにログデータを送信する必要がある場合は、「サブアカウントとメインアカウント間の転送」をご参照ください。
- サブアカウント (RAM ユーザー) が、そのメインアカウントのログデータを同メインアカウントの OSS バケットに送信する必要がある場合は、「サブアカウントに Log Service へのアクセス権限を付与」をご参照ください。

ポリシーの変更

クイック許可を完了すると、AliyunLogRolePolicy に AliyunLogDefaultRole ロールがデフォルトで割り当てられ、アカウント B の所有する全 OSS バケットに書き込むことができるようになります。

より詳細にアクセス制御する必要ある場合は、AliyunLogDefaultRole ロールより AliyunLogRolePolicy 権限を削除し、より詳細なポリシーを作成し、そのポリシーを AliyunLogDefaultRole ロールに割り当てます。 詳細は、「『OSS へのアクセス権を付与』」をご参照ください。

アカウント間の転送

Log Service プロジェクトおよび OSS バケットが、別々の Alibaba Cloud アカウントで作成したものである場合、次の方法でポリシーを設定します。

たとえば、アカウント A の Log Service データをアカウント B の作成した OSS バケットに転送するとします。

1. アカウント B は、AliyunLogDefaultRole ロールを作成して、作成したロールに OSS への書き込み権限を割り当てます。詳細は、「OSS へのログ転送」をご参照ください。
2. RAM コンソールにログインして左側のメニューのロール管理をクリックします。ロール名「AliyunLogDefaultRole」をクリックすると、ロールの基本情報が表示されます。

   ロールを行使することのできるユーザーは、ロールの `Service` に記述されています。なお、`log.aliyuncs.com` は、現在のアカウントがこのロールを行使して OSS に書き込むことができることを示します。

3. ロールの `Service` に `A_ALIYUN_I  D @ log . aliyuncs . com` を追加します。アカウント管理 > セキュリティ設定より、メインアカウント A の ID が追加されているかどうかを確認します。

   アカウント A の ID が「1654218**********」の場合、ロールには次のように記述します。

```
{
" Statement ": [
 {
   " Action ": " sts : AssumeRole ",
   " Effect ": " Allow ",
   " Principal ": {
     " Service ": [
       "**********@ log . aliyuncs . com ",
       " log . aliyuncs . com "
     ]
   }
 }
],
" Version ": " 1 "
}
```

   上記ロールには、アカウント A が Log Service を介して一時トークンを取得し、アカウント B のリソースを操作する権限が付与されていることが記述されています。ロールの記述方法については、「ポリシー管理」をご参照ください。

4. 以上で、アカウント A で転送タスクが生成されるようになります。生成される転送タスクの RAM ロールフィールドには、OSS バケット所有者の RAM ロール識別子 ARN (アカウント B の作成した RAM ロール AliyunLogDefaultRole) を指定します。

   RAM ロールの ARN は、基本情報で確認します。書式は `acs : ram :: 13234 : role / logrole` です。

サブアカウントとメインアカウント間の転送

メインアカウント A のサブアカウント a_1 が、アカウント B の OSS バケットへのログ転送タスクを生成できるようにするには、メインアカウント A が、サブアカウント a_1 に PassRole ロールを割り当てる必要があります。

設定方法は次のとおりです。

1. アカウント B はクイック許可を行い、ロールに記述を追加します。詳細は、「アカウント間の転送」をご参照ください。

2. メインアカウント A は RAM コンソールにログインして、サブアカウント a_1 に AliyunRAMFullAccess 権限を与えます。

   a. ユーザー管理ページで、サブアカウント a_1 の右側の許可をクリックします。

   b. ポリシー一覧リストよりAliyunRAMFullAccessを許可したポリシーリストに追加します。確認をクリックします。

      許可が正常に完了すると、a_1 は RAM を操作する全権限を有することになります。

      a_1 の権限を限定し、OSS へのログ転送のみを実行できるようにするには、メインアカウント A は `Action` および `Resource` パラメータを変更します。

      `Resource` の値を AliyunLogDefaultRole の ARN に置き換えます。ポリシーは次のようになります。

```
{
" Statement ": [
{
" Action ": " ram : PassRole ",
" Effect ": " Allow ",
" Resource ": " acs : ram :: 1111111 : role / aliyunlogd
 efaultrole "
}
],
" Version ": " 1 "
}
```

   c. 以上で、サブアカウント a_1 で転送タスクが生成されるようになります。生成される転送タスクの RAM ロールフィールドには、OSS バケット所有者のRAM ロール識別子である ARN (クイック許可でアカウント B の作成した RAM ロール AliyunLogDefaultRole) を指定します。

10.3 MaxCompute へのログ転送

10.3.1 DataWorks を通じてデータを MaxCompute への転送

ログを OSS ストレージに転送するだけでなく、DataWorks のデータ統合機能を使用してログデータを MaxCompute に転送することもできます。 データ統合は、Alibaba Group が外部ユーザーに提供する、安定、効率的、且つ柔軟で拡張可能なデータ同期プラットフォームです。 それはAlibaba Cloudsビッグデータコンピューティングエンジン（MaxCompute、AnalyticDB、および OSPS を含む）にオフラインのバッチデータアクセスチャネルを提供します。

この機能が使用可能なリージョンの詳細については、DataWorks を参照してください。

シナリオ

- リージョンを跨げるデータソース（LogHub と MaxCompute）間のデータ同期
- 異なる Alibaba Cloud アカウントを持つデータソース（LogHub と MaxCompute）間のデータ同期
- 同じ Alibaba Cloud アカウントを持つデータソース（LogHub と MaxCompute）間のデータ同期
- パブリッククラウドアカウントと AntCloud アカウントを使用したデータソース（LogHub と MaxCompute）間のデータ同期

前提条件

1. Log Service、MaxCompute、および DataWorks が有効にされています。
2. Log Serviceはログデータを正常に収集し、LogHub は転送するデータを持っています。
3. データソースアカウントに対応するアクセスキーペアが有効になります。
4. アカウントを跨げて転送するには、RAM 認証を構成する必要があります。

   詳細は、本ドキュメントのアカウント間でのログ配信の権限付与を実行するを参照してください。

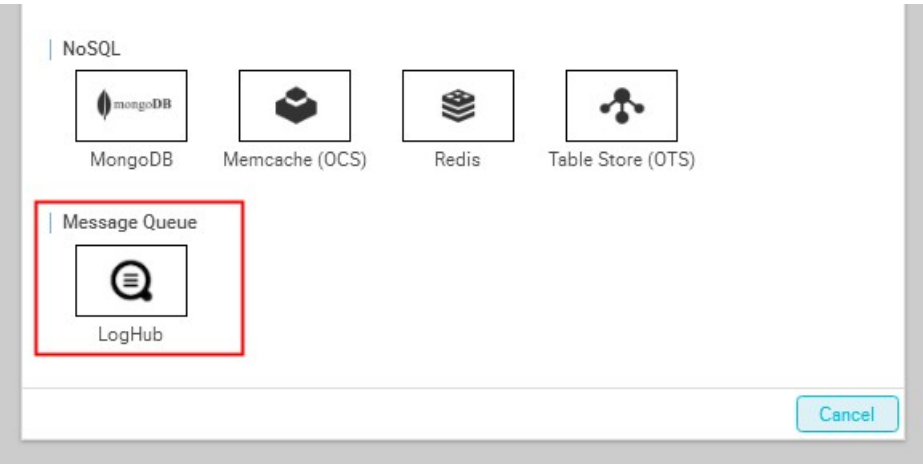
手順

手順 1：データソースを作成します。

1. DataWorksコンソールで Data Integration ページを開きます。 左側のナビゲーションバーで、Data Source をクリックします。
2. Data Source ページで、右上隅の Add Data Source をクリックします。 Add Data Source ダイアログボックスが表示されます。

3. Message Queue リストから LogHub をクリックします。 Add Data Source LogHub データソースページが表示されます。

図 10-5 : データソースの追加



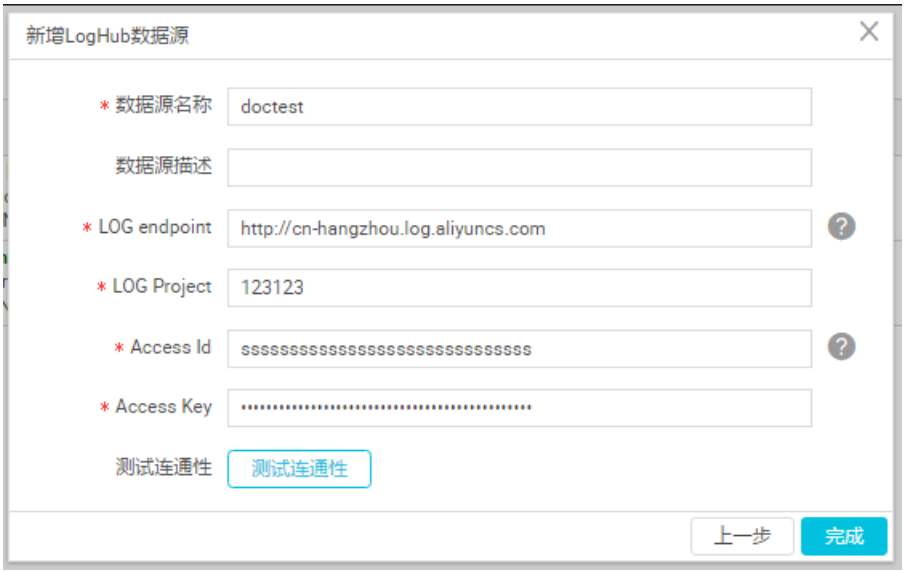
4. データソースの構成項目を設定します。

次の表に構成項目を示します。

構成項目	説明
データソース名	データソース名は、文字、数字、およびアンダースコアで構成されます。 先頭は英字または下線でなければならず、60文字を超えることはできません。
データソース説明	データソースに関する簡単な説明。最大で80文字です。
LOG Endpoint	Log Service のエンドポイント、 `http://yyy.com` の形式で、お住まいのリージョンによって決まります 。 詳細は、Service endpointを参照してください。
LOG Project	ログデータの送信先となる MaxCompute の Log Service プロジェクト。 存在している、作成済みのプロジェクトでなければなりません。

構成項目	説明
Access Id/Access Key	データソースアカウントのアクセスキーは、ログインパスワードと同じです。 プライマリアカウントのアクセスキーまたはデータソースのサブアカウントを入力できます。 構成が成功すると、現行のアカウントにデータソースのアカウントログへのアクセス権限が付与され、同期タスクを通じてデータソースアカウントのログを送信できます。

図 10-6 : LogHub データソースの作成



5. Test Connectivity をクリックします、接続テストに成功した後、ページの右上隅にある Finish をクリックします。

手順 2 同期タスクの構成

左側のナビゲーションバーで Synchronization Task をクリックし、Create a synchronization task をクリックして同期タスクを構成します。

Wizard Mode を選択してより簡単に構成できます、Script Mode を選択して、同期タスクの構成をさらにカスタマイズできます。

Wizard mode

タスク同期ノードの構成項目には、ソースの選択、ターゲットの選択、フィールドマッピング、およびチャネル制御があります。

1. ソースを選択します。

   Data source: 手順1で構成したデータソースを選択します。 構成項目を次の表のように設定します：

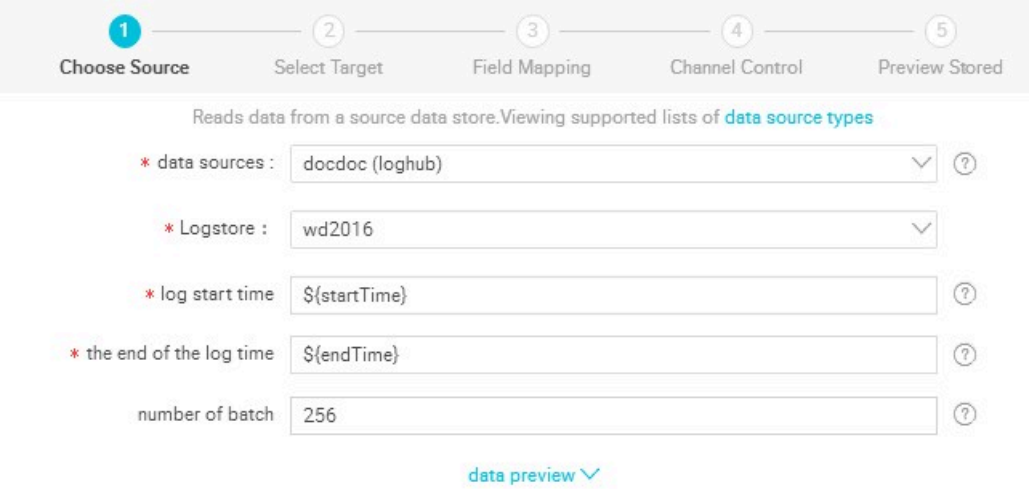
構成項目	説明
Data source	LogHub データソースの名前を選択します。
Logstore	増分データのエクスポート元のテーブルの名前。 テーブルを作成するとき、または後で UpdateTable API を使用するときは、テーブルでストリーム機能を有効にする必要があります。
ログの開始時刻	データ消費の開始時刻 このパラメーターは、yyyyMMddHHmmss（20180111013000など） の形式で時間範囲の左ボーダー（左閉右開）を定義し、DataWorks のスケジューリング時間パラメーターで機能します。
ログの終了時刻	データ消費の終了時刻 このパラメーターは、yyyyMMddHHmmss（20180111013010など） の形式で時間範囲の右ボーダー（左閉右開）を定義し、DataWorks のスケジューリング時間パラメーター形式で機能します。
バッチサイズ	毎回に読取りできるるデータ項目の数。 デフォルト値は256。

   構成項目を設定したら、Data Preview ドロップダウンボタンをクリックしてData Previewの詳細を表示します。 ログデータが取得されたことを確認して、Next をクリックします。

    注：

データプレビューは、LogHubから選択されたいくつかのデータエントリを表示します。 同期データはログの開始時刻と終了時刻で設定されるため、プレビュー結果は設定した同期データと異なる場合があります。

図 10-7 : ソースを選択します。



2. ターゲットを選択します。

   a. MaxComputeデータソースとターゲットテーブルを選択します。

      MaxComputeテーブルを作成していない場合は、右側の Generate Target Table in One Click をクリックします。 ポップアップメニューの [Create Data Table] を選択します。

   b. Partition information を入力します。

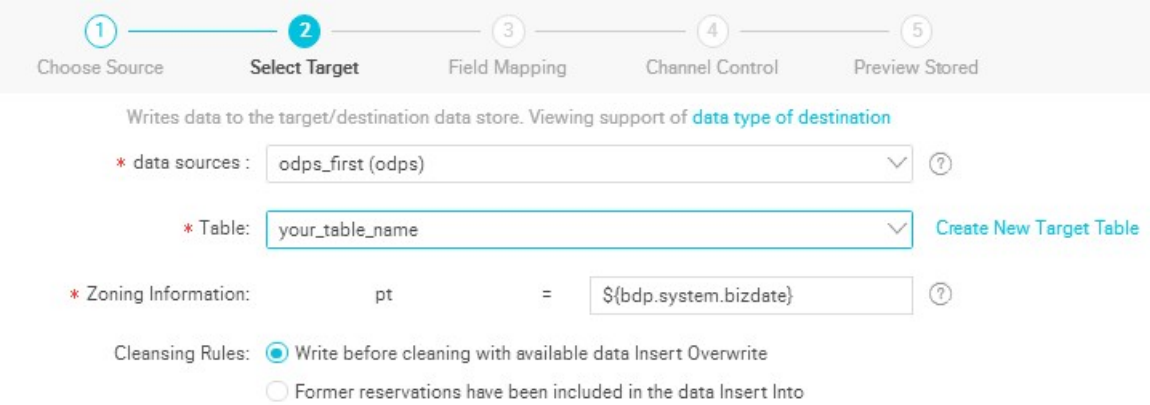
      パーティション構成は正規表現をサポートしています。 たとえば、パーティション「*」のpt値を設定して、すべてのptパーティションのデータを読み取ることができます。

   c. Clearing Rules を選択します。

      書き込み前に既存のデータを消去する（上書きモード）か、既存のデータを保持する（挿入モード）かを選択できます。

   構成後、 Next をクリックします。

   図 10-8 : ターゲットを選択します。



3. フィールドをマッピングします。

   フィールド間のマッピングを選択します。 フィールドのマッピング関係を設定する必要があります。 左側のソーステーブルフィールドは、右側のターゲットテーブルフィールドと1対1で対応しています。 Same row mapping をクリックして、 Same row mapping を選択、または選択解除できます。

   注 :

   - ログフィールドを同期列として手動で追加する必要がある場合は、構成を使用してください。
   - 定数を入力することができます。 各定数は、'abc' や '123' のように、一重引用符で囲む必要があります。

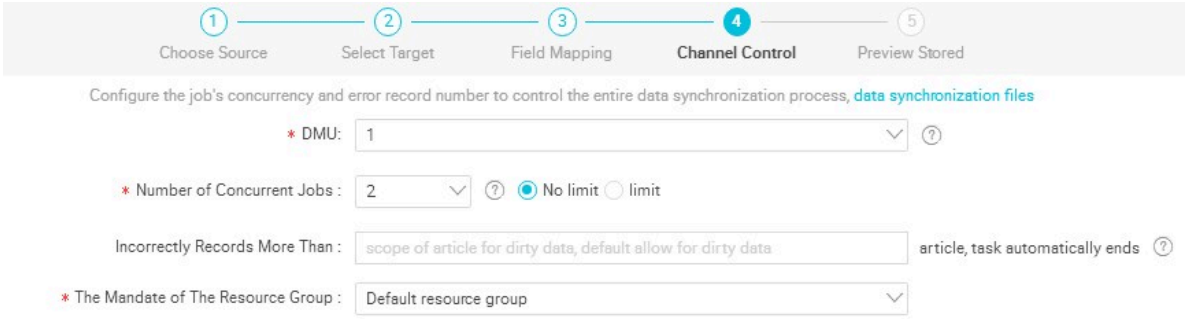
- $ {bdp.system.bizdate} などのスケジューリングパラメータを追加できます。
- now（）やcount（1）など、リレーショナルデータベースでサポートされている関数を入力できます。
- 入力した値を解析できない場合、タイプは「識別されていません」と表示されます。

図 10-9 : マッピングフィールド

4. トンネルを制御します。

次の図に示すように、最大ジョブレートとダーティデータチェックルールを設定します。

図 10-10 : トンネルを制御します。



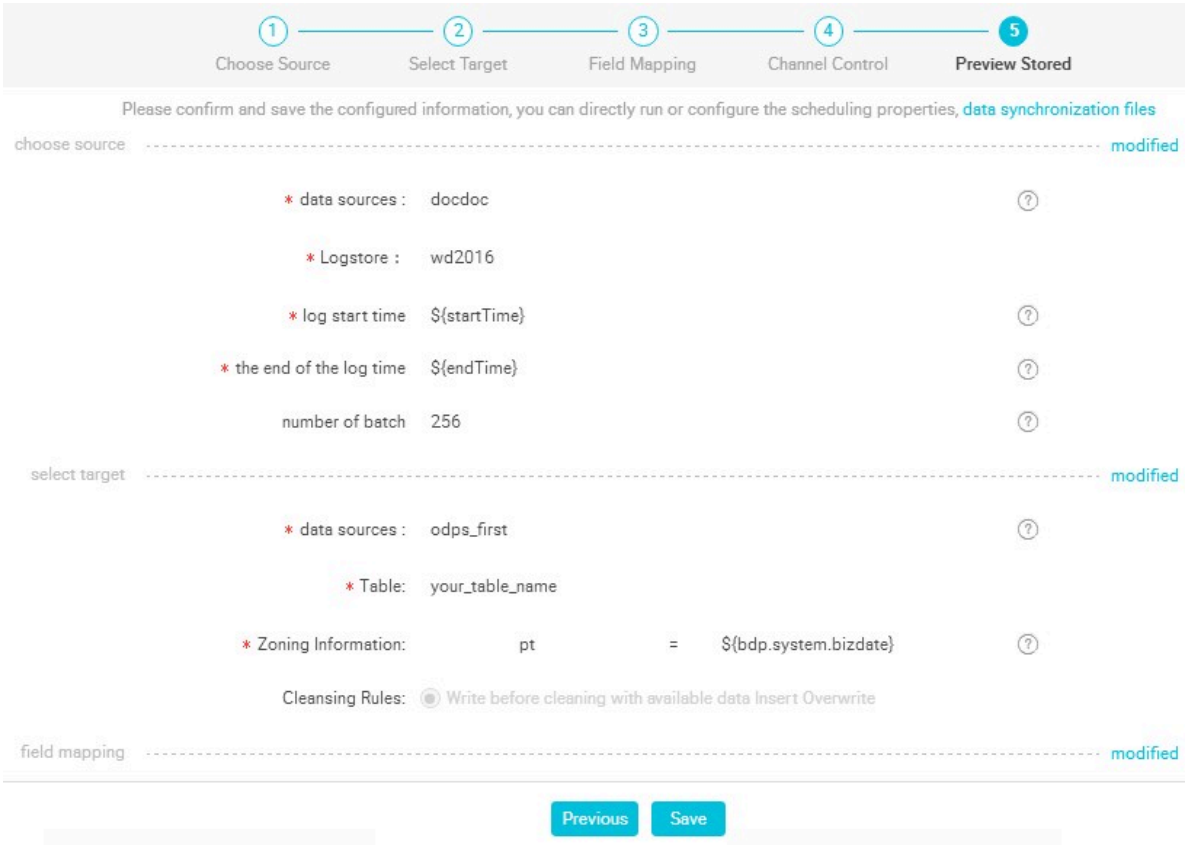
構成項目説明：

- DMU：データ統合中に消費されるリソース（CPU、メモリ、ネットワーク帯域幅など）を測定するデータマイグレーション単位。
- 同時ジョブ数：データ同期タスクでデータ記憶メディアからデータを同時に読み書きするのに使用されるスレッドの最大数

5. 構成をプレビューして保存します。

   設定が完了したら、上下にスクロールしてタスク設定を表示できます。 エラーが発生しない場合は、Saveをクリックします。

   図 10-11 : 構成をプレビューして保存します。



Script mode

スクリプトを使用してタスクを設定するには、次のスクリプトを参照してください。

```
{
" type ": " job ",
" version ": " 1 . 0 ",
" configurat  ion ": {
" reader ": {
" plugin ": " loghub ",
" parameter ": {
" datasource ": " loghub_lzz ",//データソース名。 追加したデータリソース名を使用します。
" logstore ": " logstore - ut2 ",//対象のログストア名。 ログストアは、ログサービス内のログデータの収集、保存、およびクエリ単位です。
" beginDateT  ime ": "${ startTime }",//データ消費の開始時刻。 このパラメータは、時間範囲の左のボーダーを定義します（左閉右開）
" endDateTim  e ": "${ endTime }",//データ消費の終了時刻。 このパラメータは、時間範囲の右のボーダーを定義します（左閉右開）
" batchSize ":  256 ,//毎回に読み取りできるデータ項目の数 デフォルト値は 256 。
" splitPk ": "",
" column ": [
" key1 ",
```

```
" key2 ",
" key3 "
]
}
},
" writer ": {
" plugin ": " odps ",
" parameter ": {
" datasource ": " odps_first ",//データソース名。 追加したデータリソース名を
使用します。
" table ": " ok ",//対象のテーブル名
" truncate ":  true ,
" partition ": "",//シャード情報
" column ": [//対象の列の名前
" key1 ",
" key2 ",
" key3 "
]
}
},
" setting ": {
" speed ": {
" mbps ":  8 ,//最大ジョブレート
" concurrent ":  7 //同時ジョブ数
}
}
}
}
```

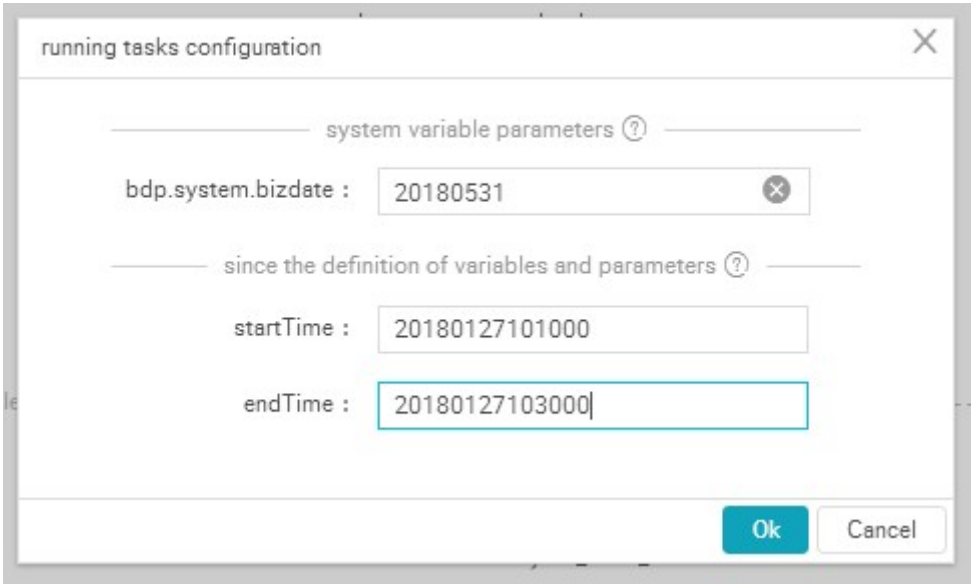
手順3 タスクの実行

次のいずれかの方法でタスクを実行できます。

- タスクを直接実行します（1回実行）。

  タスクの上にある Run をクリックして、data integration ページでタスクを直接実行します。 タスクを実行する前に、カスタムパラメータの値を設定します。

  図 10-12 : 実行中のタスク構成



  上図に示すように、10：10~17：30 の間の LogHub レコードは、MaxCompute と同期しています。

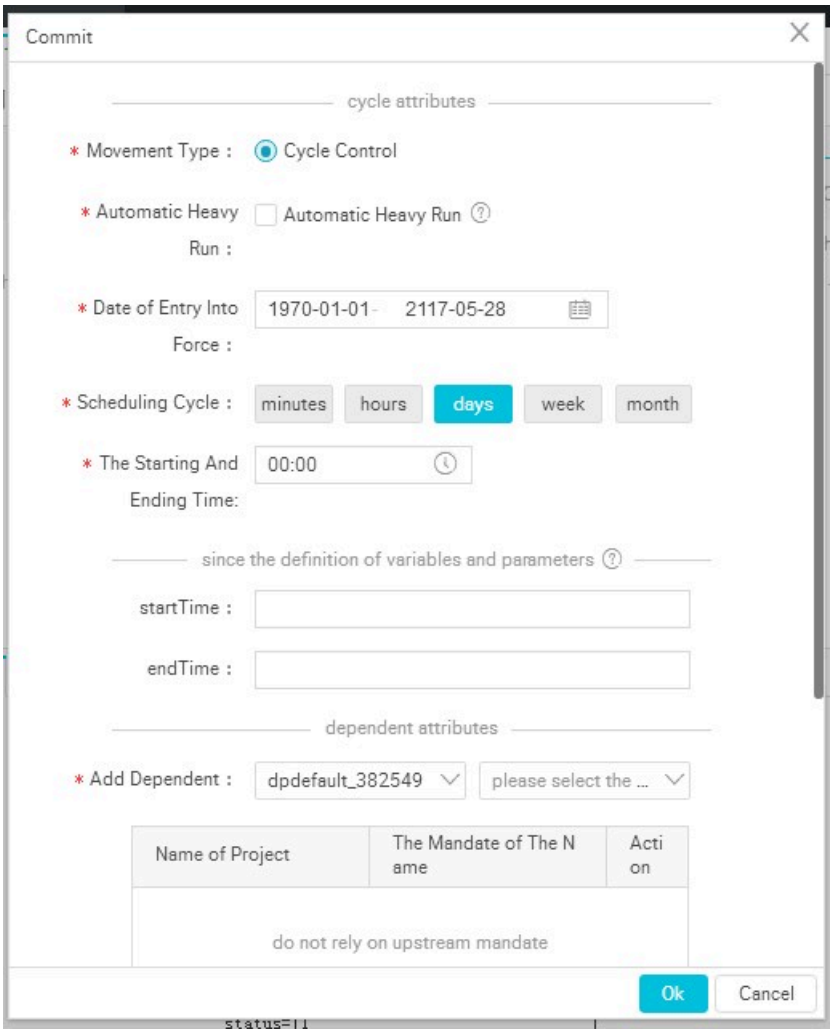
- タスクのスケジューリング

  Submit をクリックして、同期タスクをスケジューリングシステムに送信します。 スケジューリングシステムは、設定属性に従ってタスクを翌日から自動的かつ定期的に実行します。

  00：00～23：59のスケジュール期間で、 startTime = $[yyyymmddhh24miss-10/24/60]システムの10分前から endTime = $[yyyymmddhh24miss-5/24/60]システムの5分前まで、スケジュール間隔を5分に設定しま

す。 カスタムパラメータ構成の詳細については、 Parameter configuration を参照してください。

図 10-13 : スケジューリング設定



アカウント間でのログ配信の権限付与を実行する

アカウント間でログ配信タスクを構成するには、RAM上で権限付与を実行します。

- アカウント間でのログ配信の権限付与を実行する

  プライマリアカウント間でデータを送信するには、ステップの Add LogHub Data Source でデータソースのプライマリアカウントのアクセスキーを入力します。 接続テストに合格した場合、権限付与は成功です。

  たとえば、アカウントBで有効された DataWorks サービスを通じてアカウントAのログデータをアカウントBの MaxCompute テーブルに転送するには、アカウントBでデータ統合タスクを設定し、ステップの Add LogHub Data Source でアカウントAのプライマリアカウント

のアクセスキーを入力します。 構成が成功した後、アカウントBにはアカウントAの下にあるすべてのログデータを読み取る権限が付与されます。

- サブアカウント権限付与

  プライマリアカウントのアクセスキーを明らかにしたくない場合、またはサブアカウントによって収集されたログデータを転送する必要がある場合は、サブアカウントに対して明示的な承認を設定します。

  - サブアカウントに管理権限を割り当てる

    サブアカウントを通じてすべてのログデータをプライマリアカウントで転送する必要がある場合は、権限付与とアクセスキーの設定について次の手順を実行します。

    1. プライマリアカウントAを使用して、ログサービス管理権限（ `AliyunLogF ullAccess` および `AliyunLogR eadOnlyAcc ess` ）をサブアカウントA1に割り当てます。 詳細について、RAM ユーザーに Log Service へのアクセスを許可を参照してください。
    2. アカウントBを使用してデータ統合タスクを構成し、ステップの Add LogHub Data Source でデータソースのサブアカウントのアクセスキーを入力します。

    構成が成功した後、アカウントBにはアカウントAの下にあるすべてのログデータを読み取る権限が付与されます。

  - サブアカウントにカスタマイズ権限を割り当てます

    特定のログデータをサブアカウントを通じてプライマリアカウントで転送する必要がある場合は、権限付与とアクセスキーの設定について次の手順を実行します。

    1. プライマリアカウントAを使用して、サブアカウントA1のカスタマイズ権限付与ポリシーを設定します。関連する権限付与操作の詳細については、概要およびOverviewを参照してください。
    2. アカウントBを使用してデータ統合タスクを構成し、ステップの Add LogHub Data Source でデータソースのサブアカウントのアクセスキーを入力します。

    上記の手順が正常に完了すると、アカウントBにはアカウントAの指定されたログデータを読み取る権限が付与されます。

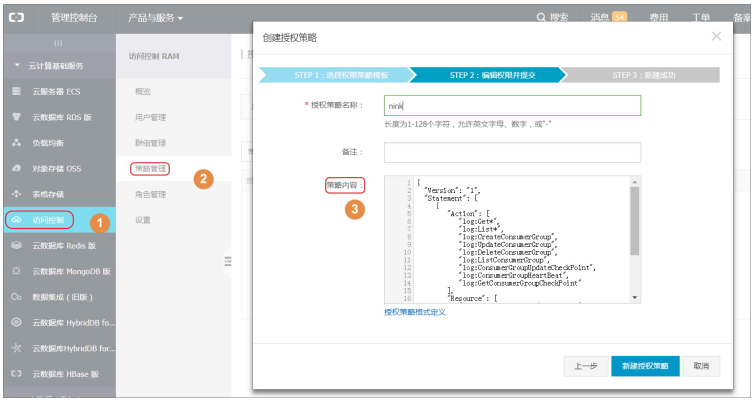
    カスタマイズ権限付与ポリシーの例：

    このように、アカウントBは、サブアカウントA1を通じてLog Service内のproject_name1およびproject_name2データのみを同期できます。

    ```
    {
    " Version ": " 1 ",
    ```

```
" Statement ": [
{
" Action ": [
" log : Get *",
" log : List *",
" log : CreateCons  umerGroup ",
" log : UpdateCons  umerGroup ",
" log : DeleteCons  umerGroup ",
" log : ListConsum  erGroup ",
" log : ConsumerGr  oupUpdateC  heckPoint ",
" log : ConsumerGr  oupHeartBe  at ",
" log : GetConsume  rGroupChec  kPoint "
],
" Resource ": [
" acs : log :*:*: project / project_na  me1 ",
" acs : log :*:*: project / project_na  me1 /*",
" acs : log :*:*: project / project_na  me2 ",
" acs : log :*:*: project / project_na  me2 /*"
],
" Effect ": " Allow "
}
]
}
```

図 10-14 : カスタマイズ権限付与ポリシー



10.4 LogShipper タスクの管理

Log Service の LogShipper 機能を使用することにより、データ価値を最大限に引き出すことができます。コンソールより、収集したログを OSS (Object Storage Service) に転送することで、データを長期間格納したり、 E-MapReduce といった他のシステムと併せてデータを活用したりすることができます。LogShipper 機能を有効にすると、Log Service はバックグランドで、Logstore に書き込まれたログを定期的に任意のクラウド製品に転送します。なお、Log Service コンソールの OSS Shipper ページでは、期間を指定して転送ステータスを照会することができます。転送ステータスを確認することで、迅速にオンライン問題に対応できます。

Logstore リスト ページの左側のナビゲーションメニューより LogShipper - OSS をクリックします。 OSS Shipper ページが表示されます。LogShipper タスクの管理方法は、次のとおりです。

LogShipper タスクを有効化/無効化

1. OSS Shipper ページで Logstore を選択します。
2. 有効化または無効化をクリックしてタスクを有効または無効にします。

   タスクを無効にして再度有効にした場合、転送ルールも再度設定する必要があります。

転送ルールを設定

LogShipper タスクを有効にしたら、設定をクリックして転送ルールを設定します。

LogShipper タスクの詳細を表示

Logstore、期間、およびタスクの転送ステータスを基に表示する LogShipper タスクを絞り込むことができます。なお、タスクのステータス、開始時間、終了時間、ログの受信時間、およびタイプが表示されます。

LogShipper タスクには 3 つのステータスがあります。

ステータス	説明	操作
正常	ログは正常に転送されています。	何も注意する必要がありません。
実行中	ログは転送処理中です。	後ほどログが正常に転送されたかどうかを確認します。
異常	ログの転送に失敗しました。LogShipper タスクは外部要因によりエラーが発生し、再試行できません。	詳細は、「OSS へのログの転送」の「LogShipper タスクの管理」をご参照ください。

転送ルールを削除

1. Logstore リストページのルールを削除をクリックします。
2. 表示されるダイアログボックスの確認をクリックします。

   削除したルールと同じ名前のルールを再び作成することはできません。削除は慎重に行ってください。

11 Log Service をモニタリング

11.1 Log Service のモニタリング

Log Service のモニタリングは、CloudMonitor コンソールまたは Log Service コンソールで行います。

- CloudMonitor コンソールで確認できる項目は、以下のとおりです。
  - Logstore の読み取り/書き込みログ
  - Logtail エージェントの収集ログ
- Log Service コンソールで確認できる項目は、以下のとおりです。
  - リアルタイムなサブスクリプション情報 (Spark Streaming、Storm、および Consumer Library)
  - ログの送信ステータス

本ドキュメントでは、CloudMonitor コンソールでの確認方法について説明します。Log Service コンソールでの確認方法については、「コンシューマーグループのステータス表示」、「LogShipper タスクの管理」および「アラーム設定」をご参照ください。

手順

> 注：
> RAM ユーザーが CloudMonitor の設定を行う場合は、RAM ユーザーに権限を付与する必要があります。

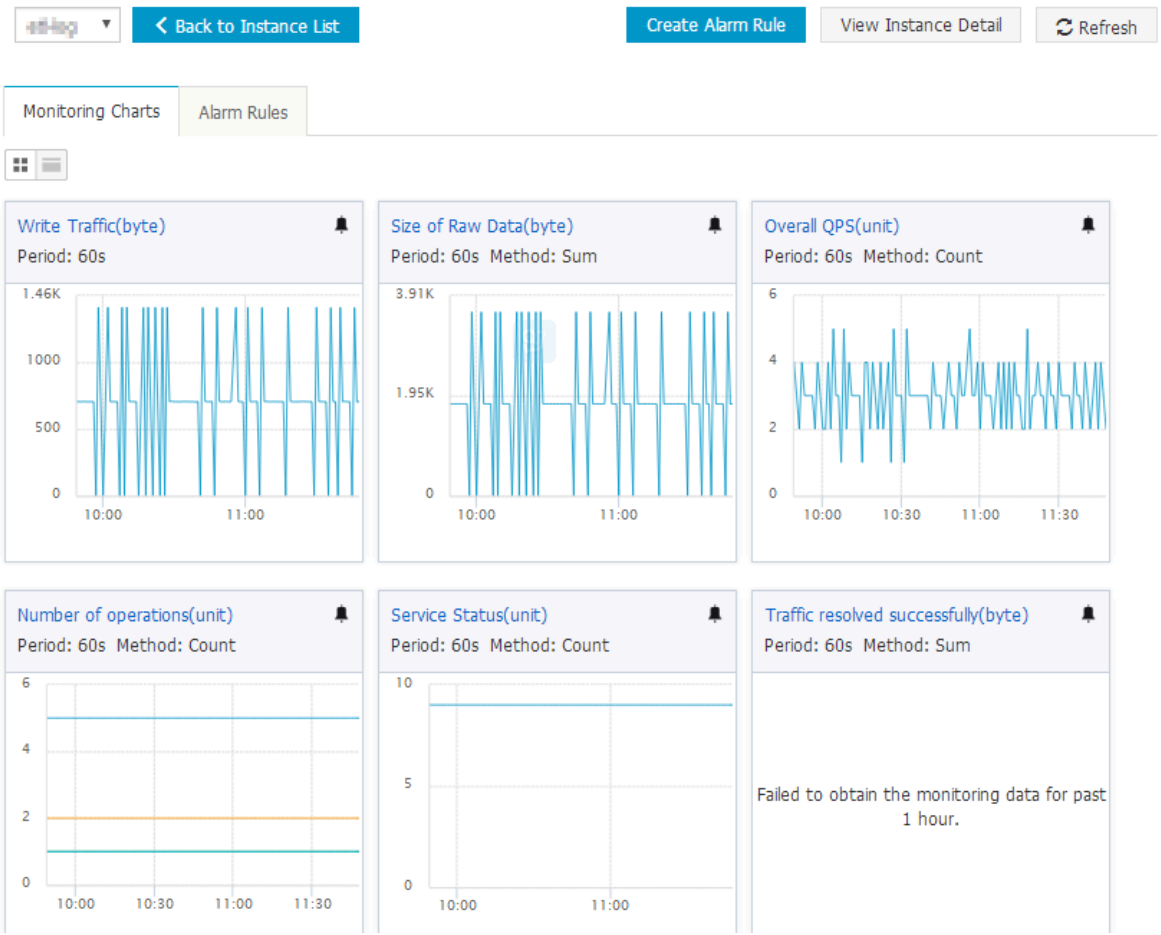
1. Log Service コンソールにログインします。
2. プロジェクト一覧ページで、プロジェクト名をクリックします。

3. Logstore の右側のモニタリングアイコンをクリックして、CloudMonitor コンソールに移動します。

   モニタリング設定ページに移動するには、CloudMonitor コンソールに直接ログインし、左側のナビゲーションメニューよりクラウドサービス、Log Serviceと順にクリックします。

   CloudMonitor のログデータをモニタリングします。 詳細については、Log Service をモニタリングをご参照してください。

   図 11-1 : モニタリング項目詳細



参考

Log Service モニタリングメトリック

アラームルールの設定

モニタリンググラフページの右上隅のアラームルールを作成をクリックします。関連するリソース、アラームルール、および通知方法を設定します。詳細については、CloudMonitor にアラームルールを設定をご参照ください。

11.2 サービスログ

11.2.1 機能の有効化、無効化、および構成

サービスログ機能を有効または無効にしたり、プロジェクトリスト内の指定したプロジェクトのログ構成を変更したりできます。 Log Service は、プロジェクトに対して生成されたすべてのログを新規または既存のプロジェクトに保存します。 デフォルトでは、この機能は無効化されています。

前提条件

1. Project が作成済み。
2. RAM サービスユーザーとして Log Service コ ンソールにログインすると、Alibaba Cloud アカウントから RAM ユーザーに権限が付与済み。

背景

Log Service は、指定された Project の操作ログやその他のログ（Logtail アラートログなど）を記録するためのサービスログ機能を提供します。 Log Service はログを新規または既存のプロジェクトに保存します。 Log Service は指定された保存場所にログストアを自動的に作成して、操作ログとその他のログを別々に保存します。 Log Service には、さまざまなログシナリオに対応する 5 つのダッシュボードも用意されているため、Log Service の実行ステータスをリアルタイムで表示およびモニタリングできます。

注 :

- サービスログ機能を有効にすると、Log Service は指定された保存場所にログストアとダッシュボードを作成します。 操作ログの保存に使用されるログストアは、指定した請求方法に基づいて請求されます。 その他のログを保存するために使用されるログストアは無料です。
- 同じリージョン内で生成されたログを、Log Service によって自動的に作成された同じ Project に格納することをお勧めします。
- 機能の有効化後に生成されたサービスログのみが記録されます。

サービスログ機能の有効化

1. Log Service コンソールにログインし、対象のプロジェクトを見つけます。
2. 操作 列での 操作ログ をクリックします。

3. 表示された「操作ログの有効化」ダイアログボックスで、操作ログの有効化 フィールドに記録するログの種類を選択します。

   必要に応じて操作ログおよびその他のログのチェックボックスにチェックを入れます。

   - 操作ログ ：作成、変更、更新、削除、書き込み、および読み取り操作を含む、対象プロジェクト内のリソースに対して実行されたすべての操作を記録します。 これらのログは、対象 Project の internal-operation_log ログストアに格納されています。
   - その他のログ ：メータリングログ、コンシューマグループ内での遅延コンシューマに関するログ、および各ログストア内の Logtail 関連のエラー、ハートビート、および統計ログが含まれます。 これらのログは、対象 Project の internal-diagnostic_log ログストアに格納されています。
4. ログストレージの設定

   - 自動作成（推奨）を選択すると、Log Service は自動的に `log - service -{ User ID }-{ Current region }` という名前の Project を対象 Project と同じリージョンに作成します。 この作成した Project の同じリージョン内で生成されたログを保存することをお勧めします。
   - 既存 Project を選択した場合、Log Service はサービスログをこの Project に保存します。
5. 確認をクリックします。

   サービスログ機能が有効になりました。 Log Service は、対象 Project 用に生成されたログを指定された場所にリアルタイムで記録します。

   図 11-2 : サービスログ機能の有効化

ログのタイプと保存場所の変更

1. Alibaba Cloud アカウントで Log Service コンソールにログインして、対象 Project を見つけます。
2. 操作 列での 操作ログ をクリックします。
3. ログタイプを変更します。

   操作ログの有効化 フィールドで、必要に応じて[操作ログ]または[その他のログ]のチェックボックスにチェックを入れます。

4. ロストレージ の値を変更します。

   ログストレージ ドロップダウンリストから別の Project を選択します。

   > 注 :
   >
   > - Log Service によって自動的に作成された Project にサービスログを保存することをお勧めします。 同じ Project 内の同じリージョンで生成されたログを保存することをお勧めします。
   > - ログストレージの値を変更すると、新しいログデータが指定した Project に保存されます。 元の Project に保存されているログデータとダッシュボードが自動的に削除されることや、新しく指定された Project に移行されることはありません。 データが不要になった場合は、手動で削除できます。

   図 11-3 : ログのタイプと保存場所の変更

サービスログ機能の無効化

> 注 :
> サービスログ機能を無効にした後、Project に保存されているログデータとダッシュボードは自動的に削除されません。 ログデータが不要になった場合は、ログデータを格納している Project またはログストアを手動で削除できます。

1. Alibaba Cloud アカウントで Log Service コンソールにログインして、対象 Project を見つけます。
2. 操作 列での 操作ログ をクリックします。
3. [操作ログ]と[その他のログ]の両方のチェックボックスのチェックを外します。
4. 確認をクリックします。

RAM ユーザーへの権限付与

RAM ユーザーでサービスログ機能を使用する前に、Alibaba Cloud アカウントから必要な権限を取得する必要があります。 詳細は RAM ユーザーに Log Service へのアクセスを許可 をご参照ください. 次のコードは、権限ポリシーを示しています。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
```

```
        " log : CreateDash  board ",
        " log : UpdateDash  board "
      ],
      " Resource ": " acs : log :*:*: project /{ Project   where
 logs   are   stored }/ dashboard /*",
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : GetProject ",
        " log : CreateProj  ect ",
        " log : ListProjec  t "
      ],
      " Resource ": " acs : log :*:*: project /*",
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : List *",
        " log : Create *"
        " log : Get *",
        " log : Update *",
      ],
      " Resource ": " acs : log :*:*: project /{ Project   where
 logs   are   stored }/ logstore /*",
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log :*"
      ],
      " Resource ": " acs : log :*:*: project /{ Project   for   which
   the   service   log   feature   is   enabled }/ logging ",
      " Effect ": " Allow "
    }
  ]
}
```

11.3 CloudMonitor でモニタリング

11.3.1 Log Service のモニタリングメトリック

メトリックの詳細については、Log Service のモニタリングをご参照ください。

1. 読み込み/書き込みトラフィック

   - 説明: Logstore からの読み取り/ Logstore への書き込みのデータトラフィック量。iLogtail、SDK、および API を介して、指定された Logstore への書き込み、Logstore から読み込みデータのトラフィック量の統計情報がリアルタイムに生成されます。トラフィック量とは、転送されたデータ (または圧縮データ) 量のことです。
   - 単位: Byte/分

2. Raw データサイズ
   - 説明: 各 Logstore に書き込まれた Raw データの量 (圧縮前)
   - 単位: Byte/分
3. 合計 QPS
   - 説明: すべての操作の QPS
   - 単位: 件/分
4. 操作件数
   - 説明: さまざまな操作タイプの QPS の数
   - 単位: 件/分
   - モニタリングできる操作は、以下のとおりです。
     - 書き込み
       - PostLogStoreLogs: API 0.5 以降
       - PutData: API 0.4 以前
     - キーワードクエリ
       - GetLogStoreHistogram: キーワードクエリの分布 (API 0.5 以降)
       - GetLogStoreLogs: キーワードクエリにマッチしたログ数 (API 0.5 以降)
       - GetDataMeta: GetLogStoreHistogram と同様 (API 0.4 以前)
       - GetData: GetLogStoreLogs と同様 (API 0.4 以前)
     - データの一括読み取り
       - GetCursorOrData: カーソルまたはデータを一括して読み取る
       - ListShards: Logstore のすべてのシャードを読み取る
     - リスト
       - ListCategory: ListLogStore と同様 (API 0.4 以前)
       - ListTopics: Logstore のすべてのトピックを読み取る

5. サービスステータス
   - 説明: すべての操作の応答 HTTP ステータスコードの QPS 統計が表示されます。迅速に、応答エラーコードに基づいて例外の発生した操作を応答エラーコードに特定し、プログラムを修正することができます。
   - ステータスコード
     - 200: 正常に処理されたことを示します。
     - 400: Host、Content-length、APIVersion、RequestTimeExpired、query time range、 Reverse、AcceptEncoding、AcceptContentType、Shard、Cursor、PostBody、Parameter、ContentType のいずれかのパラメーターによって発生したエラーです。
     - 401: 認証に失敗したことを示します。AccessKey ID が存在しない、AccessKey Secret が一致しない、または認証に使用したアカウントに権限がありません。Log Service コンソールより、プロジェクトのアクセス許可リストに AccessKey が含まれているかどうかを確認します。
     - 403: 設定上限を超えていることを示します。Logstore の数、シャードの数、または、1 分あたりの読み取り/書き込み処理が上限を超えました。応答メッセージを基にエラーを特定します。
     - 404: リクエストされたリソース (プロジェクト、Logstore、トピック、またはユーザー) が存在しないことを示します。
     - 405: 指定されたメソッドが不適切であったことを示します。リクエスト URL を確認します。
     - 500: Log Service エラーを示します。 もう一度お試しください。
     - 502: Log Service エラーを示します。 もう一度お試しください。
6. Logtail の正常に解析したたトラフィック量
   - 説明: Logtail によって正常に収集されたログのサイズ (生データ)
   - 単位: Byte
7. Logtail の正常に解析した行数
   - 説明: Logtail によって正常に収集されたログの数
   - 単位: 行
8. Logtail の解析に失敗した行数
   - 説明: エラーが発生したために Logtail によって収集されなかった行数
   - 単位: 行

9. Logtail のエラー発生件数

- 説明: Logtail のログ収集時にエラーの発生した IP アドレスの数
- 単位: 件

10.Logtail エラーの発生したマシンの数

- 説明: Logtail のログ収集エラーによって送信されたアラーム数
- 単位: 件

11.エラーの発生した IP アドレスの数 (件/5 分)

- 説明: サブカテゴリーに、各取得エラーが発生した IP アドレス数が記載されます
  - LOGFILE_PERMINSSION_ALARM: ログファイルへのアクセス権が Logtail エージェントにありません。
  - SENDER_BUFFER_FULL_ALARM: データ収集速度がネットワーク転送速度を超えたため、データは破棄されました。
  - INOTIFY_DIR_NUM_LIMIT_ALARM (INOTIFY_DIR_QUOTA_ALARM): モニタリング対象のディレクトリ数が 3,000 を超えています。モニタリング対象を下位ディレクトリに設定します。
  - DISCARD_DATA_ALARM: データの書き込まれた時間が、システム時間よりも 15 分も前であるため、そのデータは破棄されました。ログファイルへのデータ書き込み時間とシステム時間との差が 15 分未満になるようにします。
  - MULTI_CONFIG_MATCH_ALARM: 1 つのファイルを収集する設定が複数ある場合、Logtail はランダムにログ収集する設定を選択します。選択された設定以外の設定によってデータは収集されません。
  - REGISTER_INOTIFY_FAIL_ALARM: Inotify イベントの登録に失敗しました。詳細については、Logtail ログをご参照ください。
  - LOGDIR_PERMINSSION_ALARM: モニタリング対象のディレクトリーにアクセスする権限がエージェントにはありません。
  - REGEX_MATCH_ALARM: 正規表現の不一致エラー。正規表現を修正します。
  - ENCODING_CONVERT_ALARM: ログのエンコード形式変換時にエラーが発生しました。詳細については、Logtail ログをご参照ください。
  - PARSE_LOG_FAIL_ALARM: ログの解析エラーです。行の先頭の正規表現が不適切であるか、1 行のログのサイズが 512 KB を超えているためにログが不適切に分割されて

いる可能性があります。詳細については、Logtail のログを確認します。正規表現が不適切な、修正します。

- DISCARD_DATA_ALARM: Logtail がローカルのキャッシュファイルにデータを書き込むことができないため、Log Service にデータを送信できず、データは破棄されました。原因としては、ログファイルの生成に、キャッシュへのデータ書き込みが追いついていないことが考えられます。
- SEND_DATA_FAIL_ALARM: Logtail は解析済みログの Log Service への送信に失敗しました。詳細については、Logtail ログのデータ送信失敗に関するエラーコードとメッセージを確認します。一般的には、Log Service の上限超過、または Logtail エージェント側のネットワーク例外が挙げられます。
- PARSE_TIME_FAIL_ALARM: ログの時間フィールド解析時にエラーが発生しました。Logtail は、時間の書式設定に基づいて正規表現で時間フィールドを解析することに失敗しました。時間の書式設定を変更します。
- OUTDATED_LOG_ALARM: Logtail は過去のデータを破棄しました。データの書き込み時間とシステム時間との差が 5 分未満になるようにします。

・エラーを基に IP アドレスを特定します。Log Service にログインし、「/usr/logtail/ilogtail.LOG」ファイルを参照して原因を特定します。

11.3.2 CloudMonitor にアラームルールを設定

Log Service のアラームルールは、CloudMonitor で設定します。Log Service のステータスが設定したアラームルールに適合すると、SMS または電子メールでアラームが送信されます。CloudMonitor コンソールより Log Service をモニタリングするアラームルールを設定し、Logtail のログ収集状況、シャード使用状況、およびプロジェクトの書き込みトラフィックをモニタリングします。

手順

CloudMonitor コンソールで、Logstore の右側のCloudMonitor コンソール > Log Serviceアラームルールをクリックし、右上隅の新規アラームルールをクリックします。

1. 関連リソースを設定します。

   a. 製品ドロップダウンリストより、Log Serviceを選択します。

   b. リソースの範囲を選択します。

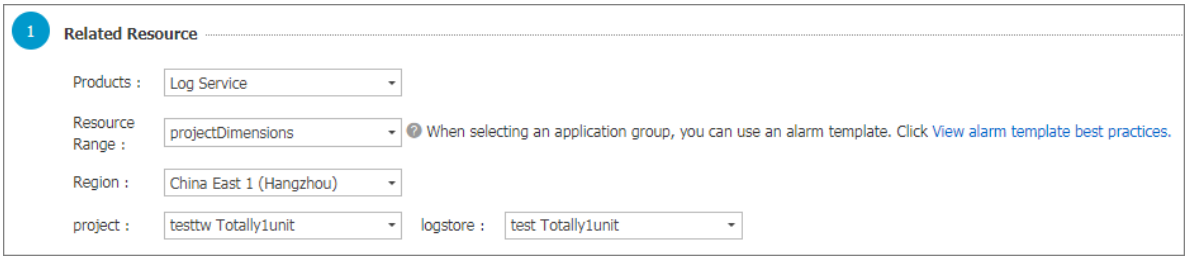
      すべてのリソース、アプリケーショングループ、またはプロジェクト単位のいずれかを選択します。

      - すべてのリソース - Log Service インスタンスがアラームルールに適合すると、アラーム通知は送信されます。
      - アプリケーショングループ - アプリケーショングループ内のインスタンスがアラームルールに適合した場合にのみ、アラーム通知は送信されます。
      - プロジェクト単位 - 選択したインスタンスがアラームルールに適合した場合にのみ、アラーム通知は送信されます。

   c. リージョンを選択します。

   d. プロジェクトおよびLogstoreを選択します。プロジェクトおよび Logstore は複数選択できます。

図 11-4 : 関連リソース



2. アラームルールを設定します。

   アラームルールは複数設定できます。

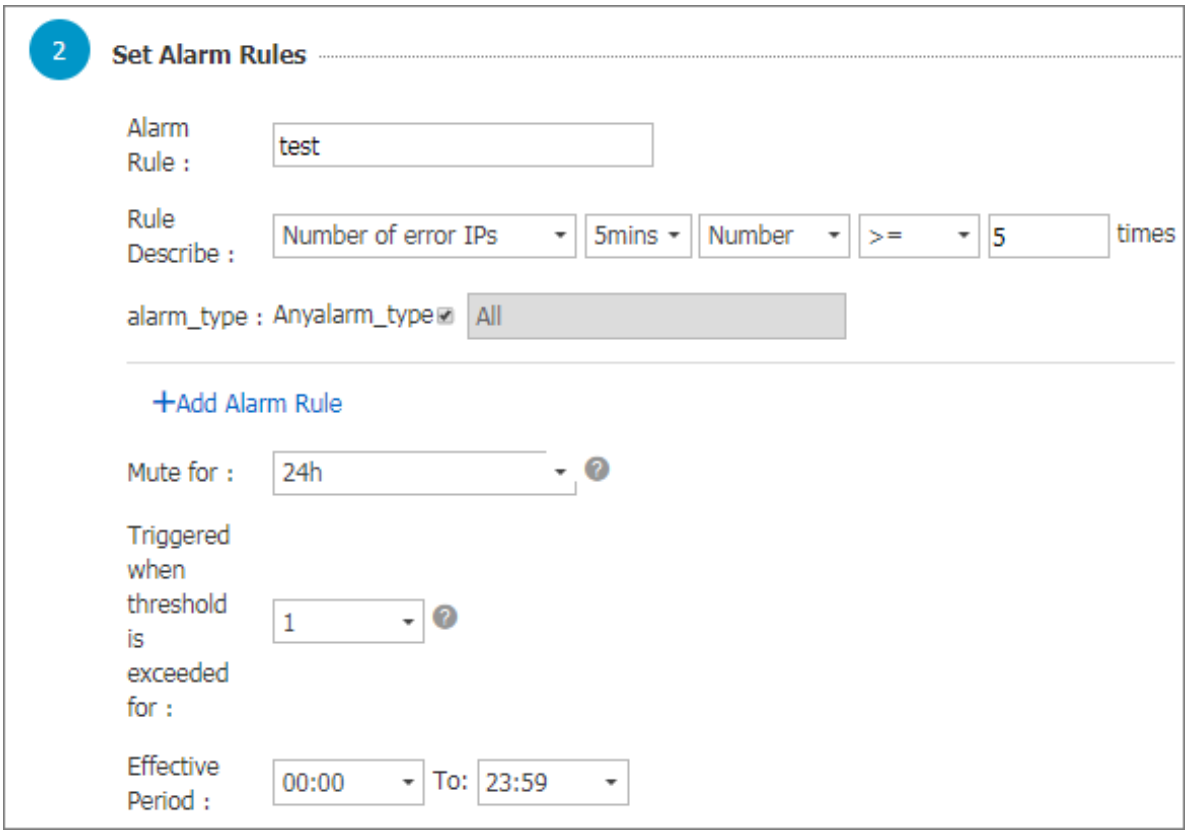
   a. アラームのルール名を入力します。
   b. ルールの詳細を設定します。

      モニタリングポリシーを定義します。モニタリング項目を選択し、モニタリング項目にしきい値を設定します。しきい値を超えると、CloudMonitor よりアラーム通知が送信されます。

      各モニタリング項目の詳細については、「Log Service モニタリングメトリック」をご参照ください。統計方法の詳細については、「Log Service のモニタリング」をご参照ください。
   c. alarm_typeを選択します。デフォルトでは、any alarm_typeが選択されています。
   d. ミュート時間を設定します。ミュート時間とは、アラーム通知を送信したにもかかわらず、依然として異常が検出される場合に、前回のアラーム送信からアラームが再送信されるまでの時間を指します。
   e. しきい値を超えた場合にトリガードロップダウンリストより数値を選択します。指定したしきい値を超過すると、つまり、指定回数連続してルールに適合した場合に、アラームは送信されます。

f. モニタリングポリシーの有効期間を選択します。モニタリングアラームポリシーは、選択した期間内のみ有効です。

図 11-5 : アラームルールの設定



3. 通知方法を設定します。

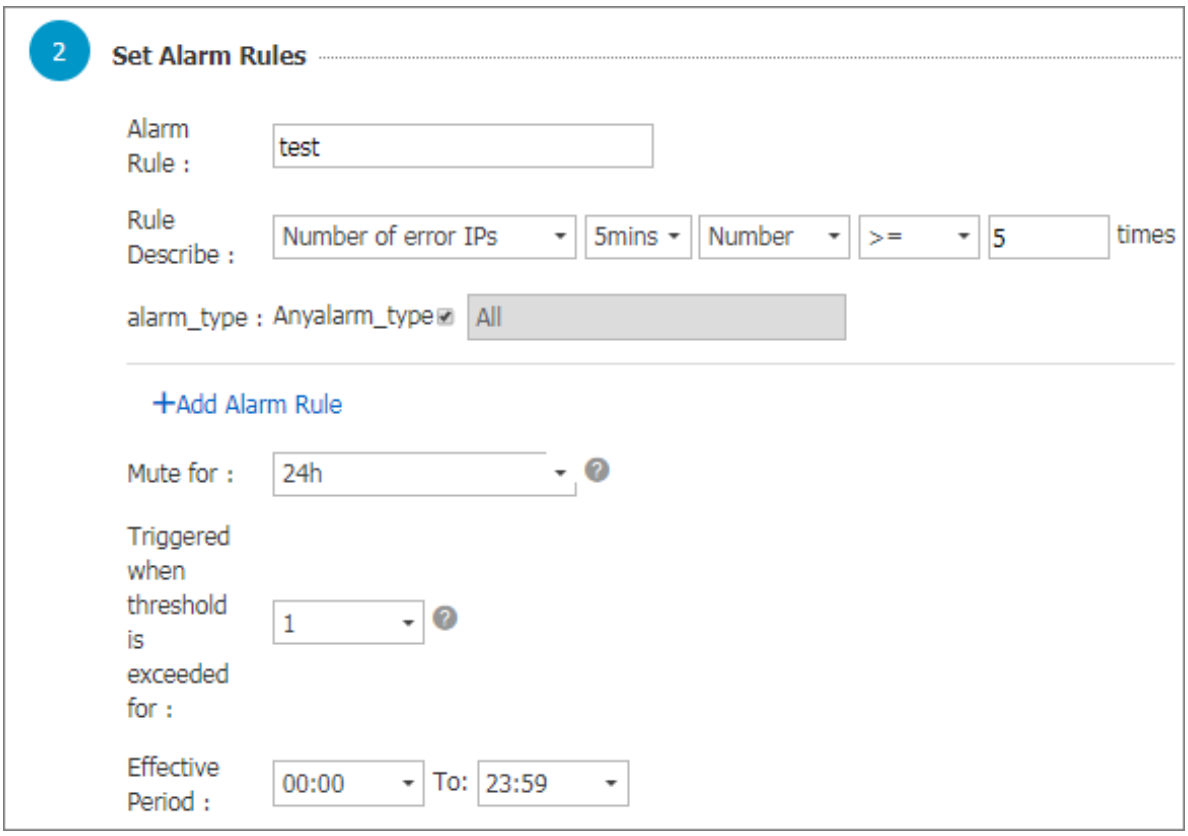
   a. 通知の送信先 - 送信先グループに通知を送信します。

   b. アラームレベル - 警告または情報を選択します。レベルによって通知方法は異なります。

   c. 通知対象と注釈 - デフォルトでは、通知対象は製品名 + モニタリング項目名 + インスタンス ID です。

   d. HTTP コールバック - インターネットからアクセスできる URL を入力します。CloudMonitor は、ここで指定するアドレスに POST リクエストでアラーム通知をプッシュします。現時点では、HTTP プロトコルのみを使用できます。

   図 11-6 : 通知方法



設定したら完了をクリックし、モニタリングポリシーの設定を完了します。

例

Logtail のログ収集状況をモニタリング

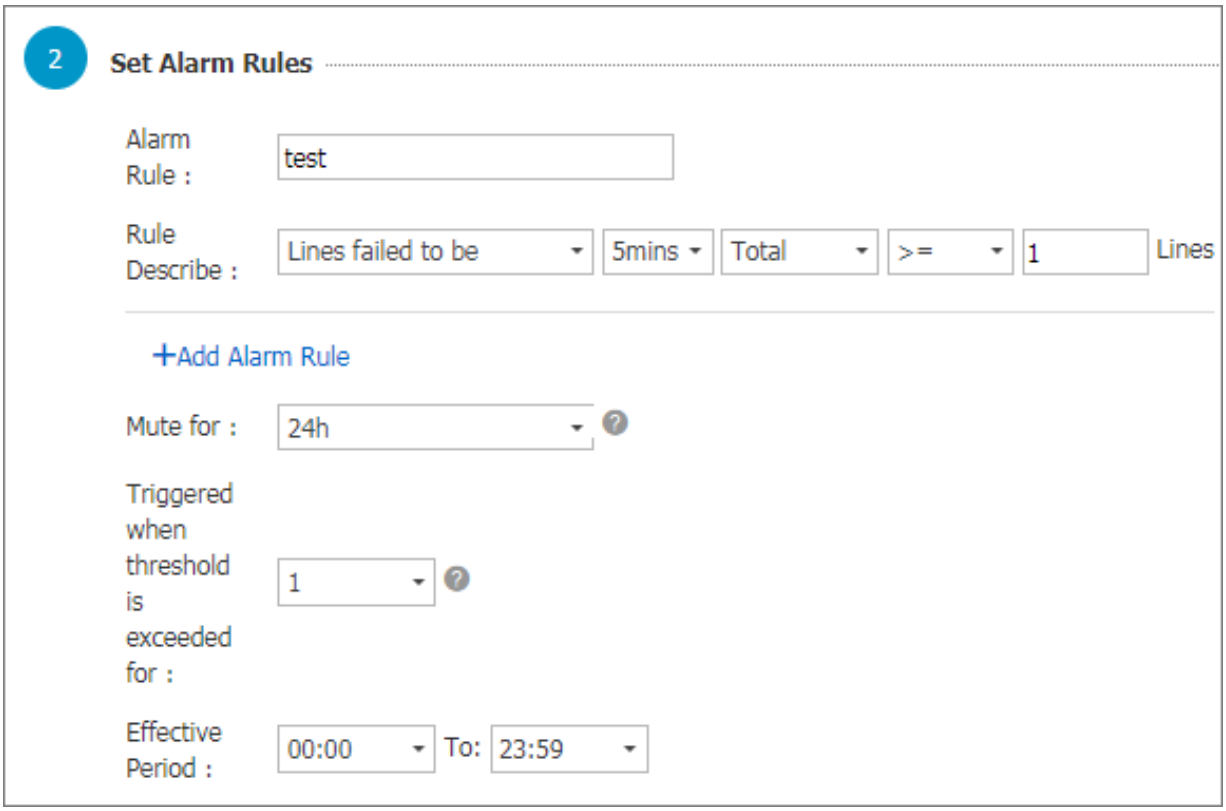
Logtail の稼働中に、ログフォーマットが一致しない、何度も収集されるログファイルがある、といった設定が不適切なために発生するエラーがあります。詳細については、Logtail のよくある質問をご参照ください。エラーを適時に検出できるよう、処理されなかった行や Logtail のエラー数といったメトリックでモニタリングします。

モニタリングルールの設定方法は、以下のとおりです。

アラームルール名を入力し、ルールの説明を設定します。処理されなかった行またはエラー数を選択します。統計期間や方法といったルールを設定します。Logtail のその他のエラーを基にアラームルールを設定することもできます。ログ収集エラーを適時に特定することができます。

下図の例では、5 分以内に 1 回以上接続に失敗するとアラームが送信されます。モニタリング時間は 24 時間です。

図 11-7 : Logtail のログ収集ステータスのモニタリング



シャードの使用状況をモニタリング

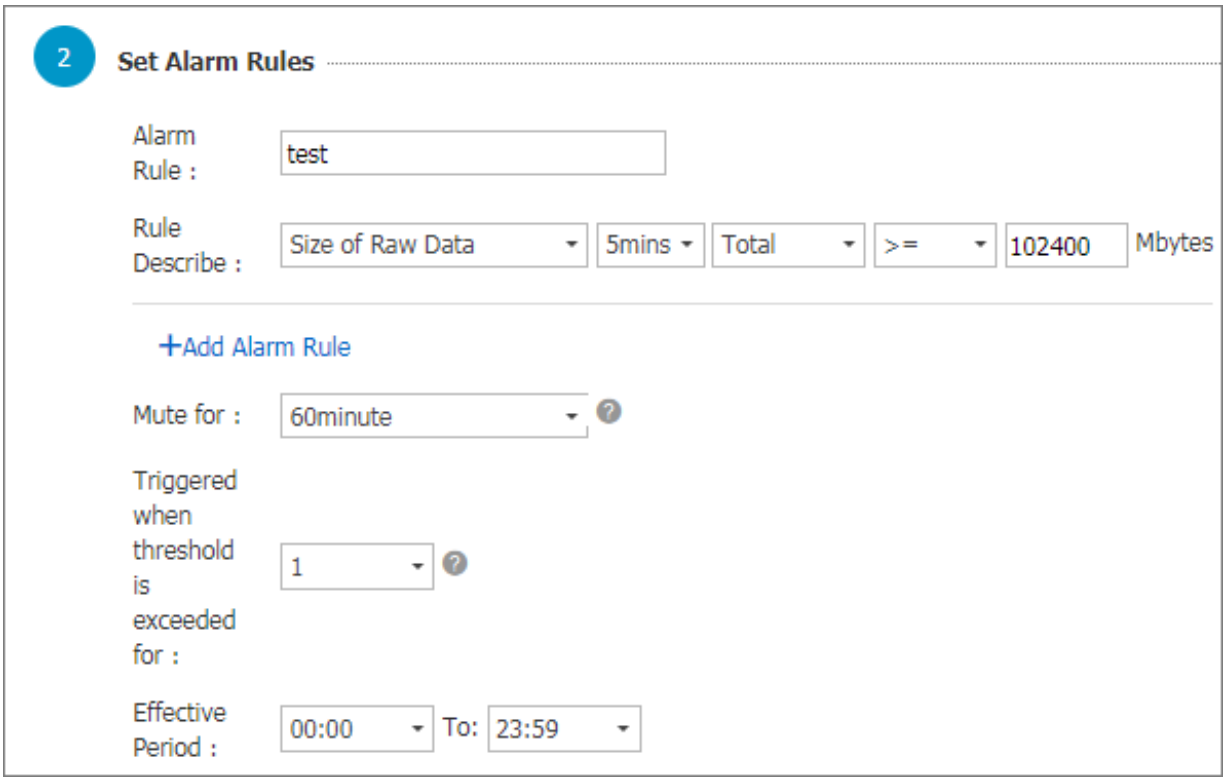
Logstore の各シャードは、最大 5 MB/秒 (毎秒 500 回) で書き込まれ、通常は、これで十分です。上限を超えた場合、 Log Service はリクエストを拒否せずに、処理を試みますが、トラフィックピーク時に上限を超えたデータは保証されません。こういった状況を検出するには、Logstore の送信トラフィックおよび受信トラフィックに対してアラームルールを設定します。ログのデータ量が多く、シャードを増やす必要がある場合は、適当なタイミングでコンソールのシャード数を調整します。

Logstore トラフィックのアラームルールを設定する方法は、以下のとおりです。

解決策 1: トラフィックにアラームルールを設定する

アラームルール名を入力します。生データのサイズを選択します。統計期間と方法を設定します。たとえば、100 GB/5 分を超えた場合にアラームが送信されるようにするには、ルールを 5 分、合計、> =、および 102400 に設定します。5 分以内の合計トラフィックが 102400 MB を超えた場合にアラームは送信されます。

図 11-8 : トラフィックアラートを設定

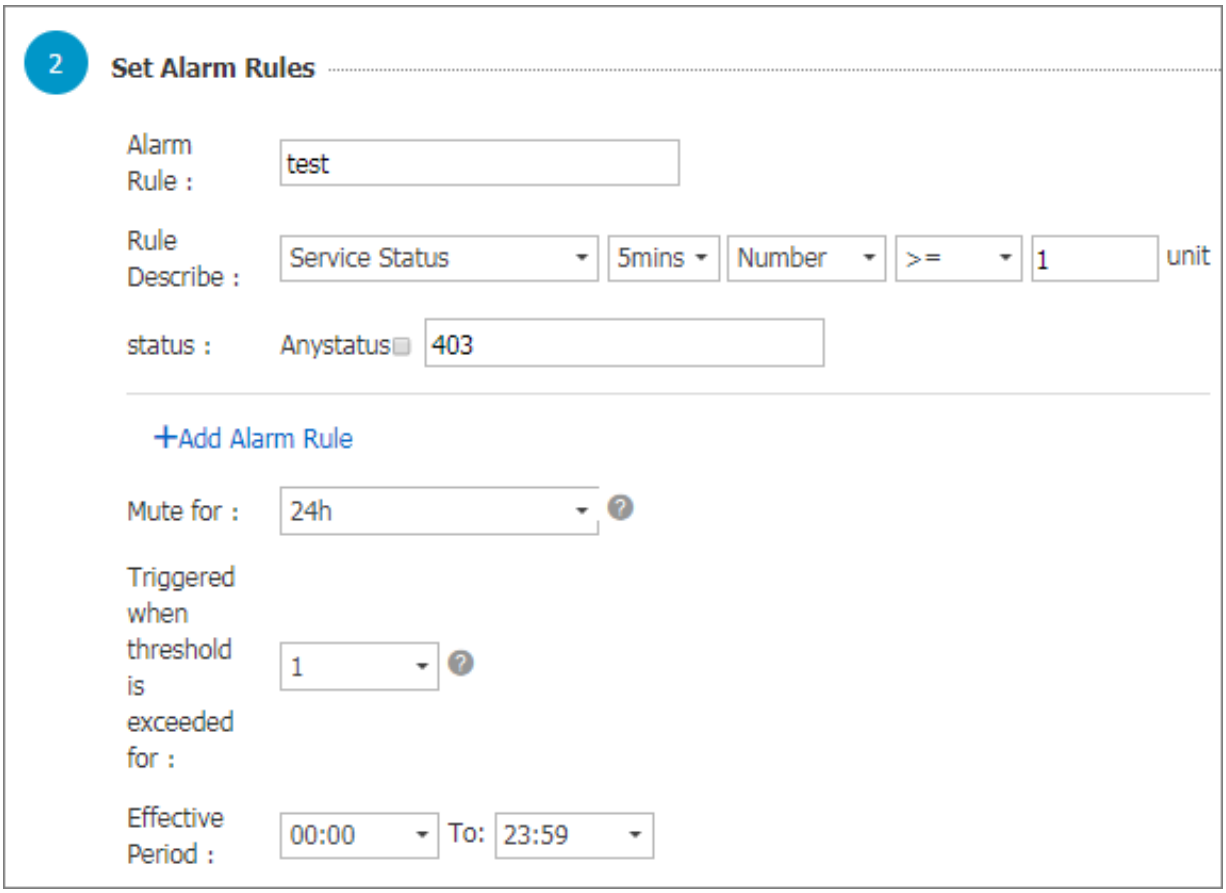


解決策 2: サービスステータスに関するアラームルールを設定する

アラームルール名を入力します。サービスステータスを選択します。統計期間と方法を設定します。たとえば、サービスステータス「403」が 5 分以内に 2 回以上発生した場合にアラームが送

信されるようにするは、ルールを 5 分、数値、>= および 1 を指定し、ステータス欄に 403 と入力します。

図 11-9 : サービスステータスアラームの設定



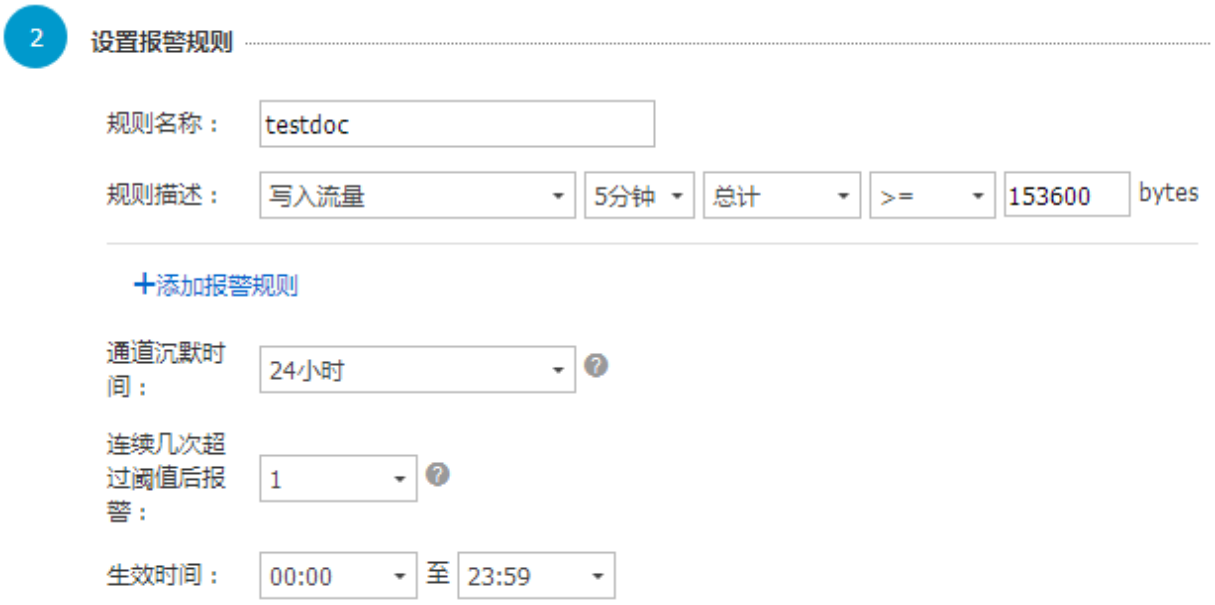
プロジェクトの書き込みトラフィックをモニタリング

プログラムエラーによって大量のログを生成されないよう、各プロジェクトの書き込み上限は 30 GB/分 (生データのサイズ) に設定されています。通常は、これで十分ですが、ログが大量にある場合には、書き込み上限を超える可能性があります。アップグレードする場合は、チケットを起票し、サポートセンターにお問い合わせください。

プロジェクトクォータのモニタリングポリシーの設定は、下図のとおりです。

5 分以内の書き込みトラフィックが 150 GB を超えると、アラーム通知が送信されます。

図 11-10 : プロジェクトの書き込みトラフィックのモニタリング



12 RAM でアクセス制御

12.1 概要

Alibaba Cloud の Resource Access Management (RAM) は、ユーザーアカウント (ユーザー ID) を管理し、リソースへのアクセスを制御できるように設計されています。Alibaba Cloud アカウント下のリソースにアクセスできるユーザーアカウント (従業員、システム、アプリケーションなどのアカウント) を作成し、作成したユーザーのアクセス権を管理することができます。企業内で複数のユーザーが協働してリソースを操作する場合、RAM を使用することにより、Alibaba Cloud アカウントの AccessKey を複数のユーザーで共有する必要がなくなります。ユーザーに必要最小限の権限を付与することで、企業の情報セキュリティリスクを減らすことができます。

Log Service リソースに対する操作を適正に運用管理するには、RAM を使用して、RAM サービスロール、Log Service ユーザーロール、および RAM ユーザーに適切なアクセス権限を与えます。

ユーザーアカウントの管理

RAM ユーザーアカウントの管理作業を行います。Alibaba Cloud アカウント下の RAM ユーザーアカウントおよび RAM ユーザーグループを作成/管理、Log Service を代表するサービスロールを作成、リソースへの操作権限のあるユーザーロールを作成、また、アカウント間の権限付与管理といったことを行うことができます。

Log Service は、API Gateway や Server Load Balancer といったクラウドプロダクトのログを収集するのに役立ちます。設定する前に、クイック権限付与ページでサービスロールを作成し、権限を付与しておきます。

ロール	デフォルトの権限	説明
AliyunLogArchiveRole	AliyunLogArchiveRolePolicy	Log Service の Server Load Balancer のログへのアクセスに使用されるデフォルトのロール。デフォルトで Server Load Balancer ログをエクスポートする権限が含まれます。すばやく権限を付与するには、クイック認可ページにアクセスします。

ロール	デフォルトの権限	説明
AliyunLogDefaultRole	AliyunLogRolePolicy	Log Service のデフォルトのロール。デフォルトで Object Storage Service (OSS) への書き込み権限が含まれます。すばやく権限を付与するには、クイック認可ページにアクセスします。
AliyunLogETLRole	AliyunLogETLRolePolicy	Log Service が ETL のため、他のクラウドプロダクトのリソースへのアクセスに使用されるデフォルトのロール。すばやく権限を付与するには、クイック認可ページにアクセスします。
AliyunMNSLoggingRole	AliyunMNSLoggingRole Policy	Log Service の MNS (Alibaba Cloud Message Service) ログへのアクセスに使用されるデフォルトのロール。OSS への書き込み権限が含まれ、MNS ログをエクスポートする権限が含まれます。すばやく権限を付与するには、クイック認可ページにアクセスします。

RAM

Alibaba Cloud アカウント下の RAM ユーザーアカウント、グループ、ロールに適切なポリシーを割り当てることができます。

また、カスタムポリシーを作成したり、カスタムポリシー/システムポリシーをテンプレートにして、より詳細な設定を行うこともできます。 詳細については、「概要」をご参照ください。

Log Service は、次のシステムポリシーをサポートしています。

ポリシー	タイプ	説明
AliyunLogFullAccess	System policy	Log Service の管理に必要なすべての権限
AliyunLogReadOnlyAccess	System policy	Log Service に対する読み取り権限

シナリオ

RAM ユーザーに Log Service へのアクセス権限を付与

通常は、Alibaba Cloud アカウントは Log Service の日常的な運用保守業務を RAM ユーザーに委任します。その際、RAM ユーザーが業務を遂行できるよう、Alibaba Cloud アカウント下の RAM ユーザーアカウントに権限を付与する必要があります。セキュリティの観点から、RAM ユーザーアカウントには必要最小限の権限を付与することをお勧めします。

設定の詳細については、RAM ユーザーに Log Service へのアクセス権限を付与をご参照ください。

サービスロールでログの読み取り権限を付与

Log Service は、ログの内容に即したアラーム機能を提供します。Log Service サービスアカウントに、ログデータを読み取る権限を付与する必要があります。

設定の詳細については、サービスロールをご参照ください。

ユーザーロールに Log Service の操作実行権限を付与

RAM ユーザーロールは、ID 認証用 AccessKey を有さない仮想ユーザーです。Alibaba Cloud アカウント、RAM ユーザーアカウント、クラウドサービスアカウントといった信頼できるユーザーにロールを割り当てる必要があります。ロールを割り当てられたユーザーは、この RAM ユーザーロールの一時的なセキュリティトークンを受け取ります。ユーザーはこのセキュリティトークンを使用することにより、RAM ユーザーロールとして許可されているリソースにアクセスできます。

- 信頼できるユーザーに Log Service の操作権限を付与し、ユーザーの RAM ロールで Log Service で操作を実行できるようにします。 設定の詳細については、サービスロールをご参照ください。
- モバイルアプリケーションクライアントより Log Service に直接接続し、アプリケーションログを Log Service に直接アップロードできるよう、モバイルアプリケーションに権限を付与します。 設定の詳細については、「モバイルアプリケーションに Log Service への直接接続権を付与」をご参照ください。

12.2 RAM ユーザーに Log Service へのアクセスを許可

Log Service を運用するにあたり、Alibaba Cloud アカウントは RAM (Resource Access Management) ユーザーに Log Service の日常の運用保守業務を委任することがあります。また、RAM ユーザーが Log Service のリソースにアクセスする必要が出てくることもあります。そういった場合、RAM ユーザーが Log Service にアクセスして操作できるよう、Alibaba

Cloud アカウントは RAM ユーザーに権限を付与しておく必要があります。なお、セキュリティ上、RAM ユーザーに付与するアクセス権は必要最小限にとどめておく事を推奨します。

Alibaba Cloud アカウントより RAM ユーザーに Log Service リソースへのアクセスを許可する手順は、次のとおりです。RAM ユーザーの詳細については、「概要」をご参照ください。

1. RAM ユーザーを作成
   a) RAM コンソールにログインします。
   b) 左側のナビゲーションメニューよりユーザーをクリックします。右上にある新規ユーザーをクリックします。
   c) ユーザー情報を入力します。アクセスキーを自動的に生成するチェックボックスをオンにして、OK をクリックします。
2. RAM ユーザーに Log Service リソースへのアクセス権を付与

   Log Service には、`AliyunLogF ullAccess` (読み書き権限) および `AliyunLogR eadOnlyAcc ess` (読み取り権限) の 2 つのシステムポリシーが用意されています。なお、RAM コンソールよりポリシーを作成することもできます。ポリシーの作成方法については、「ポリシーを管理」をご参照ください。RAM ユーザーに読み取り権限を付与する方法は、次のとおりです。

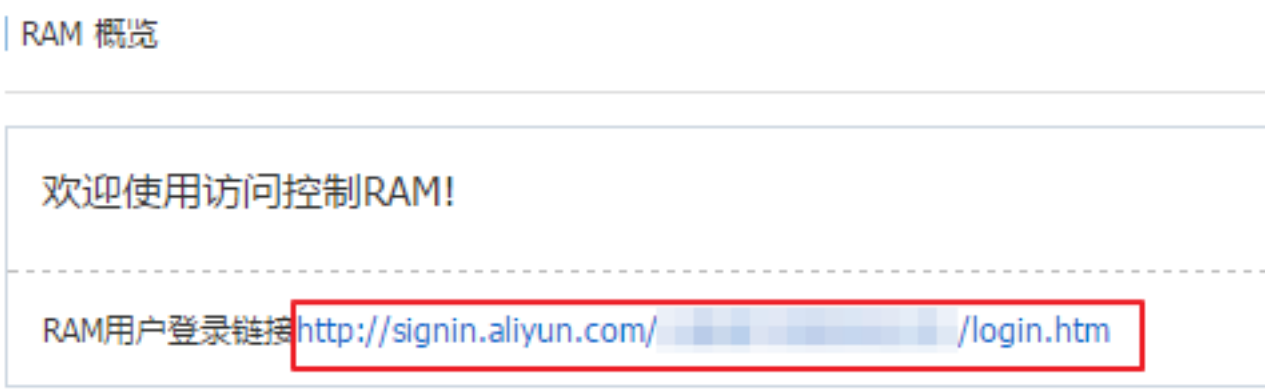
   a) ユーザー管理 ページで、RAM ユーザー名の右側の許可をクリックします。ユーザーのポリシー編集ダイアログボックスが表示されます。
   b) ポリシー一覧より AliyunLogReadOnlyAccess を選択します。

3. RAM ユーザーアカウントでコンソールにログイン

   RAM ユーザーを作成して権限を付与したら、RAM ユーザーは Log Service コンソールにアクセスできます。RAM ユーザーアカウントでコンソールにログインする方法は、次のとおりです。

   a) RAM コンソールの RAM 概要 ページで、RAM ユーザーログインのリンクをクリックします。ステップ 1 で作成したRAM ユーザーのユーザー名およびパスワードを入力し、コンソールにログインします。

   図 12-1 : RAMユーザー



   b) ログイン画面に移動し、手順 1 で作成した RAM ユーザーのユーザー名とパスワードを入力し、コンソールにログインします。

   デフォルトでは、Alibaba Cloud アカウントの ID (aliuid) が アカウント別名 になります。RAM コンソールより設定 > アカウント別名に移動し、アカウント別名を表示および設定します。

12.3 RAM カスタムポリシー

RAM より、Alibaba Cloud アカウント下の RAM ユーザーに権限を付与することができます。

Alibaba Cloud アカウントより、RAM ユーザーが Log Service にアクセスし、操作できるよう権限を付与します。 RAM ユーザーには、システムポリシーおよびカスタムポリシーを付与することができます。

注意事項

- Log Service のセキュリティを確保するには、最小権限の原則 (PoLP) に従うことを推奨します。RAM ユーザーには、必要以上に権限を与えないようにします。
- 一般的に、プロジェクトリストのリソースを確認する RAM ユーザーには、プロジェクトリストに対する読み取り権限で十分です。

- `log : ListProjec t` で、プロジェクトリストを表示する権限を付与します。
  - 本権限を有する RAM ユーザーはプロジェクト一覧を表示できますが、表示するプロジェクトを指定することはできません。
  - 本権限を有さない RAM ユーザーは一切のプロジェクトを表示することもできません。

本ドキュメントでは、次のよくあるカスタムポリシーとその詳細を説明します。

- コンソールよりプロジェクトリスト、プロジェクトを読み取る権限を付与
- コンソールより Logstore を読み取り、クイック照会を作成/利用する権限を付与
- コンソールよりプロジェクト内のすべてのクイック照会/ダッシュボード/ Logstore を読み取る権限を付与
- API よりプロジェクトにデータを書き込む権限を付与
- API よりプロジェクトを読み取る権限を付与
- API より Logstore を読み取る権限を付与

参考：

- RAM ユーザーアカウントが API を介してアクセス可能なリソース一覧
- RAM ユーザーアカウントが API を介して実行できる操作一覧
- RAM ユーザーアカウントが API を介して指定できるポリシー一覧

コンソールよりプロジェクトリスト、プロジェクトを読み取る権限を付与

Alibaba Cloud アカウントより RAM ユーザーに次の権限を付与します。

1. Alibaba Cloud アカウント下のプロジェクトリストを表示する権限
2. Alibaba Cloud アカウントの指定するプロジェクトを読み取る権限

RAM ユーザーに両方の権限を付与するポリシーは、次のとおりです。

```
{
   " Version ": " 1 ",
   " Statement ": [
     {
       " Action ": [" log : ListProjec  t "],
       " Resource ": [" acs : log :*:*: project /*"],
       " Effect ": " Alow "
      },
     {
       " Action ": [
         " log : Get *",
         " log : List *"
       ],
       " Resource ": " acs : log :*:*: project /<プロジェクト名>/*",
       " Effect ": " Allow "
     }
   ]
```

```
  }
```

コンソールより Logstore を読み取り、クイック照会を作成/利用する権限を付与

Alibaba Cloud アカウントより RAM ユーザーに次の権限を付与します。

1. Alibaba Cloud アカウント下のプロジェクトを一覧表示する権限
2. 特定の Logstore に対する読み取り権限、および、クイック照会を作成/利用する権限

RAM ユーザーに両方の権限を付与するポリシーは、次のとおりです。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " log : ListProjec  t "
      ],
      " Resource ": " acs : log :*:*: project /*",
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : List *"
      ],
      " Resource ": " acs : log :*:*: project /<プロジェクト名>/
 logstore /*",
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : Get *",
        " log : List *"
      ],
      " Resource ": [
        " acs : log :*:*: project /<プロジェクト名>/ logstore /<
 Logstore  名>"
      ],
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : List *"
      ],
      " Resource ": [
        " acs : log :*:*: project /<指定するプロジェクトの名前>/ dashboard
 ",
        " acs : log :*:*: project /<指定するプロジェクトの名前>/ dashboard
 /*"
      ],
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : Get *",
        " log : List *",
        " log : Create *"
      ],
      " Resource ": [
        " acs : log :*:*: project /<プロジェクト名>/ savedsearc  h ",
        " acs : log :*:*: project /<プロジェクト名>/ savedsearc  h /*"
```

```
      ],
      " Effect ": " Allow "
    }
  ]
}
```

コンソールよりプロジェクト内のすべてのクイック照会/ダッシュボード/Logstore を読み取る権限を付与

Alibaba Cloud アカウントより RAM ユーザーに次の権限を付与します。

1. Alibaba Cloud アカウント下のプロジェクト一覧を表示する権限
2. 特定の Logstore、すべてのクイック照会とダッシュボードを表示する権限

 注：
RAM ユーザーに、特定の Logstore に対する読み取り権限を付与する場合は、併せてすべてのクイック照会およびダッシュボードを表示する権限も付与する必要があります。

RAM ユーザーに両方の権限を付与するポリシーは次のとおりです。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " log : ListProjec  t "
      ],
      " Resource ": " acs : log :*:*: project /*",
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : List *"
      ],
      " Resource ": " acs : log :*:*: project /<プロジェクト名>/
 logstore /*",
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : Get *",
        " log : List *"
      ],
      " Resource ": [
        " acs : log :*:*: project /<プロジェクト名>/ logstore /<
 Logstore  名>"
      ],
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : Get *",
        " log : List *"
      ],
      " Resource ": [
        " acs : log :*:*: project /<プロジェクト名>/ dashboard ",
        " acs : log :*:*: project /<プロジェクト名>/ dashboard /*"
```

```
      ],
      " Effect ": " Allow "
    },
    {
      " Action ": [
        " log : Get *",
        " log : List *"
      ],
      " Resource ": [
        " acs : log :*:*: project /<プロジェクト名>/ savedsearc  h ",
        " acs : log :*:*: project /<プロジェクト名>/ savedsearc  h /*"
      ],
      " Effect ": " Allow "
    }
  ]
}
```

API 呼び出しでプロジェクトにデータを書き込む権限を付与

RAM ユーザーに、特定のプロジェクトに対する書き込み権限を付与します。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " log : Post *"
      ],
      " Resource ": " acs : log :*:*: project /<プロジェクト名>/*",
      " Effect ": " Allow "
    }
  ]
}
```

API よりプロジェクトを読み取る権限を付与

RAM ユーザーに、特定のプロジェクトに対する読み取り権限を付与します。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " log : ListShards ",
        " log : GetCursorO  rData ",
        " log : GetConsume  rGroupChec  kPoint ",
        " log : UpdateCons  umerGroup ",
        " log : ConsumerGr  oupHeartBe  at ",
        " log : ConsumerGr  oupUpdateC  heckPoint ",
        " log : ListConsum  erGroup ",
        " log : CreateCons  umerGroup "
      ],
      " Resource ": " acs : log :*:*: project /<プロジェクト名>/*",
      " Effect ": " Allow "
    }
  ]
```

```
}
```

API より Logstore を読み取る権限を付与

RAM ユーザーに、特定のプロジェクトに対する読み取り権限を付与します。

```
{
  " Version ": " 1 ",
  " Statement ": [
    {
      " Action ": [
        " log : GetCursorO  rData ",
        " log : GetConsume  rGroupChec  kPoint ",
        " log : UpdateCons  umerGroup ",
        " log : ConsumerGr  oupHeartBe  at ",
        " log : ConsumerGr  oupUpdateC  heckPoint ",
        " log : ListConsum  erGroup ",
        " log : CreateCons  umerGroup "
      ],
      " Resource ": [
        " acs : log :*:*: project /<プロジェクト名>/ logstore /<
 Logstore  名>",
        " acs : log :*:*: project /<プロジェクト名>/ logstore /<
 Logstore  名>/*"
      ],
      " Effect ": " Allow "
    }
  ]
}
```

12.4 サービスロール

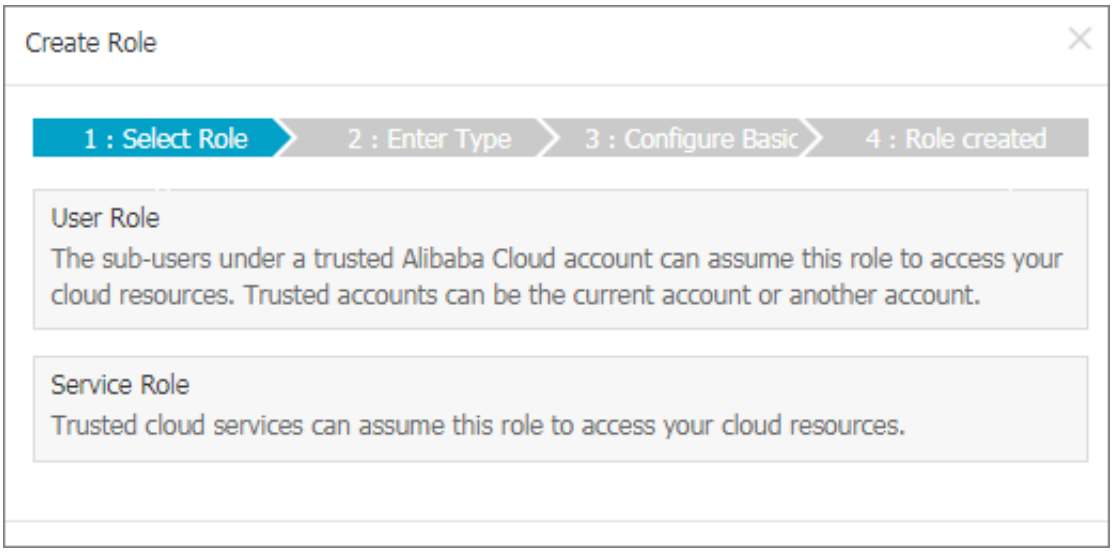
Log Service では、ログの内容に応じたアラームを設定することができます。Log Service サービスアカウントでログを読み込むには、ログへのアクセス権がそのサービスアカウントに付与されている必要があります。既に本ドキュメントに従って権限を付与している場合は、以下をスキップし、アラームルールの作成に進みます。サービスロールに権限を付与する方法は、以下のとおりです。

RAM ロールを作成する

1. RAM (Resource Access Management) コンソールにログインします。左側のナビゲーションメニューよりロール管理をクリックし、右上の新規ロールをクリックします。ロール作成ダ

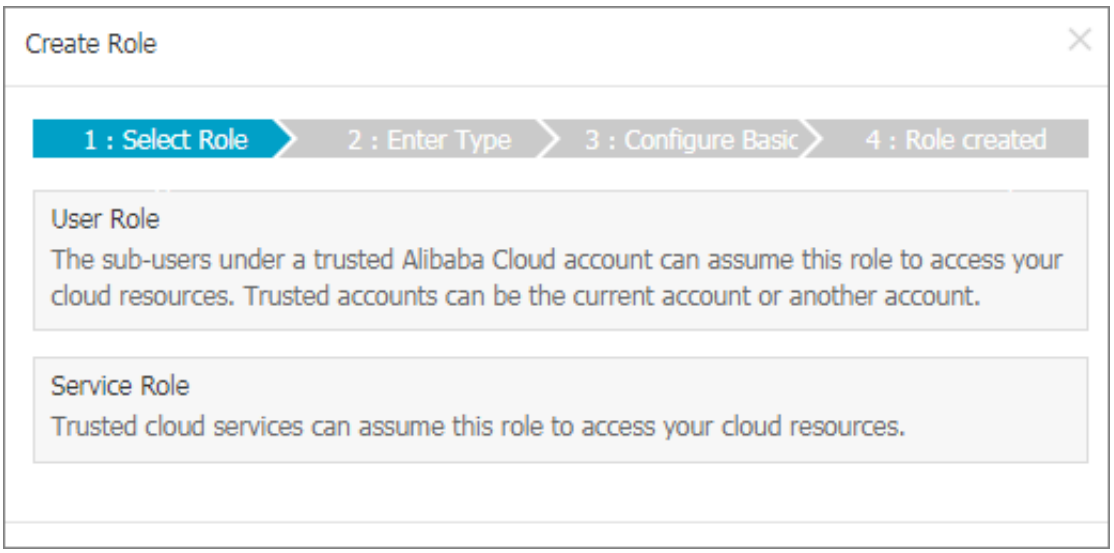
イアログボックスが表示されます。ロールタイプのステップでは、サービスロールを選択します。

図 12-2 : ロールタイプの選択



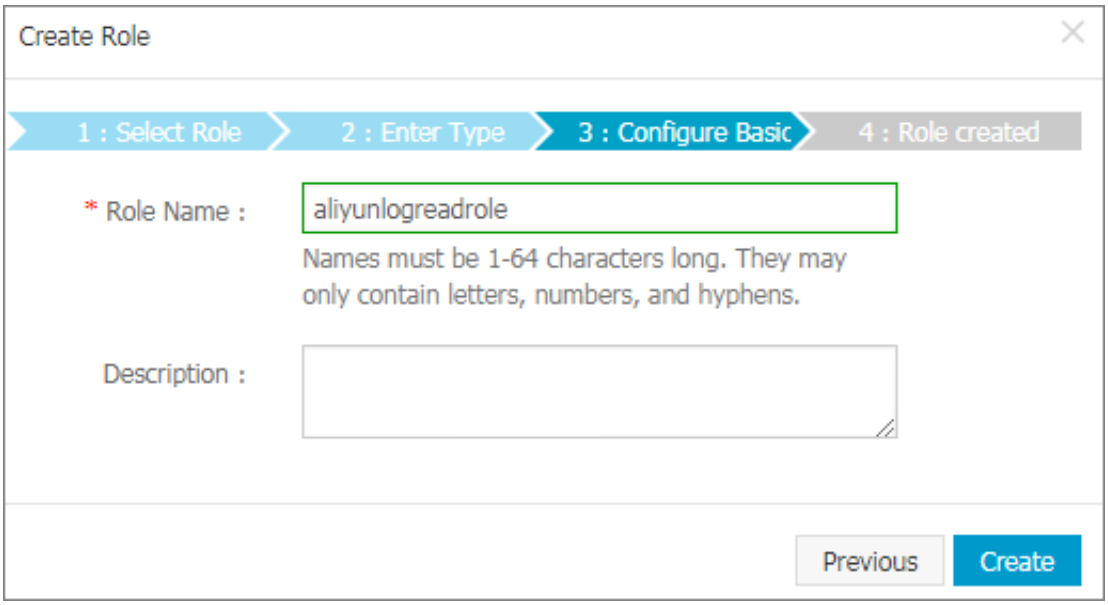
2. タイプ情報を入力ステップでは、LOG Log Serviceを選択します。

図 12-3 : タイプ情報を入力



3. ロール名欄に aliyunlogreadrole と入力し、作成 をクリックします。(データを読み取るために行使されるロールとなるため、ロール名は変更しないようにします。)

図 12-4 : ロールの基本情報の設定

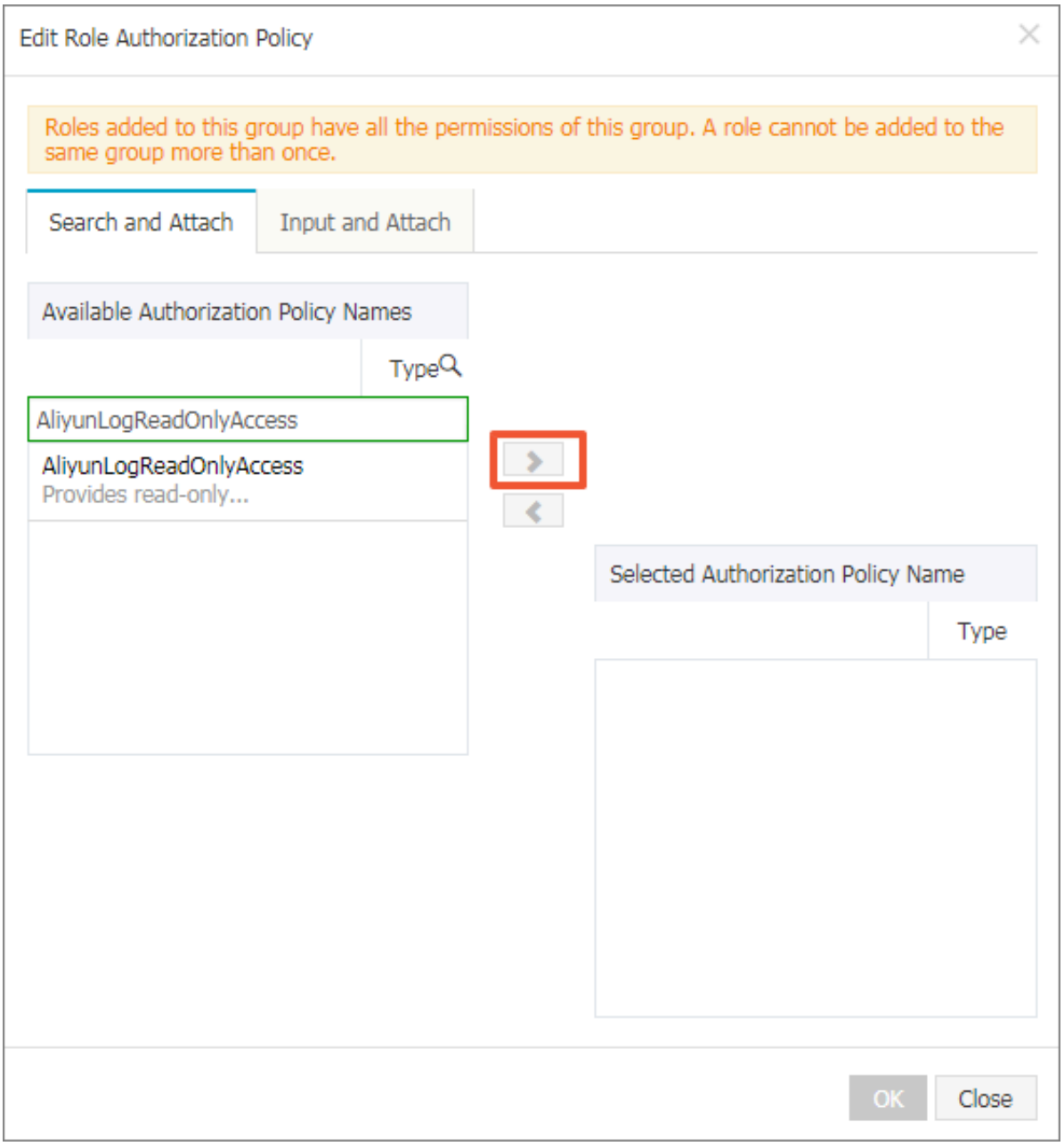


ログデータへのアクセス権をロールに付与する

ロールを作成したら、ロール管理ページで、aliyunlogreadrole の右の許可をクリックします。ポリシーの編集ダイアログボックスが表示されます。ポリシー一覧よりAliyunLogR

eadOnlyAccessを選択し、→をクリックします。選択されているポリシーに追加されますので、OKをクリックします。

図 12-5 : ロールのポリシー編集



以上で、Log Service は、指定した Logstore から定期的にデータを読み込み、アラームを確認することができます。

12.5 ユーザーロール

RAM (Resource Access Management) におけるロールは、ユーザーと同様に管理します。ただし、RAM ユーザーのように、認証に必要な AccessKey を RAM ユーザーロールに作成する

ことはできません。認証に必要な AccessKey を持つアカウントが、RAM ユーザーロールを行使する必要があります。つまり、AccessKey を持つアカウントに、RAM ユーザーロールを割り当てる必要があります。RAM ユーザーロールを割り当てられたアカウントは、リソースにアクセスするための一時的なセキュリティトークンを受け取ります。アカウントはこの一時的なセキュリティトークンを使って RAM ユーザーロールに付与されている権限を行使します。

RAM ユーザーロールに付与されている Log Service 操作権をアカウントが行使するには、以下の設定を行います (ユーザーロールを作成 → Alibaba Cloud アカウントを指定 → ユーザーロールに権限を付与 → RAM ユーザーに AssumeRole 権限を付与 → ユーザーロールより一時的なセキュリティトークンを取得)。

詳細は、「ユーザー」をご参照ください。

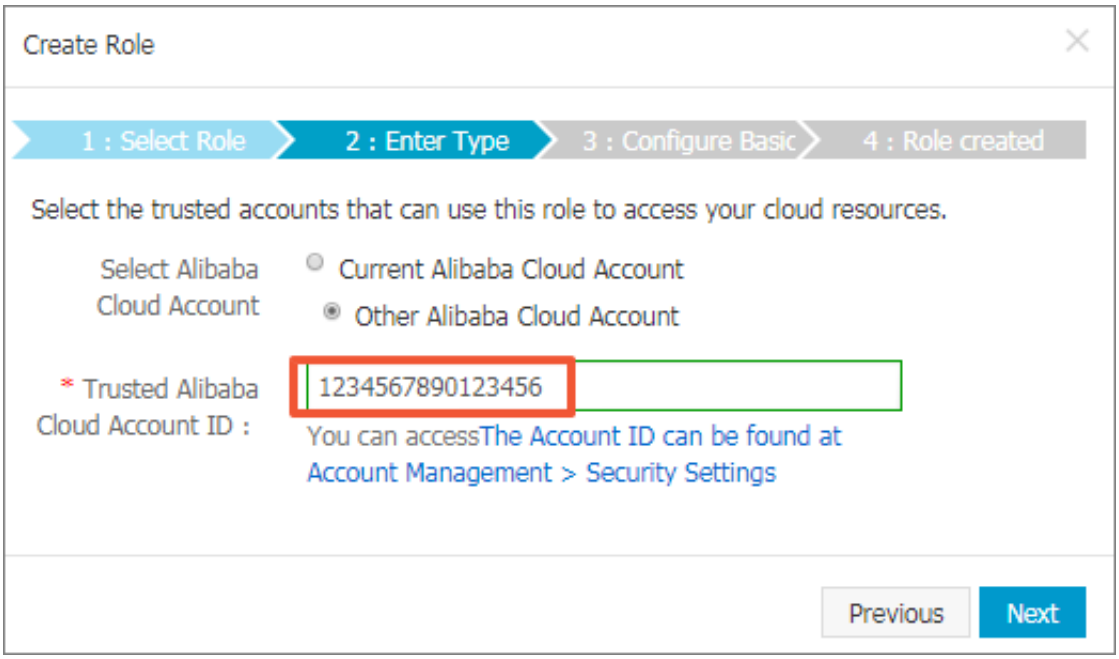
ステップ 1. ユーザーロールを作成し、認証可能なアカウントを指定

1. RAM コンソールにログインします。左側のナビゲーションメニューのロール管理をクリックします。
2. 右上隅の新規ロールをクリックします。ロール作成 ダイアログボックスが表示されます。
3. ロールタイプの選択ステップでは、ユーザーロールを選択します。
4. タイプの入力ステップでは、Alibaba Cloud アカウントを選択します。

> 注 :
>
> - Alibaba Cloud アカウント下の RAM ユーザーに割り当てるロールを作成する場合 (たとえば、モバイルアプリから直接 Log Service リソースを操作するためのロールを作成する場合) は、ログイン中の Alibaba Cloud アカウントを選択します。
> - アカウント間リソース操作といった、別の Alibaba Cloud アカウント下の RAM ユーザーが行使するロールを作成する場合は、 別の Alibaba Cloud アカウントを選択

し、Alibaba Cloud アカウント ID欄に別の Alibaba Cloud アカウントの ID を入力します。

図 12-6 : ロール作成



5. 基本情報ステップでは、ロール名および説明を入力し、作成をクリックします。

ステップ 2. ユーザーロールに権限を付与する

作成したユーザーロールには、まだ何の権限もありません。ユーザーロールに、Log Service 操作を実行する権限を付与する必要があります。ユーザーロールに権限を付与すると、前のステップで指定したアカウントが Log Service を操作できるようになります。

注 :
ユーザーロールには、システムポリシーおよびカスタムポリシーを複数付与することができます。本ドキュメントでは、ユーザーロールに、Log Service を管理する権限を付与します。

1. RAM コンソールの左側のナビゲーションメニューよりロール管理をクリックします。
2. ロール名の右の許可をクリックします。
3. AliyunLogFullAccess 権限をクリックし、OK をクリックします。

詳細は、「RAM 権限付与」をご参照ください。

ステップ 3. RAM ユーザーにユーザーロールを割り当てる

ユーザーロールは、AccessKey をもつアカウントしか行使することができません。ただし、Alibaba Cloud アカウント自身がユーザーロールを行使することはできません。Alibaba

Cloud アカウント下の RAM ユーザーに権限を委譲し、RAM ユーザーにユーザーロールを行使させる必要があります。RAM ユーザーのみがユーザーロールを行使することができます。

したがって、Alibaba Cloud アカウントは、その RAM ユーザーに AssumeRole 権限を付与する必要があります。RAM ユーザーに STS (Security Token Service) の AssumeRole API を呼び出す権限を付与しない限り、RAM ユーザーは Alibaba Cloud アカウントの代わりに、ステップ 1 で作成したユーザーロールを行使できません。

1. Alibaba Cloud アカウントで RAM コンソールにログインします。
2. ユーザー管理ページで、RAM ユーザーの右側の許可をクリックします。

   RAM ユーザーをまだ作成していない場合は、「ユーザー」を参照してユーザーを作成します。
3. システムポリシー AliyunSTSAssumeRoleAccess を選択し、OKをクリックします。

ステップ 4. RAM ロールの一時セキュリティトークンを取得

RAM ユーザーに AssumeRole を付与すると、RAM ユーザーは AccessKey を使用して STS (Security Token Service) の AssumeRole API を呼び出し、ロールの一時的なセキュリティトークンを取得することができるようになります。

AssumeRole API の呼び出し方法については、「はじめに」をご参照ください。

STS SDK を使用して AccessKeyId、AccessKeySecret、および SecurityToken を取得すると、Log Service SDK を使用して Log Service にアクセスできるようになります。

AccessKeyId、AccessKeySecret、および SecurityToken を使用してログクライアントを初期化する方法は、次の例のとおりです。Java SDK の使用方法については、「Java SDK」をご参照ください。

```
package   sdksample ;
import   java . util . ArrayList ;
import   java . util . List ;
import   java . util . Vector ;
import   java . util . Date ;
import   com . aliyun . openservic  es . log . Client ;
import   com . aliyun . openservic  es . log . common .*;
import   com . aliyun . openservic  es . log . exception .*;
import   com . aliyun . openservic  es . log . request .*;
import   com . aliyun . openservic  es . log . response .*;
import   com . aliyun . openservic  es . log . common . LogGroupDa
ta ;
import   com . aliyun . openservic  es . log . common . LogItem ;
import   com . aliyun . openservic  es . log . common . Logs . Log ;
import   com . aliyun . openservic  es . log . common . Logs . Log .
Content ;
import   com . aliyun . openservic  es . log . common . Logs .
LogGroup ;
import   com . aliyun . openservic  es . log . common . Consts .
CursorMode ;
```

```
 public   class   sdksample  {
     public   static   void   main ( String   args [])  throws
 LogExcepti  on ,  Interrupte  dException  {
         String   endpoint  = "< log_servic  e_endpoint >"; // 前のス
テップで作成したプロジェクトの属するリージョンのエンドポイントを指定
         String   accessKeyI  d  = ""< your_acces  s_key_id >"; // お客
様の  Alibaba   Cloud  アカウントの  AccessKey   ID  を指定
         String   accessKeyS  ecret  = ""< your_acces  s_key_secr  et
 >"; // お客様の  Alibaba   Cloud  アカウントの  AccessKey   Secret  を指
定
     String   securityTo  ken  = ""< your_secur  ity_token >"; // ロー
ルの  Security   Token  を指定
         String   project  = ""< project_na  me >"; // 前のステップで作成
したプロジェクトの名前を指定
         String   logstore  = ""< logstore_n  ame >"; // 前のステップで作
成した  Logstore  の名前を指定
        // クライアントインスタンスを作成
         Client   client  =  new   Client ( endpoint ,  accessKeyI  d
 ,  accessKeyS  ecret );
    //  SecurityTo  ken  を指定
     client . SetSecurit  yToken ( securityTo  ken );
        // ログを書き込む
         String   topic  = "";
         String   source  = "";
        //  10  パケットを続けて送信、 1  パケットにつきログ  10  件
         for  ( int   i  =  0 ;  i  <  10 ;  i ++) {
             Vector < LogItem >  logGroup  =  new   Vector < LogItem
 >();
             for  ( int   j  =  0 ;  j  <  10 ;  j ++) {
                 LogItem   logItem  =  new   LogItem (( int ) ( new
 Date (). getTime () /  1000 ));
                 logItem . PushBack (" index "+ String . valueOf ( j
 ),  String . valueOf ( i  *  10  +  j ));
                 logGroup . add ( logItem );
            }
             PutLogsReq  uest   req2  =  new   PutLogsReq  uest (
 project ,  logstore ,  topic ,  source ,  logGroup );
             client . PutLogs ( req2 );
    }
    }
}
```