

Zabbix 最新 LTS バージョン 6.0 の新機能解説

SRA OSS, Inc. 日本支社
OSS 事業本部 基盤技術グループ
赤松 俊弘



whoami

- ☑ 赤松 俊弘 (Toshihiro Akamatsu)
- ☑ SRA OSS, Inc. 日本支社
OSS 事業本部 基盤技術グループ
- ☑ Zabbix 認定プロフェッショナル

職務

- ☑ PostgreSQL 以外の OSS 全般の技術サポート、構築
- ☑ 主に Zabbix を担当

Zabbix 6.0

- 長期サポート（LTS）バージョン
 - フルサポート 3 年、リミテッドサポート 2 年
- Alpha 6 までリリース（2021/11/18 現在）
- 正式版のリリースは今冬予定

5.0 LTS

フルサポート 2023/05

リミテッドサポート 2025/05

4.0 LTS

リミテッドサポート 2023/10

Zabbix 6.0 対応データベース

DBMS	バージョン
MySQL/Percona	8.0.X
MariaDB	10.5.X ~ 10.6.X
PostgreSQL	13.X
Oracle	19.c ~ 21.c
TimescaleDB	2.0.1 ~ 2.3
SQLite	3.3.5 ~ 3.34.X

Zabbix 6.0 の新機能 (2021/11/18 現在)



ビジネスサービス監視



HA とスケーラビリティ



高度な障害検知



監視の簡易化と
制限解除



セキュリティ向上

ビジネスサービス監視

Zabbix のサービス監視とは

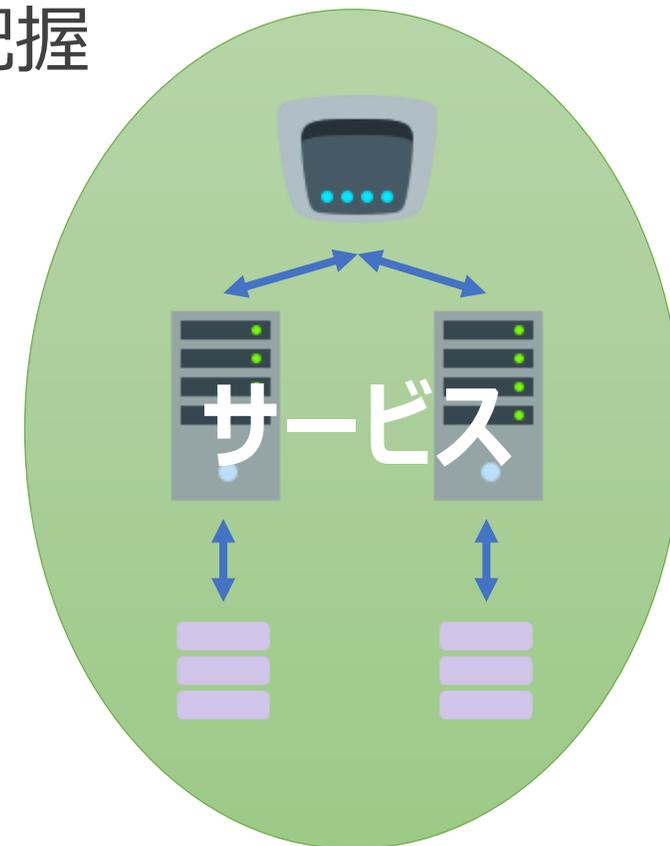
- 監視対象のインフラをハイレベル（ビジネスレベル）で把握

IT サービスの可用性

IT サービスの SLA

IT インフラの構造

IT インフラの弱点特定



サービス

名前 タグ And/Or Or

ステータス

タグ 含む 値

追加

名前	ステータス	根本原因	SLA	タグ
Corporate Web site 2	重度の障害	PostgreSQL is down	99.9990	
ERP 3	正常		99.9000	
SFA 2	正常		99.9000	

3件のうち3件を表示しています

サービス監視の機能更新

- タグによる障害の紐づけ
- 柔軟なステータス計算および伝播
- 根本原因の表示
- サービスのステータス変化によるアラート通知
- サービスへの権限付与
- サービスの複製

タグによる障害の紐づけ

- サービスへの障害の紐づけがトリガーからタグに変更

サービス

サービス **SLA ●** タグ 子サービス

* 名前

親サービス
検索文字列を入力

障害タグ

名前	処理内容	値	アクション
<input type="text" value="Service"/>	等しい ▼	<input type="text" value="Corp Web site"/>	削除
<input type="text" value="Server"/>	等しい ▼	<input type="text" value="DB"/>	削除

[追加](#)

柔軟なステータス計算および伝播（1）

- 子サービスのステータスから親サービスへのステータスを計算する方法が一新

ステータス計算ルール	説明
子サービスで最も重要	子サービスの障害のうち最も高い深刻度を親サービスのステータスとする
すべての子が障害であったときに最も重要	すべての子サービスが障害であった場合に、その障害のうち最も高い深刻度を親サービスのステータスとする
ステータスを正常に設定	ステータスは計算せず常に正常

* ソート順(0->999)

ステータス計算ルール ⓘ

- すべての子が障害であったときにもっとも重要
- 子サービスでもっとも重要
- すべての子が障害であったときにもっとも重要
- ステータスを正常に設定

柔軟なステータス計算および伝播 (2)

- さらに柔軟なステータス計算のための追加ルール
 - ルールに当てはまる場合に親に伝播する深刻度も合わせて指定

追加のルール

少なくとも **N** 個の子サービスの状態が**ステータス以上**

少なくとも **N%** の子サービスの状態が**ステータス以上**

N 個未満の子サービスのステータスが**ステータス以下**

子サービスの **N%** 未満のステータスが**ステータス以下**

ステータスが**ステータス以上**の子サービスの重みが少なくとも **W**

ステータスが**ステータス以上**の子サービスの重みが少なくとも **N%**

ステータスがステータス以下の子サービスの重みが **W** 未満

ステータスがステータス以下の子サービスの重みが **N%** 未満

ステータス計算ルール i すべての子が障害であったときにもっとも重要 ▼

高度な設定

追加のルール

名前

警告 - 少なくとも1個の子サービスの状態が警告以上

軽度の障害 - 少なくとも子サービスの50%の状態が警告以上

[追加](#)

ステータス伝搬ルール 深刻度を増やす ▼

1 2 3 4 5

重み

柔軟なステータス計算および伝播 (3)

- ステータス伝播時のルールも詳細に設定可能

ステータス伝播ルール	説明
そのまま	子サービスのステータスをそのまま伝播
深刻度を増やす	深刻度を 1 ~ 5 増やして伝播
深刻度を減らす	深刻度を 1 ~ 5 減らして伝播
このサービスを無視	ステータスを伝播しない
固定ステータス	常に同じステータスを伝播

ステータス計算ルール ⓘ すべての子が障害であったときにもっとも重要 ▼

高度な設定

追加のルール

名前

警告 - 少なくとも1個の子サービスの状態が警告以上

軽度の障害 - 少なくとも子サービスの50%の状態が警告以上

[追加](#)

ステータス伝播ルール 深刻度を増やす ▼

1 2 3 4 5

重み

根本原因の表示

- サービスのステータスに影響を与えている障害のリストを「根本原因」列に表示

<input type="checkbox"/> 名前	ステータス	根本原因	SLA
<input type="checkbox"/> Corporate Web site 2	重度の障害	PostgreSQL is down	99.9990
<input type="checkbox"/> ERP 3	正常		99.9000
<input type="checkbox"/> SFA 2	正常		99.9000



根本原因に表示される障害は {SERVICE.ROOTCAUSE} マクロで取得可能

サービスのステータス変化によるアラート通知

- トリガーによる障害と同様にサービスのステータス変化時にもアクションによる通知が可能

サービスアクション ▾

名前 ステータス

<input type="checkbox"/> 名前 ▲	実行条件	実行内容
<input type="checkbox"/> Corporate Web site action	サービス 等しい Corporate Web site	ユーザーグループにメッセージを送信: Zabbix administrators via Email

サービスへの権限付与

- ユーザロール単位でサービスへの Read-Write 権限が付与可能
 - 個々のサービス、またはタグによる指定が可能
 - サービスのマルチテナント対応

サービスへのアクセス

サービスの表示/設定 なし すべて サービスのリスト

検索文字列を入力

タグとサービスの表示/設定

サービスの表示のみ なし すべて サービスのリスト

HA とスケーラビリティ

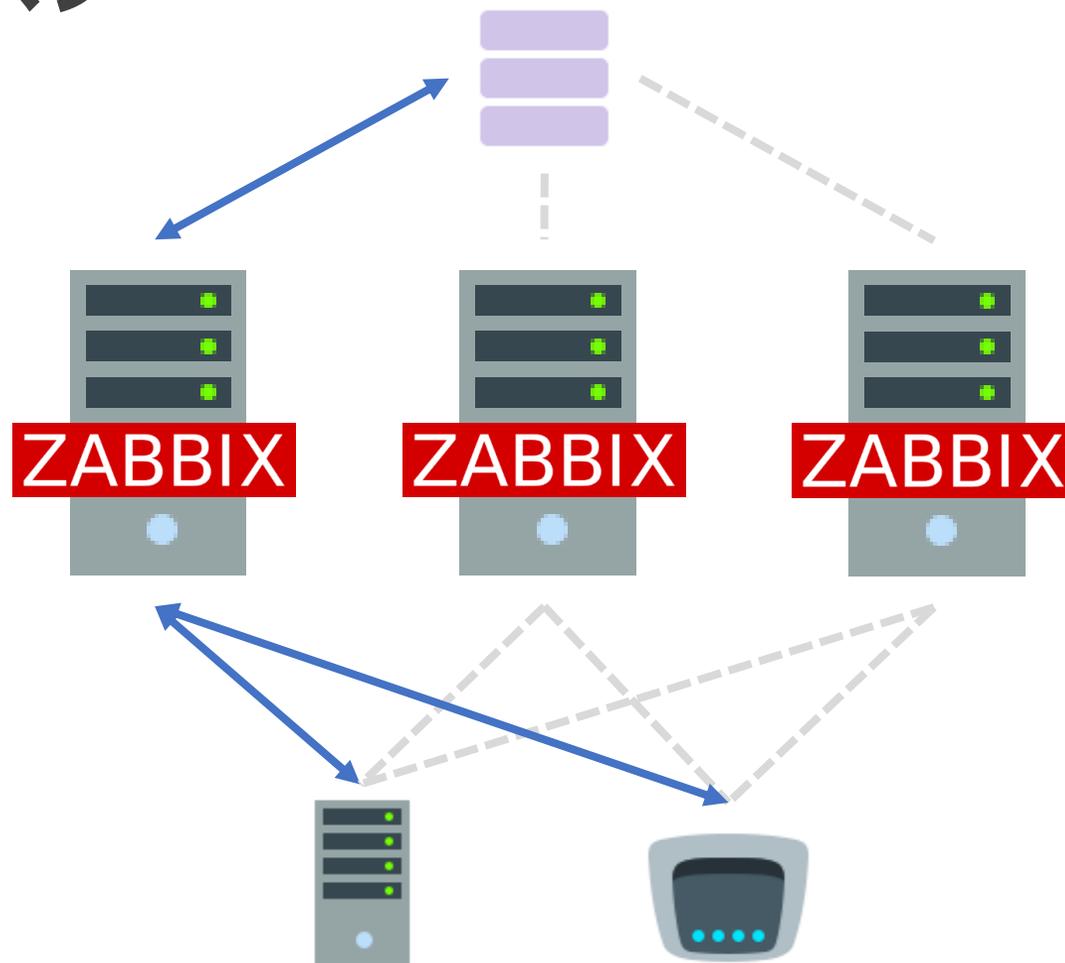
Zabbix server の HA クラスタ

- Active/Standby
- 1 つの DB を共有
- マイナーバージョン間での互換性あり



設定方法

- 各 HANodeName にノード名を設定
- 各 DBHost に同じ DB を設定



HA クラスタの状態確認および監視

- 「システム情報」ウィジェット
 - HA クラスタの状態
- 「レポート」⇒「システム情報」画面
 - HA クラスタの状態と各ノードの状態
- ha_status ランタイムコントロールオプション
 - 各ノードの状態 (zabbix_server.log に出力)
- zabbix[cluster,discovery,nodes] インターナルアイテム
 - 各ノードの状態 (JSON 形式)

1秒あたりの監視項目数(Zabbixサーバーの要求パフォーマンス)	1.69	
HAクラスター	有効	フェールオーバーの遅延: 1分

➡ 詳細は [Appendix I](#) へ

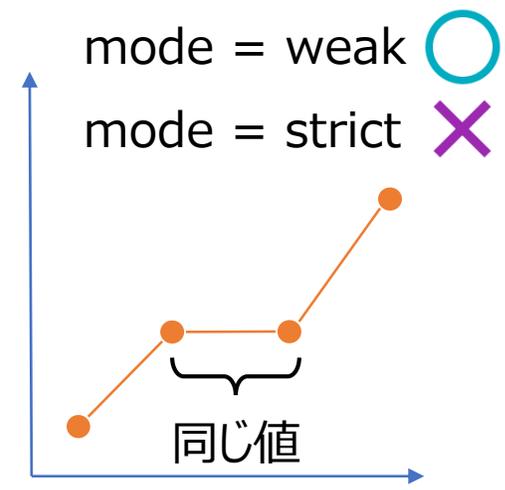
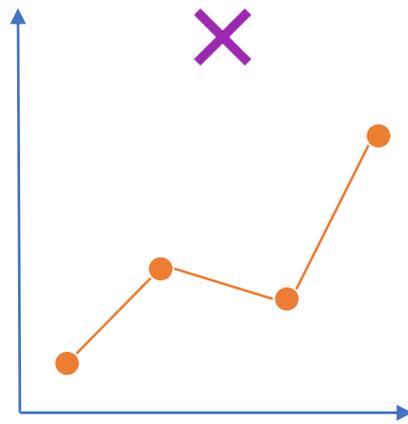
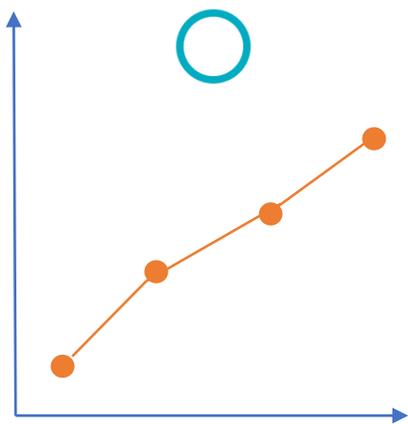
**Zabbix server の HA クラスタに関する
より詳細な情報は
11/19 (金) 14:30 からの
Zabbix 社の講演で！**

高度な障害検知

単調変化の検知

- `monoinc()` 関数 : 任意期間の単調増加を検知
- `monodec()` 関数 : 任意期間の単調減少を検知

i `monoinc()` の場合



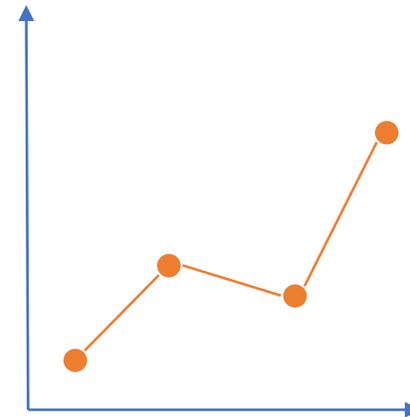
変化回数の検知

- `changecount()` 関数 : 任意期間に隣接する値の間で変化があった回数を返す

i

mode

- `all` : 増減問わずカウント ⇒ 3
- `inc` : 増加のみカウント ⇒ 2
- `dec` : 減少のみカウント ⇒ 1



Prometheus パターンの保存前処理の演算子追加

- != : 指定した文字列と等しくないラベルを選択
- !~ : 指定した正規表現と等しくないラベルを選択

i

例

- `node_filesystem_avail_bytes{device!="tmpfs"}`
: device ラベルが tmpfs 以外
- `wmi_service_state{name!~"^dhcp"}==1`
: name ラベルが dhcp から始まるもの以外

Prometheus histogram 対応の関数

Prometheus histogram タイプのメトリクスは解析が困難だった



- `rate()` : 指定期間の 1 秒ごとの増加率を計算
- `histogram_quantile()` : 指定したヒストグラムのバケットのクォンタイルを計算
PromQL の `histogram_quantile` に相当

上記 2 関数の補完用として以下関数も追加

- `bucket_rate_foreach()` : `histogram_quantile` のパラメータに利用
- `bucket_percentile()` : ヒストグラムのバケットからパーセンタイルを計算



監視の簡易化と制限解除

新規追加されたアイテムキー

- agent.hostmetadata : エージェントのホストメタデータ
- kernel.openfiles : 現在開いているファイルディスクリプタ数
- net.tcp.socket.count[] : パラメータに一致する TCP ソケット数
- net.udp.socket.count[] : パラメータに一致する UDP ソケット数
- vfs.dir.get[] : JSON 形式のディレクトリ情報
- vfs.file.get[] : JSON 形式のファイル情報
- vfs.file.owner[] : ファイルの所有者情報
- vfs.file.permissions[] : ファイルのパーミッション

 詳細は [Appendix II](#) へ

更新されたアイテムキー

- `vfs.file.cksum[]` : 第 2 パラメータ `<mode>` が追加
(`crc32` (デフォルト), `md5`, `sha256`)
- `vfs.file.size[]` : 第 2 パラメータ `<mode>` が追加
(`bytes` (デフォルト), `lines`)
- `vfs.fs.discovery` : Windows の場合、ボリューム名を
{#FSLABEL} マクロで取得
- `vfs.fs.get` : Windows の場合、ボリューム名を
{#FSLABEL} マクロで取得

新規追加された関数

- `count()` : `foreach` 関数によって返される配列の値を集計
- `item_count()` : フィルタ条件に合うアイテム数を集計
`count(exists_foreach(item_filter))` に相当
- `exists_foreach` : フィルタ条件に合う有効なアイテム数を配列で返す



例

```
count(max_foreach(/*/proc.num[*], 1h))  
⇒ 過去 1 時間でデータ収集が行われた  
proc.num アイテム数
```



`count` と履歴系の `foreach` 関数 (`max_foreach`, `avg_foreach` など) を併用すると性能に影響が出る可能性があるが、`exists_foreach` では影響はない

トリガー関数デバッグ用マクロ

- トリガー条件式/復旧条件式を実際の値で解決した結果
 - {TRIGGER.EXPRESSION.EXPLAIN}
 - {TRIGGER.EXPRESSION.RECOVERY.EXPLAIN}
- トリガー条件式/復旧条件式の N 番目の関数を実際の値で解決した結果
 - {FUNCTION.VALUE<1-9>}
 - {FUNCTION.RECOVERY.VALUE<1-9>}



例

```
last(/host/system.cpu.util)>80 and monoinc(host/system.cpu.util, #2)=1
```

```
{TRIGGER.EXPRESSION.EXPLAIN} ⇒ 95>80 and 1=1
```

Github メディアタイプの追加

- Webhook で Github の issue を登録可能



Zabbix get、Zabbix sender のタイムアウト

zabbix_get、zabbix_sender コマンドで
タイムアウトを指定するパラメータをサポート

```
-t <seconds>  
--timeout <seconds>
```

- Zabbix get : 1 ~ 30 秒 (デフォルト: 30 秒)
- Zabbix sender : 1 ~ 300 秒 (デフォルト: 60 秒)

その他

- MikroTik のテンプレートの追加
- Web 監視で圧縮コンテンツをサポート
- 保存前処理等で使用できる JavaScript で PATCH、HEAD、OPTIONS、TRACE、CONNECT メソッド および customRequest が利用可能
- TCP キューの最大サイズが設定可能
- シンプルマクロがエクスプレッションマクロに変更
- アイテムのポジショナルマクロ (\$1, \$2...) の削除



セキュリティ向上

Zabbix ユーザのパスワードポリシー設定

- 最小パスワード長
- 必須項目
 - 大文字小文字
 - 数字
 - 記号
- 推測が容易な文字列の禁止
 - よくあるパスワード
 - ユーザ名を含む

The screenshot shows the Zabbix authentication settings page. The title is "認証" (Authentication). There are four tabs: "認証", "HTTP認証の設定", "LDAP認証の設定", and "SAML認証の設定". The "認証" tab is selected. Under "デフォルトの認証" (Default authentication), there are two options: "Zabbixデータベース内のユーザー情報" (selected) and "LDAP". Under "パスワードポリシー" (Password policy), there is a "最小パスワード長" (Minimum password length) field set to "8". Below that, there are three checkboxes for "パスワード必須項目" (Password required items): "英字の大文字と小文字" (English uppercase and lowercase), "数字" (Numbers), and "特殊記号" (Special characters). All three are currently unchecked. At the bottom, there is a checkbox for "推測されやすいパスワードの回避" (Avoidance of easily guessable passwords), which is checked. A blue "更新" (Update) button is at the bottom right.

監査ログ

- ディスカバリールールやアクション、スクリプトの実行を含むすべての処理が記録されるように改良
- 1 つの操作の結果として複数のログレコードが作成された場合、それらのレコードは同じレコード ID を持つように改良
- レコード ID でのフィルタ機能を追加
- 監査ログの有効/無効、保存期間の設定が一般設定の 1 セクションに分離

その他の新機能

データベースのバージョンチェック機能

- Zabbix server/proxy 起動時 DB のバージョンをチェック
- 対応バージョン外なら起動しない

zabbix_server.log

Unable to start Zabbix server due to unsupported PostgreSQL database server version (10.17)

Must be at least (13.0)

Use of supported database version is highly recommended.

Override by setting AllowUnsupportedDBVersions=1 in Zabbix server configuration file at your own risk.

データベースのバージョンチェック無効化 (非推奨)

- 設定ファイルで「AllowUnsupportedDBVersions=1」を設定

zabbix_server.conf

```
### Option: AllowUnsupportedDBVersions
# Allow server to work with unsupported database versions.
# 0 - do not allow
# 1 - allow
#
# Mandatory: no
# Default:
# AllowUnsupportedDBVersions=0

AllowUnsupportedDBVersions=1
```



どのような障害が発生するか分からないので、サポートバージョン内のDBを使用することを**強く推奨**

アイテムのデータ型の自動選択

- アイテムキーの選択時に
対応するデータ型を自動選択
 - agent.ping ⇒ 数値 (整数)
 - proc.cpu.util ⇒ 数値 (浮動小数)
 - logrt ⇒ ログ



キーが複数のデータ型を返す
可能性がある場合は
自動選択はされない

誤ったデータ型を選択すると
警告が表示

* キー agent.ping

データ型 文字列 

ポート番号 127.0.0.1:10050 

This type of information may not match the key. 

Zabbix agent 2 のプラグイン設定ファイルの分離

- プラグインの設定ファイルが
/etc/zabbix/zabbix_agentd2.d/plugins.d 以下に
プラグイン毎に分離して配置

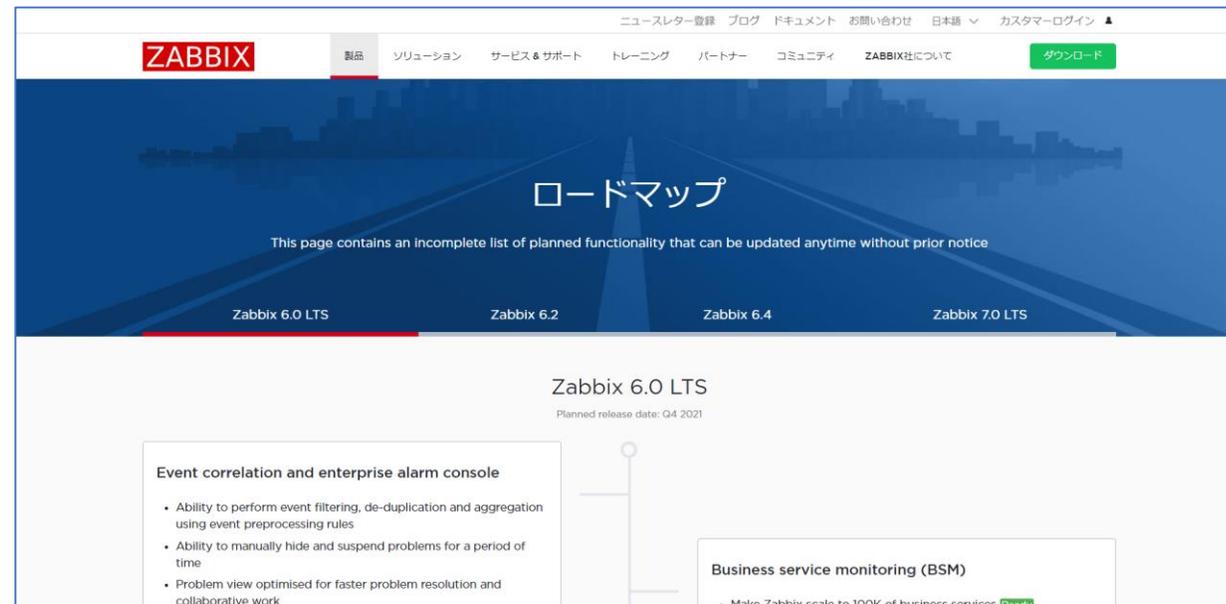
```
# ls /etc/zabbix/zabbix_agent2.d/plugins.d/  
ceph.conf  docker.conf  memcached.conf  modbus.conf  mongodb.conf  
mqtt.conf  mysql.conf  oracle.conf  postgres.conf  redis.conf  smart.conf
```

フロントエンドの変更点

- 「監視」⇒「ホスト」画面からホストの作成が可能
- ホスト編集画面がポップアップに変更
- 「監視」⇒「概要」画面が削除
 - データの概要、トリガーの概要ウィジェットで代替
- 言語からイギリス英語 (en_GB) が削除、デフォルトがアメリカ英語 (en_US) に変更
- メニューの Share のリンクがインテグレーション (Zabbix 公式サイト of インテグレーションページへのリンク) に変更
 - リンク先の言語もフロントエンドの設定言語のページがあればそれに合わせて開く

今後 6.0 に実装予定の機能

- ベースライン監視
- 異常値監視
- Kubernetes クラスタ監視
- 新しいグラフやウィジェット
 - 積み上げグラフ
 - データテーブルウィジェット
 - ジオメトリックマップウィジェット
- Escalation cancelled メッセージの抑制
- etc...



参考

- What's new in Zabbix 6.0.0
 - <https://www.zabbix.com/documentation/6.0/manual/introduction/whatsnew600>
- Zabbix roadmap
 - <https://www.zabbix.com/jp/roadmap>

弊社 Tech blog でも今後 Zabbix 6.0 の情報を掲載予定です。

SRA OSS Tech Blog

- 弊社 Tech Blog では Zabbix を始め、様々な OSS の技術情報を掲載
- Zabbix 6.0 に関する技術情報も随時掲載予定
- SRA OSS Tech Blog
 - <https://www.sraoss.co.jp/tech-blog/>



オープンソースとともに



SRA OSS, INC.

Appendix I

HA クラスタの状態確認と監視

「システム情報」ウィジェット

- HA クラスタの有効/無効
- フェイルオーバーの遅延時間

1秒あたりの監視項目数(Zabbixサーバーの要求パフォーマンス)	1.69	
HAクラスター	有効	フェイルオーバーの遅延：1分

「レポート」⇒「システム情報」画面

- ノード名
- アドレス
- 最終アクセスからの時間
- ステータス（アクティブ/スタンバイ/停止中/利用不可）

1秒あたりの監視項目数(Zabbixリハーの要求パフォーマンス)		1.69	
HAクラスター		有効	フェールオーバーの遅延：1分
名前	アドレス	最終アクセスからの時間	ステータス
Zabbix server #1	133.137.175.61:10051	2s	アクティブ
Zabbix server #3	133.137.175.151:10051	1m 44s	停止中
Zabbix server #2	133.137.175.28:10051	3s	スタンバイ

ha_status ランタイムコントロールオプション

- zabbix_server.log に各ノードの状態を出力

```
# zabbix_server -R ha_status
```

zabbix_server.log

#	ID	Name	Address	Status	Last Access
1.	ckvupihk70001z8mkpw5cg0u3	Zabbix server #1	133.137.175.61:10051	active	0s
2.	ckvupsmeq0001egpvksl6c5ih	Zabbix server #2	133.137.175.28:10051	standby	1s
3.	ckvupwi8n0001rwnxvdlwfv0s	Zabbix server #3	133.137.175.151:10051	stopped	8m 12s

zabbix[cluster,discovery,nodes]

- 各ノードの情報を JSON 形式で取得
 - id
 - name
 - status
 - 0: standby
 - 1: stopped
 - 2: unavailable
 - 3: active
 - lastaccess (最終アクセス時刻 Unixtime)
 - address
 - db_timestamp (現在時刻 Unixtime)
 - lastaccess_age (最終アクセスからの経過秒)



```
[  
  {  
    "id":"ckvupihk70001z8mkpw5cg0u3",  
    "name":"Zabbix server #1",  
    "status":3,  
    "lastaccess":1636623048,  
    "address":"133.137.175.61:10051",  
    "db_timestamp":1636623050,  
    "lastaccess_age":2  
  },  
  ...  
]
```

HA の状態をアイテムで取得してみよう (1)

- 値のマッピング
 - status の数値と対応する状態をマッピング

値のマッピング

* 名前 Zabbix HA status

* マッピング

タイプ	値	マッピング文字列	アクション
等しい	0	Standby	削除
等しい	1	Stopped	削除
等しい	2	Unavailable	削除
等しい	3	Active	削除

[追加](#)

[更新](#) [キャンセル](#)

HA の状態をアイテムで取得してみよう (2)

- アイテム
 - zabbix[cluster,discovery,nodes] で各ノードの状態一覧を取得

アイテム

すべてのホスト / Zabbix server 有効 ZBX アイテム 122 トリガー 63 グラフ 24 ディスカバリールール 4 Webシナリオ

アイテム タグ 保存前処理

* 名前

タイプ

* キー

データ型

* 監視間隔

HA の状態をアイテムで取得してみよう (3)

- LLD ルール
 - Zabbix HA Status の依存アイテムとして作成
 - LLD マクロで id と name を抽出

ディスカバリルール

すべてのホスト / Zabbix server 有効 ZBX ディスカバリリスト / Zabbix server HA status discovery
アイテムのプロトタイプ 1 トリガーのプロトタイプ グラフのプロトタイプ ホストのプロトタイプ

ディスカバリルール 保存前処理 LLDマクロ 2 フィルター オーバーライド

* 名前 Zabbix HA status discovery

タイプ 依存アイテム

* キー zabbix.ha.status.discovery

* マスターアイテム Zabbix server: Zabbix HA status 選択

* 存在しなくなったリソースの保持期間 30d

保存前処理 LLDマクロ 2 フィルター オーバーライド

LLDマクロ

LLDマクロ	JSONPath	
{#ID}	\$.id	削除
{#NAME}	\$.name	削除

[追加](#)

HA の状態をアイテムで取得してみよう (4)

• アイテムのプロトタイプ

- Zabbix HA Status の依存アイテムとして作成
- アイテム名に {#NAME} キーに {#ID} を使用

- 保存前処理に JSONPath で `$[?(@.id == "{#ID}")].status.first()` を設定

アイテムのプロトタイプ

すべてのホスト / Zabbix server 有効 ZBX ディスカバリリスト / Zabbix server HA status discovery

アイテムのプロトタイプ 1 トリガーのプロトタイプ グラフのプロトタイプ ホストのプロトタイプ

アイテムのプロトタイプ タグ 保存前処理 1

* 名前 {#NAME} HA status

タイプ 依存アイテム

* キー zabbix.ha.status[{#ID}] 選択

データ型 数値 (整数)

* マスターアイテム Zabbix server: Zabbix HA status 選択

タイプ タグ 保存前処理 1

保存前処理の設定

名前	パラメータ
1: JSONPath	<code>\$[?(@.id == "{#ID}")].status.first()</code>

追加

HA の状態をアイテムで取得してみよう (5)

- 取得結果
 - トリガー設定はよしなに

最新データ				
<input type="checkbox"/> ホスト	名前 ▲		最新のチェック時刻	最新の値
<input type="checkbox"/> Zabbix server	Zabbix server #1 HA status		2021/11/12 14:51:51	Active (3)
<input type="checkbox"/> Zabbix server	Zabbix server #2 HA status		2021/11/12 14:51:51	Standby (0)
<input type="checkbox"/> Zabbix server	Zabbix server #3 HA status		2021/11/12 14:51:51	Stopped (1)

Appendix II

新規追加されたアイテムキー

agent.hostmetadata

- エージェントの**ホストメタデータ**を返す
 - タイプ : Zabbix エージェント
Zabbix エージェント (アクティブ)
 - データ型 : 文字列
 - パラメータ : なし

?

ホストメタデータ

1. zabbix_agentd.conf の HostMetadata に設定した文字列
2. HostMetadataItem に設定したアイテムキーの取得結果
3. 両方設定していなければ空文字列

kernel.openfiles

- 現在開いている**ファイルディスクリプタ数**を返す
 - タイプ : Zabbix エージェント
Zabbix エージェント (アクティブ)
 - データ型 : 数値 (整数)
 - パラメータ : なし

net.tcp.socket.count[]

- パラメータに一致する **TCP ソケット数**を返す
 - タイプ : Zabbix エージェント
Zabbix エージェント (アクティブ)
 - データ型 : 数値 (整数)
 - パラメータ : <laddr> ローカルの IPv4/v6 アドレス (CIDR 表記可)
<lport> ローカルのポート番号もしくはサービス名
<raddr> リモートの IPv4/v6 アドレス (CIDR 表記可)
<rport> リモートのポート番号もしくはサービス名
<state> 接続状態 (established, syn_sent, syn_recv, fin_wait1, fin_wait2, time_wait, close, close_wait, last_ack, listen, closing)

net.udp.socket.count[]

- パラメータに一致する **UDP ソケット数**を返す
 - タイプ : Zabbix エージェント
Zabbix エージェント (アクティブ)
 - データ型 : 数値 (整数)
 - パラメータ : <laddr> ローカルの IPv4/v6 アドレス (CIDR 表記可)
<lport> ローカルのポート番号もしくはサービス名
<raddr> リモートの IPv4/v6 アドレス (CIDR 表記可)
<rport> リモートのポート番号もしくはサービス名
<state> 接続状態 (established, unconn)

vfs.dir.get[]

- ディレクトリの情報を JSON 形式で返す

- タイプ : Zabbix エージェント
Zabbix エージェント (アクティブ)
- データ型 : テキスト
- パラメータ : dir ディレクトリのパス
 <regex_incl> 含めるエンティティの正規表現
 <regex_excl> 除外するエンティティの正規表現
 <types_incl> リストする種類 (file, dir, sym,
 sock, bdev, cdev, fifo, dev,
 all)
 <types_excl> リストしない種類
 <max_depth> 探索するサブディレクトリの深さ
 <min_size> リストする最小 byte 数
 <max_size> リストする最大 byte 数
 <min_age> リストする最小タイムスタンプ
 <max_age> リストする最大タイムスタンプ
 <regex_excl_dir> 除外するディレクトリの正規表現



```
[
  {
    "basename": "zabbix_agentd.d",
    "pathname": "/etc/zabbix/zabbix_agentd.d",
    "dirname": "/etc/zabbix",
    "type": "dir",
    "user": "root",
    "group": "root",
    "permissions": "0755",
    "uid": 0,
    "gid": 0,
    "size": 4096,
    "time": {
      "access": "2021-11-10T04:16:58+0900",
      "modify": "2021-11-10T00:50:35+0900",
      "change": "2021-11-10T04:16:56+0900"
    },
    "timestamp": {
      "access": 1636485418,
      "modify": 1636473035,
      "change": 1636485416
    }
  },
],
```

vfs.file.get[]

- ファイルの情報を JSON 形式で返す
 - タイプ : Zabbix エージェント
Zabbix エージェント (アクティブ)
 - データ型 : テキスト
 - パラメータ : file ファイルのフルパス



```
{
  "type": "file",
  "user": "root",
  "group": "zabbix",
  "permissions": "0600",
  "uid": 0,
  "gid": 975,
  "size": 25903,
  "time": {
    "access": "2021-11-09T11:22:32+0900",
    "modify": "2021-11-09T11:22:28+0900",
    "change": "2021-11-09T11:22:28+0900"
  },
  "timestamp": {
    "access": 1636424552,
    "modify": 1636424548,
    "change": 1636424548
  }
}
```

vfs.file.owner[]

- ファイルの所有者情報を返す
 - タイプ : Zabbix エージェント
Zabbix エージェント (アクティブ)
 - データ型 : 文字列
 - パラメータ : file ファイルのフルパス
 <ownertype> 所有者の種類 (user, group)
 <resulttype> 所有者の表示形式 (name, id)

vfs.file.permissions[]

- ファイルのパーミッション情報（8 進数 4 桁）を返す
 - タイプ : Zabbix エージェント
Zabbix エージェント（アクティブ）
 - データ型 : 文字列
 - パラメータ : file ファイルのフルパス



Windows の Zabbix agent ではサポートされない