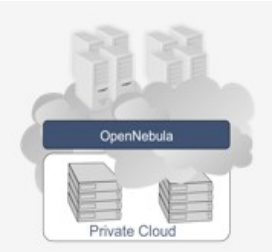
OpenNebula
プライベート・クラウド構築

2010 年 11 月 4 日

B to J Pty Ltd

OpenNebulaを用いたプライベート・クラウドの構築・・・・・・・・・・

プライベート・クラウドとは

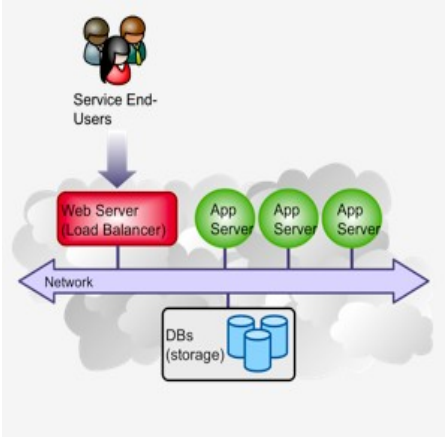


プライベート・クラウドを構築する目的は、ドメイン内に仮想サービスを運用するプライベートなインフラを構築し、社内ユーザーが柔軟かつ迅速に仮想サービスを利用する事を可能にすることにあります。

ユーザーやアドミニストレーターは、OpenNebula の仮想インフラ・インターフェースを用いることにより、仮想、ネットワーキング、イメージや物理リソースのコンフィギュレーション、マネジメント、モニタリング、アカウンティング機能を用いる事ができるようになります。

ユーザー・ビュー

OpenNebula プライベート・クラウドは、エンドユーザーの必要に応じて迅速にサービスを提供するための、柔軟なプラットフォームを提供します。サービスは仮想マシン上に構築され、以下の仮想インフラ・インターフェースよりクラウド・サービスとして提供・管理されます。



- コマンドライン・インターフェース
- XML-RPC API
- Libvirt 仮想 API、Libvirt マネジメント・ツール

それでは「OpenNebula CLI for Private Clould COmputing」機能を理解するために、簡単なセッションのサンプルをみてましょう。まずまじめに、物理クラスターのホストを確認してみます。

```
$ onehost list
  ID NAME                CLUSTER               RVM   TCPU   FCPU   ACPU    TMEM    FMEM STAT
   0 host01              default                 0    800    800    800 8194468 7867604   on
   1 host02              default                 0    800    797    800 8387584 1438720   on
```

次に、「oneimage」を用いて OpenNebula にイメージを登録します。イメージ・テンプレートを作成して、イメージ・ファイルを登録します。イメージ・ファイルは「/home/cloud/images」ディレクトリにすでに保存されています。

```
NAME           = "Ubuntu Desktop"
PATH           = /home/cloud/images/ubuntu-desktop/disk.0
PUBLIC         = YES
DESCRIPTION    = "Ubuntu 10.04 desktop for students."
```

```
$ oneimage register ubuntu.oneimg
$ oneimage list
  ID     USER                NAME TYPE             REGTIME PUB STAT  #VMS
   1 oneadmin      Ubuntu Desktop   OS   Jul 11, 2010 15:17 Yes  rdy     0
```

このイメージは仮想マシンにて利用可能となっています。では、「onevm」コマンドを用いて 仮想マシン・テンプレートを定義します。

```
CPU     = 1
MEMORY = 2056

DISK = [ image  = "Ubuntu Desktop" ]

DISK = [ type    = swap,
         size    = 1024 ]

NIC     = [ NETWORK = "Public network" ]
```

必要に応じて CPU や MEMORY のフィールドを調整した後、仮想マシンがいずれかのホストにフィットする事を確認します。そして、ホーム・フォルダから仮想マシンを構築します。

```
$ onevm create myfirstVM.template
```

すると、ID が提供されます。この ID はモニタリングやコントローリングの際に仮想マシンを特定する者となります。その際にも「onevm」コマンドが用いられます。

```
$ onevm list
  ID     USER     NAME STAT CPU     MEM        HOSTNAME        TIME
   0 oneadmin    one-0 runn   0   65536          host01  00 0:00:02
```

「STAT」フィールドは仮想マシンのステータスを示します。「runn」は仮想マシンが立ち上がっている事を示します。イメージをどのように設定したかによりますが、もし IP アドレスを知っている場合には、仮想マシンにログインする事もできます。

マイグレーションを実行する際にも、「onevm」コマンドを用います。それでは仮想マシン（VID=0）を host02（HID=1）にマイグレーションしましょう。

```
$ onevm livemigrate 0 1
```

仮想マシンが host01 から host02 に移動しました。「onevm list」はこのようになります。

```
$ onevm list
  ID     USER     NAME STAT CPU     MEM        HOSTNAME        TIME
   0 oneadmin    one-0 runn   0   65536          host02  00 0:00:06
```

システム動作の仕組み

OpenNebula はこのような機能を提供します。

仮想ネットワークの管理
それぞれの仮想ネットワークの関連づけがなされ、定義が加えられます。

仮想マシンの構築
仮想マシンの説明がデータベースに保存されます。

仮想マシンの展開
スケジューラーは、割り振りのポリシーに従って、どこで仮想マシンを実行するか判断します。

仮想イメージの管理
実行する前に、イメージを登録する事ができます。提出された後、仮想イメージはホストに転送され、スワップのディスク・イメージが作成されます。実行後は、仮想イメージはレポジトリにコピーされます。

起動中の仮想マシンの管理
起動中の仮想マシンのリソース使用状況、ステータス情報が定期的に抽出されます。仮想マシンのシャットダウン、サスペンド、ストップ、マイグレーションが可能です。

OpenNebula プライベート・クラウドのおもなコンポーネントはこちらです。

ハイパーバイザー
OpenNebula は、クラスタのリソースににインストールされた仮想マネージャーを利用して、それぞれのホスト内の仮想マシンを管理します。

仮想インフラ・マネージャ
すべての仮想マシンとリソースを一括管理します。仮想ネットワークの管理、仮想マシンのライフサイクル管理、仮想マシンのイメージ管理、耐障害性の管理などが含まれます。

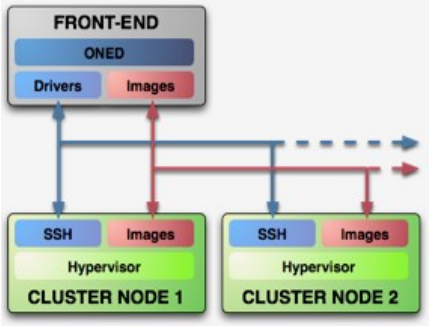
スケジューラー
リソースの利用状況に基づいた仮想マシンの展開ポリシー、サーバー統合、設置時の条件設定、アフィニティ、キャパシティの確保、SLA に応じた利用等の機能を提供します。

OpenNebula 2.0のインストレーションの準備をする・・・・・・・・・・

概観

OpenNebula を利用する前に、すでに社内において、フロントエンドと複数のクラスタ・ノードから構成される物理インフラが構築されていることが前提となります。そのクラスタ・ノード上で仮想マシンが実行されることになります。

またフロントエンドとすべてのクラスタ・ノードが少なくとも１つのネットワークにて接続されていることも必要です。



OpenNebula システムの基本的なコンポーネントはこちらです。

フロントエンド
OpenNebula とクラスタ・サービスを実行する

ノード
ハイパーバイザー利用可能なホストで、仮想マシンに必要とされるリソースを提供します

イメージ・レポジトリ
仮想イメージを保存することができるストレージ

OpenNebula デーモン
システムのコアとなる部分で、仮想マシンのライフサイクル管理する。ネットワーク、ストレージ、ハイパーバイザーから構成されるクラスタ・サブシステムもまとめて管理する。

ドライバ
コアが特定のサブシステム（ハイパーバイザー、ストレージのファイルシステム）を利用するために用いるインターフェース

oneadmin
プライベート・クラウド全体のオペレーションを管理する役割をもつアドミニストレーター。仮想マシン、仮想ネットワーク、ノード、ユーザー等の管理をする。

ユーザー
OpenNebula の機能を用いて、仮想マシンや仮想ネットワークを構築、管理する

システム上必要とされるもの

クラスタ・フロントエンド

この部分では、OpenNebula をインストールし、実行するために必要とされるソフトウェアの詳細を述べます。フロントエンドはイメージ・レポジトリにアクセスします。このイメージ・レポジトリは仮想マシン用の仮想イメージを保存するに十分の容量を備えている必要があります。通常は、仮想マシンを起動する際に、マスターとなるイメージがクローン（コピー）されます。ですから、仮想インフラ内で展開する仮想マシン数にあわせてストレージ容量を確保する必要があります。OpenNebula 自体のインストレーションは 10MB ほどです。

インストレーション・モード

OpenNebula は２つのモードにてインストールすることができます。

システム全体モード（System-wide）
バイナリ、ログファイル、コンフィギュレーション・ファイルが UNIX のルートファイルに保存されます。
このモードでインストールするには、すでにルートにアクセスする権限を持っていることが必要です。

システム一部モード（Self-contained）
システムの一部にて OpenNebula を用います。こちらのモードでの利用をお勧めします。

どちらの場合においても、OpenNebula を用いるためのルート・アカウントは必要ありません。

ソフトウェア・パッケージ

OpenNebula のサーバーとなるマシンにはこちらのソフトウェアをインストールする必要があります。

- **ruby：1.8.6 から 1.9.0**
- **sqlite3： 3.5.2 以上**
- **xmlrpc-c：1.06 以上**
- **openssl：0.9 以上**
- **ssh**

OpenNebula をソースより構築する場合には以下が必要となります。

- **Sqlite3、xmlrpc-c、openssl パッケージの開発バージョン**
- **scons：0.97 以上**
- **g++：4 以上**
- **flex：2.5 以上（オプショナル：パーサーを再構築する場合）**
- **bison：2.3 以上（オプショナル：パーサーを再構築する場合）**
- **libxml2-dev**

オプショナル・パッケージ

こちらのパッケージは、OpenNebula を用いる上で必要とはされず、コアのパフォーマンスを高めるわけでもありませんが、いくらかのツールのパフォーマンスを高めることができます。CLI のパフォーマンスが向上する場合もあります。

まずはじめに「rubygems」と「ruby development libraries」をインストールします。

- ruby-dev
- rubygems
- rake
- make

その後、以下のパッケージをインストールします。

- ruby xmlparser

いくらかのディストリビューションにはバイナリ・パッケージが含まれます。もし用いようとしているディストリビューションに含まれていない場合には、デベロップメント・ファイルと共に「expat」ライブラリーをインストールし、「gem」を用いて「xmlparser」をインストールします。

```
# gem install xmlparser --no-ri --no-rdoc
```

（注）「gem」インストールの追加パラメターに注意してください。いくらかの「xmlparser」のバージョンではドキュメンテーションを作成する際に問題が生じます。ドキュメンテーションのインストールなしでも利用できます。

- ruby nokogiri

インストールするためには「libxml2」と「libxslt ライブラリー」、またそれぞれのデベロップメント・バージョンが必要とされます。その後、「nokogiri」ライブラリーをインストールできるよになります。

```
# gem install nokogiri
```

クラスタ・ノード

ノードにて仮想マシンを実行します。特別なストレージ設定等、必要ありません。

ソフトウェア・パッケージ

仮想マシンを実行するためにクラスタ・ノードにて必要とされるソフトウェアはこちらです。

- ssh サーバー：すでに実行しているもの
- ハイパーバイザー：すでに設定、実行されているもの
- ruby 1.8.5 以上

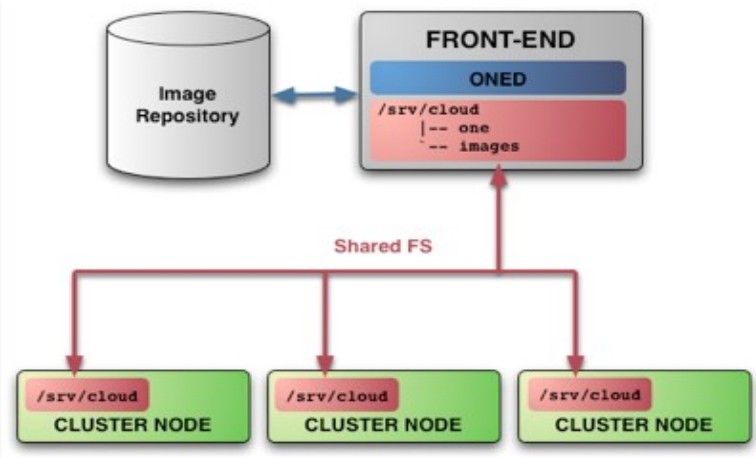
クラスタを準備する

ストレージ

（注）このガイドでは、共有ファイルシステムなどを用いて、クラスタ・ノードからイメージ・レポジトリと OpenNebula の「var」ディレクトリにアクセスできることを前提としています。そうすることにより、ハイパーバイザーの機能（例：ライブ・マイグレーション）や OpenNebula のストレージ・モジュール機能（例：常にクローンすることを避ける）を十分に活用することができます。

（注）OpenNebula は共有ファイルシステムなしでも実行できますが、その際には常にクローンが作成されてしまい、またコールド・マイグレーションのみの利用が可能となります。しかしながら、共有ファイルシステムなしの設定にはそれほど時間がかかる訳ではありません。もし共有ファイルシステムなしの設定とする場合にはこのセクションをとばしてください。

クラスタのフロントエンドは、イメージ・レポジトリと OpenNebula インストレーション・ディレクトリを、クラスタ・ノードにエクスポートします。イメージ・レポジトリにて必要とされる容量はイメージの総数とサイズによります。また、仮想マシンを実行する場合にはクローニング（コピーイング）が用いられるので、すべての実行中の仮想マシンのイメージを保存するための十分な容量があるかどうか確かめてください。



フロントエンドのルート・ファイルシステムにこのような階層を作成してください：

- 「/srv/cloud/one」：OpenNebula インストレーションと実行中の仮想マシンのクローンを入れる
- 「/srv/cloud/images」：マスター・イメージとレポジトリを入れる

```
$ tree /srv
/srv/
|
`-- cloud
    |-- one
    `-- images
```

（例）：1 つの 64 コア・クラスタでは通常、約 80 もの仮想マシンを実行できます。それぞれの仮想マシンは平均 10GB のディスク容量が必要です。よって、「/srv/cloud/one」に 800GB 必要とされます。また 10 から 15 のマスター・イメージを保存するために 200GB「/srv/cloud/images」必要とされます。ですからこの場合には、「/srv/cloud/」に 1TB 割り当てることで十分でしょう。

すべてのクラスタ・ノードに「/srv/cloud/」をエクスポートします。例えば、アドレス「192.168.0.0/24」のローカル・ネットワーク上にすべての物理ノードがある場合、「/etc/exports」ファイルにこのようなラインを追加します：

```
$ cat /etc/exports
/srv/cloud 192.168.0.0/255.255.255.0(rw)
```

それぞれのクラスタ・ノードで「/srv/cloud」を作成し、フロントエンドよりこのディレクトリをマウントしてください。

ユーザー・アカウント

仮想インフラは「oneadmin」アカウントにより管理されます。このアカウントにより、OpenNebula を実行したり、その他さまざまな管理作業やメンテナンス作業がなされます。

（注）OpenNebula では、複数のユーザーが仮想マシンを構築・管理する事が可能です。
後ほど、OpenNebula 設定の部分で、その方法を説明します。

それではこの手順に従ってください。

- 「cloud」グループを作成します。このグループ内に、OpenNebula のアドミニストレーター・ユー

ザーが含まれることになります。

```
# groupadd cloud
```

- OpenNebula のアドミニストレーター・アカウント「oneadmin」を作成します。OpenNebula のディレクトリーを、このアカウントのホーム・ディレクトリーとして設定します。

```
# useradd -d /srv/cloud/one -g cloud -m oneadmin
```

- OpenNebula のアドミニストレーター・アカウントのユーザー ID とグループ ID を取得します。この ID は、後ほどクラスター・ノードに同じ ID を用いてユーザーを作成する際に使います。

```
$ id oneadmin
uid=1001(oneadmin) gid=1001(cloud) groups=1001(cloud)
```

このケースの場合には、ユーザー ID もグループ ID も共に「1001」となっています。

- 仮想マシンを実行するすべてのノードにグループ・アカウントを作成します。ID はフロントエンドと同じにしてください。つまりこの場合には「1001」となります。

```
# groupadd --gid 1001 cloud
# useradd --uid 1001 -g cloud -d /srv/cloud/one oneadmin
```

（注）その他の方法で各ノードに共通のクラウド・グループと「oneadmin」アカウントを作成する事も可能です。例えば、NIS を用いる事もできます。

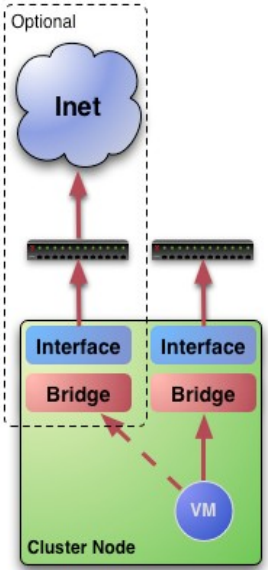
ネットワーク

前述のコンフィギュレーションのステップにて準備したもの以外で、ネットワーキングを設定するために特別に必要とされるものはありません。しかしながら、より効果的に仮想マシンを利用するためには、さらにもう１つか２つの物理ネットワークと接続することをお勧めします。

例えばイーサネット・ブリッジングを用いる事により、より効果的に仮想マシンを展開することができます。この方法につきましては、一般に公開されている資料等にて確認してください。

例えば、通常のクラスタ・ノードは２つの物理ネットワークと接続されます。１つはパブリック IP アドレス用（eth0 NIC に結びつけられている）で、もう１つはプライベート仮想 LAN 用（NIC eth1）です。この場合、２つのブリッジを設定する事ができます。

```
$ brctl show
bridge name bridge id         STP enabled interfaces
vbr0        8000.001e682f02ac no          eth0
vbr1        8000.001e682f02ad no          eth1
```



より詳細な情報につきましては、別途「Virtual Network Usage Guide」ならびに「Network Customization Guide」をご覧ください。

セキュア・シェル・アクセス

「oneadmin」ユーザー用に ssh キーを作成して、マシンを設定する必要があります。キーを用いる事により、パスワードなしで ssh を用いてマシンにコネクトすることができるようになります。

- 「oneadmin」用にに ssh キーを作成する

```
$ ssh-keygen
```

パスワードを求められた場合には、「enter」を押してください。そうすることによりプライベート・キーが暗号化されるのを防ぐ事ができます。

- パブリック・キーを「/.ssh/authorized_keys」と結びつけます。そうすることにより、「oneadmin」がパスワードなしでもユーザー・ログできます。

```
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- 多くのディストリビューション（RHEL/CentOS 等）では、パブリック・キーの承認が正常に動作するためのパミッションが必要となります。
  - **.ssh/**: 700
  - **.ssh/id_dsa.pub**: 600
  - **.ssh/id_dsa**: 600
  - **.ssh/authorized_keys**: 600
- ssh クライアントにホストを「known_hosts」に加える前に ask しないように指示します。そうすることにより「/.ssh/config」に加えられます。

```
$ cat ~/.ssh/config
Host *
    StrictHostKeyChecking no
```

（注）「sshd」デーモンがクラスタ・ノードで実行しているか確認してください。「oneadmin」はパスワードなしでクラスタ・ノードにログインできる必要があります。また、クラスタ・ノードの「sshd_config」ファイルからすべての「Banner」オプションを削除してください。

ハイパーバイザー

仮想テクノロジーがすでにクラスタ・ノードにインストールされ、設定が完了しているか確かめてください。そうする事により、「oneadmin」が仮想マシンをスタート、コントロール、モニターすることができるようになります。つまり、ルート権限を用いてコマンドを実行したり、グループ内に「oneadmin」を作成したりすることが可能となります。以下の仮想ガイドを参考にしてください。

- Xen Configuration Guide
- KVM Configuration Guide
- VMware Configuration Guide

プラニング・チェックリスト

これまでに述べたステップをふまえる事により、OpenNebula をインストールし、設定するためのクラスタの準備が整っているはずです。以下のリストを参考にして、これまでのステップを確認してください。

必要とされるソフトウェア

- インストール形態：システム全体モードか、システム一部モードか（self-contained）
- インストレーション・ディレクトリーは「/srv/cloud/one」となっているか
- OpenNebula ソフトウェアの「/srv/cloud/one/SRC」へのダウンロードはなされているか
- sqlite, g++, scons, ruby その他の必要なソフトウェアはインストールされているか

ユーザー・アカウント

- 各ノードとフロントエンドにて「oneadmin」アカウントと「cloud」グループが設定されているか

ストレージのチェック

- フロントエンドに「/srv/cloud」階層が作成されているか
- 「/srv/cloud」がエクスポートされており、クラスタ・ノードよりアクセス可能か
- 異なる場合にはノードに「/srv/cloud」ポイントがマウントされているか

クラスタ・ノードのチェック

- クラスタ・ノードのホストネームはあるか
- ノードに「sshd」と「ruby」はインストールされているか
- 「oneadmin」は ssh をパスワードなしで利用できるか

プラットフォーム情報　2.0

Ubuntu/Kubuntu

OpenNebula を構築するために、以下のパッケージをインストールしてください。

- **libsqlite3-dev**
- **libxmlrpc-c3-dev**
- **scons**
- **g+=**
- **ruby**
- **libssl-dev**

オプショナル・ソフトウェア

- **ruby-dev**
- **make**
- **rake**
- **rubygems**
- **libxml-parser-ruby1.8**
- **libxslt1-dev**
- **libxsl2-dev**
- **nokogiri (ruby gem)**

Debian Lenny

OpenNebula を構築するために、以下のパッケージをインストールしてください。

- **gcc c++ compiler:** g++
- **ruby:** ruby
- **sqlite3:** libsqlite3-0, sqlite3
- **sqlite3-dev:** libsqlite3-dev
- **sqlite3-ruby:** libsqlite3-ruby
- **libxmlrpc-c:** libxmlrpc-c3-dev, libxmlrpc-c3
- **libssl:** libssl-dev
- **scons:** scons

ちなみに Lenny の Xen パッケージは壊れているように思われます。回避策はこちらです。

```
# ln -s /usr/lib/xen-3.2-1/bin/tapdisk /usr/sbin
# echo xenblktap >> /etc/modules
# reboot
```

Fedora 8

現在確認できている問題はありません。

Gentoo

「libxmlrpc」インストールの際には、スレッド・サポートとコンパイルする必要があります。

```
# USE="threads" emerge xmlrpc-c
```

Arch

追加レポジトリで利用可能な「xmlrpc-c」パッケージは、abyss server をサポートするためのコンパイルがなされていません。Server をサポートするために Arch Build System (ABS)を用いてください。PKGBUILD ファイルのコンフィギュア・コマンドから「-disable-abyss」を削除し、こちらのパッケージをインストールしてください。

```
    cd $srcdir/$pkgname-$pkgver
    ./configure --prefix=/usr \
                --enable-libxml2-backend \
                --disable-cgi-server \
                --disable-libwww-client \
                --disable-wininet-client
```

CentOS 5 / RHEL 5

バイナリ・パッケージからのインストール

バイナリ・パッケージはこちらからダウンロードできます。
http://downloads.dsa-research.org/opennebula

バイナリ・パッケージをインストールする前に、「CentOS Karan Repo」を「yum repos」に加えることが必要です（ガイドはこちら：http://centos.karan.org/）。追加した後、「etc/yum.repos.d/kbsingh-CentOS-Extras.repo」の「testing repo」を使用可能としてください。

こちらの OpenNebula のパッケージをインストールすることをお勧めします。

```
yum localinstall --nogpgcheck opennebula-2.0-1.<arch>.rpm
```

ソースからのインストール

Ruby

スタンダード・パッケージを yum と共に直接インストールします。

```
$ yum install ruby ruby-devel ruby-docs ruby-ri ruby-irb ruby-rdoc
```

rubygems をインストールするには、EPEL レポジトリーをアクティベートする必要があります。

```
$ wget http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-4.noarch.rpm
$ yum localinstall epel-release-5-3.noarch.rpm
$ yum install rubygems
```

rubygems がインストールされた後、こちらの gems もインストールします。

```
gem install nokogiri rake xmlparser
```

Scone

Centos と共に提供されるバージョンでは、こちらの提供するビルド・スクリプトとの互換性がありません。Scone の最新バージョンをインストールするには、こちらにて RPM をダウンロードしてください。

http://www.scons.org/download.php

```
$ wget http://prdownloads.sourceforge.net/scons/scons-1.2.0-1.noarch.rpm
$ yum localinstall scons-1.2.0-1.noarch.rpm
```

xmlrpc-c

xmlrpc-c と xmlrpc-c パッケージはこちらの rpm レポジトリでダウンロードできます。

http://centos.karan.org/

```
$ wget http://centos.karan.org/el5/extras/testing/i386/RPMS/xmlrpc-c-1.06.18-1.el5.kb.i386.rpm
$ wget http://centos.karan.org/el5/extras/testing/i386/RPMS/xmlrpc-c-devel-1.06.18-1.el5.kb.i386.rpm
$ yum localinstall xmlrpc-c-1.06.18-1.el5.kb.i386.rpm xmlrpc-c-devel-1.06.18-1.el5.kb.i386.rpm
```

sqlite

このパッケージはソースよりインストールする必要があります。「tar.gz」はこちらよりダウンロードできます。
http://www.sqlite.org/download.html

sqlite 3.5.9 でテストがなされています。

```
$ wget http://www.sqlite.org/sqlite-amalgamation-3.6.17.tar.gz
$ tar xvzf sqlite-amalgamation-3.6.17.tar.gz
$ cd sqlite-3.6.17/
$ ./configure
$ make
$ make install
```

システム全体のロケーション（/user または/usr/local）でインストールしない場合には、「LD_LIBRARY_PATH」を追加する必要があります。そうすることにより、scons がファイルを見つける事ができるようになります。

```
$ scons sqlite=<path where you installed sqlite>
```

OpenSUSE 11.3

バイナリ・パッケージからのインストール

パッケージをインストールする前に「Packman repo」を 「zypper repos」に追加する必要があります。
http://packman.links2linux.org/

ソースからのインストール

ビルディング・ツール

openSUSE 11 には標準的なビルティング・ツールがディフォルトでは含まれていません。ですから、コンパイル前に、こちらをインストールする必要があります。

```
$ zypper install gcc gcc-c++ make patch
```

求められるライブラリー

OpenNebula と効果的に関連づけるために、以下のパッケージをインストールしてください。

```
$ zypper install libopenssl-devel libcurl-devel scons pkg-config sqlite3-devel libxslt-devel libxmlrpc_server_ab
fig sqlite3-devel libxslt-devel libxmlrpc_server_abyss++3 libxmlrpc_client++3 libexpat-devel libxmlrpc_server++3
```

Ruby

こちらの標準的なパッケージを zypper と共にインストールすることができます。

```
$ zypper install ruby ruby-doc-ri ruby-doc-html ruby-devel rubygems
```

rubygems は 1.3.1 以上でなければなりません。安全策として、最新版にアップデートしてください。

```
$ wget http://rubyforge.org/frs/download.php/45905/rubygems-1.3.1.tgz
$ tar zxvf rubygems-1.3.1.tgz
$ cd rubygems-1.3.1
$ ruby setup.rb
$ gem update --system
```

rubygems インストール後は、以下の gems をインストールします。

```
gem install nokogiri rake xmlparser
```

xmlrpc-c

xmlrpc-c を構築するには、最新版の svn リリースをダウンロードし、コンパイルする必要があります。追加情報のパッケージとともに README ファイルを読んでください。

```
svn co http://xmlrpc-c.svn.sourceforge.net/svnroot/xmlrpc-c/super_stable xmlrpc-c
cd xmlrpc-c
./configure
make
make install
```

MAC OSX 10.4 10.5

OpenNebula フロントエンドは MAC OSX にインストールすることができます。ここでは、10.5(Leopard)にて構築する際の情報となります。

必要とされるもの：

- **xcode**（MAC OS X DVD よりインストール可能）
- **macports**（http://www.macports.org/）

Getopt

このパッケージは BSD スタイルとともに提供される getopt が必要です。

```
$ sudo port install getopt
```

xmlrpc

```
$ sudo port install xmlrpc-c
```

scons

macport を用いて scons をインストールすることができます。

```
$ sudo port install scons
```

残念ながら、この方法では python やその他のパッケージもコンパイルしてしまいます。回避策としては、スタンドアロンのパッケージをこちらからダウンロードすることもできます。
http://www.scons.org/download.php

「scons-local」パッケージを探し、「Gzip tar」ファイルをダウンロードしてください。
こちらの例では、バージョン 1.2.0 を用いています。

```
$ mkdir -p ~/tmp/scons
$ cd ~/tmp/scons
$ tar xvf ~/Downloads/scons-local-1.2.0.tar
$ alias scons='python ~/tmp/scons/scons.py'
```

OpenNebula

```
$ scons -j2
```

（注）OpenNebula を構築するために必要とされるソフトウェアにつきましては「Platform Notes」を参考にしてください。

OpenNebula をインストールするために、以下の簡単なステップに従ってください。

- OpenNebula tarball をダウンロードし、解凍（untar）してください
- Created folder に変更し、OpenNebula を圧縮するために scons を実行してください

```
$ scons [OPTION=VALUE]
```

[OPTIONAL]の部分ではディフォルト外のバリューを設定できます。

オプション	バリュー
sqlite_db	path-to-sqlite-install
sqlite	sqliteのサポート構築しない場合には、**no**を選択
mysql	mysqlのサポートを構築する場合には、**yes**を選択
xmlrpc	path-to-xmlrpc-install
parsers	flex/bisonファイルを構築する場合には、**yes**を選択

- OpenNebula はシステム全体モード（system-wide）、またはシステム一部モード（self-contained）にてインストールできます。どちらの場合においても、OpenNebula をルートとする必要はありません。インストール・スクリプトを実行する際には、以下のようなオプションを特定する事ができます。

```
./install.sh <install_options>
```

<install_options>の部分では、以下を 1 つ、または複数用いる事ができます。

オプション	バリュー
-u	OpenNebulaを実行するユーザー、install.shを実行するユーザーにデフォルトで提供される
-g	OpenNebulaを実行するユーザーのグループ、、install.shを実行するユーザーにデフォルトで提供される
-k	現在のコンフィギュレーションを保つためのもので、アップグレード時に便利
-d	インストレーション・ディレクトリーをターゲットにする。定義されることにより、システム一部モード（self-contained）のインストレーションのパスが固定される。定義されていない場合には、システム全体モード（system wide）でのインストレーションが実行される。
-r	OpenNebulaを削除する。-dが定義されていない場合にのみ用いる。 または、rm -rf $ONE_LOCATIONを用いることもできる。
-h	インストーラーのヘルプをプリントする

ここではシステム一部モード（self-contained のインストレーションを用います。oneadmin ユーザーにて：

```
~$ wget <opennebula tar gz>
~$ tar xzf <opennebula tar gz>
~$ cd one-2.0
~/one-2.0$ scons -j2
[ lots of compiling information ]
scons: done building targets.
~/one-2.0$ ./install.sh -d /srv/cloud/one
```

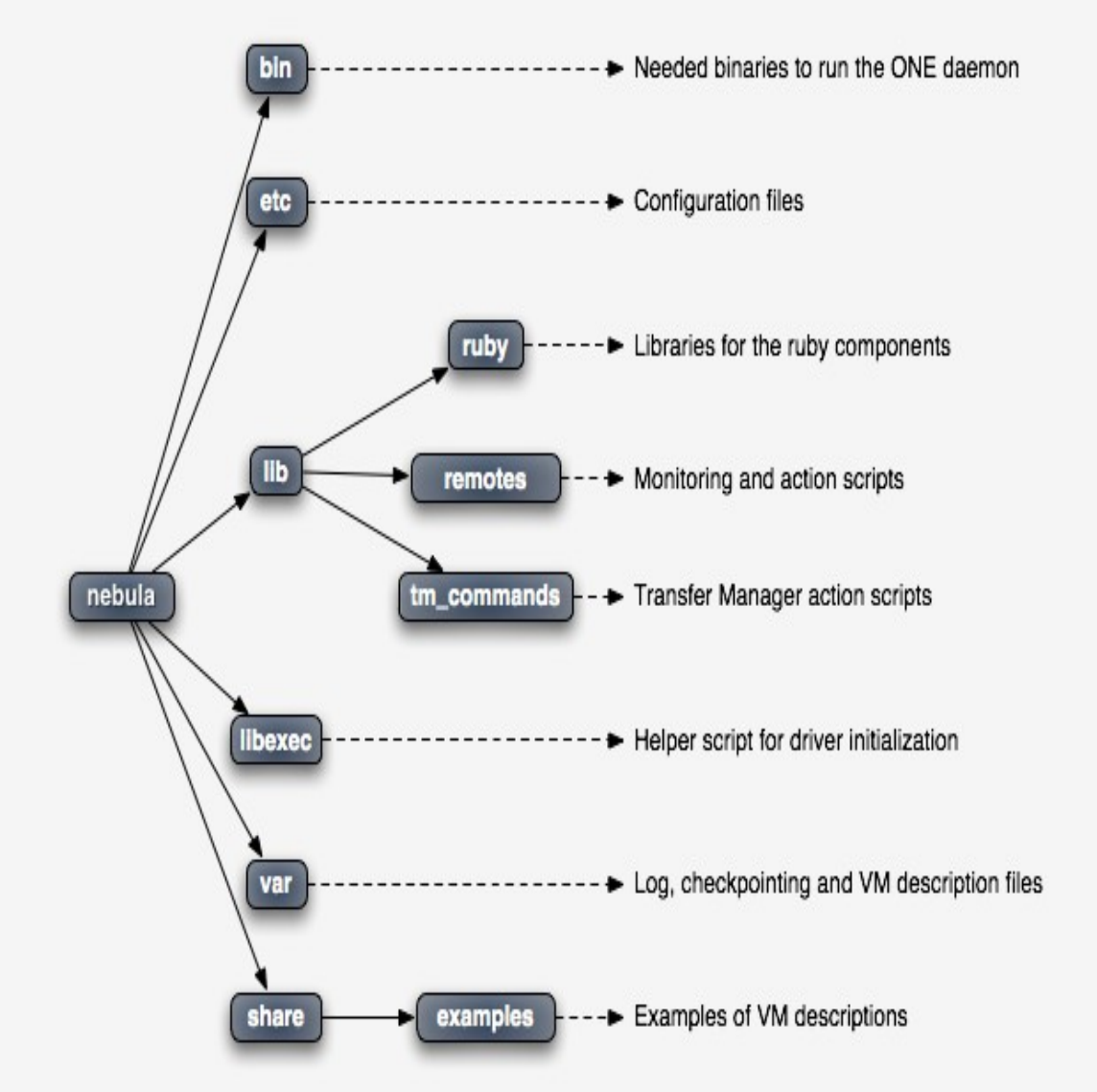
次は、特定のクラスタを調整するために OpenNebula をコンフィギュレーションする必要があります。

インストレーションを確認する

インストレーション・スクリプト内の、-d オプションの有無により定義されたインストレーションのモードによりますが、OpenNebula のファイル構成は以下の２つのケースのいずれかとなります。

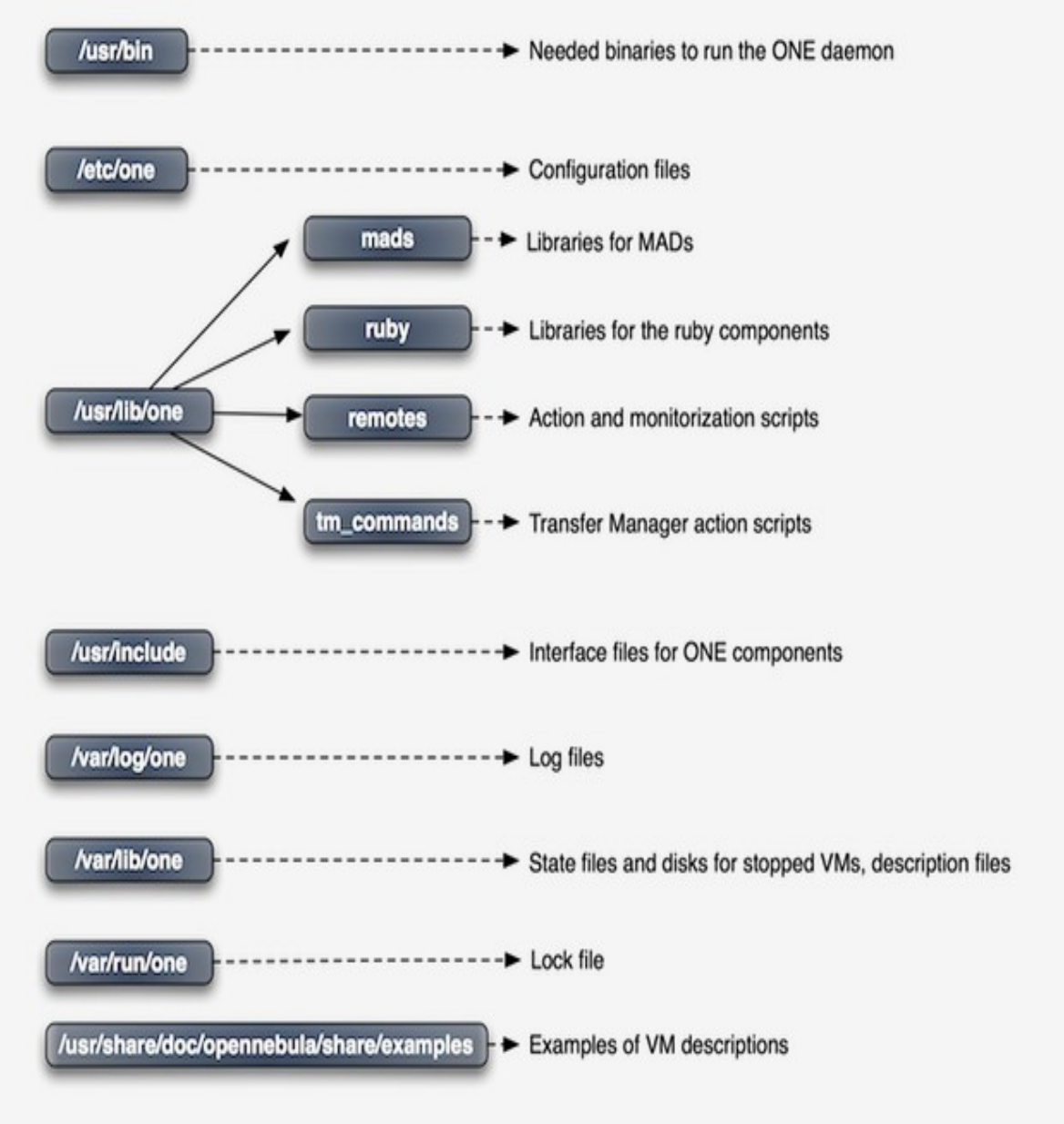
システム一部モード（Self contained）

-d オプションによりディレクトリが特定されて OpenNebula ソフトウェアがインストールされた場合には、$ONE_LOCATION 以下は、このような階層となります。



システム全体モード（System wide）

-d オプションによりディレクトリが特定されずに OpenNebula ソフトウェアがインストールされた場合には、このようなファイル構成となります。



OpenNebula のコンポーネント

OpenNebula では以下の３タイプのプロセスを実行します。

- OpenNebula デーモン（oned）：すべてのモジュールの管理、仮想マシンのライフサイクル管理
- ドライバー：特定のクラスタ・システム（ストレージ、ハイパーバイザーなど）へのアクセス
- スケジューラー：仮想マシン展開の判断

このセクションでは、これらのサービスをどのように設定し、開始するのかを学びます。

OpenNebula デーモン

デーモンのコンフィギュレーション・ファイルは oned.conf です。このファイルは$ONE_LOCATION/etc ディレクトリー（システム全体モードの場合は/etc/one）にあります。

（注）OpenNebula デーモンのすべてのコンフィギュレーション・オプションニ関する詳細な情報は「the oned.conf reference document」にてご確認ください。

oned.conf ファイルは以下のセクションにより構成されます。

- General configuration attributes：クラスタ・ノードと仮想マシンの監視する際、または MAC 接頭辞を用いる際に利用します。
- Information Drivers：クラスタ・ノードを監視する際のアダプターとして用いる
- Virtualization Drivers：ハイパーバイザーのインターフェースとなるアダプター
- Transfer Driver：ストレージ・システムのインターフェースとなり、クローン、削除、仮想マシンのイメージの移動の際に用いられる
- Image Repository：仮想マシンのイメージを保管するために用いる
- Hooks：特定のイベント（例：仮想マシンの構築）を実行するためのもの

以下の例は、KVM と共有ファイルシステムと共に用いる場合に OpenNebula をどのように設定するかを示しています。

（注）フロントエンドの$ONE_LOCATION/var がクラスタ・ノードにマウントされている VM_DIR がパスとして設定されていることを確かめてください。もしこのパスがフロントエンドとクラスタ・ノードで同じである場合（例えばワーカー・ノードが$ONE_LOCATION/var にマウントされており、これがフロントエンドのパスと同じ場合 ）には、VM_DIR 変数は必要ありません。

```
# Attributes
HOST_MONITORING_INTERVAL = 60
VM_POLLING_INTERVAL      = 60

VM_DIR = /srv/cloud/one/var #Path in the cluster nodes to store VM images

SCRIPTS_REMOTE_DIR=/tmp/one
DB = [ backend = "sqlite" ]
VNC_BASE_PORT = 5000

NETWORK_SIZE = 254     #default
MAC_PREFIX   = "00:03"

DEFAULT_IMAGE_TYPE    = "OS"
DEFAULT_DEVICE_PREFIX = "hd"

#Drivers
IM_MAD = [name="im_kvm", executable="one_im_ssh", arguments="kvm"]
VM_MAD = [name="vmm_kvm", executable="one_vmm_sh", arguments="kvm,
          default="vmm_sh/vmm_sh_kvm.conf", type= "kvm" ]
TM_MAD = [name="tm_nfs", executable="one_tm", arguments="tm_nfs/tm_nfs.conf" ]
```

スケジューラー

スケジューラーのモジュールは、待機中の仮想マシンとクラスタ・ノード間の割り振りを担当します。OpenNebula のアーキテクチャでは、このモジュールは別プロセスとして見なされます。よって oned と無関係に開始することができます。OpenNebula ではマッチメーキング・スケジューラー（mm_sched）がすでに搭載されており、ランク別スケジューリング・ポリシーを実装します。

このポリシーの目的は仮想マシンに応じてリソースの優先順位をつけることです。Virtual Machine definition ファイル内の RANK 条件を特定するのみで複数のリソースや使用状況に応じたポリシーを設定することができます。「スケジューリング・ポリシー 2.0」を参考にしてください。

（注）OpenNebula はスケジューリング機能なしでも利用できます。その場合には、仮想マシンの構築ならびにマイグレーションをする場合に onevm コマンドを用いることができます。

（注）OpenNebula のスケジューリング・モジュールとして Haizea lease manager を用いる事も可能です。Haizea を用いる事により、より高度なリソース確保やリクエスト・キューを利用する事ができるようになります。

ドライバー

ドライバーは独立したプロセスとして、OpenNebula のコアと ASCII プロトコルを用いて連携します。ドライバーをロードする前に、２つの run commands（RC）ファイルが用いられ、オプションとして環境変数が取得（environmental variable）されます。

２つの RC ファイルとは：

- $ONE_LOCATION/etc/mad/defaultrc：すべてのドライバーのグローバル環境、タスク。変数は sh シンタックスを用いることにより定義され、読み込みがなされることにより、ドラーバー環境がエクスポートされます。

```
# Debug for MADs [0=ERROR, 1=DEBUG]
# If set, MADs will generate cores and logs in $ONE_LOCATION/var.
ONE_MAD_DEBUG=
# Nice Priority to run the drivers
PRIORITY=19
```

それぞれのドライバーの特定のファイル。それにより defaultrc 変数が再定義されます。特定のオプションについては、それぞれのドライバーのコンフィギュレーション・ガイドを確認してください。

OpenNebula の開始、停止

（注）OpenNebula を始めて実行する場合、アドミニストレーターのアカウントが作成されます。その際に $ONE_AUTH ファイルにユーザーとパスワードを一行で「user:password」と入力してください。

OpenNebula デーモンとスケジューラーは$ONE_LOCATION/bin/one スクリプトで容易に開始できます。<oneadmin>ユーザーとして実行してください。

```
$ one start
```

OpenNebula はディフォルトで古いログを消去します。OpenNebula のメイン・ログをバックアップしたい場合には-b オプションを用いてください。自動的にバックアップがなされます。

```
$ one -b start
```

もしスケジューラーを開始したくない場合には、oned のみを使用して、オプションとして oned -h をチェックします。

現在、２つのプロセスが実行されているはずです。

- oned：コア・プロセス、CLI リクエストの扱い、プールとすべてのコンポーネントの管理
- mm_sched：スケジューラーのプロセス、仮想マシンとクラスタ・ノードのマッチングの管理

これらのプロセスが実行している場合、ログファイルのコンテントを確認する事ができます。（システム全体モードで OpenNebula がインストールされている場合には/var/log/one にてログを確認できます。）

- $ONE_LOCATION/var/one.log
- $ONE_LOCATION/var/sched.log

OpenNebula のユーザー

OpenNebula システムでは２つのタイプのアカウントがあります。

- **oneadmin** は OpenNebula が ONE_AUTH データを用いて始めて開始される場合に作成されます。oneadmin は,いかなるオペレーションやオブジェクト（仮想マシン、ネットワーク、ホスト、ユーザー）をも実行できます。

- **Regular user** アカウントは<oneadmin>によって作成され、自身のオブジェクト（仮想マシン、ネットワーク）のみ管理することができます。

（注）oneadmin により作成される仮想ネットワークは「パブリック」とみなされ、すべてのユーザーにより利用可能です。

OpenNebula ユーザーには以下の環境変数セットがあるはすです。

ONE_AUTH	「username:password」という 1 行が含まれるファイルにポイントする必要があります。もしONE_AUTHが定義されていない場合には、代わりとして$HOME/.one/one_authが用いられる事になります。
ONE_LOCATION	OpenNebulaがシステム一部モード（self-contained）でインストールされている場合には、この変数は<destination_folder>として設定されます。システム全体モード<system wide>の場合には、この変数は設定されません。
ONE_XMLRPC	http://localhost:2633/RPC2
PATH	システム一部モード（self-contained）の場合は$ONE_LOCATION/bin:$PATHとなります。それ以外では必要ありません。

ユーザーの追加、削除

OpenNebula システム内のユーザー・アカウントは<oneadmin>により oneuser ユーティリティを用いて管理されます。ユーザーのシステムへの追加は容易です。このようにして追加できます。

```
$ oneuser create helen mypass
```

このケースでは、ユーザー helen が、このようなコンテンツを$ONE_AUTH ファイルに追加する必要があります。

```
$ export ONE_AUTH="/home/helen/.one/one_auth"
$ cat $ONE_AUTH
helen:mypass
```

ユーザーの削除は容易にできます。

```
$ oneuser delete john
```

システム内のユーザーを列挙する場合には、こちらのコマンドを用いてください。

```
> oneuser list
 UID NAME            PASSWORD                                 ENABLE
   0 oneadmin        c24783ba96a35464632a624d9f829136edc0175e   True
   1 paul            e727d1464ae12436e899a726da5b2f11d8381b26   True
   2 helen           34a91f713808846ade4a71577dc7963631ebae14   True
```

（注）oneuser ユーティリティのより詳細な情報は「Command Line Interface 2.0」にて確認できます。

OpenNebula のホスト

最後に、物理ノードをシステム内に OpenNebula ホストとして追加する必要があります。onehost ユーティリティを用いる事により、いつでもホストを追加することができます。このようになります。

```
$ onehost create host01 im_kvm vmm_kvm tm_nfs
$ onehost create host02 im_kvm vmm_kvm tm_nfs
```

（注）ホストを追加する前に、パスワードなしで ssh を用いる事ができるか確認してください。

物理ホストのモニタリングはプローブ（Probes）によってなされます。プローブとは、ホスト OS よりいくらかの情報を抽出するために作成されたスクリプトのことです。このスクリプトはシステムにホストが追加された際に、リモートで実行中のノード内の SCRIPTS_REMOTE_DIR（$ONE_LOCATION/etc/oned.conf 内のセット）にコピーされます。もしプローブがフロントエンド（$ONE_LOCATION/lib/remotes）に追加ならびに削除された場合には、さらにコピーされます。もしスクリプトに変更があった場合、またはアドミニストレーターが特定のホストにプローブをコピーしたい場合には、「onehost sync」機能を用いる事ができます。

ロギング、デバッグ

それぞれの OpenNebula コンポーネント用に対して、異なるログファイルがあります。

- **ONE デーモン**：OpenNebula のコア・コンポーネントのすべてのログ情報は $ONE_LOCATION/var/oned.log にダンプされます。冗長さは$ONE_LOCATION/ect/oned.conf 内の DEBUG_LEVEL によって規制されます。

- **スケジューラー**：すべてのスケジューラー情報は$ONE_LOCATION/var/sched.log に集められます。

- **仮想マシン**：OpenNebula によってコントロールされるすべての仮想マシンには $ONE_LOCATION/var/<VID>/（システム全体のインストレーションの場合には/var/lib/one/<VID>）フォルダが存在します。そのフォルダ内にて、以下のような情報が確認できます。

    - **ログ・ファイル**：仮想マシン情報はファイルのディレクトリ内の **vm.log** にダンプされます。システム全体モードの場合には、/var/log/one にダンプされます。

    - **展開情報ファイル**：deployment.<EXECUTION>に保存されます。<EXECUTION>番号は、仮想マシンの実行歴順（最初のログは deployment.0、次のログは deployment.1、といった具合）となります。

    - **転送情報ファイル**：transfer.<EXECUTION>.<OPERATION>に保存されます。<EXECUTION>番号は、仮想マシンの実行歴順となり、<OPERATION>はスクリプトが用いられたステージ（transfer.0.prolog、transfer.0.epilog、tranfer.1.cleanup）となります。

    - **保存イメージ**：images/サブディレクトリ内に保存されます。イメージはフォーム disk.<id>内にあります。

    - **再構築ファイル**：チェックポイントの情報がこのディレクトリ内に保存され、仮想マシンに障害が生じた際に用いられます。ステータス情報は checkpoint ファイルに保存されます。

- **ドライバー**：それぞれのドライバーは RC ファイル内の ONE_MAD_DEBUG 変数 をアクティベートできます。 その場合に、エラー情報は$ONE_LOCATION/var/name-of-the-driver-executable.log にダンプされます。ドライバーのログ情報は oned.log にあります。