Eucalyptus 調査報告書

クリエーションライン株式会社

2009 年 6 月 24 日

◎免責条項

本報告書はクリエーションライン株式会社(以下「クリエーションライン」）が作成したものですが、報告書の内容及び情報の正確性、完全性、有用性について、クリエーションラインは保証を行なっておらず、また、いかなる責任を持つものでもありません。

以上の点をご了承の上、ご利用ください。

◎本報告書に関する問い合わせ先

クリエーションライン　info@creationline.com

1. Eucalyptus 6
1.1. 概要 6
1.2. アーキテクチャ 6
1.2.1. クラウドコントローラ 7
1.2.2. クラスタコントローラ 7
1.2.3. ノードコントローラ 7
1.3. ネットワークモード 8
1.3.1. SYSTEM MODE 8
1.3.1.1. SYSTEM MODE の設定 8
1.3.1.2. SYSTEM MODE の注意点 9
1.3.2. STATIC MODE 9
1.3.2.1. STATIC MODE の設定 9
1.3.2.2. STATIC MODE の注意点 10
1.3.3. MANAGED MODE 10
1.3.3.1. MANAGED MODE の設定 11
1.3.3.2. MANAGED MODE の注意点 12
1.3.4. MANAGED-NOVLAN MODE 12
1.3.4.1. MANAGED-NOVLAN MODE の設定 12
1.4. Eucalyptus のインストール手順及び動作確認 14
1.4.1. 環境構築に用いたコンピュータ 14
1.4.2. Cent OS 5.3 によるインストール手順 14
1.4.2.1. Cent OS のインストール 14
1.4.2.2. ソフトウェアのインストール 15
1.4.2.3. Eucalyptus の初期設定 17
1.4.3. Ubuntu 9.04 Server でのインストール手順 22
1.4.3.1. Ubuntu のインストール 22
1.4.3.2. Eucalyptus の初期設定 24
1.4.4. Eucalyptus の動作確認 25
1.5. Eucalyptus 利用手順 33
1.6. Eucalyptus のログファイル 35
1.6.1. Cloud Controller に関するログファイル 35
1.6.1.1. cloud-debug.log 36
1.6.1.2. cloud-error.log 36
1.6.1.3. cloud-output.log 36
1.6.2. Cluster Controller に関するログファイル 36

1.6.2.1. cc.log 37
1.6.2.2. httpd_cc_error_log 37
1.6.3. Node Controller に関するログファイル 38
1.6.3.1. nc.log 38
1.6.3.2. httpd_nc_error_log 38
1.6.3.3. euca_test_nc.log 38
1.6.4. その他のログファイル 39
1.6.4.1. Axis2c.log 39
1.6.4.2. Jetty-request-yyyy_mm_dd.log 39
1.7. eucalyptus.conf 40
1.7.1. 基本設定 40
1.7.2. クラウドコントローラの設定 41
1.7.3. クラスタコントローラの設定 41
1.7.4. ノードコントローラの設定 42
1.7.5. ネットワークの設定 43
1.8. イメージファイルの作成 44
1.8.1. Web 上からダウンロードする 44
1.8.2. 新しく OS をインストールして作成する 45
1.8.3. 既存のコンピュータからイメージを抽出して作成する 46
1.8.3.1. 事前準備 47
1.8.3.2. Virt-P2V の起動 48
1.8.3.3. イメージファイルの縮小 54
2. Amazon EC2（Amazon Elastic Compute Cloud） 56
2.1. 概要 56
2.2. 特徴 57
2.3. 料金体系 58
2.3.1. 基本料金 58
2.3.2. 追加料金 59
2.4. Amazon EC2 利用手順 60
2.4.1. 事前準備 60
2.4.1.1. アカウントのセットアップ 60
2.4.1.2. Amazon EC2 Command-Line Tools のセットアップ 60
2.4.1.3. Elasticfox のセットアップ(EC2 用の GUI) 61
2.4.1.4. SSH アクセスの準備(Elasticfox を用いた手順) 61
2.4.1.5. S3 Organizer のセットアップ（S3 用の GUI） 62
2.4.2. 利用手順 62

2.5. 機能一覧 66
2.5.1. 基本機能 67
2.5.1.1. インスタンスの起動 67
2.5.1.2. インスタンスの停止 67
2.5.1.3. インスタンスの再起動 67
2.5.1.4. インスタンスの起動状態の確認 68
2.5.1.5. 秘密鍵の発行 68
2.5.1.6. 秘密鍵の削除 68
2.5.1.7. セキュリティグループの作成 68
2.5.1.8. AMI の作成 68
2.5.1.9. AMI の削除 68
2.5.1.10. AMI のアップロード 69
2.5.1.11. AMI の登録 69
2.5.1.12. AMI の登録解除 69
2.5.1.13. AMI の一覧の確認 69
2.5.2. EBS 69
2.5.2.1. ボリュームの作成 69
2.5.2.2. ボリュームの削除 70
2.5.2.3. ボリュームのアタッチ 70
2.5.2.4. ボリュームのデタッチ 70
2.5.2.5. スナップショットの作成 70
2.5.2.6. スナップショットの削除 70
2.5.3. Elastic IP 70
2.5.3.1. 固定 IP の設定 70
2.5.3.2. 固定 IP の解除 71
2.5.4. CloudWatch 71
2.5.4.1. インスタンス・ロードバランサのモニタリング 71
2.5.4.2. 稼動状況の確認 71
2.5.5. Auto Scaling 72
2.5.5.1. 設定の作成 72
2.5.5.2. オートスケーリングの設定 72
2.5.5.3. 閾値の設定 72
2.5.5.4. オートスケーリングの状態の確認 72
2.5.5.5. オートスケーリングの履歴の確認 72
2.5.5.6. オートスケーリングの設定の確認 73
2.5.5.7. オートスケーリングの解除 73

2.5.5.8. 設定の削除 73
2.5.5.9. 閾値の削除 73
2.5.5.10. グループの削除 73
2.5.6. ELB(Elastic Load Balancing) 74
2.5.6.1. ロードバランサの作成 74
2.5.6.2. ロードバランサの削除 74
2.5.6.3. 接続するインスタンスの設定 74
2.5.6.4. バックエンドのヘルスチェック 74
3. まとめ 75

1. Eucalyptus

1.1. 概要

Eucalyptus はカリフォルニア大学サンタバーバラ校が開発を進めているオープンソースのソフトウェアであり、ユーザが所有するコンピュータリソース上に仮想のコンピュータを構築するクラウドシステムを実装する事が可能である。Eucalyptus は Amazon が提供する Amazon Elastic Compute Cloud(EC2)とのインターフェイス互換性があり、ストレージサービスである Amazon Simple Storage Service (S3)や Amazon Elastic Block Store (EBS)との互換性も実装されている。Eucalyptus では、S3 は Walrus、EBS は Block Storage にそれぞれ対応する。

1.2. アーキテクチャ

Eucalyptus は3層の階層構造からなる。

Eucalyptus の全体構成図を以下に示す。

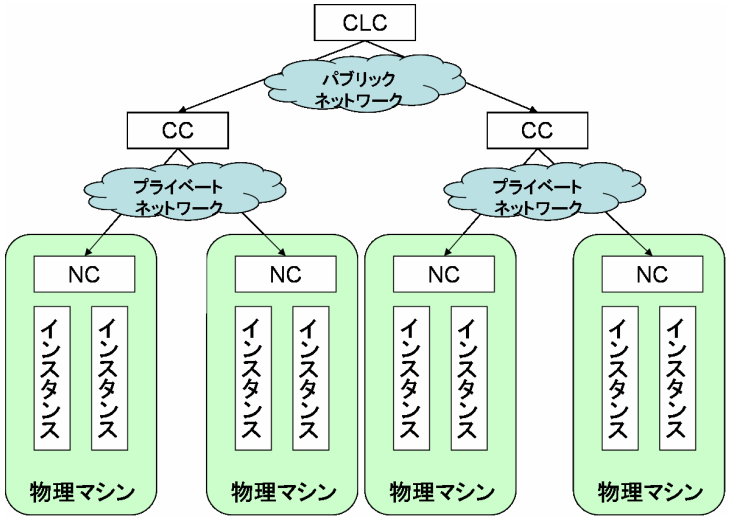


図 1 全体構成図

Eucalyptus は以下の3つのコンポーネントで構成される。

- クラウドコントローラ(CLC)
- クラスタコントローラ(CC)
- ノードコントローラ(NC)

それぞれのコンポーネントについて説明する。

1.2.1. クラウドコントローラ

クラウドコントローラ(CLC)は、クラウドシステムのフロントエンドでユーザとのインターフェイスや管理用のブラウザインタフェースを提供する。EC2 互換のインターフェイスを備え、EC2向けのAPI を用いて下位層の制御を行う。また、ストレージサービスである Walrus も CLC に同居している。

1.2.2. クラスタコントローラ

クラスタコントローラ(CC)は、フロントエンドもしくは、NC や CLC が稼動しているマシン双方にパブリック・プライベートアクセス可能なマシン上で動作する。CC の主要な機能は複数の NC の状態の管理・インスタンスを実行する NC のスケジューリング管理・インスタンスの仮想ネットワーク管理の3つである。CC は NC に対して物理マシンのリソース(CPU のコア数やメモリ及びストレージの容量)を問い合わせ、起動可能なインスタンス数を算出し、CLC に報告する。インスタンスを起動する物理マシンは、物理マシンのリソースに合わせて振り分ける。NC が稼動する物理マシンとの接続が追加・切断された際は、起動可能なインスタンス数を再計算する。

1.2.3. ノードコントローラ

ノードコントローラ(NC)は、物理マシン上で動作するインスタンスを管理するコンポーネントである。NC は上位のコンポーネントからの指示で、インスタンスの起動・停止や停止に伴うデータの削除を行い、物理マシンのリソース(コア数・メモリ・利用可能なディスク容量)やインスタンスの状態を監視している。また、物理マシン一台につき一つの NC は動作している必要があり、一つの NC で複数のインスタンスを起動する事が可能である。但し、現バージョンでは、複数の物理マシンにまたがるインスタンスの起動や物理マシンのスペックを越えるインスタンスの起動は出来ないため、冗長性を持たせるためには、RAID を組むなど物理的な構成を考慮する必要がある。

インスタンスを起動する際、NC はイメージファイル(マシンイメージ・カーネルイメージ・ラムディスクイメージ)をストレージである Walrus やリモートレポジトリからコピーする。Walrus は CLC に同居している為、NC から CLC に通信が必要であり、イメージファイルの転送は、平文で行われるので、セキュアなネットワークである必要がある。

インスタンスを停止すると、NC はインスタンスに関連するファイルを全て削除し、インスタンスの停止後はイメージファイルも保存されない。

1.3. ネットワークモード

Eucalyptus には、仮想マシンをネットワーク上で自動的に構築する為に、4つのモードが用意されている。本項ではこれらのモードについて、特徴、設定方法、注意点などについて説明を行う。ネットワークモードの設定は、eucalyptus.conf（$EUCALYPTUS/etc/eucalyptus/eucalyptus.conf）の設定を変更する事で行う。

1.3.1. SYSTEM MODE

SYSTEM MODE は最も単純なネットワークモードである。特に小規模のネットワーク環境を構築する際に利用される。このモードでは、仮想マシンにランダムに MAC アドレスを割り振り、物理マシンと仮想マシンのネットワークデバイスを仮想ブリッジによって接続する。仮想マシンは物理マシンと同様にDHCPサーバによってIPが自動的に割り振られ、物理マシンと同じネットワーク領域に仮想マシンを構築する。

デスクトップ PC などで実験的に Eucalyptus の構築を行う場合に有効であるが、SYSTEM MODE を用いる際には、DHCP サーバが、構築する仮想マシンに対して充分な数の IP アドレスを保有していなければならない。

1.3.1.1. SYSTEM MODE の設定

SYSTEM MODE を設定する為には、front-end、node それぞれの eucalyptus.conf の項目を以下の様に変更する必要がある。

front-end での eucalyptus.conf の設定例

```
VNET_MODE="SYSTEM"
```

node での eucalyptus.conf の設定例

```
VNET_MODE="SYSTEM"
VNET_BRIDGE="virbr0"
```

1.3.1.2. SYSTEM MODE の注意点

SYSTEM MODE を利用する場合、仮想マシンは DHCP から割り振られる IP アドレスによって、物理マシンと区別する事無くネットワーク上に配置しようとする。この為、CC のネットワーク設定やネットワークの構造などによって、仮想マシンの IP が設定されない場合がある。

IP が設定されない場合、”ec2-describe-instances” で仮想マシンの状態を表示させると、プライベート IP とパブリック IP がともに ”0.0.0.0” と表示される。このような問題は、ネットワークの構成や CC のネットワーク設定を変更する事で解決する事がある。

1.3.2. STATIC MODE

STATIC MODE では、予め用意された複数の IP アドレスと MAC アドレスのペアを、起動した仮想マシンに割り振る。IP / MAC アドレスのペアは CC 上に確保されており、CC 上の DHCP を利用して仮想マシンに IP / MAC アドレスのペアを順に割り振る。仮想マシンは、仮想ブリッジを用いてネットワークと接続するため、SYSTEM MODE と同様にネットワーク上の物理マシンと区別する事無く IP アドレスを取得する。

このモードは、 IP アドレスと MAC アドレスを定めて仮想マシンのネットワークを構築する際に利用される。

1.3.2.1. STATIC MODE の設定

STATIC MODE を設定する為には、front-end、node それぞれの eucalyptus.conf の項目を以下の様に変更する必要がある。

front-end での eucalyptus.conf の設定例

```
VNET_MODE="STATIC"
VNET_INTERFACE="eth0"
VNET_DHCPDAEMON="/usr/sbin/dhcp3"
VNET_DHCPUSER="<dhcpusername>"
VNET_SUBNET="192.168.1.0"
VNET_NETMASK="255.255.255.0"
VNET_BROADCAST="192.168.1.255"
VNET_ROUTER="192.168.1.1"
VNET_DNS="192.168.1.2"
VNET_MACMAP=        "00:AA:DD:11:CE:FE=192.168.1.3
                     00:AA:DD:11:CE:FF=192.168.1.4
                     …"
```

node での eucalyptus.conf の設定例

```
VNET_MODE="STATIC"
VNET_BRIDGE="virbr0"
```

STATIC MODE では、front-end が内部の DHCP サーバを利用するため、 DHCP に関する設定を Eucalyptus 上で行う必要がある。以下、各設定項目について説明する。

"VNET_INTERFACE" では、各 node と接続している front-end のイーサネットポートを指定する。

"VNET_DHCPDAEMON" では、 front-end 内の DHCP デーモンのパスを指定する。

"VNET_DHCPUSER" では、 DHCP デーモンが root 以外のユーザで実行される設定の場合に、実行されるユーザ名を指定する必要がある。

"VNET_SUBNET"、"VNET_NETMASK"、"VNET_BROADCAST"、"VNET_ROUTER"、"VNET_DNS" には、 DHCP サーバが割り振るネットワークの情報や DNS サーバのアドレスを指定する。

"VNET_MACMAP" には、MAC アドレスと IP アドレスの対応を記述する。

1.3.2.2. STATIC MODE の注意点

STATIC MODE では、Eucalyptus が front-end 内の DHCP デーモンを利用して、 IP / MAC アドレスを割り振る。このため、DHCP デーモンは Eucalyptus が対応している ISC DHCP DAEMON ver. 3.0.x か、それに互換性のあるものを使う必要がある。

1.3.3. MANAGED MODE

MANAGED MODE では、仮想マシンに Eucalyptus 用に割り振られたネットワーク IP を割り当てる。Amazon EC2 の Security Group の様に、管理者はネットワークを分割する事で、複数のネットワークを Eucalyptus 上に構築し、ユーザはネットワークの設定を変更する事で、別の Eucalyptus 上のネットワークや外部ネットワークからの通信を制限する事が出来る。また、Amazon EC2 の Elastic IP の様に、ネットワーク上に配置された仮想マシンのパブリック IP を動的に変化させる事も出来る。

MANAGED MODE は、Eucalyptus 上に Security Group や Elastcic IP、仮想マシンのネットワークからの分離などを行いたい場合に有効である。

1.3.3.1. MANAGED MODE の設定

MANAGED MODE でネットワークを構築するには以下の条件が必要となる。

- 絶対に使用されない、充分に広い IP アドレス空間がある事
- Eucalyptus 上で VLAN が完全に利用可能である事
- Front-end の firewall が無効か、または動的な変更が可能な事

MANAGED MODE を設定する為には front-end、node それぞれの eucalyptus.conf の項目を以下の様に変更する必要がある。

front-end での eucalyptus.conf の設定例

```
VNET_MODE="MANAGED"
VNET_INTERFACE="eth0"
VNET_DHCPDAEMON="/usr/sbin/dhcp3"
VNET_DHCPUSER="<dhcpusername>"
VNET_SUBNET="192.168.1.0"
VNET_NETMASK="255.255.0.0"
VNET_DNS="192.168.1.2"
VNET_ADDRSPERNET="64"
VNET_PUBLICIPS= 10.0.0.0 10.0.0.1 …
```

Node での eucalyptus.conf の設定例

```
VNET_MODE="MANAGED"
VNET_INTERFACE="peth0"
```

"MANAGED MODE" では、front-end 内の DHCP デーモン の設定とパブリックネットワークの設定を行う必要がある。

"VNET_INTERFACE" では、各 node と接続している front-end のイーサネットポートを指

定する。

VNET_DHCPDAEMON では、 front-end 内の DHCP デーモンのパスを指定する。

VNET_DHCPUSER では、 DHCP デーモンが root 以外のユーザで実行される設定の場合に、実行されるユーザ名を指定する必要がある。

VNET_SUBNET、VNET_NETMASK、VNET_DNS では、MANAGED MODE で利用するパブリック IP アドレス空間や、DNS サーバを指定する。

VNET_ADDRSPERNET では一つのネットワークに含まれる IP アドレスの数を設定する。IP アドレスの総数と VNET_ADDRSPERNET の値によってネットワークの最大数が設定される。

VNET_PUBLICIPS では上記のパブリック IP アドレス以外の IP アドレスを指定する事で、IP アドレス空間を拡張することができる。

1.3.3.2. MANAGED MODE の注意点

MANAGED MODE を利用する場合、一つの PC 上で Eucalyptus System を構築する事は出来ない。また、MANAGED MODE の設定を行う時、front-end の iptables のうち、filter と nat の設定をクリアし、filter カテゴリの FORWARD ポリシーを DROP に設定する必要がある。

1.3.4. MANAGED-NOVLAN MODE

MANAGED-NOVLAN MODE は、VLAN を用いずに MANAGED MODE と同様に Security group と Elastic IP を実装する事が出来る。VLAN が利用できない環境で MANAGED MODE を構築する場合に有効だが、仮想マシンと物理マシンのネットワークでの分離は出来ない。

1.3.4.1. MANAGED-NOVLAN MODE の設定

MANAGED-NOVLAN MODE でネットワークを構築するには以下の条件が必要となる。

- 絶対に使用されない、充分に広い IP アドレス空間がある事
- Front-end の firewall が無効か、または動的な変更が可能な事

MANAGED-NOVLAN MODE を設定する為には front-end 、node それぞれの eucalyptus.conf の項目を以下の様に変更する必要がある。

front-end での eucalyptus.conf の設定例

```
VNET_MODE="MANAGED-NOVLAN"
VNET_INTERFACE="eth0"
VNET_DHCPDAEMON="/usr/sbin/dhcp3"
VNET_DHCPUSER="<dhcpusername>"
VNET_SUBNET="192.168.1.0"
VNET_NETMASK="255.255.0.0"
VNET_DNS="192.168.1.2"
VNET_ADDRSPERNET="64"
VNET_PUBLICIPS= 10.0.0.0 10.0.0.1 …
```

node での eucalyptus.conf の設定例

```
VNET_MODE="MANAGED-NOVLAN"
VNET_BRIDGE="virbr0"
```

MANAGED-NOVLAN MODE の設定は、node の設定が“VNET_INTERFACE“ ではなく ”VNET_BRIDGE“ となっている点以外は、基本的にはMANAGED MODE と同様である。

1.4. Eucalyptus のインストール手順及び動作確認

本項では、Eucalyptus を Cent OS 5.3 及び、Ubuntu 9.04 Server 上でインストールした場合のインストール手順について示した後、Eucalyptus の動作確認を行う。

1.4.1. 環境構築に用いたコンピュータ

Cloud-Controller、Cluster-Controller、Node-Controller(1)

CPU：Core2Quad Q9450(2.66GHz)
Memory：4GB (DDR2SDRAM 2GB×2)
HDD：500GB×1

Node-Controller (2)

CPU：Pentium 4 2.8GHz
Memory：2GB (DDRSDRAM 1GB ×1)
HDD：250GB×1

1.4.2. Cent OS 5.3 によるインストール手順

1.4.2.1. Cent OS のインストール

Graphical Mode でインストールを開始する。
以下にインストール時の環境設定を示す。

インストール時の言語・・・Japanese
キーボードタイプ・・・jp106
インストール方法・・・FTP

TCP/IP を設定する。

IPv4 は Manual configuration (グローバル IP)
IP アドレス　　***.***.***.***
ゲートウェイ　　***.***.***.***
DNS サーバ　　***.***.***.***
IPv6 のチェックは外す。

FTP を設定する。

FTP サーバ　　ftp.riken.jp
ディレクトリ　　/Linux/centos/5.3/os/i386

non-anonymouse ftp のチェックは入れない。(匿名でのログインが許可されている)

パーティションを設定する。

「選択したドライブ上のすべてのパーティションを削除してデフォルトのレイアウトを作成します。」を選択する。

ローカル IP・ホスト名を設定する。

ローカル IP	192.168.1.100/255.255.255.0
ホスト名	front-end

リージョンを設定する。

Asia/Tokyo を選択する。

root のパスワードを設定する。

パッケージを選択する。

Server 及び、仮想化(Virtualization)を選択する。

1.4.2.2. ソフトウェアのインストール

アップデートし、DHCP・RUBY をインストールする。

```
# yum update
# yum install dhcp
# yum install ruby
```

・JAVA(jdk)のインストール

URL:http://java.sun.com/javase/downloads/index.jsp

上記 URL より/jdk/jdk-6u14-linux-i586-rpm.bin をダウンロードし、~/jdk フォルダにおく。

rpm を実行し、JAVA をインストールする。

```
# cd ~/jdk
# chmod 755 ./jdk-6u14-linux-i586-rpm.bin
# ./jdk-6u14-linux-i586-rpm.bin
```

・ant のインストール

URL:http://ant.apache.org/bindownload.cgi

上記 URL より apache-ant-1.7.1-bin.tar.gz をダウンロードし、~/ant フォルダにおく。
yum コマンドを用いて、ant をインストールする。

```
# cd ~/opt
# tar zxvf ~/ant/apache-ant-1.7.1-bin.tar.gz
# ln -s apache-ant-1.7.1 apache-ant
# yum install ant-nodeps
```

・Eucalyptus のインストール

CentOS 用の Eucalyptus のパッケージをダウンロードする。

```
# mkdir ~/eucalyptus
# cd eucalyptus
# wget http://open.eucalyptus.com/downloads/89
```

rpm を実行し、Eucalyptus をインストールする。

```
# tar xzf eucalyptus-1.5.1-centos-i386.tar.gz
# cd eucalyptus-1.5.1-centos-i386
# cd eucalyptus-1.5.1-rpm-deps-i586
# rpm -Uvh *.rpm
# cd ..
```

・Amazon EC2 の API と AMI のツールをダウンロード

```
# mkdir ~/ec2tools
# wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools-1.3-30349.zip
# wget http://s3.amazonaws.com/ec2-downloads/ec2-ami-tools-1.3-26357.zip
# unzip ec2-api-tools-1.3-30349.zip -d ~/ec2tools
# unzip ec2-ami-tools-1.3-26357.zip -d ~/ec2tools
```

1.4.2.3. Eucalyptus の初期設定

一旦ファイアウォールを停止して Eucalyptus を起動する。

```
# /etc/rc.d/init.d/iptables stop
# /opt/eucalyptus/etc/init.d/eucalyptus-cc start
# /opt/eucalyptus/etc/init.d/eucalyptus-cloud start
# rpm -Uvh *.rpm
```

ブラウザで https://<front-end-ip-address>:8443/ にアクセスし環境設定する。
Eucalyptus のログイン画面より、Username、Password を入力する。
初期設定ではともに admin となっている。

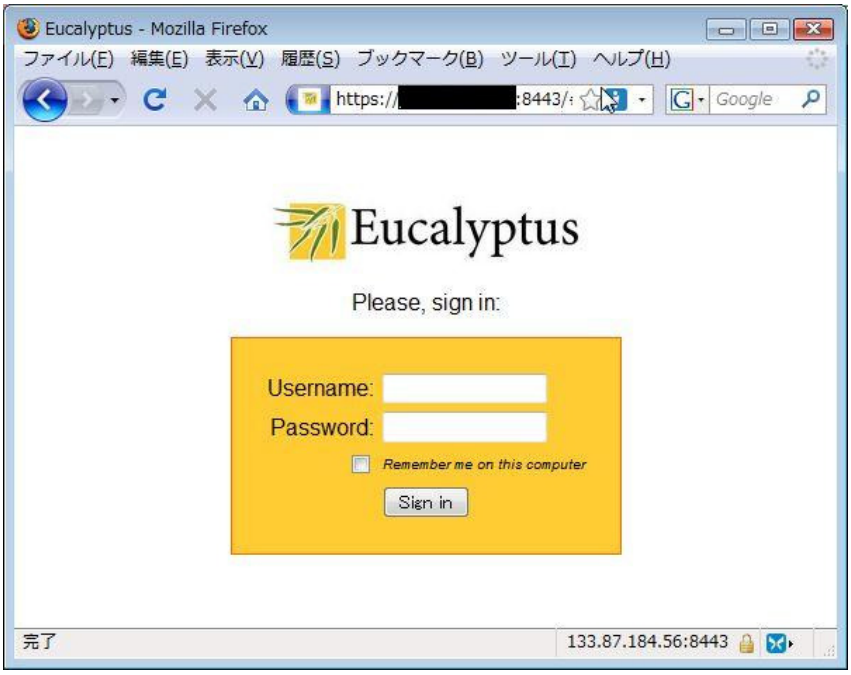


図 2 ログイン画面

ログイン後、パスワード変更画面よりパスワードを変更する。

図 3 パスワード変更画面

パスワードを変更し、OK を選択する。

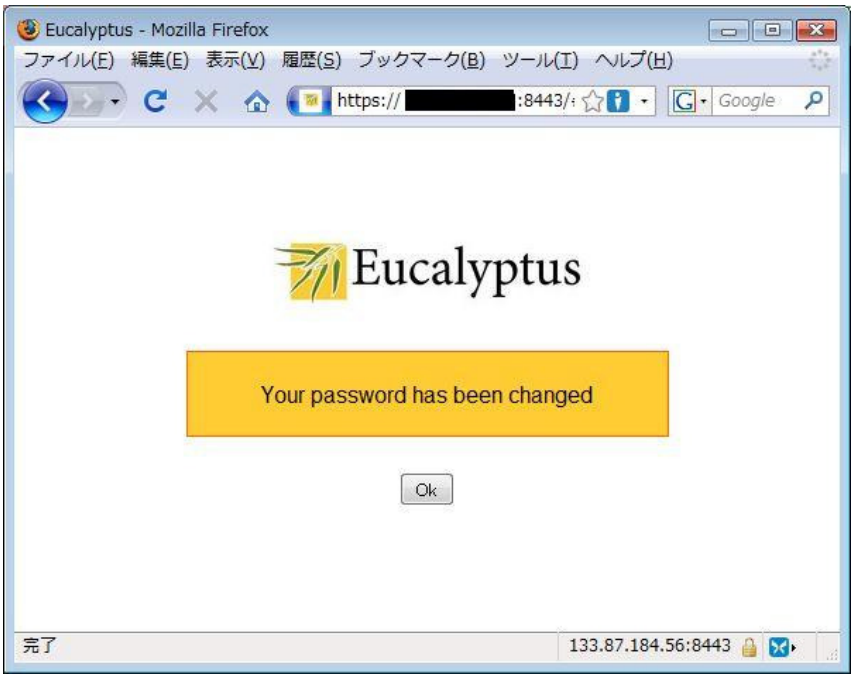


図 4 パスワード変更完了画面

メールアドレスを入力し、Change address を選択する。

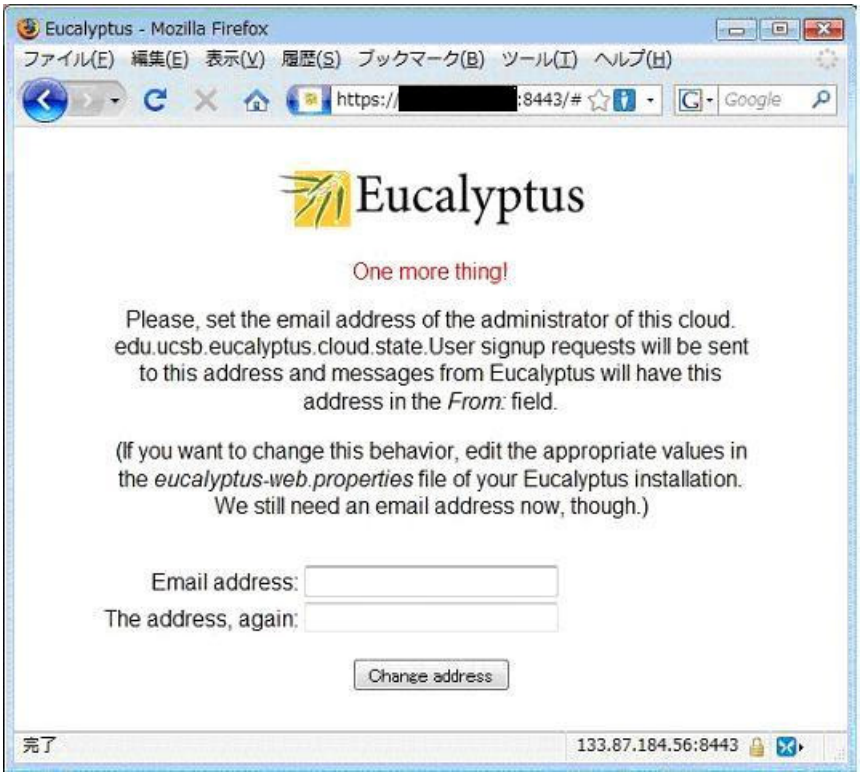


図 5 メールアドレス入力画面

ストレージサーバである Walrus の URL を入力し、Confirm URL を選択する。

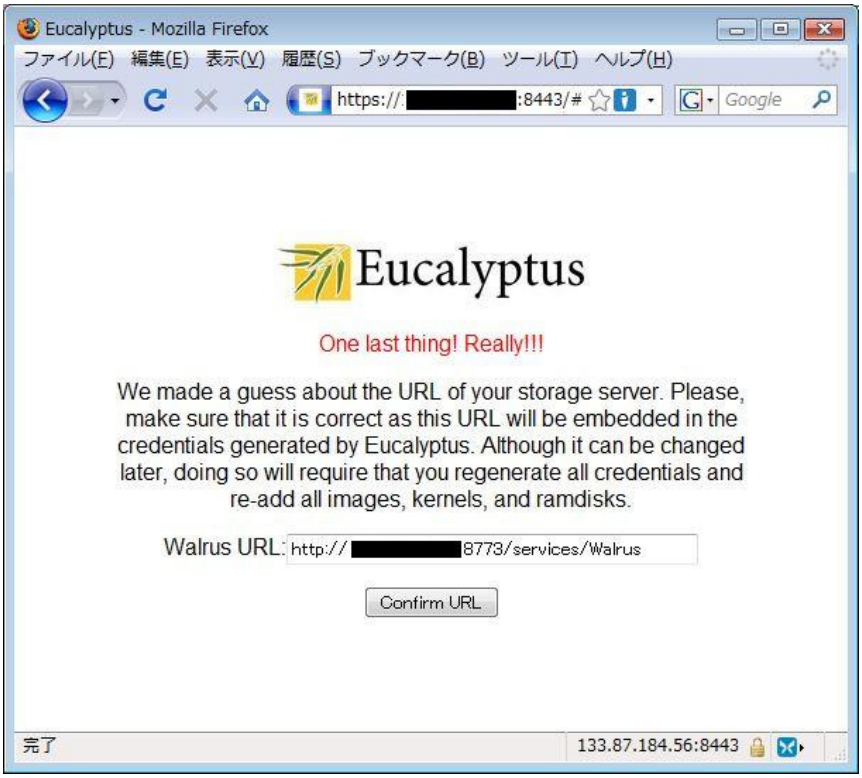


図 6 Walrus URL 設定画面

Configuration を選択し、設定を行う。

Walrus configuration（図7）より Walrus のバケットパスや、バケットのサイズ・キャッシュのサイズ等を設定する。

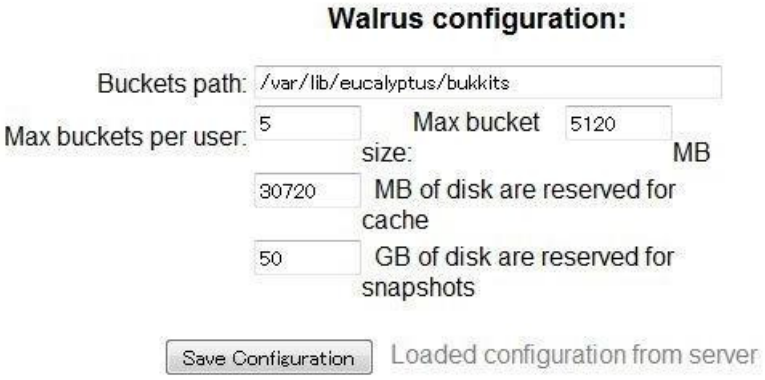


図 7 Walrus 設定画面

Cluster（図 8）の Add Cluster ボタンを押し、 新しいクラスターを追加する。
作成するクラスター名を Name: に書き込み、 Save cluster configuration ボタンを押す。

図 8 Cluster 設定画面（クラスター未設定）

クラスターを追加した後は、設定の変更も可能（図9）。CC の設定やボリュームに関する設定はこの項目からも行える。

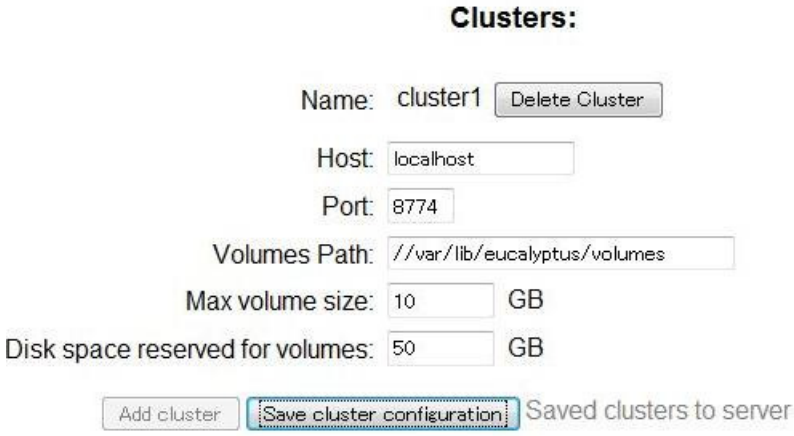


図 9 Cluster 設定画面（クラスター設定済み）

仮想マシンのスペックの設定は VM Type（図10）から行える。
下位の仮想マシンのスペックが上位を超えない様に各項目を設定しなければならない。

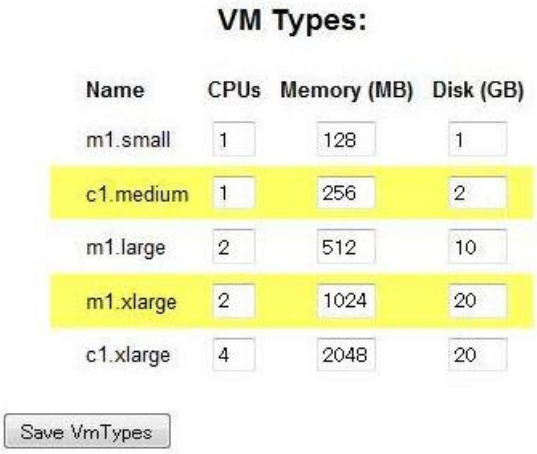


図 10 VM Type 設定画面（クラスター未設定）

X.509 certificate をダウンロードし、~/.euca フォルダに展開する。

```
# mkdir ~/.euca
# cd ~/.euca
# unzip euca2-admin-x509.zip
# chmod 0700 ~/.euca
# chmod 0600 ~/.euca/*
```

環境変数を設定する。

```
# vi ~/.bash_profile
```

```
export JAVA_HOME=/usr/java/default
export ANT_HOME=/opt/apache-ant
export EC2_HOME=/root/ec2tools/ec2-api-tools-1.3-30349
export EC2_AMITOOL_HOME=/root/ec2tools/ec2-ami-tools-1.3-26357
export PATH=$PATH:$EC2_HOME/bin:$EC2_AMITOOL_HOME/bin:
        $ANT_HOME/bin
source ~/.euca/eucarc
```

[.bash_profile ファイル内]

1.4.3. Ubuntu 9.04 Server でのインストール手順

1.4.3.1. Ubuntu のインストール

デスクトップ環境をインストールする。

```
# apt-get install ubuntu-desktop
```

ネットワークを設定する。interfaces を編集し、下記を追記する。
DHCP であれば変更する必要はない。

```
# vi /etc/network/interfaces
```

```
auto eth0
iface eth0 inet static
address ***.***.***.***
netmask 255.255.255.0
gateway ***.***.***.***
```

[/etc/network/interfaces 内]

DNS サーバを設定する。

resolv.conf を編集し、nameserver に DNS サーバの IP アドレスを記述する。

```
# vi /etc/resolv.conf
```

・KVM 環境をインストール

```
# sudo apt-get install ubuntu-vm-builder kvm qemu
# sudo apt-get install libvirt-bin
# sudo apt-get install virt-manager
```

・Eucalyptus をインストール

```
# sudo apt-get install eucalyptus-cloud
# sudo apt-get install eucalyptus-cc
# sudo apt-get install eucalyptus-nc
```

・その他インストール(全部必要かどうかは不明)

```
# sudo pat-get install openssl
# sudo apt-get install libssl-dev
# sudo apt-get install ruby
# sudo apt-get install libopenssl-ruby
# sudo apt-get install curl
```

・Amazon EC2 の API と AMI のツールをダウンロード

```
# mkdir ~/ec2tools
# wget http://s3.amazonaws.com/ec2-downloads/ec2-api-tools-1.3-30349.zip
# wget http://s3.amazonaws.com/ec2-downloads/ec2-ami-tools-1.3-26357.zip
# unzip ec2-api-tools-1.3-30349.zip -d ~/ec2tools
# unzip ec2-ami-tools-1.3-26357.zip -d ~/ec2tools
```

1.4.3.2. Eucalyptus の初期設定

eucalyptus.conf を編集する。
VNET_BRIDGE を変更して、ブリッジを設定する。
Xen ではなく libvirt の管理するブリッジを使用する。
また、同じホストで cc と nc を稼動させる為、NODES に localhost を追加する。

```
# vi /etc/eucalyptus/eucalyptus.conf
```

```
#VNET_BRIDGE="xenbr0"
VNET_BRIDGE="virbr0"

NODES="localhost"
```

[/etc/eucalyptus/eucalyptus.conf 内]

Eucalyptus を起動する。

```
# sudo service eucalyptus-cloud restart
# sudo service eucalyptus-cc restart
# sudo service eucalyptus-nc restart
```

ブラウザで https://localhost:8443/にアクセスする。
ブラウザ上での Eucalyptus の初期設定は、Cent OS の 1.4.2.3 を参照して設定する。
X.509 certificate をダウンロードする。

環境変数を設定する。

```
# mkdir .euca
# mv euca2-admin-x509.zip .euca
# cd .euca
# unzip euca2-admin-x509.zip
```

```
# vi ~/.bashrc
```

```
export JAVA_HOME=/usr/lib/jvm/default-java
export EC2_HOME=/home/user/ec2tools/ec2-api-tools-1.3-30349
export EC2_AMITOOL_HOME=/home/user/ec2tools/ec2-ami-tools-1.3-26357
export PATH=$PATH:$EC2_HOME/bin:$EC2_AMITOOL_HOME/bin
source ~/.euca/eucarc
```

[.bashrc 内]

1.4.4. Eucalyptus の動作確認

1.4.2 の Cent OS 上でインストールした Eucalyptus の動作確認を行う。
ハイパーバイザは Xen を用いて、Fedora9 を起動する。

・Xen 用のサンプルイメージファイル（今回は Fedora9）をダウンロード
URL:*http://jailtime.org/lib/exe/fetch.php?cache=cache&media=download%3Afedora%3Afedora.fc9.20080706.img.tar.bz2*
上記 URL より、ダウンロードし、~/image-fedora フォルダにおく。

```
# mkdir image-fedora
# cd image-fedora
# tar xvjf fedora.fc9.20080706.img.tar.bz2
```

SCSI モジュールなしの ramdisk イメージを構築する。
（標準のものだと Xen で起動時にエラーが出る模様）

```
# cd ~
# mkinitrd --omit-scsi-modules --with=xennet --with=xenblk --preload=xenblk
initrd-$(uname -r)-no-scsi.img $(uname -r)
# mv initrd-2.6.18-128.el5xen-no-scsi.img /boot
```

fedora.fc9.xen3.cfg を編集する。

```
# vi ~/image-fedora/fedora.fc9.xen3.cfg
```

```
kernel = "/boot/vmlinuz-2.6.18-128.el5xen"
ramdisk = "/boot/initrd-2.6.18-128.el5xen-no-scsi.img"
memory = 428
name = "fedora.fc9"
vif = [ '' ]
dhcp = "dhcp"
disk = ['file:/root/image-fedora/fedora.fc9.img、sda1、w']
root = "/dev/sda1 ro"
```

[fedora.fc9.xen3.cfg ファイル内]

Xen でイメージを起動する。(多少起動には時間がかかる)

```
# xm create ~/image-fedora/fedora.fc9.xen3.cfg
# xm console fedora.fc9
```

fedora の VM 内でユーザ・パスワードを追加し、IP アドレスを確認する。

```
# adduser user
# passwd user
# ifconfig
```

Ctrl ^]で domain 0 に戻る。

ifconfig で確認した IP アドレスでログイン可能かどうか確認する。

```
# ssh user@IP アドレス
```

・Eucalyptus のイメージ作成

Amazon EC2 の API を用いて、カーネルイメージを作成する。

```
# cd ~
# mkdir kernel
# ec2-bundle-image -i /boot/vmlinuz-2.6.18-128.el5xen -d ./kernel --kernel true
# ec2-upload-bundle -b kernel -m
  ./kernel/vmlinuz-2.6.18-128.el5xen.manifest.xml
# EKI=`ec2-register kernel/vmlinuz-2.6.18-128.el5xen.manifest.xml |
    awk '{print $2}'`
```

Amazon EC2 の API を用いて、ラムディスクのイメージを作成する。

```
# mkdir ramdisk
# ec2-bundle-image -i /boot/initrd-2.6.18-128.el5xen-no-scsi.img -d ./ramdisk
--ramdisk true
# ec2-upload-bundle -b ramdisk -m
 ramdisk/initrd-2.6.18-128.el5xen-no-scsi.img.manifest.xml
# ERI=`ec2-register ramdisk/initrd.img-2.6.28-11-generic.manifest.xml |
 awk '{print $2}'`
```

Amazon EC2 の API を用いて、仮想マシンのイメージを作成し、アップロードする。アップロードする際の注意点として、アップロード可能なサイズは 10GB までである。これは、EC2 の制約であり、ami-tools のソースコードを修正する事で、変更可能だと予想されるが、今回の調査では、実施していない。

```
# mkdir image
# ec2-bundle-image -i image-fedora/fedora.fc9.img -d ./image --kernel $EKI
 --ramdisk $ERI
# ec2-upload-bundle -b image -m ./image/fedora.fc9.img.manifest.xml
# EMI=`ec2-register image/fedora.fc9.img.manifest.xml | awk '{print $2}'`
```

秘密鍵を発行し、保存する。

```
# ec2-add-keypair mykey > ~/.euca/mykey.priv
# chmod 0600 ~/.euca/mykey.priv
```

・インスタンスの起動

秘密鍵を用いて、インスタンスを起動する。

```
# ec2-run-instances $EMI -k mykey
```

ノードコントローラを起動する。

```
# /opt/eucalyptus/etc/init.d/eucalyptus-nc start
# /opt/eucalyptus/usr/sbin/euca_conf -addnode localhost
 /opt/eucalyptus/etc/eucalyptus/eucalyptsu.conf
```

eucalyptus.conf を編集する。

```
# vi /opt/eucalyptus/etc/eucalyptus/eucalyptus.conf
```

```
VNET_BRIDGE="xenbr0"
↓
VNET_BRIDGE="virbr0"
```

[eucalyptus.conf ファイル内]

・cluster を作成

Web のインターフェイスより cluster を作成する。

名前を cluster1、Host を localhost として　save cluster configuration を選択する。

VM Types　のメモリ、ディスク容量を設定し、save VM Types を選択する。

Eucalyptus 上でイメージを動かす為には、メモリは 512MB 以上、ディスク容量は 4GB 以上を推奨する。ディスク容量は最低イメージが収まるサイズ必要であり、デフォルトでは、ディスク容量は 1GB に設定されているので、注意が必要である。

（ディスク容量が不足している場合、pending→terminated になる）

サーバを再起動する。

```
# /opt/eucalyptus/etc/init.d/eucalyptus-nc restart
# /opt/eucalyptus/etc/init.d/eucalyptus-cc restart
# /opt/eucalyptus/etc/init.d/eucalyptus-cloud restart
```

インスタンスを起動する。

```
# ec2-run-instances $EMI -k mykey
```

【参考① S3 コマンドを用いたアップロード】

イメージのアップロードがうまくいかないときは s3cmd ツールを使う。アップロードに失敗すると、ec2-upload-bundle で 409（conflict error）が起きる。一度 409 エラーに遭遇した場合は s3cmd ツールを使って、アップロードしたファイルを全て消すと 409 エラーが起きなくなる。rm などでアップロード先のファイルやフォルダを削除してはいけない。（削除した場合でも s3cmd ツールで ls してみるとファイルやフォルダが残っているので s3cmd del コマンドで削除することが可能）

どうしてもアップロードがうまくいかないときは、eucalyptus-cloud を再起動してみるとうまくいくことがある。なぜアップロードが失敗するのかは不明である。

・s3cmd のインストール

URL:http://open.eucalyptus.com/wiki/s3cmd

URL:http://sourceforge.net/project/showfiles.php?group_id=178907&package_id=218690#files

【注意】最新版ではなく 0.9.8.3 を DL すること

```
# mkdir s3cmd
# cd s3cmd
# tar zxvf s3cmd-0.9.8.3.tar.gz
```

URL:http://open.eucalyptus.com/wiki/s3cmd

上記 URL を参考に patch を作成する。

```
# cd s3cmd-0.9.8.3
# patch -p1 < s3cmd-0.9.8.3.patch
```

URL:http://open.eucalyptus.com/wiki/s3cmd

上記 URL を参考に s3cfg.walrus を作成する。

（access_key と secret_key は.euca/eucarc を参考にする。）

```
[default]
access_key = ISMvKXpXpadDiUoOSoAfww
acl_public = False
bucket_location = US
debug_syncmatch = False
default_mime_type = binary/octet-stream
delete_removed = False
dry_run = False
encrypt = False
force = False
gpg_command = /usr/bin/gpg
gpg_decrypt = %(gpg_command)s -d --verbose --no-use-agent --batch --yes
--passphrase-fd %(passphrase_fd)s -o %(output_file)s %(input_file)s
gpg_encrypt = %(gpg_command)s -c --verbose --no-use-agent --batch --yes
--passphrase-fd %(passphrase_fd)s -o %(output_file)s %(input_file)s
gpg_passphrase =
guess_mime_type = False
host_base = localhost:8773
host_bucket = localhost:8773
service_path = /services/Walrus
human_readable_sizes = False
preserve_attrs = True
proxy_host =
proxy_port = 0
recv_chunk = 4096
secret_key = 10-7I4FPGYb_6CXzBcJYFa0gUnidD-EdG5_q-A
send_chunk = 4096
use_https = False
verbosity = WARNING
```

[s3cfg.walrus 内]

【参考② LinuxとEucalyptusの仮想化技術対応状況】

現在、Eucalyptus 1.5 が対応し、パッケージを配布している Distribution は以下の通り。

- Cent OS 5.3
- Open SUSE 11
- Debian Lenny 5.0
- Debian Squeeze/sid
- Ubuntu Jaunty 9.04

このうち、Cent OS 5.3 と Ubuntu Jaunty 9.04 について、仮想化のソフトウェアの対応状況を以下に示す。

・Cent OS 5.3

Cent OS 5.3 は、Xen に対応している。また、CPU が仮想化支援機能に対応している場合、カーネルを Linux Kernel 2.6.20 以降に更新する事で KVM に対応できる

Eucalyptus 上では、Xen による仮想化にのみ対応している。kernel のバージョンや CPU の機能に関わらず、Eucalyptus 上で KVM を用いた仮想化は**できない**。

・Ubuntu Jaunty 9.04

Ubuntu Jaunty 9.04 は、KVM による仮想化のみ対応している。Ubuntu 8.10 以降 Xen には対応していない。

Eucalyptus 上では、KVM による仮想化のみに対応している。Xen による仮想化には対応していない。

【参考③　KVM と Xen について】

ハイパーバイザ型の仮想化では、PC のリソースを分割管理する VMM (Virtual Machine Monitor) と呼ばれるソフトウェアの上に仮想マシンを立ち上げる。Xen と KVM はどちらもこの VMM に含まれる。本項では この２つの仮想化ソフトウェアの違いについて述べる。

・Xen について

Xen の仮想化のコンセプトは準仮想化である。Xen は ホスト OS の kernel に対して変更を加え、Xen Hypervisor の上に複数の kernel を構築する。このため、仮想化する OS の kernel は全て Xen 用に修正される必要がある。

・KVM について

KVM は CPU の仮想化支援機能を利用し、ハードウェアを含む完全仮想化を行う。CPU がハードウェアの仮想化を行う事で、仮想化のオーバーヘッドを抑え、高速な完全仮想化を実現する。CPU に Intel VT-x や AMD-V などの仮想化支援機構が必要となる。

1.5. Eucalyptus 利用手順

・インスタンスの起動

Eucalyptus を利用する際は、iptables の設定に注意が必要であり、SELinux は止めておく。

SSH で<front-end-ip-address>にログインする。

利用可能なインスタンス数・タイプを確認する。

```
# ec2-describe-availability-zones verbose
```

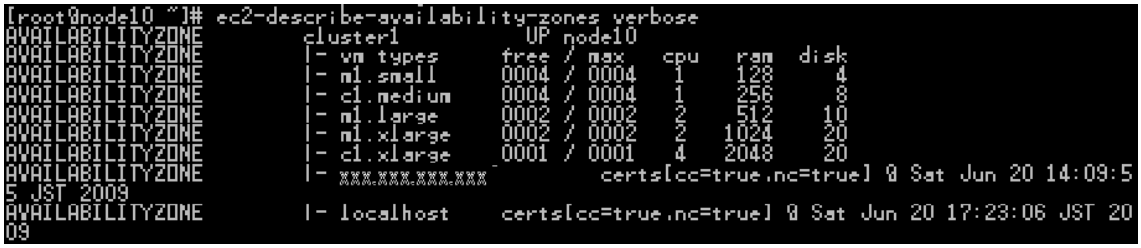


図 11 利用可能なインスタンス

vm types・・・設定したスペックタイプ

free / max・・・起動可能なインスタンス数 / 最大インスタンス数

cpu、ram、disk・・・スペック

登録されているイメージファイルを確認する。

```
# ec2-describe-images
```

図 12 登録済みのイメージ

秘密鍵を用いてインスタンスを起動する。

```
# ec2-run-instances [EMI ID]  -k [秘密鍵]
```

-t オプションでスペックタイプを選択可能

インスタンスの起動状態を確認する。

```
# ec2-describe-instances
```

起動が成功すると状態が pending→running となる。
他にも、インスタンスの ID やスペックタイプ・IP アドレス等が確認可能で、
利用しているラムディスクやカーネルイメージも確認可能である。

・インスタンスの停止

インスタンス ID を指定し、起動中のインスタンスを停止させる。

```
# ec2-terminate-instances  [インスタンス ID]
```

・EMI の登録

EMI をアップロードする。

```
# ec2-upload-bundle -bucket <バケットパス> --mainfest image.mainfest.xml
```

アップロードした EMI を登録する。

```
# ec2-regster <バケットパス> /image.mainfest.xml
```

・EMI の削除

EMI の登録を解除する。

```
# ec2-deregister [EMI ID]
```

EMI を削除する。

```
# ec2-delete-bundle   -bucket <バケットパス> --mainfest image.mainfest.xml
```

1.6. Eucalyptus のログファイル

Eucalyptus のログファイルは下記の様に構成されている。

表 1．ログファイルの構成（ $EUCALYPTUS/var/log/eucalyptus ）

ログファイル	記載される内容
Axis2c.log	Axis2/C の例外処理
cc.log	Cluster Controller が実行したコマンド
cloud-debug.log	Cloud Controller の DEBUG レベルでの動作
cloud-error.log	Cloud Controller の ERROR レベルでの動作
cloud-output.log	Cloud Controller の入出力
euca_test_nc.log	node Controller の起動時の動作
jetty-request-yyyy_mm_dd.log	日付毎の Jetty の動作
nc.log	Node Controller が実行したコマンド
httpd_cc_error_log	外部から Cluster Controller への標準
httpd_nc_error_log	Node Controller 上の httpd の動作

ここからは、特に Eucalyptus を運用する上で重要な、Cloud Controller、Cluster Controller、Node Controller に関するものを中心に、ログファイルの説明を行う。

1.6.1. Cloud Controller に関するログファイル

Cloud Controller の動作は、以下の3つのログファイルに記される。

- cloud-debug.log
- cloud-error.log
- cloud-output.log

Cloud Controller は、Eucalyptus の管理ページの構築や、Walrus の管理、クライアントツール（ EC2 API Tools ）を用いた命令の処理などを行う。EC2 API のコマンドが反映されない場合や、例外処理が発生した場合にこれらのログファイルを参照する必要がある。

1.6.1.1. cloud-debug.log

Cloud Controller が行った処理に関する詳細が記される。

例) 構築可能なインスタンス数の取得

```
21:50:01[<ホスト名>] DEBUG Sending to http://local:8774/axis2/services/EucalyptusCC: DescribeResourcesType
21:50:01[<ホスト名>] DEBUG <DescribeResource sxmlns = "http://eucalyptus.ucsb.edu/"> <correlationId>
74f380aa-46ea-4d3e-a1bf-bec2c9801cdb</correlationId><userId>eucalyptus</userId><instanceTypes><name>
m1.small</name><memory>128</memory><cores>1</cores><disk>4</disk></instanceTypes><instanceTypes>
<name>c1.medium</name>...
21:50:01[<ホスト名>] DEBUG <n:DescribeResourcesResponse xmlns:n="http://eucalyptus.ucsb.edu/">
<n:correlationId>74f380aa-46ea-4d3e-a1bf-bec2c9801cdb</n:correlationId><n:userId>eucalyptus</n:userId><n
:return>true</n:return><n:resources><n:instanceType><n:name>m1.small</n:name><n:memory>128</n:mem
ory><n:cores>1</n:cores><n:disk>4</n:disk></n:instanceType><n:maxInstances>5</n:maxInstances><n:availa
bleInstances>4</n:availableInstances></n:resources><n:resources><n:instanceType><n:name>c1.medium</n:n
ame>...
```

上の例ではCloud Controller が利用可能なリソースの情報をCluster Controller に要求し、その応答をもとにインスタンスの最大量と現在構築可能な量を計算する。

1.6.1.2. cloud-error.log

Cloud Controller に発生した例外処理が記される。

1.6.1.3. cloud-output.log

Cloud Controller が外部に向けて送ったメッセージが記される。

1.6.2. Cluster Controller に関するログファイル

Cluster Controller の動作は、以下の2つのログファイルに記される。

- cc.log
- httpd_cc_error_log

Cluster Controller は、クラスターやその基盤となる仮想ネットワークの構築、管理、計算リソースの管理などを行う。ネットワークの接続が確立できない場合など、ネットワークが原因で Eucalyptus が終了した場合などにこれらのログファイルを参照する必要がある。

1.6.2.1. cc.log

Cluster Controller が実行した処理に関する詳細が記される。

例) Cluster 全体の利用可能な計算リソースの取得

```
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ] DescribeResources(): called 5
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ] refresh_resources(): called
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ] calling http://node1:8775/axis2/services/EucalyptusNC
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ]  time left for next op: 300
[Fri Jun 19 21:50:11 2009][004170][EUCAINFO ] node=node1 mem=1758/1630 disk=3752/3236 cores=1/0
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ] calling http://localhost:8775/axis2/services/EucalyptusNC
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ] time left for next op: 300
[Fri Jun 19 21:50:11 2009][004170][EUCAINFO ]  node=node0 mem=3806/3806 disk=3105/3105 cores=4/4
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ] refresh_resources(): done
[Fri Jun 19 21:50:11 2009][004170][EUCADEBUG ] DescribeResources(): done
```

上の例では Cloud Controller からの要求を受けて、各 Node Controller のリソースの状態を要求し、その結果からクラスタ全体のリソースの状態を返している。

cc.log 内の記述は、内容の重要度によって [FATAL]、[ERROR] 、[WARN]、[INFO]、[DEBUG] に分けられている. eucalyptus.conf の設定を変更する事で、ログファイルに記載される内容を設定する事が出来る。初期設定では全ての内容を記述する。

1.6.2.2. httpd_cc_error_log

Cluster Controller が外部から受けた命令の標準出力や標準エラー出力などが記録される。

1.6.3. Node Controller に関するログファイル

Node Controller の動作は、以下の3つのログファイルに記される。

- nc.log
- httpd_nc_error_log
- euca_test_nc.log

Node Controller は、インスタンスの実行や物理マシンのリソースの監視などを行う。インスタンスが起動後、直ぐに強制終了してしまうなど、インスタンスの動作に問題がある場合はこれらのログを参照する必要がある。

1.6.3.1. nc.log

Node Controller が実行した処理に関する詳細が記される。

例) Node Controller の計算リソースの取得

```
[Fri Jun 19 21:50:01 2009][007740][EUCAINFO  ] doDescribeResource() invoked
```

上の例では、Node Controller が Cluster Controller からの要求を受けて、計算リソースの状態を返している。

nc.log 内の記述も cc.log と同様に内容の重要度によって分けられており、設定を変更する事で記載される内容を選択する事が出来る。初期設定では全ての内容を記録する。

1.6.3.2. httpd_nc_error_log

Node Controller が外部から受けた命令の標準出力や標準エラー出力などが記録される。

1.6.3.3. euca_test_nc.log

Node Controller の起動時の処理結果が表示される。正常に動作が終了した場合は、仮想環境の最小限必要なメモリ値が記録される。例外が発生した場合は、例外処理の結果が記録される。

1.6.4. その他のログファイル

Eucalyptus では、コントローラ間での通信の処理や、管理ページの構築などに、オープンソースのソフトウェアを利用している。これらの処理のログは以下の 2 つのログファイルに記述されている。

1.6.4.1. Axis2c.log

Axis2/C に発生した例外処理が記される。

1.6.4.2. Jetty-request-yyyy_mm_dd.log

日付毎に Jetty の HTTP プロトコルでの入出力に関する動作が記される。このログファイルは Cloud Controller 上にのみ作られる。

1.7. eucalyptus.conf

eucalyptus.conf は、インストールディレクトリ内/etc/eucalyptus に存在し、Eucalyptus の基本設定、各コントローラの基本設定、ネットワークの基本設定を行うファイルである。

ここでは、各項目の説明と設定例を記載する。

1.7.1. 基本設定

基本設定の設定例を以下に示す。

設定例

```
EUCALYPTUS=”/opt/eucalyptus”
EUCA_USER=”root”
#DISABLE_EBS="Y"
ENABLE_WS_SECURITY=”Y”
LOGLEVEL="DEBUG"
```

EUCALYPTUS は、Eucalyptus のインストール先を設定する。

EUCA_USER はユーザ名を設定する。

#DISABLE_EBS は EBS の利用するか否か設定を行う。デフォルトではコメントアウトされており、利用する設定となっている。利用しない場合はコメントアウトを消去する。

ENABLE_WS_SECURITY は Eucalyptus コンポーネント間での WS-Security の設定を行う。有効にする場合は、Y を無効にする場合は、N を選択する。

LOGLEVEL はログファイルに記載される内容を設定する。

内容の重要度に応じて FATAL、ERROR、WARN、INFO、DEBUG の五段階で設定可能である。

1.7.2. クラウドコントローラの設定

クラウドコントローラの設定例を以下に示す。

設定例

```
CLOUD_PORT="8773"
CLOUD_SSL_PORT="8443"
```

クラウドコントローラのポートを設定する。クラウドコントローラのみ2つのポートを必要とし、ストレージである Walrus やコンポーネント間で通信する際に用いる CLOUD_PORT とブラウザ上で管理画面を提供する際に SSL 通信を行うポートである CLOUD_SSL_PORT の設定を行う。

1.7.3. クラスタコントローラの設定

クラスタコントローラの設定例を以下に示す。

設定例

```
CC_PORT="8774"
SCHEDPOLICY="ROUNDROBIN"
NODES="localhost"
NC_SERVICE="axis2/services/EucalyptusNC"
```

CC_PORT はクラスタコントローラのポートの設定を行う。

SCHEDPOLICY はインスタンスを起動するスケジューリングの設定を行う。
ROUNDROBIN と GREEDY の2種類から選択可能で、ROUNDROBIN は起動するマシンを順番に選択、GREEDY は起動可能な中で最適なマシンを選択する。

NODES はクラスタに登録されているノードコントローラのアドレスの設定を行う。

NC_SERVICE はノードコントローラのパスの設定を行い、プラグインを追加した独自のノードコントローラのパスを設定し、利用可能である。

1.7.4. ノードコントローラの設定

ノードコントローラ設定例を以下に示す。

設定例

```
NC_PORT="8775"
HYPERVISOR="xen"
MAX_MEM=2048
MAX_CORES="2"
SWAP_SIZE=512
MANUAL_INSTANCES_CLEANUP=0
INSTANCE_PATH="/opt/eucalyptus/var/lib/eucalyptus/instances"
NC_CACHE_SIZE=99999
```

NC_PORT はノードコントローラのポートを設定する。

HYPERVISOR は、仮想マシンを制御するためのハイパーバイザを設定する。
現在のバージョンでは Xen もしくは KVM から選択可能である。

MAX_MEM 及び MAX_CORES は、Eucalyptus で使用可能な最大メモリ容量(MB)と CPU の最大コア数を設定する。コメントアウトした場合は使用可能なリソースを全て使用する。

SWAP_SIZE はスワップパーティション(MB)を設定する。

MANUAL_INSTANCES_CLEANUP は、終了したインスタンスのファイルクリーンアップを手動で行うか自動で行うかを設定する。手動で行う場合は 1、自動で行う場合は 0 を選択する。自動で行う事が推奨されており、手動はインスタンス起動の際のデバック等での利用に有効である。

INSTANCE_PATH は、インスタンスのイメージをコピーするパスを設定する。起動しているイメージはインスタンスが停止すると全て消去されるが、キャッシュにコピーされたイメージは残る。キャッシュは LRU により管理されている。

NC_CACHE_SIZE は、キャッシュのサイズ(MB)を設定する。キャッシュのパスは $INSTANCE_PATH/eucalyptus/cache である。

1.7.5. ネットワークの設定

ネットワークのモードの設定や DHCP 等ネットワークの基本設定を行う。
詳細な内容や設定例は、1.3 項で先述した通りである。

1.8. イメージファイルの作成

1.8.1. Web 上からダウンロードする

Xen で動作する Linux の VM イメージは，インターネット上から入手することが可能である。Eucalyptus を試しに使ってみる場合など，最も手軽に環境を構築することができる。

例えば、http://jailtime.org/ では、下記の Linux ディストリビューションの VM イメージをダウンロードすることが可能である。

- CentOS 5.2
- CentOS 5.2 64-bit
- Debian 4.0
- Debian4.0 64-bit
- Fedora 9
- Fedora 9 64-bit
- Gentoo 2008.0
- Gentoo 2008.0 64-bit
- Slackware 12.1
- Ubuntu 8.04 (Hardy)
- Ubuntu 8.04 64-bit (Hardy)

ダウンロードしたファイルには、Xen の設定ファイルが含まれている。下記に示すのは Fedora 9 の設定ファイルである。

```
kernel = "/boot/vmlinuz-2.6-xen"
memory = 428
name = "fedora.fc9"
vif = [ '' ]
dhcp = "dhcp"
disk = ['file:/xen/fedora/fedora.fc9.img,sda1,w']
root = "/dev/sda1 ro"
```

設定ファイルは、そのままでは使用できないため、kernel, disk について書き換える必要があり、また ramdisk を加える必要がある。設定を変更した例を以下に示す。http://jailtime.org/download:start などに使い方が記載されている

```
kernel = "/root/xen_kernel/vmlinuz-2.6.18-128.el5xen"
ramdisk = "/root/xen_ramdisk/initrd-2.6.18-128.el5xen-no-scsi.img"
memory = 428
name = "fedora.fc9"
vif = [ '' ]
dhcp = "dhcp"
disk = ['file:/root/xen_image/fedora/fedora.fc9.img,sda1,w']
root = "/dev/sda1 ro"
```

1.8.2. 新しく OS をインストールして作成する

Xen の VM イメージを作成する方法はいくつかあるが、ここでは virt-install を使用する方法と、QEMU によりイメージを作成し、KVM で CD ブートして作成する方法について説明する。

① virt-install を使う方法

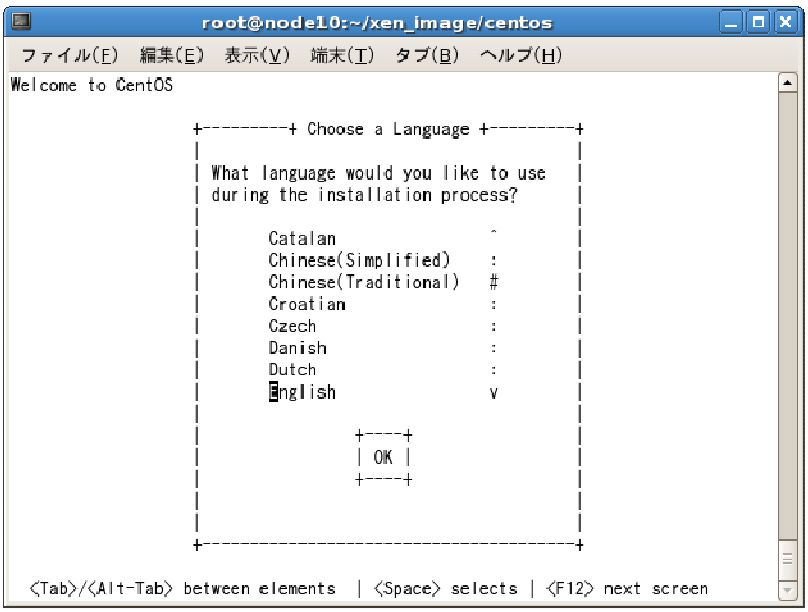
今回調査環境で用いた CentOS 上での VM 作成では、virt-install を用いた作成が簡単である。virt-install の実行例を次に示す.

```
# virt-install -n test -r 512 -f /xen/test -s 16 --nographics -l
ftp://ftp.riken.go.jp/Linux/centos/5.0/os/i386
```

-n VM イメージにつける名前
-r VM イメージに割り当てるメモリの容量
-f VM イメージを保存するファイル
-s VM イメージのディスクサイズ
-l インストール元の URL
--nographics コンソール上でインストールする場合はこのオプションを付ける

実行すると,「完全仮想化」のイメージを作成するか(yes)、「準仮想化」のイメージを作成するか(no)訪ねられるので、Eucalyptus で用いる場合は yes を入力する。

以下はインストールが開始された画面



インストールは通常と同じように行う。インストールが終了すると自動的に VM イメージが起動する。

② QEMU で仮想ディスクを作成し、KVM で CD ブートしてインストールする方法

もう一つの方法として、QEMU で仮想ハードディスクを作成し、OS のインストールディスクから CD ブートさせて VM を作成する方法を説明する。

まず、下記のコマンドで仮想ハードディスクを作成する

```
$ qemu-img create -f qcow image.img 4G
```

次に作成したイメージと、インストールディスクを使用して KVM を起動し、OS をインストールする。

```
$ sudo kvm -no-acpi -m 512 -cdrom /dev/cdrom –had image.img -boot d
```

OS のインストール手順は通常と同じである。

1.8.3. 既存のコンピュータからイメージを抽出して作成する

既存の物理的なコンピュータ上で動作するシステムを、そのまま VM イメージとして取り込む方法

について説明する。物理的なコンピュータから VM イメージを作成することを P2V (physical to virtual)と呼ぶこともある。

ここでは、Virt-P2V と呼ばれるツールを用いて既存のコンピュータから VM イメージを作成する方法について説明する。Virt-P2V は CD ブートで起動し、対象マシンのハードディスクからデータを吸い上げ、作成したイメージファイルをSSH 経由で他のマシンに転送する。複数のハイパーバイザに対応したイメージを作成できるという特徴がある。

1.8.3.1. 事前準備

・Virt-P2V のダウンロード

URL: http://et.redhat.com/~rjones/virt-p2v/download.html

上記 URL より、Virt-P2V の ISO イメージをダウンロードし、CD-ROM に焼く。

・転送先サーバの準備

SSH 経由で仮想マシンのイメージを転送する為、SSH でアクセスできる環境を用意する。

また、イメージは対象マシンのハードディスクの容量となる為、必要な容量を確保する。

・ルートパーティションの容量確認

ec2-tool の制約で、10GB を超えるイメージファイルは Eucalyptus にアップロードすることができない。そのため、予め転送元となるマシンのルートパーティションを 10GB より小さくしておく必要がある（余裕を持って 9GB 程度にしておくほうが好ましい）パーティションサイズの変更は、Virt-p2v によるイメージ化の後でも行えるが、ディスク使用量などは予め確認しておくこと。

1.8.3.2. Virt-P2V の起動

対象マシンを Virt-P2V の ISO イメージを焼いた CD-ROM より起動する。

図 13 起動中画面

Virt-P2V の説明を読み、OK を選択する。

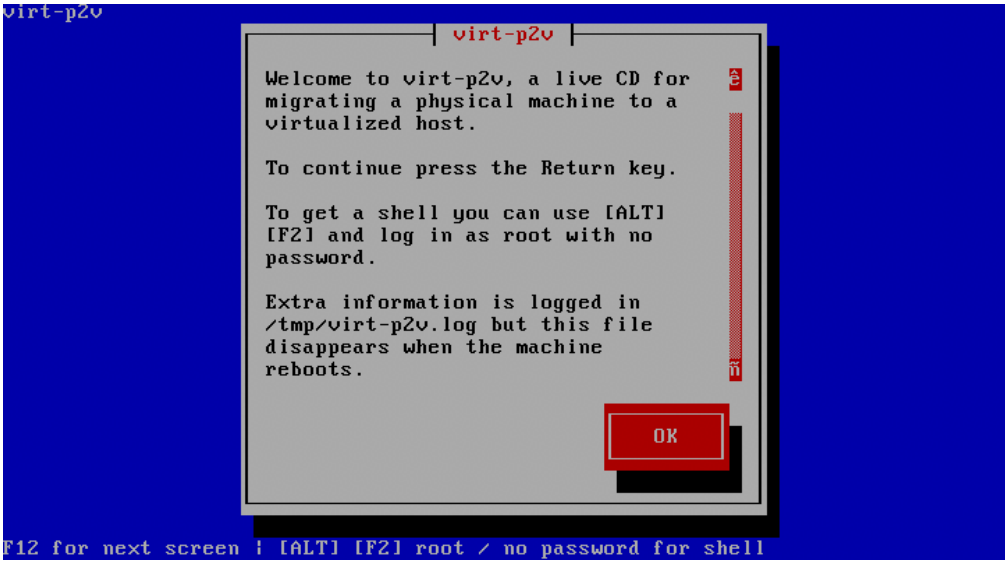


図 14 説明画面

転送タイプを選択する。

Physical to Virtual(P2V)にチェックを入れ、OK を選択する。

Physical to Virtual(P2V)・・・物理マシンから仮想マシンへの転送

Virtual to Virtual(V2V)・・・仮想マシンから仮想マシンへの転送

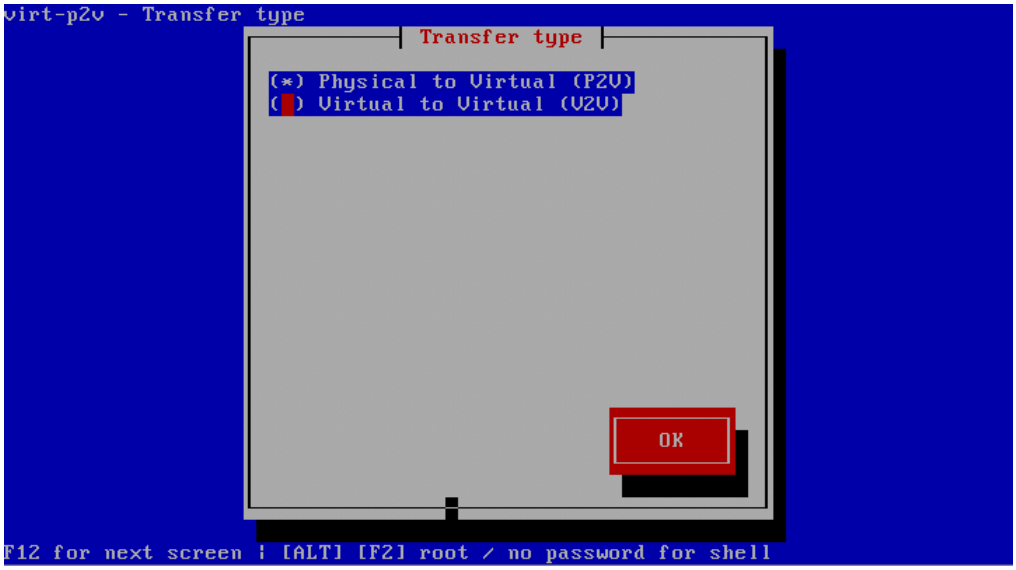


図 15 転送タイプの選択画面

ネットワークの設定を行う。

Static configuration を選択し、対象マシンのネットワーク設定を入力し OK を選択する。

Automatic from を選択すると、対象マシンの設定を利用する。

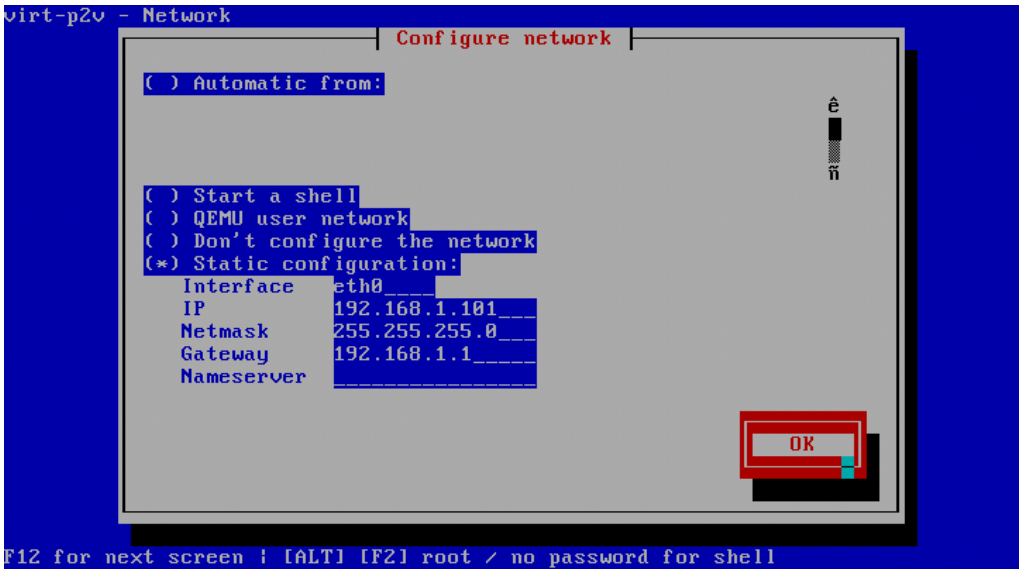


図 16 ネットワーク設定

SSH の設定を行う。転送先のホスト名・ディレクトリパス・ユーザ名等を設定する。

Test SSH connection にチェックを入れ、OK を選択する。

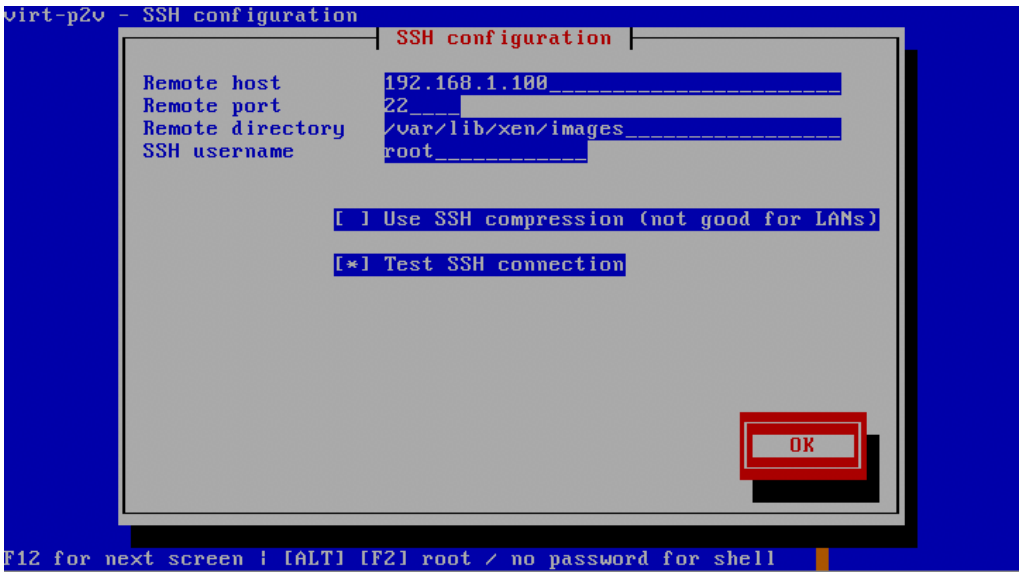


図 17 SSH 設定

SSH でのログインが可能である事を確認する。

接続するか確認されるので yes と入力し、パスワードを入力する。

SSH が動作している事を確認しyを入力する。

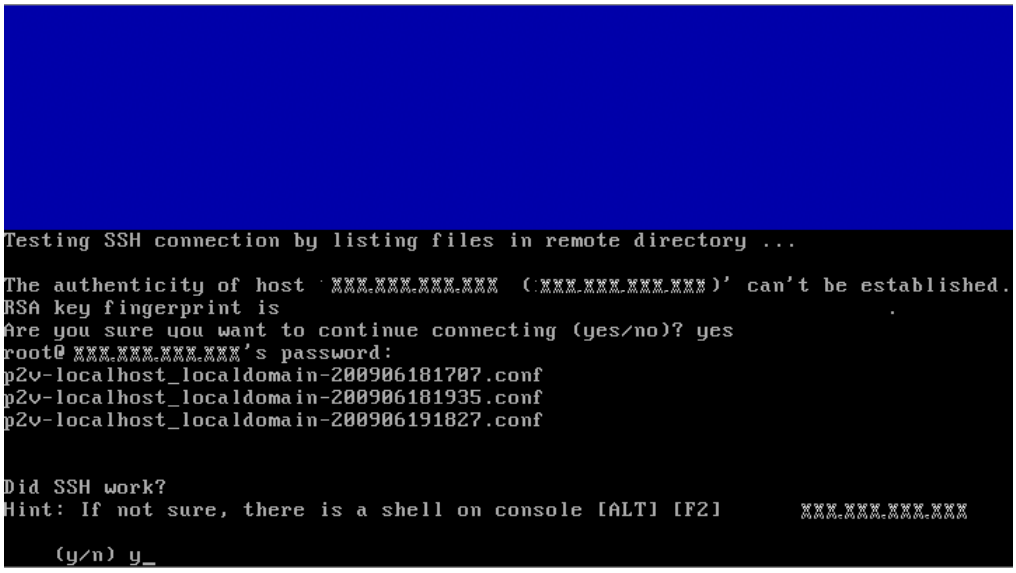


図 18 SSH ログイン確認画面

イメージ化する対象ディスクを選択し、OK を選択する。

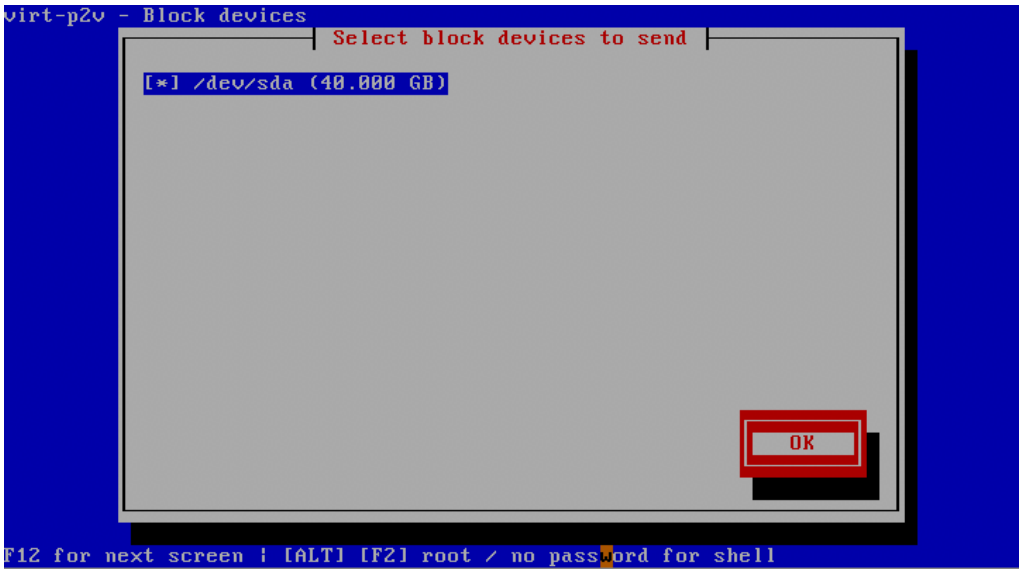


図 19 ディスクの選択画面

ルートファイルシステムがあるパーティションを選択し、OK を選択する。

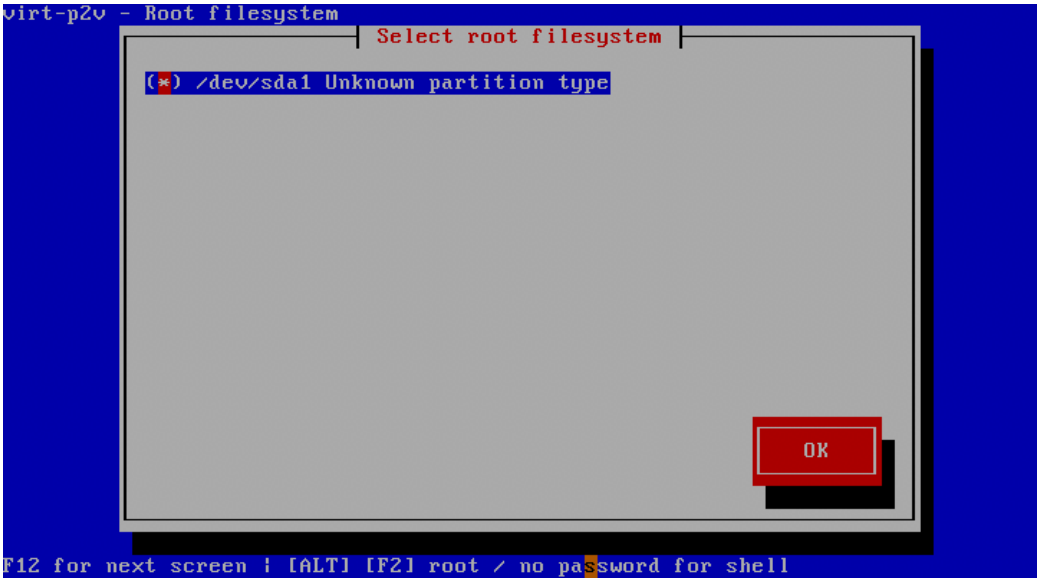


図 20 パーティションの選択画面

仮想マシン環境の設定を行う。

ハイパーバイザやアーキテクチャ・メモリや CPU 数を入力し OK を選択する。

ハイパーバイザは Xen、QEMU、KVM が選択可能である。

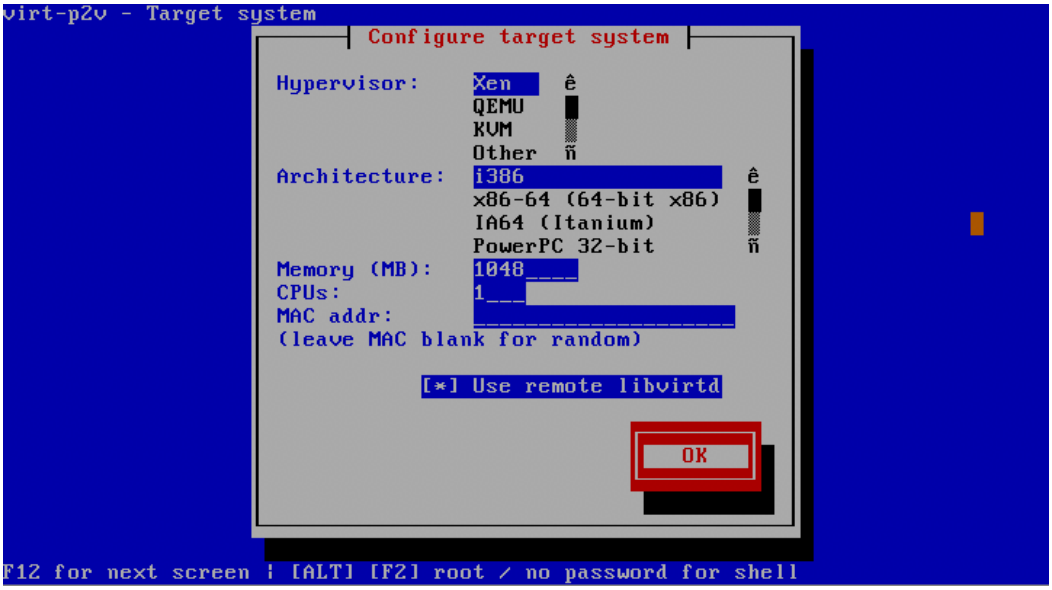


図 21 仮想マシン環境の設定画面

選択後、イメージファイル吸い出され、SSH での転送が開始される。転送が終了すると、仮想イメージファイルと、起動用の設定ファイルが作成されている。virtp2v によって作成される KVM 用の設定ファイルの例を以下に示す。

```
<!--
   This is an automatically generated libvirt configuration file.
   It was written by the virt-p2v program.

   Please check the values in this configuration file carefully,
   particularly maxmem, memory, vcpu and any paths.

   To start the domain, do:
      virsh -c qemu:///system define p2v-localhost_localdomain-200906201607.conf
      virsh -c qemu:///system start localhost_localdomain
-->

<domain type="kvm">
   <name>localhost_localdomain</name>
   <uuid>857b6eae6ab7deb59cc512a8fc3900b2</uuid>
   <memory>522240</memory>
   <maxmem>522240</maxmem>
   <vcpu>1</vcpu>
   <os>
      <type>hvm</type>
   </os>
   <clock sync="localtime"/>
   <devices>
      <emulator>/usr/bin/kvm</emulator>
      <interface type="user">
         <mac address="00:16:3e:63:35:bb"/>
      </interface>
      <graphics type="vnc"/>
      <disk type="file" device="disk">
         <source
file="/home/user/kvm_image/p2v-localhost_localdomain-200906201607-hda.img"/>
         <target dev="hda"/>
      </disk>
   </devices>
</domain>
```

1.8.3.3. イメージファイルの縮小

転送されるイメージファイルは、物理的なハードディスクと同じ容量になる。しかし、ec2-tools では容量が 10GB 以上のイメージは転送することができない。そのため、Virt-P2V によって抽出したイメージファイルが 10GB を超えている場合は、10GB 以下に縮める必要がある。

Virt-P2V で抽出したイメージファイルをここでは便宜上 org.img、作成する 10GB 以下のイメージファイルを new.img と呼ぶ。まず、コピー先となる 10GB 以下のイメージファイル（new.img）を予め空の状態で作成する。

```
# dd if=/dev/zero of=/home/user/new.img bs=1024k count=1 seek=102400
```

Virt-P2V が作成した起動用の設定ファイルに下記を加えて、作成したディスク領域をマウント可能にする。

```
<disk type="file" device="disk">
  <source file="/home/user/new.img"/>
  <target dev="hdb"/>
</disk>
```

以下のコマンドで仮想マシンを起動する。

```
# virsh create [virsh-p2v で作成したイメージの設定ファイル]
```

デスクトップから Virtual Machine Manager を起動し、localhost_localdomain として起動しているのを確認する。ダブルクリックで仮想マシンの画面を表示し、仮想マシンに root でログインする。この作業はコンソール上からでも可能である。

以下の作業は仮想マシン上で行う。もし、仮想化したいルートパーティションが 10GB（おそらく 10G ぎりぎりよりは 9G 程度が好ましい）を超えている場合はパーティションを小さくする。例えば下記のようなコマンドでパーティションを小さくすることができる。

```
# e2fsck -f /dev/hda
# resize2fs -p /dev/hda 9000M
```

その後、hda の領域を hdb にコピーする

```
# dd if=/dev/hda of=/dev/hdb
```

最後に、仮想マシンをシャットダウンし、ホストマシン上で Virt-P2V で作成した設定ファイルを new.img のみをマウントするように修正し、起動することを確認する。

【補足】

今回の調査では、上記の方法で作成したイメージファイルが Eucalyptus 上で動作することを確認できた。しかし、ネットワークの設定に関しては未調査の部分が残っている。今回の調査は、Ubuntu 9.04 上で KVM を設定し、Eucalyptus のネットワーク設定は SYSTEM モードとし、外部 DHCP によって IP アドレスを割り当てる方法で動作確認を行った。イメージをアップロードする作業までは問題なく行えるものの、起動時に IP が割り当てられないという現象が頻発し、この問題については未だ原因が不明である。(IP アドレスが、起動後も 0.0.0.0 のまま変わらない現象が起こる)ただし、そのような現象が起こる設定から変更を加えていないにもかかわらず IP が DHCP サーバより割り当てられることもあった。その際には、一度 IP アドレスが割り当てられれば、SSH などによって仮想マシンにログインすることも可能である。このことから、物理マシンからのイメージ作成方法については問題なく行えるものの、ネットワークの設定については再調査が必要である。

2. Amazon EC2（Amazon Elastic Compute Cloud）

2.1. 概要

Amazon EC2 は、Amazon が所有するコンピュータリソース上に仮想のコンピュータを構築するクラウドシステムであり、ユーザには Web サービスとして仮想のコンピュータを操作するインタフェースを提供するものである。起動する仮想マシンのイメージは、AMI（Amazon Machine Image）という独自の形式で、あらかじめ Amazon によって OS の種類やインストールされているアプリケーションの組み合わせなど複数用意され、またユーザが独自に構築することも可能である。ユーザは、まず Amazon EC2 上にリソースとして仮想マシン（インスタンス）を用意し、起動時に AMI を指定する事で、マシンのイメージを起動し、API を通じて操作できる。また、起動時に割り当てられる IP によって SSH などでインスタンスと通信することも可能である。

ユーザが Amazon EC2 上に用意する仮想マシンのスペックは Amazon によりあらかじめ用意された以下の5つの「タイプ」から選択する。5つのタイプは、それぞれ CPU のコア数やメモリ及びストレージの容量などが異なっており、また各タイプによって料金体系も異なる。

また、20インスタンス以上利用する際には、事前申請が必要となる。

表 2 インスタンスのスペック一覧

種類	CPU	メモリ	ストレージ	プラットフォーム
Small (Default)	1 ECU (1core * 1ECU)	1.7GB	160GB	32-bit
Large	4 ECU (2core * 2ECU)	7.5GB	850GB	64-bit
Extra Large	8 ECU (4core * 2ECU)	15GB	1690GB	64-bit
High-CPU Medium	5 ECU (2core * 2.5ECU)	1.7GB	350GB	32-bit
High-CPU Extra Large	20 ECU (8core * 2.5ECU)	7GB	1690GB	64-bit

1 ECU は 1.0-1.2 GHz に相当する仮想 CPU(2007 Opteron または 2007 Xeon processor)

ここでのストレージは、インスタンスからアクセス可能な一時的なもので、インスタンスを停止するとストレージも消滅し内容は失われる。インスタンスを終了後もデータを保存するには、Amazon EC2と連携可能なストレージシステムであるElastic Block Device (EBS)にデータを保存するか、データもしくはイメージ自体をSimple Storage Service(S3)へ保存する必要がある。

EBS は直接ボリュームをフォーマットしてマウントし、利用する。1 つの EBS ボリュームあたり、1TBまで容量を大きく出来る。S3は容量に上限は無く、HTTP経由でアクセス可能なストレージサービスである。 EBSボリュームのスナップショットをS3 上に取るAPIも準備されている。

インスタンスには、ひとつのNIC(Network Interface Card)が仮想的に装備されている。インスタンスを起動した際に、Amazon EC2 内で有効なプライベートIPアドレスが割り当てられ、同時に1 対 1 で対応するパブリックな IP アドレスと DNS 名も割り当てられる。プライベート IP アドレスとパブリック IP アドレス及び、DNS 名はインスタンスが動作している間のみ、有効であり、インスタンスを再起動すると異なるIPアドレスが割り当てられる。

Elastic IP Addressという追加サービスによって、ユーザは固定のIPアドレスを取得し、インスタンスと関連付ける事で、インスタンスを再起動しても不変のIPアドレスを利用する事が可能である。5つ以上のIPアドレスを取得する際には、事前申請が必要となる。

インスタンスの稼動状況をモニタリングし、負荷状況に応じて動的にインスタンス数を増減させるAuto Scaling や、フロントエンドに立ち状況に合わせて、トラフィックをインスタンスに分散されるElastic Load Balancingという追加サービスがあり、負荷状況に応じた適切なシステム構築が可能となる。

2.2. 特徴

Amazon EC2は次の特徴を備える。

- スケーラビリティ
  スケールアウト・スケールアップ(ダウン)を負荷状況に応じて柔軟に変更可能
- ファシリティ
  サーバ・ストレージの場所が複数箇所から選択可能
- 信頼性
  稼働率99.95%のサービスレベル合意(SLA)を導入

2.3. 料金体系

2.3.1. 基本料金

Amazon EC2 の料金体系は、従量課金制となっており、起動時間と転送量により課金される。起動時間による課金は、インスタンスのスペックにより異なる。事前購入による割引制度も用意されている。Amazon EC2 の基本料金について以下に記載する。

表 3 起動時間による課金

種類	Linux/UNIX	Windows
Small (Default)	$0.1/時	$0.125/時
Large	$0.4/時	$0.5/時
Extra Large	$0.8/時	$1.0/時
High-CPU Medium	$0.2/時	$0.3/時
High-CPU Extra Large	$0.8/時	$1.2/時

表 4 転送量による課金

アップロード		$0.10/GB
ダウンロード (1 ヵ月の転送量)	0～10TB	$0.17/GB
	10～50TB	$0.13/GB
	50～150TB	$0.11/GB
	150TB～	$0.10/GB

表 5 EC2 内の通信による課金

Private IP による通信	同じ Avalability Zone 内	無料
	同じ Region の異なる Availability Zone 間	$0.01/GB
Public IP や Elastic IP による通信		$0.01/GB

2.3.2. 追加料金

利用する機能によって別途課金される追加料金について以下に記載する。

表 6 Elastic IP による課金

使用中		無料
未使用時の維持費(1IP)		$0.01/時
アドレスの変更	100 回/月	無料
	101～回/月	$0.10/回

表 7 Amazon EBS による課金

使用量(1 ヶ月あたり)	$0.10 / GB
I/O リクエスト	$0.10 / 1,000,000 回
S3 への Snap Shot	S3 の利用料金

表8 Amazon CloudWatchによる課金

使用量(1 インスタンス)	$0.015 / 時

表 9 Elastic Load Balancing による課金

使用料金	$0.025 / 時
データ処理料金	$0.008 / GB

表 10 Amazon S3 による課金

		アメリカ	ヨーロッパ
ストレージ使用量 (1 ヶ月あたり)	0～50TB	$0.15/GB	$0.18/GB
	50～100TB	$0.14/GB	$0.17/GB
	100～500TB	$0.13/GB	$0.16/GB
	500TB～	$0.12/GB	$0.15/GB
PUT,POST,LIST リクエスト(1000 回あたり)		$0.01	$0.012
GET その他のリクエスト(1000 回あたり)		$0.01	$0.012

別途転送量による課金も行われる

2.4. Amazon EC2 利用手順

本項では、Amazon EC2 の利用手順について記載する。

2.4.1. 事前準備

2.4.1.1. アカウントのセットアップ

・Amazon Web Services(AWS)のアカウントを取得

URL:*http://aws.amazon.com/*

URL 先より AWS のアカウントを取得し、「Your Account」→「Access Identifiers」内のアカウントナンバー, アクセスキー及びシークレットアクセスキーを記録する。

図 22　アクセスキー

・X.509 証明書及び秘密鍵ファイルを作成

証明書ファイル(cert-******.pem)及び秘密鍵(pk-******.pem)を保存する。

図 23　証明書ファイル

2.4.1.2. Amazon EC2 Command-Line Tools のセットアップ

・Amazon EC2 Command-Line Tools のダウンロード

解凍したファルダを適当な箇所(<ディレクトリパス>)に配置する。

・初期設定

コマンドプロンプトを起動し, 以下のコマンドを入力する。

ディレクトリパスを設定し、PATH を追加する。

```
C:/>set EC2_HOME=<ディレクトリパス>/ec2
C:/>set PATH=%PATH%;%EC2_HOME%/bin
```

先述の X.509 証明書ファイル及び秘密鍵のパスを設定する。(ディレクトリパス内に配置)

```
C:/>set EC2_PRIVATE_KEY=%EC2_HOME%//pk-******.pem
C:/>set EC2_CERT=%EC2_HOME%//cert-******.pem
```

JAVA のパスを設定する。

```
C:/>set JAVA_HOME=C:/"Program Files"/Java/jre1.6.0_07
```

2.4.1.3. Elasticfox のセットアップ(EC2 用の GUI)

・Firefox のプラグイン「Elasticfox Firefox Extension for Amazon EC2」をインストール
Firefox のメニューのツールより「Elasticfox」を起動し、「Credentials」を選択し、アカウント名、アクセスキー、シークレットアクセスキーを入力する。

2.4.1.4. SSH アクセスの準備(Elasticfox を用いた手順)

・秘密鍵の発行
①「Key Pairs」②「Create a new keypair」より、秘密鍵を発行する。
秘密鍵の適当な name を入力し、name.pem を保存する。

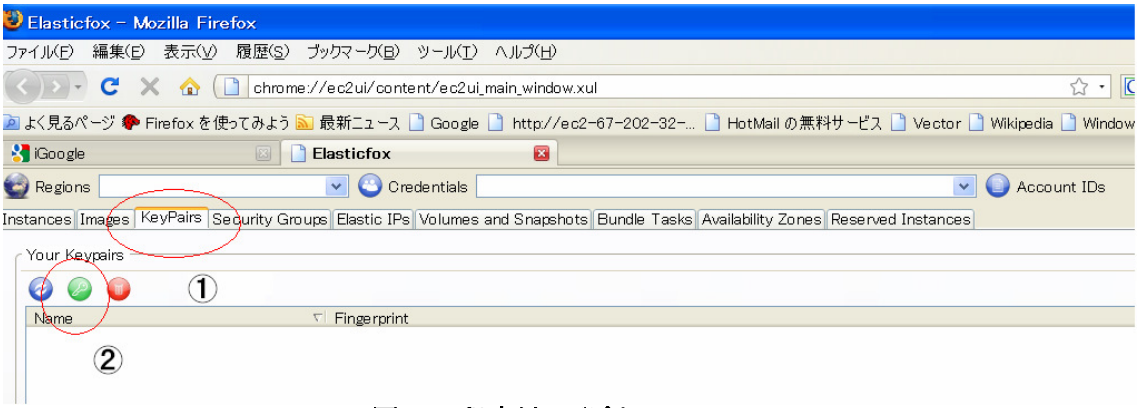


図 24 秘密鍵の発行

PuTTY を使う場合、作った秘密鍵を PuTTY 形式に変換する。(PuTTYgen を利用)

2.4.1.5. S3 Organizer のセットアップ（S3 用の GUI）

・Firefox のプラグイン「S3 Firefox Organizer」をインストール

Firefox のメニューのツールより「S3 Organizer」を起動し、「Manage Account」を選択し、アカウント名、アクセスキー、シークレットアクセスキーを入力する。

2.4.2. 利用手順

・インスタンスの起動

起動するインスタンスの AMI を選択し、「Launch Instance」を選択する。

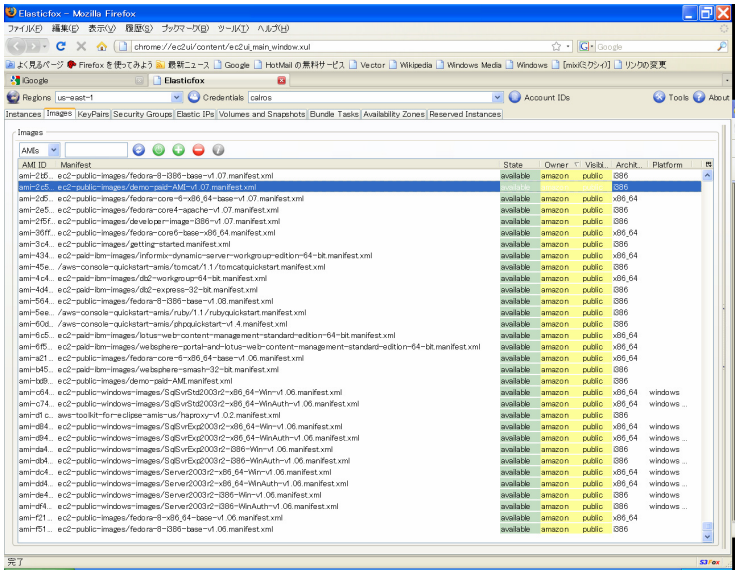


図 25　AMI 一覧

スペックやインスタンス数を選択し、作成した秘密鍵を選択する。

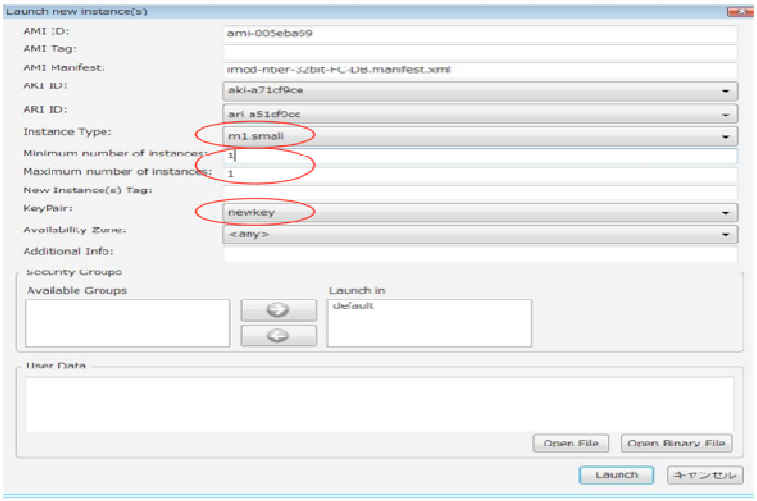


図 26 インスタンスの設定

「Instances」を選択し，インスタンスが起動した事を確認する。(starting →running)

Public DNS がグローバルなアドレスで、アドレス内に含まれる数列がグローバル IP である。

「Security Groups」を選択し，ポートを設定する。

・SSHによるアクセス

Putty を起動する。

ホスト名に起動したインスタンスの Public DNS を入力する。

メニュー内「SSH」→「認証」 で作成済みの name.ppk のパスを入力する。

・AMI の作成

X.509 証明書及び秘密鍵ファイルの転送する。

インスタンスのディレクトリ「/mnt」に証明書ファイル(cert-******.pem)

及び秘密鍵(pk-******.pem) を転送する。(winSCP を利用)

インスタンス内の「/mnt」に AMI を作成する。(SSH を利用)

```
#ec2-bundle-vol -d /mnt --privatekey /mnt/pk-******.pem
--cert /mnt/cert-******.pem --user <アカウントナンバー>
```

<アカウントナンバー>: AWS 登録時に割り振られた 12 桁の数字

作成したAMIをS3にアップロードする。(SSHを利用)

```
#ec2-upload-bundle --bucket <バケットパス>
--manifest image.manifest.xml
```

<バケットパス>:「S3 Organizer」で予め作成しておいたAMI保存用のディレクトリのパス

S3にアップロードしたAMIを登録する。(コマンドプロンプトを利用)

```
C:/>ec2-register <バケットパス>/image.manifest.xml
```

・インスタンスのモニタリング

Amazon CloudWatch API Toolsをダウンロードする。

ディレクトリのパスを設定し、PATHへ追加する。

```
C:/>set AWS_CLOUDWATCH_HOME=<ディレクトリパス>
C:/>set PATH=%AWS_CLOUDWATCH_HOME%/bin
```

指定したインスタンスのモニタリングを開始する。

```
C:/>ec2-monitor-instances [インスタンスID]
```

モニタリングの状態を確認する。

```
C:/>mon-get-stats  [パラメータ]
```

・オートスケーリング

Auto Scaling API Toolsをダウンロードする。

ディレクトリのパスを設定し、PATHへ追加する。

```
C:/>set AWS_AUTO_SCALING_HOME=<ディレクトリパス>
C:/>set PATH=%AWS_AUTO_SCALING_HOME%/bin
```

スケールアウトするAMIとインスタンスのタイプを設定する。

```
C:/>as-create-launch-config [設定名] --image-id [AMI_ID]
--instance-type [インスタンスタイプ]
```

グループの構成を設定する。

```
C:/>as-create-auto-scaling-group [グループ名]
--launch-configuration [設定名] --availability-zones [サーバエリア]
--min-size [最小インスタンス数] --max-size [最大インスタンス数]
```

オートスケーリングの閾値を設定する。

```
C:/>as-create-or-update-trigger  [トリガー名]
```

オートスケーリングを停止する。

```
C:/>as-terminate-instance-in-auto-scaling-group [インスタンス　ID]
```

・ロードバランサ

Elastic Load Balancing API Tools をダウンロードする。

ディレクトリのパスを設定し、PATH へ追加する。

```
C:/>set AWS_ELB_HOME=<ディレクトリパス>
C:/>set PATH=%AWS_ELB_HOME%bin
```

ロードバランサを作成する。

```
C:/>elb-create-lb [ロードバランサ名] --availability-zones [サーバエリア]
--listener "protocol=HTTP, lb-port=80, instance-port=80"
```

"--listener"は、プロトコルの指定、ロードバランサのポート番号、
振り分け先のバックエンドのポート番号を指定可能

ロードバランサに接続するインスタンスを設定する。

```
C:/>elb-register-instances-with-lb [ロードバランサ名]
--instances [インスタンス ID1][インスタンス ID2]…
```

ロードバランサを削除する。

```
C:/>elb-delete-lb [ロードバランサ名]
```

ヘルスチェックの設定

```
C:/>elb-configure-healthcheck [ロードバランサ名]
```

2.5. 機能一覧

Amazon EC2 での機能の一覧と Eucalyptus との対応を以下の表に示す。

表 11 機能一覧

		Eucalyptus での実装
基本機能	インスタンスの起動	○
基本機能	インスタンスの停止	○
基本機能	インスタンスの再起動	○
基本機能	インスタンスの起動状態の確認	○
基本機能	秘密鍵の発行	○
基本機能	秘密鍵の削除	○
基本機能	セキュリティグループの作成	○※1
基本機能	AMI の作成	○
基本機能	AMI の削除	○
基本機能	AMI のアップロード	○
基本機能	AMI の登録	○
基本機能	AMI の登録解除	○
基本機能	AMI の一覧の確認	○
EBS	ボリュームの作成	○
EBS	ボリュームの削除	○
EBS	ボリュームのアタッチ	○
EBS	ボリュームのデタッチ	○
EBS	スナップショットの作成	○
EBS	スナップショットの削除	○
Elastic IP	固定 IP の設定	○※2
Elastic IP	固定 IP の解除	○※2
CloudWatch	インスタンス・ロードバランサのモニタリング	×
CloudWatch	稼動状況の確認	×
Auto Scaling	設定の作成	×
Auto Scaling	オートスケーリングの設定	×
Auto Scaling	閾値の設定	×
Auto Scaling	オートスケーリングの状態の確認	×
Auto Scaling	オートスケーリングの履歴の確認	×
Auto Scaling	オートスケーリングの設定の確認	×
Auto Scaling	オートスケーリングの解除	×

Auto Scaling	設定の削除	×
Auto Scaling	閾値の削除	×
Auto Scaling	グループの削除	×
ELB	ロードバランサの作成	×
ELB	ロードバランサの削除	×
ELB	接続するインスタンスの設定	×
ELB	バックエンドのヘルスチェック	×

※1 Eucalyptus では、ネットワークモードが MANAGED MODE の場合のみ、セキュリティグループが有効である。
※2 Eucalyptus では、ネットワークモードが STATIC MODE の場合のみ、固定 IP をもつ。

各機能の概要と対応する API について以下に記載する。

2.5.1. 基本機能

2.5.1.1. インスタンスの起動

機能概要

コンピュータリソース上でインスタンスを起動する。

対応する API

ec2-run-instances [AMI ID] –k [秘密鍵]

ec2run [AMI ID] –k [秘密鍵]

2.5.1.2. インスタンスの停止

機能概要

起動したインスタンスを停止する。

対応する API

ec2-terminate-instances [インスタンス ID]

ec2kill [インスタンス ID]

2.5.1.3. インスタンスの再起動

機能概要

起動しているインスタンスを再起動する。

対応する API

ec2-reboot-instances [インスタンス ID]

ec2reboot [インスタンス ID]

2.5.1.4. インスタンスの起動状態の確認

機能概要

インスタンスの起動状態やホスト名を確認する。

対応する API

ec2-describe-instances

2.5.1.5. 秘密鍵の発行

機能概要

SSH アクセス用の秘密鍵を発行する。

対応する API

ec2-add-keypair [秘密鍵名]

2.5.1.6. 秘密鍵の削除

機能概要

SSH アクセス用の秘密鍵を削除する

対応する API

ec2-delete-keypair [秘密鍵名]

2.5.1.7. セキュリティグループの作成

機能概要

EC2 のファイアウォール機能を設定するグループを作成し、ポートを開く。

対応する API

ec2-add-group [グループ名]

ec2-authorize [グループ名] -p [ポート番号]

2.5.1.8. AMI の作成

機能概要

イメージファイルを作成する。

対応する API

ec2-bundle-vol -d <作成するディレクトリパス> --privatekey <X.509 秘密鍵のパス>
--cert <X.509 証明書のパス> --user [アカウントナンバー]

2.5.1.9. AMI の削除

機能概要

イメージファイルを削除する。

対応する API

ec2-delete-bundle –-bucket <バケットパス> --manifest image.manifest.xml

2.5.1.10. AMI のアップロード

機能概要

作成したイメージファイルをストレージ上にアップロードする。

対応する API

ec2-upload-bundle --bucket <バケットパス> --manifest image.manifest.xml

2.5.1.11. AMI の登録

機能概要

作成したイメージファイルを AMI として登録する。

対応する API

ec2-register <バケットパス>/image.manifest.xml

2.5.1.12. AMI の登録解除

機能概要

AMI の登録を解除する。

対応する API

ec2-deregister [AMI ID]

2.5.1.13. AMI の一覧の確認

機能概要

公開されている AMI の一覧を確認する。

対応する API

ec2-describe-images

2.5.2. EBS

2.5.2.1. ボリュームの作成

機能概要

任意の領域のボリュームを作成する。

対応する API

ec2-create-volumu –size [領域(GB)] –availability-zone [サーバエリア]

2.5.2.2. ボリュームの削除

機能概要

作成したボリュームを削除する。

対応する API

ec2-delete-volume [ボリューム ID]

2.5.2.3. ボリュームのアタッチ

機能概要

作成済みのボリュームとインスタンスを紐付ける。

対応する API

ec2-attach-volume [ボリューム ID] –instance [インスタンス ID] –device <ディレクトリパス>

2.5.2.4. ボリュームのデタッチ

機能概要

インスタンスとボリュームの紐付けを解除する。

対応する API

ec2-detach-volume [ボリューム ID]

2.5.2.5. スナップショットの作成

機能概要

スナップショットを作成する。

対応する API

ec2-create-snapshot [ボリューム ID]

2.5.2.6. スナップショットの削除

機能概要

スナップショットを削除する。

対応する API

ec2-delete-snapshot [スナップショット ID]

2.5.3. Elastic IP

2.5.3.1. 固定 IP の設定

機能概要

固定 IP を獲得し、インスタンスに割り当てる。

対応する API

ec2-allocate-address

ec2-associate-address -i[インスタンス ID][IP アドレス]

2.5.3.2. 固定 IP の解除

機能概要

固定 IP の割り当てを解除し、返却する。

対応する API

ec2-disassociate-address [IP アドレス]

ec2-release-address [IP アドレス]

2.5.4. CloudWatch

2.5.4.1. インスタンス・ロードバランサのモニタリング

機能概要

インスタンスの CPU 利用率やディスク領域等インスタンスの稼動状況を監視する。

レイテンシや要求数等ロードバランサの稼動状況を監視する。

対応する API

ec2-monitor-instances [インスタンス ID]

2.5.4.2. 稼動状況の確認

機能概要

モニタリング中のステータスを取得し稼動状況を確認する。

対応する API

mon-get-stats [ステータス] --start-time [モニタリング起点]

--end-time [モニタリング終点] --period [モニタリング間隔]

--namespace “AWS/EC2 or AWS/ELB”--statistics [モニタリング取得データ]

--dimensions [モニタリング対象]

[モニタリング取得データ]は Average,Minimum,Maximum,Samples,Sum を指定可能

[ステータス]は以下の値を指定可能

モニタリング対象がインスタンスの場合(namespace “AWS/EC2”)

CPUUtilization、 NetworkIn、 NetworkOut、

DiskWriteOps、 DiskReadOps、 DiskReadByte、 DiskWriteBytes

モニタリング対象がロードバランサの場合(namespace “AWS/ELB”)

Latency、RequestCount、HealthyHostCount、UnHealthyHostCount

2.5.5. Auto Scaling

2.5.5.1. 設定の作成

機能概要

スケールアウトする AMI とインスタンスタイプ等を設定する。

対応する API

as-create-launch-config [設定名] –image-id [AMI_ID] –instance-type [インスタンスタイプ]

2.5.5.2. オートスケーリングの設定

機能概要

オートスケーリングを適用するグループを作成する。負荷状況に応じて、インスタンス数を自動で増減したり、最小インスタンス数を n 以上に設定する事で、インスタンスが停止しても自動で新しいインスタンスが起動し、インスタンス数を n 以上に保つ。

対応する API

as-create-auto-scaling-group [グループ名] --launch-configuration [設定名]
--availability-zones [サーバエリア] --min-size [最小インスタンス数]
--max-size [最大インスタンス数]

2.5.5.3. 閾値の設定

機能概要

オートスケーリングの閾値を設定する。

対応する API

as-create-or-update-trigger [トリガー名] –auto-scaling-group [グループ名]
--namespace “AWS/EC2 or AWS/ELB” –measure [ステータス]

2.5.5.4. オートスケーリングの状態の確認

機能概要

グループの状態や稼動しているインスタンスを確認する。

対応する API

as-describe-auto-scaling-groups

2.5.5.5. オートスケーリングの履歴の確認

機能概要

オートスケーリングの動作履歴を確認する。

対応する API

as-describe-scaling-activities [グループ名]

2.5.5.6. オートスケーリングの設定の確認

機能概要

オートスケーリングの設定や閾値を確認する。

対応する API

as-describe-launch-configs

as-describe-triggers

2.5.5.7. オートスケーリングの解除

機能概要

オートスケーリングを解除する。

対応する API

as-terminate-instance-in-auto-scaling-group [インスタンス ID]

--decrement-desired-capacity

2.5.5.8. 設定の削除

機能概要

オートスーリングの設定を削除する。

対応する API

as-delete-launch-config [設定名]

2.5.5.9. 閾値の削除

機能概要

設定した閾値を削除する。

対応する API

as-delete-trigger [トリガー名]

2.5.5.10. グループの削除

機能概要

設定したグループを削除する。

対応する API

as-delete-auto-scaling-group [グループ名]

2.5.6. ELB(Elastic Load Balancing)

2.5.6.1. ロードバランサの作成

機能概要

フロントエンドでトラフィックを分散させるロードバランサを作成する。

対応する API

elb-create-lb [ロードバランサ名] --availability-zones [サーバエリア]

2.5.6.2. ロードバランサの削除

機能概要

ロードバランサを削除する。

対応する API

elb-delete-lb [ロードバランサ名]

2.5.6.3. 接続するインスタンスの設定

機能概要

ロードバランサに接続するインスタンスを設定する。

対応する API

elb-register-instances-with-lb [ロードバランサ名] --instances [インスタンス ID…]

2.5.6.4. バックエンドのヘルスチェック

機能概要

バックエンドのヘルスチェックを行う。

対応する API

elb-describe-instance-headlth [ロードバランサ名] --target [チェック方法]
--healthy-threshold [チェック OK になる回数(2-10 回)]
--unhealthy-threshold [チェック NG になる回数(2-10 回)]
--interval [チェック間隔(5-600 秒)]
--timeout [タイムアウト時間(2～60 秒))

3. まとめ

本調査書では、Amazon EC2 とインタフェース互換性を持つオープンソースクラウド基盤システム Eucalyptus について、その基本的な仕組みや、インストール及び使用方法、また Amazon EC2 との違いについて報告した。Eucalyptus には 3 つのネットワークモードが備わっており、特に MANAGED モードではセキュリティグループが設定可能など柔軟なクラウドシステムを構成可能である。また、ノードコントローラが管理する計算ノードのリソース情報については、クラスタコントローラが自動的に収集し、それぞれの計算ノードで動作可能なインスタンス数を自動的に集計する仕組みも備わっている。そのため、クラウドシステムを運用したまま新たに計算ノードを追加することも可能であるなど、一定の柔軟性を備えている。

■仮想化技術と Eucalyptus

仮想化技術としては XEN 及び KVN の両方に対応しており、環境に合わせた仮想化方法を選択することができるが、仮想マシンとして動作可能な条件など、実際の運用を考慮すれば KVM で構築する方が妥当であると考えられる。XEN で構築する場合は準仮想化されたイメージファイルで動作するため、原則として XEN 専用に新たにインストールしたイメージファイルを用意する必要がある。物理マシン上で動作している環境をそのまま Eucalyptus で動かすためにはカーネルの変更など、Xen 用に様々な変更を加えなければならない。ただし KVM で構築するためには計算ノードの CPU が VT（Virtualization Technology）に対応している必要があるため、比較的新しい計算機を用意する必要がある。

■Amazon EC2 との比較

Amazon EC2 と比較してみると、地理的に離れた計算ノードを含むクラウド構成が可能な点や、仮想マシンのネットワーク構成の柔軟性ではほぼ同等の機能を提供している。一方で、現バージョンの Eucalyptus では、システム全体の冗長性を確保できず、ロードバランサやオートスケーラは搭載していない。そのため大規模な計算用途やトラフィックが非常に大きい web システムの構築に関しては、別の手段を併用することで信頼性の高いシステムの構築が可能となる。Amazon EC2 では稼働率 99.95%のサービスレベルを確保し、大規模な商用システムにも耐えられるようなサービスを提供していることを考えると、現状では Amazon EC2 を利用した方が良いと思われる。ただし、Amazon EC2 が保持する計算ノードは欧米が中心であるため、ネットワークを介したレスポンスの悪さが指摘されている。また日本企業において、データを社外に置くことに非常にセンシティブな面もある。この点においては、Eucalyptus を用いて

自前でクラウドを構築するメリットと捉えられる。

■Eucalyptus の Linux ディストリビューション対応状況

Eucalyptus を利用可能な Linux ディストリビューションについては、現時点では限られている。Ubuntu では Ubuntu9.04 で KVM を用いた Eucalyptus が構築可能である。ただし、Ubuntu8.10 以降は Xen がサポートされなくなったため、VT を搭載した CPU を計算ノードにすることが必須である。もし Ubuntu で Xen を使う場合は Ubuntu8.04 などを使用する方が良いと思われる。一方 CentOS でも Eucalyptus を利用可能であるが、こちらは現時点で最新版の CentOS5.3 でも Eucalyptus は KVM に対応していない。Linux カーネル 2.6.20 より KVM が組み込まれているが、CentOS5.3 のカーネルは 2.6.18 であり、KVM はカーネルのモジュールとして提供される。そのため CentOS 上で KVM を動作させることは可能であるものの、Eucalyptus の KVM の設定ではノードコントローラの起動に失敗する。

■計算機に求められる性能

Eucalyptus を動作させる場合は計算機の性能もある程度必要である。実運用を考えれば、4GB 以上のメモリと Intel Core2 か同程度以上の CPU を搭載したマシンが好ましいと思われる。また Eucalyptus は冗長性を持たないため、計算機の故障などを考慮してストレージは Raid を構成するなどの対策をしておくことが必要である。

【主な参考文献】

[1] http://open.eucalyptus.com/

[2] https://help.ubuntu.com/community/Eucalyptus

[3] Eucalyptus : A Technical Report on an Elastic Utility Computing Archietcture Linking Your Programs to Useful Systems, Daniel Nurmi, Rich Wolski, Chris Grzegorczyk Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, UCSB Computer Science Technical Report Number 2008-10.

[4] The Eucalyptus Open-source Cloud-computing System, Daniel Nurmi, Rich Wolski, Chris Grzegorczyk Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov.

[5] http://builder.japan.zdnet.com/sp/09-personal-cloud/story/0,3800097247,20394582,00.htm