クラウド基盤構築演習
第二部: Eucalyptusによるクラウド基盤構築

第14回: Eucalyptusの監視と運用

ver1.1 2012/07/20

目次

- 監視するポイントと監視ツール
- 監視ツールのインストールと設定演習
- 運用のポイントと運用ツール

本講で学ぶこと、実施すること -1-

- 監視するポイントを把握しましょう
  - どこを監視すべきかを知ることで、トラブル発生時に素早く「何が起きているか」「どんな状態なのか」を把握することができます
- 監視ツールのインストールや設定を通して
  - 監視ツールがどんな値を閾値として設定されているのか？などを把握することができます

本講で学ぶこと、実施すること -2-

- 運用ポイントを把握しましょう
  - 運用するにあたり、普段どのような作業が発生するのかを認識し、その作業を集約することで運用面での改善点や自動化すべき部分が見えてきます
- 運用ツールを使ってみましょう
  - 知識だけでなく実際に運用ツールを使ってみて、どのような運用ツールがあれば運用作業が楽かを考えてみましょう

監視するポイントと監視ツール

監視するポイント

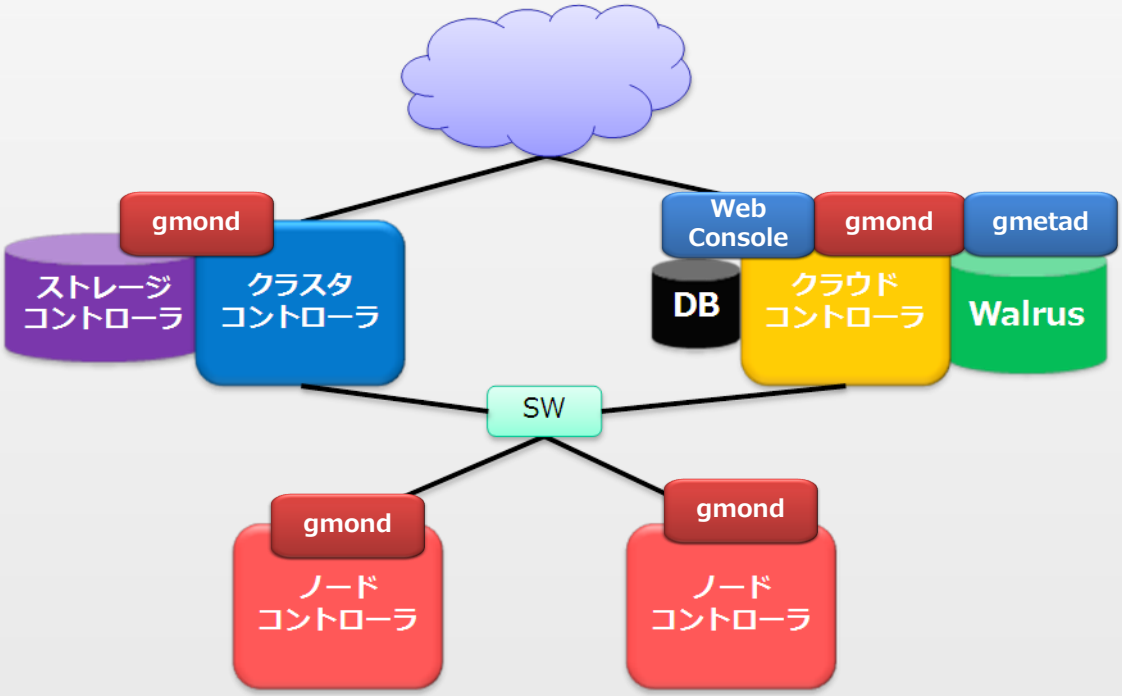
- Eucalyptusが正常に稼動しているかを確認するために以下を監視対象とします
  - 各物理マシンの物理ディスクの容量
  - 各物理マシンのメモリの容量
  - 各コンポーネントのメモリ使用量
  - 各物理マシンのCPUの使用率
  - 各物理マシンのNICのI/O使用量

監視ツール

- 色々な監視ツールがありますが、Eucalyptusでは Ganglia用の監視スクリプトが提供されており、NCと SCとWalrusが使用しているハードウェアリソースについて監視できるため、本講ではGangliaを使用します。
- Gangliaはクラスタやグリッドなどのシステム向けに特化して作られた分散システム監視ソフトウェアです。ライセンスはBSDライセンスで配布されています 。Gangliaは大きくわけて3つのコンポーネントで構成され、各ホストで指定されたリソースなどを監視する Ganglia モニタリングデーモン (gmond)、各ホストでの監視結果を受けとるGanglia メタデーモン(gmetad)、PHPによるWebコンソールを提供するGanglia PHP Webフロントエンド(Web Console)があります

Gangliaの構成例

- 以下に本講で構築するGangliaの構成例を示します



監視ツールのインストールと設定演習

Gangliaのインストール

- 前述の構成に従って、各マシン(clc, cc, nc1, nc2)にgmondを以下のように実行してインストールします

```
パッケージのインストール
# yum install ganglia-gmond.x86_64 ganglia-gmond-python.x86_64
サービスを有効化
# chkconfig gmond on
```

- 次にclcのマシンにgmetadとWebConsoleをインストールします

```
パッケージのインストール
# yum install ganglia-gmetad.x86_64 ganglia-web.x86_64
サービスを有効化
# chkconfig gmetad on
```

Gangliaの設定

- 各マシン(clc, cc, nc1, nc2)のgmondの設定ファイル(/etc/ganglia/gmond.conf)内のclusterとhostを以下のように設定します

```
cluster {
  name = "Eucalyptus Cloud"
  owner = "Taro Kaso"
  latlong = "39.19853, 139.9083"
  url = "https://clc.user30.ip-157-1-206-3.ecloud.nii.ac.jp:8443/"
}

/* The host section describes attributes of the host, like the location */
host {
  location = "clc.user30.ip-157-1-206-3.ecloud.nii.ac.jp"  # ← 別紙のホスト名を設定してください
}
```

Gangliaの設定

- 各マシン(clc, cc, nc1, nc2)のgmondの設定ファイル(/etc/ganglia/gmond.conf)内のudp_send_channelを以下のように設定します

```
udp_send_channel {
  #bind_hostname = yes # Highly recommended, soon to be default.
  (中略)
  mcast_join = 239.2.11.71
  port = 8649
↓
udp_send_channel {
  #bind_hostname = yes # Highly recommended, soon to be default.
  (中略)
  #mcast_join = 239.2.11.71
  host = clcのeth1のIPアドレス
  port = 8649
```

Gangliaの設定

- 各マシン(clc, cc, nc1, nc2)のgmondの設定ファイル(/etc/ganglia/gmond.conf)内のudp_recv_channelを以下のように設定します

```
udp_recv_channel {
  mcast_join = 239.2.11.71
  port = 8649
  bind = 239.2.11.71
}
↓
udp_recv_channel {
  #mcast_join = 239.2.11.71
  port = 8649
  #bind = 239.2.11.71
}
```

Gangliaの設定

hostのlocationは各ホスト毎に設定値を変更します。なお、上記設定の各項目の概要は以下になります。

項目	概要
cluster.name	Gangliaの監視対象とするホスト群をひとまとめにした際に命名するクラスタ名を設定。Eucalyptusのクラスタとは関係ない。
cluster.owner	クラスタの所有者の名前(任意)を設定。
cluster.latlong	クラスタが置かれている場所の緯度経度を設定。
cluster.url	クラスタに対するURL。本章では便宜的にEucalyptusのWeb管理画面のURLを設定。
host.location	ホストのロケーションを設定。例えば「CentralTower 25F RiverSide」など。本章では便宜的にホスト名を設定。

次にGanglia メタデーモンを配置するホストでgmetadの設定を行ないます。

gmetadの設定ファイル/etc/ganglia/gmetad.confのdata_sourceを以下のように変更します。

```
data_source "Eucalyptus Cloud" 10 192.168.30.10 192.168.30.20 192.168.30.100 192.168.30.101
                                  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                    ↑ 適宜環境にあわせて変更してください ↑
```

Gangliaの設定

- clcの/etc/httpd/conf.d/ganglia.confを以下のように変更し、Webインターフェイスにアクセスできるように設定します。

```
<Location /ganglia>
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
    Allow from ::1
    # Allow from .example.com
</Location>

↓
<Location /ganglia>
    Order deny,allow
    Allow from 127.0.0.1
    Allow from ::1
    # Allow from .example.com
    Allow from all
</Location>
```

Gangliaの設定

なお、data_sourceは以下の書式で設定することができますが、第2引数のポーリング間隔は省略すると15秒間隔でポーリングを実施し、第3引数以降のポート番号は省略すると8649番ポートに対して通信を行ないます。

```
data_source "クラスタ名" [ポーリング間隔(秒)] ホスト:ポート ホスト:ポート ...
```

以上でGangliaの基本的な設定は完了ですが、冒頭で示した図のようにGangliaを導入するホストがマルチホームの場合はGangliaが使用するマルチキャストアドレスに対する通信にどのインターフェイスを使用するかを設定する必要があります。例えば本章の環境では/etc/sysconfig/network-scripts/route-eth1というファイルに以下の設定を記述しています 。

```
239.2.11.71 dev eth1
```

必要な設定が完了したのち、以下のようにgmetadとgmondを起動します。

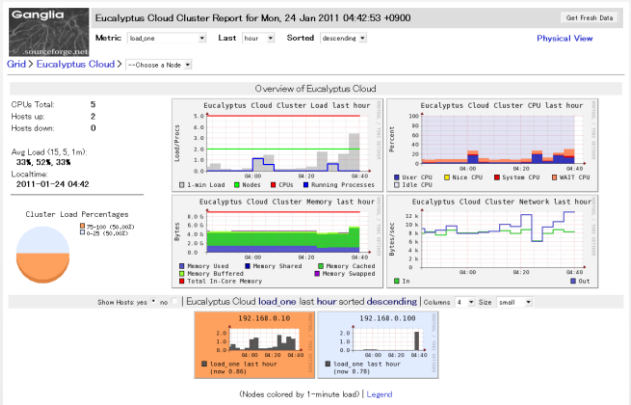
```
# gmetadを配置したホストにて
/etc/init.d/gmetad start
/etc/init.d/httpd restart

# gmondを配置した各ホストにて
/etc/init.d/gmond start
```

Gangliaの使い方 -1-

GangliaはGanglia PHP Webフロントエンドを配置したホストの以下のようなURLに対してアクセスすることで監視内容を閲覧することができます。

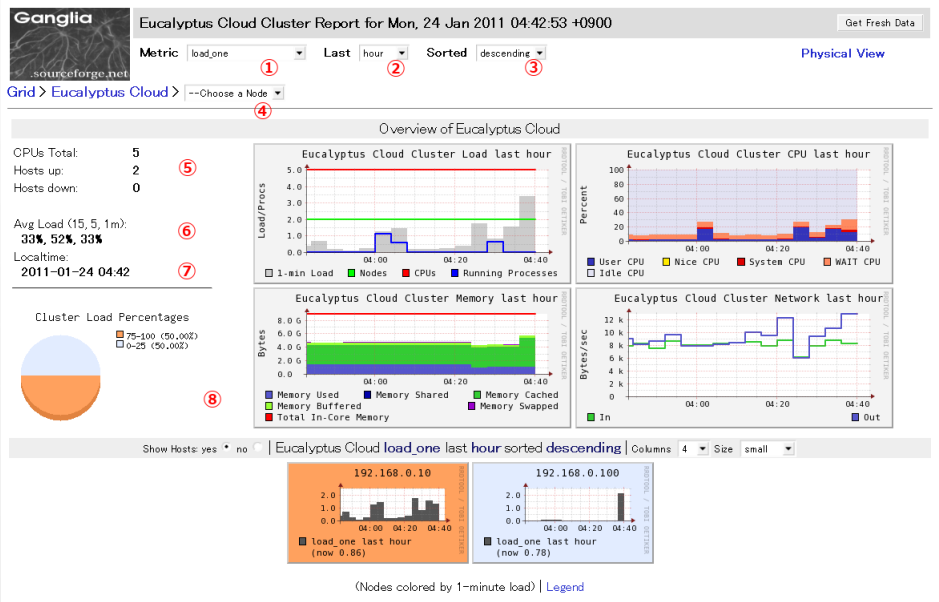
```
http://WebConsoleのIPアドレス/ganglia/
```



なおGanglia PHP Webフロントエンドのトップ画面にはクラスタ全体のリソース消費の概要(ロードアベレージ1分,CPU使用率,メモリ使用率,ネットワーク帯域)のグラフが表示され、その下にホスト毎のロードアベレージ(1分)のグラフが表示されます。ホスト毎のグラフはページ上部にあるMetricのリストで切り換えることが可能です。

以下にトップ画面の簡単な説明を示します。

Gangliaの使い方 -2-



① 画面に表示する各ホストのグラフの種類を指定
② グラフの表示対象を指定。直近の1時間,1日,1週間,1ヶ月,1年から選択します。
③ 各ホストのグラフの並びを指定。ascending(降順),descending(昇順),by name(名前でソート)から選択します。
④ 詳細を表示するホストを選択します。
⑤ クラスタ全体のCPUコア数と起動しているホスト数と停止しているホスト数が表示されます。
⑥ クラスタ全体のロードアベレージ(15分,5分,1分)が表示されます。
⑦ クラスタの時刻が表示されます。
⑧ クラスタ全体のロードアベレージを0～25%,25%～50%,50～%75%,75%～100%で分けた場合の円グラフが表示されます。Eucalyptusのクラスタではなく、Gangliaに設定したホスト群を意味します。

Gangliaの使い方 -3-

EucalyptusではGangliaのgmetricコマンドを使用した監視スクリプトを提供しており、SCとWalrusとNCのハードウェアリソースを監視することが簡単に行なえるようになっています。ただし現在Eucalyptus社が配布しているRHEL/CentOS用パッケージにはこの監視スクリプト「ganglia.sh」が含まれていないため、githubから入手 する必要があります。

ganglia.shを入手し、SC,Walrus,NCをインストールしている各ホストの/usr/local/bin/配下に配置し、以下のように実行し、ユーザ「eucalyptus」のcronjobに登録します。

```
# wget https://raw.github.com/eucalyptus/eucalyptus/master/extras/ganglia.sh
# mv ganglia.sh /usr/local/bin/
# chmod +x /usr/local/bin/ganglia.sh
# sudo -u eucalyptus crontab -e

# SCのハードウェアリソースを毎分監視する場合
* * * * * /usr/local/bin/ganglia.sh -type sc -d /

# Walrusのハードウェアリソースを毎分監視する場合
* * * * * /usr/local/bin/ganglia.sh -type walrus -d /

# NCのハードウェアリソースを毎分監視する場合
* * * * * /usr/local/bin/ganglia.sh -type nc -d /
```

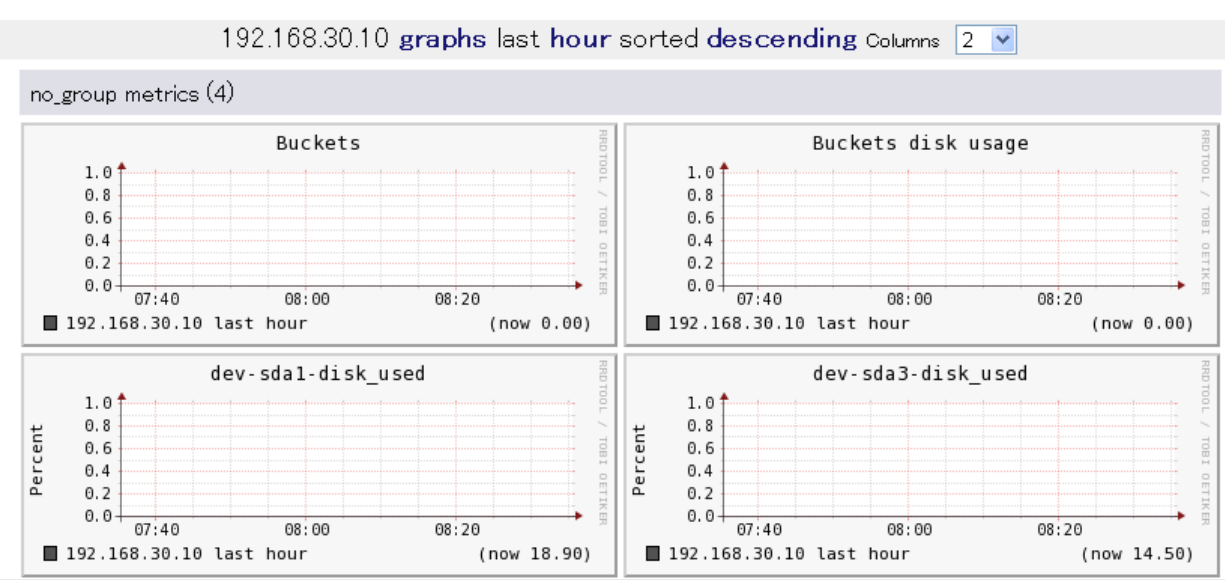
Gangliaで監視できる項目

Eucalyptusが提供する監視スクリプトによりGangliaで監視できる項目を以下に示します。

対象	項目	概要
SC	Volumes	ボリュームの数
SC	Volumes disk usage	ボリュームが使用しているディスクの容量
Walrus	Buckets	バケットの数
Walrus	Buckets disk usage	バケットが使用しているディスクの容量
NC	Running VMs	起動しているインスタンス数
NC	VMs available cores	インスタンスが利用可能なCPUコアの残数
NC	VMs available disks	インスタンスが利用可能なディスクの残容量 (GB)
NC	VMs available memory	インスタンスが利用可能なメモリの残容量 (MB)
NC	VMs used cores	インスタンスが利用しているCPUのコア数
NC	VMs used disks	インスタンスが利用しているディスクの容量 (GB)
NC	VMs used memory	インスタンスが利用しているメモリの容量 (MB)

Gangliaで監視できる項目

- 1分ちょっと経過してから各ホストのグラフを閲覧すると、前述で追加したリソース情報に関するグラフが確認できます。(下記の図はWalrus)



Pythonによる監視モジュールプラグイン -1-

Eucalyptusが提供する監視スクリプトはgmetricコマンドを使用するものですが、もう一つの監視方法としてPythonによる監視モジュールプラグインを作成する方法があります。

gmetricコマンドによる監視方法は簡単に監視対象を設定することができる反面、監視スクリプトをcronjobに登録する必要があり、gmetad.confに設定したポーリング間隔が毎分以上の間隔である場合に更新タイミングに差異が発生します。一方で、Pythonによる監視モジュールプラグインを使用した場合はgmetad.confに設定したポーリング間隔で更新されるほか、より柔軟に監視項目を追加することが可能です。

サンプルとして、NCのサービスのプロセスが使用する物理メモリを取得するプラグインを説明します。

以下のコードを各NCに/usr/lib64/ganglia/python_modules/eucalyptus-nc.pyという名前で配置します。(wget http://157.1.207.1/eucalyptus-nc.py で入手してください)

```
import os, sys, popen2

def Get_RSS(cmd):
    cp = popen2.Popen3(cmd)
    cp.tochild.close()
    child_stdout_fp = cp.fromchild
    ret = cp.wait()
```

Pythonによる監視モジュールプラグイン -2-

（前ページの続き）

```
    if ( ret == 0 ):
        l = child_stdout_fp.read()
        line = l.split()
        rss = int(line[0])
    else:
        rss = 0

    return rss

def Eucalyptus_Parent_Process_RSS(name):
    global ppid

    pid_file = '/var/run/eucalyptus/eucalyptus-nc.pid'

    try:
        f = open(pid_file, 'r')

    except IOError:
        return 0

    for l in f:
        ppid = l.split()[0]

    cmd = "ps --pid %s -o rss=" % ppid
    return Get_RSS(cmd)
```

Pythonによる監視モジュールプラグイン -3-

（前ページの続き）

```
def Eucalyptus_Child_Process_RSS(name):

    cmd = "ps --ppid %s -o rss=" % ppid
    return Get_RSS(cmd)

def metric_init(params):
    global descriptors

    d1 = {'name': 'Eucalyptus_Parent_Process_RSS',
        'call_back': Eucalyptus_Parent_Process_RSS,
        'time_max': 90,
        'value_type': 'uint',
        'units': 'KB',
        'slope': 'both',
        'format': '%u',
        'description': 'RSS size of Parent Process in Eucalyptus',
        'groups': 'eucalyptus'}
```

Pythonによる監視モジュールプラグイン -4-

（前ページの続き）

```
    d2 = {'name': 'Eucalyptus_Child_Process_RSS',
        'call_back': Eucalyptus_Child_Process_RSS,
        'time_max': 90,
        'value_type': 'uint',
        'units': 'KB',
        'slope': 'both',
        'format': '%u',
        'description': 'RSS size of Child Processes in Eucalyptus',
        'groups': 'eucalyptus'}

    descriptors = [d1,d2]

    return descriptors

def metric_cleanup():
    '''Clean up the metric module.'''
    pass

#This code is for debugging and unit testing
if __name__ == '__main__':
    metric_init(None)
    for d in descriptors:
        v = d['call_back'](d['name'])
        print 'value for %s is %u' % (d['name'],  v)
```

Pythonによる監視モジュールプラグイン -5-

次に上記のコードが正しく動作できるかを以下のように実行して、プロセスのRSSが取得できるかを確認します。

```
python /usr/lib64/ganglia/python_modules/eucalyptus-nc.py
value for Eucalyptus_Parent_Process_RSS is 1644
value for Eucalyptus_Child_Process_RSS is 107212
```

次に以下のコードを各NCに/etc/ganglia/conf.d/eucalyptus-nc.pyconfという名前で配置します。

```
modules {
  module {
    name = "eucalyptus-nc"
    language = "python"
  }
}

collection_group {
  collect_every = 10
  time_threshold = 50
  metric {
    name = "Eucalyptus_Parent_Process_RSS"
    title = "RSS size of Parent Process in Eucalyptus"
    value_threshold = 1048576
  }
  metric {
    name = "Eucalyptus_Child_Process_RSS"
    title = "RSS size Child Processes in Eucalyptus"
    value_threshold = 1048576
  }
}
```

Pythonによる監視モジュールプラグイン -6-

次に以下のように実行し、NC上のgmondを再起動します。

```
/etc/init.d/gmond restart
```

gmondを再起動してしばらくしてからGanglia PHP Webフロントエンドに接続し、監視モジュールプラグインを設置したNCの詳細グラフを参照すると、以下のようにグラフが描画されていることが確認できます。

運用のポイントと運用ツール

運用について

- 本章では、Eucalyptusで構築したクラウド環境を運用する際に考慮する必要のある、ユーザデータとバックアップについて説明します。また本章では、CentOS環境下のEucalyptusが使用するディレクトリをベースに説明します。一部のディレクトリはEucalyptusの設定値を変更することによってディレクトリパスが異なりますが、本章ではデフォルト設定の場合を前提として説明します。

ユーザデータについて -1-

- 実際にEucalyptus環境を運用すると作成されるユーザのデータ(仮想マシンイメージ、EBSボリューム、スナップショット、その他Walrus上のデータ)について説明します。ユーザデータを知ることは、クラウド構築時のディスク設計やバックアップを考える事に役立ちます。
- Eucalyptusにあるユーザデータと概要は下記の通りです。

項目	概要
ザーユ情報	のザーユIDや名前とAPIを呼び出とーキのめたす証明書
仮想ジーメインシマ	仮想のジーメインシマIDや属性
プールグィテリュキセ	ルールのそとプールグィテリュキセのスンタスンイ
アペーキ	るすンイグロにスンタスンイ際に使用るすSSH鍵
PublicIP	がザーユ確保たしPublicIP
ムーュリボ	EBSムーュリボの
トッョシプッナス	トッョシプッナスのムーュリボ
トクェジブオとトッケバ	Walrusに保存トクェジブオとトッケバるす

ユーザデータについて -2-

- ユーザ情報
  - ユーザ情報とは、Web管理画面から作成する際に入力するユーザの情報と、EucalyptusのAPI呼び出しに必要なアクセスキー、シークレットキー、X.509証明書の情報を指します。Eucalyptusでは、これらのユーザ情報すべてをCLC内のDBに保存しています。これらのデータはユーザ作成時に同時に作成されますが、X.509証明書のみWeb管理画面にログイン後のZIPファイルのダウンロード時に作成されます。さらにX.509証明書はダウンロードする度に新規に証明書が発行される点に注意が必要です。X.509証明書はSOAPを使用したAPIの呼び出しに使われるだけでなく、仮想マシンイメージのバンドルにも使用されます。 アップロードしたイメージを手元で複号するには、バンドルに使用したX.509証明書が必要になります。そのため、初めにダウンロードした証明書でバンドルしたイメージは、二度目にダウンロードした証明書では元に戻せないことに注意してください。

ユーザデータについて -3-

- 仮想マシンイメージ
  - 仮想マシンイメージは比較的サイズの大きなユーザデータです。仮想マシンイメージの登録情報や権限などの情報はCLCのDBで管理されますが、仮想マシンイメージの実体はWalrus上にアップロードしたバンドルファイルとマニフェストファイルとなります。また、仮想マシンイメージの登録を解除しただけでは、Walrus上のファイルは削除されないため、euca-delete-bundleコマンドやCyberduckなどのWalrusのAPIを呼び出す事ができるアプリケーションを使用して削除する必要があります。

ユーザデータについて -4-

- セキュリティグループ
  - セキュリティグループとそのパーミッションルールの情報はすべてCLC上のDBで管理されます。ユーザには、必ずdefaultというセキュリティグループが作成されます。
- キーペア
  - キーペアは仮想マシンにログインする際に使用するSSHの鍵で、CLCのDBに情報が保存されます。その名前の通り、公開鍵と秘密鍵のペアが作成されますが、CLCのDBに保存される情報は、公開鍵とフィンガープリントのみです。秘密鍵はキーペアの作成時に一度だけ作られる情報であるため、紛失したり削除した場合は二度と同じファイルを作成することはできません。キーペア作成時には注意して作成してください。
- Public IP
  - Public IPは、Elastic IP機能で利用するユーザだけが使うことができるIPアドレスで、CLCのDBに情報が保存されます。

ユーザデータについて -5-

- ボリューム
  - ボリュームはサイズの大きいユーザデータの1つです。ボリュームのIDや作成時刻、サイズ、所有者についての情報はCLCのDBに保存されますが、実体のデータはデフォルトの設定では、SCが動作するマシンの「/var/lib/eucalyptus/volumes」配下に保存されます。EucalyptusではLVMとAoEもしくはiSCSIと組み合わせ、SCが動作するマシン上のローカルファイルをボリュームとして動的に使用できるようにしています。
- スナップショット
  - スナップショットはボリュームのある時点での情報を保存したもので、このスナップショットからボリュームを作成することが出来ます。スナップショットのIDや作成時刻、元となったボリュームのIDについての情報はCLCのDBに保存されますが、実体のデータとしてデフォルトの設定ではSCが動作するマシンの「/var/lib/eucalyptus/volumes」配下とWalrusが動作するマシンの「/var/lib/eucalyptus/bukkits」配下にファイルを作成します。

ユーザデータについて -6-

- バケットとオブジェクト
  - バケットとオブジェクトはWalrusが格納および管理するデータです。バケットとオブジェクトの名前や作成時刻、所有者、権限などの情報はCLCのDBに保存されますが、実体のデータはデフォルトの設定では、Walrusが動作するマシンの「/var/lib/eucalyptus/bukkits」配下に保存されます。仮想マシンイメージを登録するには、必ずWalrus上のいずれかのバケットにバンドルされたファイルとマニフェストファイルをオブジェクトとして保存する必要があります。そのため、EBS機能で使用するボリュームとスナップショットと同様に大きなサイズのユーザデータとなります。
- その他のユーザデータについて
  - 上記で紹介した以外にもインスタンスというユーザデータが存在します。インスタンスはどこにも永続化されることのない情報で、CLCやCCのメモリ上で管理されます。ただし、実際にインスタンスを起動しているNCでは、ディスクやCPUやネットワークなどのコンピュータリソースをユーザが使用します。

ユーザデータの削除について -1-

ドウラクがザーユ環境を使用りなくながとこるす、をトンウカア削除てし良い状態たっなに場合、のターデザーユ削除るすを必要すまりあが。のトンウカアのザーユ削除は、管理者がWeb管理画面てしンイグロに行がすまきでがとこう、をトンウカアのザーユでここ削除もてし、のてべすがターデザーユ削除。んせまりあはでけわるれさ

はでここ、トンウカアのザーユ情報を削除るす前に管理者が実施るす必要ーデザーユるあののタ削除方法を説明。すまし

- **削除手順**
  - のターデザーユ削除は下記の手順で実施。すまし

1. のトンウカアにザーユ使用を停止うらもてし
2. てしとトンウカアのザーユ作業るす環境を整るえ
3. をターデザーユ削除るす
4. をトンウカアのザーユ削除るす

管理者権限はターデザーユでザーユの削除出来がすま、がムーュリボのど削除対象ーユのんせまりかわがかのなムーュリボのザ。でこそ本手順はで、ーキスセクアのトンウカアのザーユを使用てしとザーユてしAPIを呼び出をターデザーユのてべすてし削除。すましAPIを呼び出たすにめeuca2oolsを使用。すまし

ユーザデータの削除について -2-

ユーザのアクセスキーの取得

ユーザとしてAPIを呼び出すために、ユーザのアクセスキーとシークレットキーを入手する必要があります。入手する方法は2つあり、1つは管理者権限でWeb管理画面にログインし当該ユーザのパスワードを管理者権限で変更し、当該ユーザとしてWeb管理画面にログインしてアクセスキーとシークレットキーを入手する方法と、もう1つはCLCのサーバにログインし、下記のコマンドを実行します。

```
grep "INSERT INTO AUTH_USERS" ¥
/var/lib/eucalyptus/db/eucalyptus_auth.* | ¥
gawk -F, '{print $6, $8, $9}' | uniq
```

上記のコマンドを実行すると下記のような実行結果を取得することができます。

```
'user01' 'eXaKIsDZ1vIZRAg' '4ZadWXRefyb9ysAfny3y8Wmu2Nkw'
'user02' 'j0aJCy6gwmvaDH' 'fEP9vgGN4Wp0XCNdDmN7F34F9w'
```

左から順番にシングルクォートで囲まれた文字列がユーザID、アクセスキー、シークレットキーとなります。ここで出力されている削除対象のユーザのアクセスキーとシークレットキーを使用して下記のコマンドを実行して環境変数を設定しておきます。

```
export EC2_URL='http://your-cloud-ip:8773/Eucalyptus'
export EC2_ACCESS_KEY='アクセスキー'
export EC2_SECRET_KEY='シークレットキー'
```

ユーザデータの削除について -3-

- ユーザデータを削除する

ここからはeuca2oolsを使用してユーザデータを削除しますが、まずはユーザが起動しているインスタンスを停止しておきます。

次に、以下のようにスナップショットの一覧を取得し、表示されたスナップショットの数だけ削除コマンドが実行されるようにします。

```
for snapid in `euca-describe-snapshots | gawk '{print $2}'`; do
   euca-delete-snapshot ${snapid}
done
```

ユーザデータの削除について -4-

同様にして以下のようにボリューム、キーペア、セキュリティグループについて取得した一覧に基いて削除コマンドが実行されるようにします。

```
# ボリュームの削除
for volid in `euca-describe-volumes | gawk '{print $2}'`; do
   euca-delete-volume ${volid}
done

# キーペアの削除
for key in `euca-describe-keypairs | gawk '{print $2}'`; do
   euca-delete-keypair ${key}
done

# セキュリティグループの削除
for sg in ` euca-describe-groups | grep ^GROUP | gawk '{print $3}'`; do
   euca-delete-group ${sg}
done
```

続いてユーザが確保しているPublic IPの一覧を表示して、一つずつ解放します。

```
for ip in `euca-describe-addresses | gawk '{print $2}'`; do
   euca-release-address ${ip}
done
```

ユーザデータの削除について -5-

次に仮想マシンイメージを削除しますが、ここでは登録解除だけを行いWalrusにアップロードしている仮想マシンイメージの実体の削除は別途実施します。また、ユーザの登録したイメージのみを表示するには、下記のコマンドを実行します。

```
euca-describe-images -o <ユーザID>
```

表示された対象ユーザのイメージのロケーション部分（バケット名/マニフェストファイル）を使用して登録解除コマンドを実行します。

```
euca-deregister <ロケーション>
```

最後に、Walrus上のデータであるバケットとオブジェクトを削除します。バケットとオブジェクトを削除するには、WalrusのAPIを使用する必要があります。WalrusのAPIを使用してバケットとオブジェクトを削除するには、4.3.4で紹介したCyberduckを使用して削除するか、5.3章で紹介したライブラリを使用したプログラムを作成して削除します。Cyberduckを使用した場合もライブラリを使用する場合もアクセスキー、シークレットキーとWalrusのURLがあればAPIを呼び出すことができますので、前述の方法で取得したユーザのアカウントを使用して削除します。