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13回: EC2 APIとS3 APIの演習

ver1.1 2012/07/20

目次

- APIの概要と認証
- EC2 APIの演習
- S3 APIの演習
- メタデータに関する演習

本講で学ぶこと、実施すること -1-

- APIの概要と認証を理解しましょう
  - EC2 APIとS3 APIはどのような方法で利用できるのか？について知ることができます
  - APIのアーキテクチャと認証について知ることができます
- EC2 APIを実際に利用してみましょう
  - 各APIを叩いてみることによって、euca2oolsとAPIの粒度が同じことを知ることができます

本講で学ぶこと、実施すること -2-

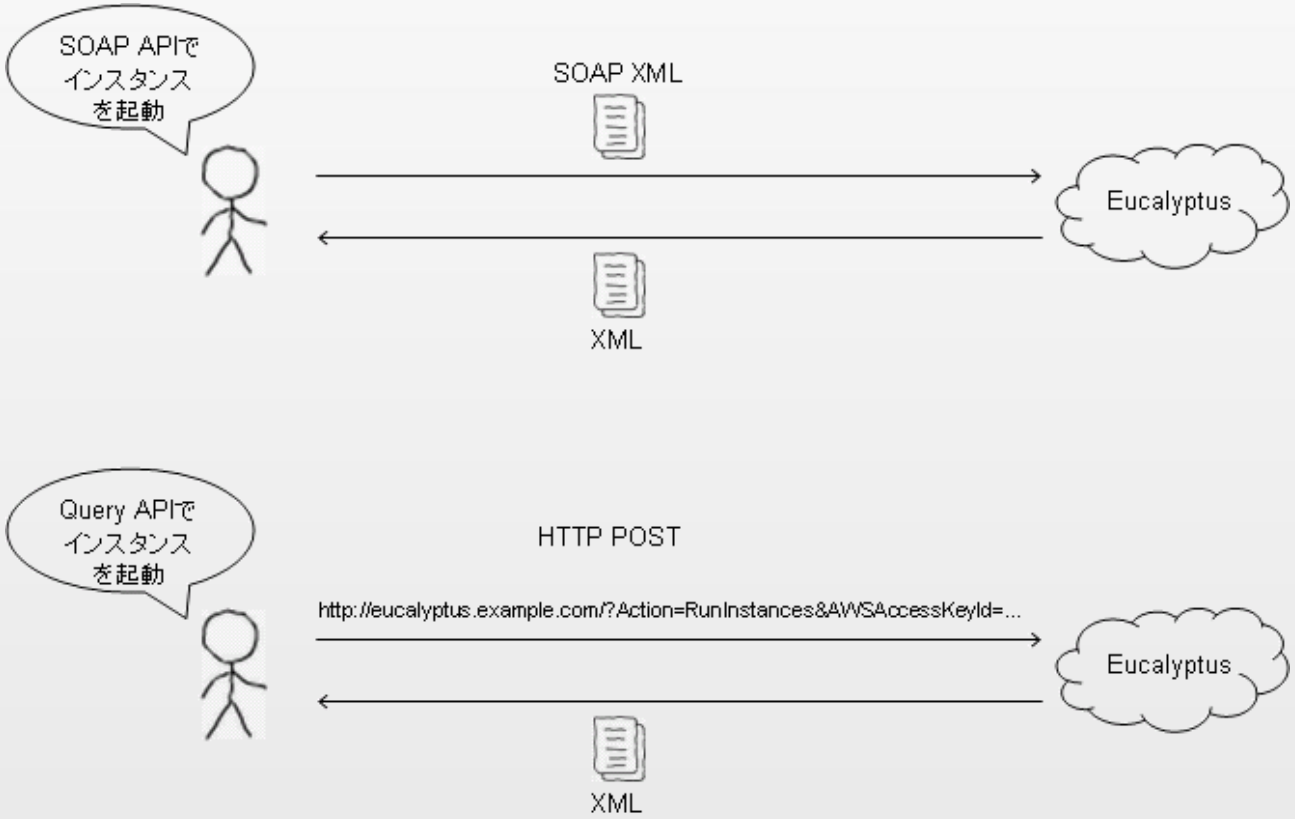
- S3 APIを実際に利用してみましょう
  - 何気なく利用していたWalrusが様々なAPIによって動作していたことを知ることができます
- メタデータを実際に利用してみましょう
  - EC2 APIやS3 APIに比べると、プログラマブルの要素は少ないですが、ただしプログラマブルにインスタンスを操作しようとすると欠かせない機能であることが理解できます

APIの概要と認証

3つのAPI提供方式 -1-

- EucalyptusはAmazon EC2/S3/IAMと互換のAPIを提供しています。このため同じAPIが利用できるならばクライアントも流用可能であると思いがちですが、実際は既存のAWS用クライアントをそのままEucalyptusで利用することはできません。
- API互換であるのに何故クライアントを流用できないのか？それにはいくつかの理由があります。
- Amazon EC2もEucalyptusも二つのAPI――SOAP APIとQuery APIを提供しています。どちらのAPIもできることはほとんど同じですが、APIを利用するための方式が異なります。SOAP APIは実行するAPI名やパラメータを記述したSOAP-XMLメッセージをエンドポイントに送付し、APIの実行結果としてXMLメッセージを受け取ります。
- 一方Query APIではエンドポイントURLとリクエストパラメータで構成されるリクエストを送付し、API実行結果としてXMLメッセージを受け取ります。
- Amazon S3とWalrusはEC2同様にSOAP APIがありますが、Query APIはありません。そのかわり、REST APIがあります。
- Amazon IAMとEucalyptus IAMもSOAP APIとREST APIがあります。

3つのAPI提供方式 -2-



3つのAPI提供方式 -3-

- euca2oolsを利用する際に読み込むeucarcファイルの中身を確認してみてください。eucarcにはSOAP/Query/REST APIそれぞれを利用するために必要な情報が記載されています。

```
# cat ~/.euca/admin/eucarc
EUCA_KEY_DIR=$(dirname $(readlink -f ${BASH_SOURCE}))
export EC2_URL=http://[CLCのIPアドレス]:8773/services/Eucalyptus
export S3_URL=http://[CLCのIPアドレス]:8773/services/Walrus
export EUARE_URL=http://[CLCのIPアドレス]:8773/services/Euare
export EUSTORE_URL=http://emis.eucalyptus.com/
export EC2_PRIVATE_KEY=${EUCA_KEY_DIR}/euca2-admin-b56c81f8-pk.pem
export EC2_CERT=${EUCA_KEY_DIR}/euca2-admin-b56c81f8-cert.pem
export EC2_JVM_ARGS=-Djavax.net.ssl.trustStore=${EUCA_KEY_DIR}/jssecacerts
export EUCALYPTUS_CERT=${EUCA_KEY_DIR}/cloud-cert.pem
export EC2_ACCOUNT_NUMBER='856120862843'
export EC2_ACCESS_KEY='OQMRSSBAXDCPC51GHQYUZ'
export EC2_SECRET_KEY='TjgVKULJEFTAcN53zK0e6rRDRS5q5hskHKcbpOQP'
export AWS_CREDENTIAL_FILE=${EUCA_KEY_DIR}/iamrc
export EC2_USER_ID='856120862843'
alias ec2-bundle-image="ec2-bundle-image --cert ${EC2_CERT} --privatekey
${EC2_PRIVATE_KEY} --user ${EC2_ACCOUNT_NUMBER} --ec2cert ${EUCALYPTUS_CERT}"
alias ec2-upload-bundle="ec2-upload-bundle -a ${EC2_ACCESS_KEY} -s ${EC2_SECRET_KEY} --
url ${S3_URL}"
```

SOAP APIの電文

- SOAP APIとして実際に送信するSOAPメッセージは、Amazon Web Serviceで定義されたWSDLに従った形式で構成され、WS-Securityの仕様に従って暗号化されたメッセージとなります。以下に、暗号化される前のインスタンス起動要求の例を示します。送信時にはこのXMLメッセージ全体が暗号化され、セキュアなSOAPメッセージとして送信されます。

```
<RunInstances xmlns="http://ec2.amazonaws.com/doc/2009-04-04/">
    <instancesSet>
        <item>
            <imageId>emi-CE56899A</imageId>
            <minCount>1</minCount>
            <maxCount>1</maxCount>
        </item>
    </instancesSet>
    <groupSet/>
</RunInstances>
```

Query APIの電文

- Query APIはSOAP APIよりも単純なメッセージ構成になります。Eucalyptusのエンドポイントに対して以下のリクエストパラメータを付与し、HTTPリクエストをPOSTします。

```
Action=RunInstances&AddressingType=public&AWSAccessKeyId=XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXX&ImageId=emi-CE56899&MaxCount=1&MinCount=1&SignatureVersion=1&Timestamp=2010-
12-28T12%3A31%3A00Z&Version=2009-04-04&Signature=XXXXXXXXXXXXXXXXXXXXXXXXXXX
```

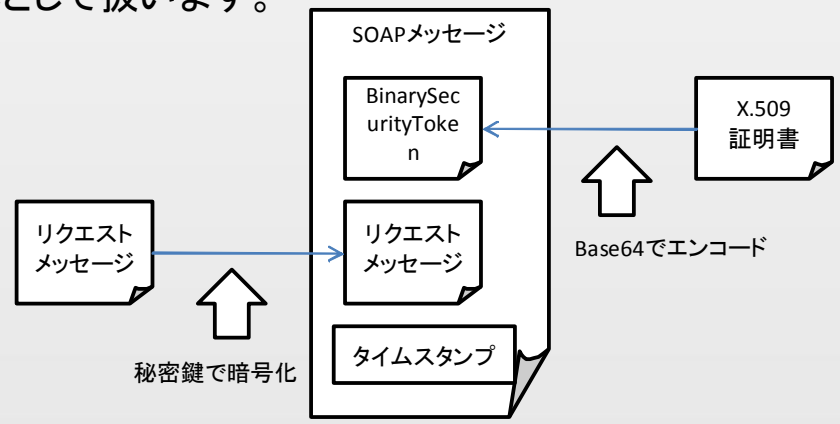
APIの認証

- Eucalyptusで提供されるAPIは第三者からの攻撃を防ぐため、またAPIを利用するユーザを特定するために認証を行います。認証の方式はSOAP/Query/REST APIで異なりますが、ユーザ情報は両APIで共通のユーザとなります。認証のための情報はEucalyptusのWeb UIから取得することができます。以下にWeb UIからダウンロードできるユーザの認証情報の一覧を提示します。

認証情報	説明
AccessKey	Query APIで利用する認証情報です。ユーザごとにAccessKeyが割当てられ、リクエストパラメータに含められます
SecretKey	Query APIで利用する認証情報です。ユーザごとにSecretKeyが割当てられ、このSecretKeyを用いて生成したSignatureがリクエストパラメータに含められます
X509証明書	SOAP APIで利用する認証情報です。ユーザごとにX509証明書が割当てられ、SOAPメッセージのBinarySecurityTokenにユーザの正当性を証明するものとして添付されます。
秘密鍵	SOAP APIで利用する秘密鍵です。ユーザごとに秘密鍵が割当てられ、XMLメッセージの署名のために利用されます。

SOAPメッセージの認証

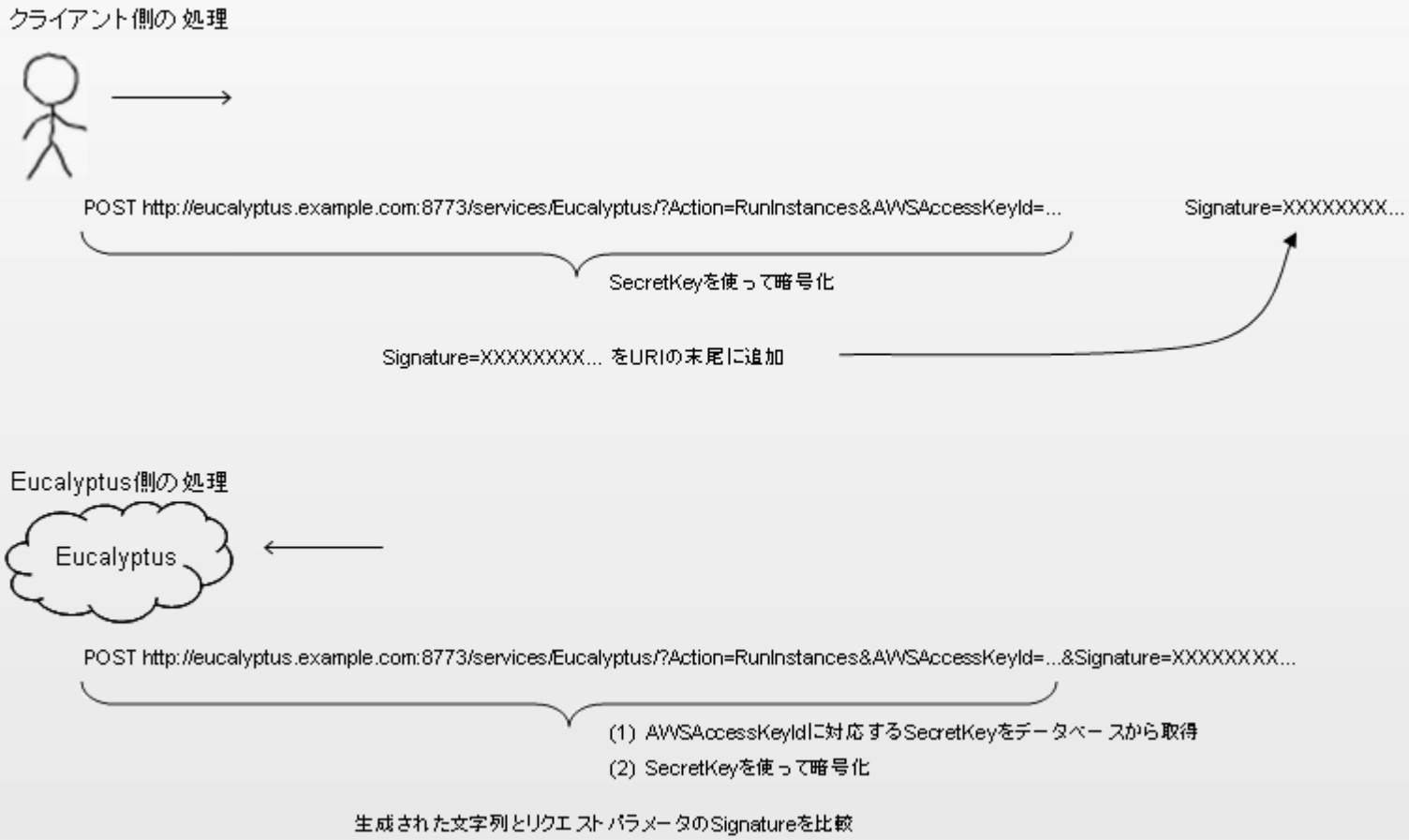
- SOAPメッセージを認証しリクエストを発行したユーザを特定するためにX.509証明書を利用します。また、WS-Securityの仕様に基づき、メッセージの暗号化と署名がおこなわれるため、ネットワークの盗み見やなり済ましなどの行為に対して安全性を保つことができます。
- 要求するリクエストメッセージはXML署名の仕様に従い秘密鍵で暗号化されます。暗号化されたリクエストメッセージを復号するためには秘密鍵に対応する公開鍵が必要となり、これはEucalyptusのデータベースに保存されています。ただし、このままだとどのユーザからのリクエストなのかを判別することができないため、X.509証明書をBase64でエンコードした状態でSOAPメッセージ内に配置されます。これをBinarySecurityTokenと呼びます。SOAPメッセージを受け取ったEucalyptusは、最初にX.509証明書を見てどのユーザから発行されたメッセージなのかを特定します。次にユーザの持つ公開鍵をデータベースから取り出し、リクエストメッセージを復号して処理します。
- なお、SOAPメッセージは発行時にタイムスタンプが付与されます。Eucalyptusではメッセージ発行から5分以上たったメッセージは期限切れとして扱います。



Query APIの認証 -1-

- Query APIを利用した場合、Eucalyptusに渡される情報はリクエストパラメータのみとなります。このため、リクエストパラメータにユーザIDに相当する情報を付与しています。これがAccessKeyです。ユーザごとに対応するAccessKeyがEucalyptusのデータベースに保存されているため、AccessKeyを送付することでそのリクエストが誰から発行されたものなのかを特定することができます。
- ただし、そのままではAccessKeyが盗まれてしまうと誰でもそのユーザになり済ますことができてしまいます。このため、そのメッセージを送付したユーザが正当であることを保証するためにAccessKeyと対になるSecretKeyを利用します。
- HTTP POSTでリクエストを発行する際、AccessKeyとともにSecretKeyを用いて生成したSignatureをパラメータに追加して、リクエストを送付します。リクエストを受け取ったEucalyptusはデータベース内に保持されている対応するSecretKeyを同じ手順で暗号化し、同じSignatureが生成されるかどうかで認証を行っています。これにより正しいSecretKeyを知るユーザからのメッセージか、不正なメッセージかを判断しています。

Query APIの認証 -2-



Query APIの認証 -3-

- Query APIではSignatureを生成する方式をSignatureVersionというバージョンで指定することができます。Query APIでは実際に利用するAPI名の他に、どのバージョンのSignature生成方式を利用したか、暗号化のアルゴリズムとして何を利用したかをそれぞれサーバに送信します。

SignatureVersion	署名を作るために必要なパラメータ
0	実行するAPI名である"Action"と"Timestamp"
1	HTTPパラメータすべて(HTTPメソッドや接続先のURIは含まない)
2	HTTPリクエスト全体(HTTPメソッドから接続先のURIすべてを含む) 推奨値

EC2 APIの演習

EC2 APIを利用できるライブラリ

- Amazon EC2のAPIを呼び出すためのライブラリは、Python、Ruby、Javaなどの各言語向けに多様な種類のライブラリが開発および公開されています。主なライブラリを下記に示します。

ライブラリ	言語	ライセンス	公式サイト
boto	Python	MIT License	http://code.google.com/p/boto/
RightAws	Ruby	MIT License	http://rubyforge.org/projects/rightscale
Typica	Java	Apache License 2.0	http://code.google.com/p/typica/
libcloud	Python	Apache License 2.0	http://incubator.apache.org/libcloud/
AWS SDK for Java	Java	Apache License 2.0	http://aws.amazon.com/sdkforjava/

主なライブラリの紹介

- boto
  - botoはPythonで記述され、EC2/S3/SQS/SDBなどに対応したAmazon Web Services用のAPIライブラリです。MITライセンスで提供されているOSSのため無償で利用することができます。euca2oolsにも採用されており、十分な利用実績があります。
- RightAws
  - RightAwsはRubyで記述され、EC2/S3/SQS/SDBなどに対応しており、boto同様MITライセンスで公開されています。RightAwsはRightScale社が開発・公開しています。RightScale社は、Amazon EC2やEucalyptusのようなクラウド上の仮想マシンを効率よく管理、運用する仕組みやWebインタフェースを提供しており、ライブラリも自社のサービスで使用しています。
- Typica
  - TypicaはJavaで記述され、EC2/SQS/CloudWatchなどに対応しており、Apache License 2.0で公開されているライブラリです。Typicaは動作確認にEucalyptus Community Cloudを使用しており、Eucalyptusで動作するAPIライブラリとして信頼があります。

RightAwsのセットアップ -1-

- RightAwsはRubyForge からダウンロードできますが、gemで簡単にインストールすることができます。なおRubyForgeで公開されているバージョンは1.10.0ですが、gemでインストールすると何故か3.0.4がインストールされます。gemを使用する場合は下記のコマンドを実行しインストールします。

```
# yum install rubygems.noarch
# gem install right_aws
```

RightAwsのセットアップ -2-

- RightAwsをEucalyptusで使用するための方法を説明します。基本的にはどのAPIを使用する場合も下記の手順を実行し作成したコネクションを介します。RightAwsもQuery APIを使用するためアクセスキーとシークレットキーを使用します。なお、ちょっと試してみる程度であればコード内でEC2エンドポイントやアクセスキーやシークレットキーをハードコーディングしてもよいですが、以下のように記述してeucarcで設定された環境変数を参照するか、もしくは実行時のオプション(ruby –s –ec2_access_key=VALUE, –ec2_secret_key=VALUE, –ec2_url=VALUE)で定義できるようにしたほうが取り回しがよくなります。

```
#!/usr/bin/ruby -s

# RightAwsのライブラリをロード
require 'right_aws'

# このスクリプトの実行時に -ec2_access_key, -ec2_secret_key,
# -ec2_url のオプションが設定されていない場合は、環境変数から
# 情報を取得する
$ec2_access_key ||= ENV['EC2_ACCESS_KEY'] if ENV.has_key?('EC2_ACCESS_KEY')
$ec2_secret_key ||= ENV['EC2_SECRET_KEY'] if ENV.has_key?('EC2_SECRET_KEY')
$ec2_url ||= ENV['EC2_URL'] if ENV.has_key?('EC2_URL')

# 上記の値を使用して、Eucalyptus に接続
@ec2 = RightAws::Ec2.new(
   $ec2_access_key, $ec2_secret_key, {
      :endpoint_url => $ec2_url,
      :eucalyptus => true,
   }
)
```

インスタンスの操作

- EC2 APIの内、もっとも重要なAPIがインスタンスを操作するAPIです。インスタンスを操作するAPIには下記の4つがあります。これらのうちRunInstances APIとDescribeInstances APIは非常に良く使用するAPIです。これより、インスタンスの起動、一覧の取得、再起動、停止を順番に解説します。

API	説明
RunInstances	インスタンスを起動します。一度に複数のインスタンスを起動することができます。
DescribeInstances	インスタンスの情報を取得します。ユーザの全インスタンスの情報を取得しますが、インスタンスIDを指定して特定のインスタンスの情報を取得することができます。
RebootInstances	インスタンスを再起動します。一度に複数のインスタンスを再起動することができます。
TerminateInstances	インスタンスを停止します。一度に複数のインスタンスを停止することができます。

インスタンスを起動 -1-

- インスタンスを起動するには、RunInstances APIを使用します。RunInstances APIを呼び出す際のパラメータと説明は下記の通りです。

パラメータ	説明	例	必須/任意
ImageId	起動するインスタンスのマシンイメージのIDを指定します。	emi-12345678	必須
MinCount	インスタンスの最小起動台数を設定します。クラウドに大して最低でも起動して欲しいインスタンスの台数を指定します。この台数を下回るキャパシティしかクラウドにない場合は、インスタンスの起動リクエストは受け付けられず、エラーが返却されます。	1	必須
MaxCount	インスタンスの最大起動台数を設定します。MinCountとセットで指定されます。MinCountとMaxCountを同じ値にすることで指定した台数起動できる場合はリクエストが受け付けられるリクエストを送付できます。	1	必須
KeyName	キーペア名を指定します。	key001	任意
SecurityGroup	セキュリティグループを指定します。	default	任意
InstanceType	インスタンスタイプを指定します。Eucalyptusでは下記の5つが使用できます。 ・ m1.small ・ c1.medium ・ m1.large ・ m1.xlarge ・ c1.xlarge	m1.small	任意
Placement	ゾーンを指定します。Eucalyptusでは、クラスタ名を指定します。	cluster01	任意
KernelId	カーネルイメージのIDを指定します。指定がない場合は、マシンイメージに紐づくカーネルイメージが使用されますが、そちらもない場合は、クラウドのデフォルト値に設定されているカーネルイメージが使用されます。	eki-12345678	任意
RamdiskId	ラムディスクイメージのIDを指定します。指定がない場合は、マシンイメージに紐づくラムディスクイメージが使用されますが、そちらもない場合は、クラウドのデフォルト値に設定されているラムディスクイメージが使用されます。	eri-12345678	任意
UserData	インスタンスから取得できるメタデータを設定します。様々なデータを自由に設定することができます。	user-data	任意

インスタンスを起動 -2-

- RunInstances APIを呼び出すとレスポンスとして下記のようなXMLが返却されます。

```
<RunInstancesResponse xmlns="http://ec2.amazonaws.com/doc/2010-08-31/">
  <requestId>582b4fab-3fbc-473b-adef-dde39655aa65</requestId>
  <reservationId>r-467008F0</reservationId>
  <ownerId>user001</ownerId>
  <groupSet>
    <item>
      <groupId>default</groupId>
    </item>
  </groupSet>
  <instancesSet>
    <item>
      <instanceId>i-442208CD</instanceId>
      <imageId>emi-37981294</imageId>
      <instanceState>
        <code>0</code>
        <name>pending</name>
      </instanceState>
      <privateDnsName>euca-0-0-0-0.eucalyptus.internal</privateDnsName>
      <dnsName>euca-0-0-0-0.eucalyptus.localhost</dnsName>
      <reason>NORMAL:  -- []</reason>
      <keyName>key01</keyName>
      <amiLaunchIndex>0</amiLaunchIndex>
      <productCodes/>
      <instanceType>m1.small</instanceType>
      <launchTime>2010-12-14T13:27:26.159Z</launchTime>
      <placement>
        <availabilityZone>cluster0</availabilityZone>
      </placement>
      <kernelId>eki-BB6A13B9</kernelId>
      <ramdiskId>eri-0DE3150A</ramdiskId>
      <monitoring>
        <state>false</state>
      </monitoring>
    </item>
  </instancesSet>
</RunInstancesResponse>
```

インスタンスを起動 -3-

- XMLに含まれる情報のうち重要な情報は下記の表の通りです。

情報	説明
requestId	どのAPIのレスポンスにも含まれるリクエスト毎に割り当てられるIDです。
reservationId	インスタンス起動リクエスト毎に発行されるIDです。1つのリザベーションに複数のインスタンスが所属します。
ownerId	インスタンスを起動したユーザのIDです。
groupSet	インスタンス起動時に指定したセキュリティグループです。
instanceId	インスタンスに一意に付けられるIDです。
imageId	マシンイメージのIDです。
instanceState	インスタンスの状態です。起動途中を表す「pending」、起動済みを表す「running」、停止中を表す「shutting-down」、停止を表す「terminated」の4つの状態が存在します。
privateDnsName	インスタンス同士の通信のみに利用可能なプライベートIP・DNSです。
dnsName	インスタンスにアクセスする際のパブリックIP・DNSです。
keyName	インスタンス起動時に指定したキーペアです。
amiLaunchIndex	リザベーション単位でインスタンスに割り振る番号です。一度のリクエストで複数台のインスタンスを起動した場合、0からインクリメントされたIDが付与されます。
instanceType	インスタンスのインスタンスタイプです。
launchTime	インスタンスの起動時刻です。
placement	インスタンスが起動したゾーン名です。Eucalyptusではクラスタ名が該当します。
kernelId	カーネルイメージのIDです。
ramdiskId	ラムディスクイメージのIDです。

インスタンスを起動 -4-

- RightAwsでRunInstancesを実行

```
# run_instancesメソッドによるインスタンスの起動
# 引数は、以下の5つが必須
# イメージID, 最小インスタンス数, 最大インスタンス数, グループID, キー名
instances = @ec2.run_instances(
   'emi-12345678', 1, 1, 'default', 'key001'
)

# launch_instancesメソッドによるインスタンスの起動
# 引数は、マシンイメージIDが必須
# launch_instances(image_id, options={})
instances = @ec2.launch_instances(
   'emi-12345678', {
      :group_ids => 'default', :key_name => 'key001',
      :instance_type => 'm1.small'
   }
)
```

インスタンスを起動 -5-

- RightAwsでRunInstancesを実行した戻り値

```
# 戻り値は以下のようなインスタンス情報が返る
pp instances
#=> [{:aws_kernel_id=>"eki-12345678",
#     :aws_reservation_id=>"r-12345678",
#     :aws_owner=>"vtaro",
#     :aws_instance_id=>"i-12345678",
#     :aws_ramdisk_id=>"eri-12345678",
#     :aws_groups=>["vtaro-default"],
#     :aws_state_code=>0,
#     :aws_image_id=>"emi-12345678",
#     :dns_name=>"0.0.0.0",
#     :monitoring_state=>"false",
#     :aws_instance_type=>"m1.small",
#     :aws_product_codes=>[],
#     :private_dns_name=>"0.0.0.0",
#     :aws_launch_time=>"2011-01-08T16:02:30.814Z",
#     :ami_launch_index=>"0",
#     :aws_availability_zone=>"cluster0",
#     :aws_state=>"pending",
#     :ssh_key_name=>"key001",
#     :aws_reason=>"NORMAL:  -- []"}]
```

インスタンス情報の取得 -1-

- 起動したインスタンスの一覧や情報を取得するには、DescribeInstances APIを使用します。DescribeInstancesはパラメータ無で実行することができますが、インスタンスIDを指定して特定のインスタンスの情報を取得することもできます。その際インスタンスIDは複数個指定することができます。
- DesribeInstances APIレスポンスのXMLには、RunInstancesのレスポンスと同様の情報が含まれています。ただし、RunInstancesのレスポンスには1つのReservationのみ含まれるのに対して、DescribeInstancesのレスポンスには複数のReservationが含まれています。

インスタンス情報の取得 -2-

- RightAwsでRunInstancesを実行

```
# 引数を省略した場合は当該ユーザのインスタンス情報が返る
# 引数は、インスタンスID を渡す。複数渡す場合は Array でも可
instance_ids = ['i-12345678', 'i-87654321']
instances = @ec2.describe_instances(instance_ids)

# 取得した情報を表示する
pp instances
#=> [{:aws_kernel_id=>"eki-12345678",
#     :aws_reservation_id=>"r-12345678",
#     :aws_owner=>"vtaro",
#     :aws_instance_id=>"i-12345678",
#     :aws_ramdisk_id=>"eri-12345678",
#     :aws_groups=>["default"],
#     :aws_state_code=>16,
#     :aws_image_id=>"emi-12345678",
#     :dns_name=>"192.0.2.100",
#     :monitoring_state=>"false",
#     :aws_instance_type=>"m1.small",
#     :aws_product_codes=>[],
#     :private_dns_name=>"172.19.1.34",
#     :aws_launch_time=>"2011-01-08T12:20:30.108Z",
#     :ami_launch_index=>"0",
```

インスタンス情報の取得 -3-

- RightAwsでRunInstancesを実行（続き）

```
#     :aws_availability_zone=>"cluster0",
#     :aws_state=>"running",
#     :ssh_key_name=>"key001",
#     :aws_reason=>"NORMAL:  -- [UPDATE]"},
#    {:aws_kernel_id=>"eki-87654321",
#     :aws_reservation_id=>"r-87654321",
#     :aws_owner=>"vtaro",
#     :aws_instance_id=>"i-87654321",
#     :aws_ramdisk_id=>"eri-87654321",
#     :aws_groups=>["default"],
#     :aws_state_code=>16,
#     :aws_image_id=>"emi-87654321",
#     :dns_name=>"192.0.2.101",
#     :monitoring_state=>"false",
#     :aws_instance_type=>"m1.small",
#     :aws_product_codes=>[],
#     :private_dns_name=>"172.19.1.35",
#     :aws_launch_time=>"2011-01-08T16:02:30.814Z",
#     :ami_launch_index=>"0",
#     :aws_availability_zone=>"cluster0",
#     :aws_state=>"running",
#     :ssh_key_name=>"key001",
#     :aws_reason=>"NORMAL:  -- [UPDATE]"}]
```

インスタンスの再起動

- 起動したインスタンスを再起動するには、RebootInstances APIを使用します。インスタンスを停止した場合はインスタンスのルートディスクに保存したデータはリセットされてしまいますが、RebootInstancesではインスタンスのルートディスクに保存したデータはリセットされません。RebootInstances APIでは複数個、最低1つのインスタンスIDをパラメータとして指定する必要があります。
- RightAwsでRebootInstancesを実行

```
# 引数は、インスタンスID を渡す。複数渡す場合は Array でも可
instance_ids = ['i-12345678', 'i-87654321']
result = @ec2.reboot_instances(instance_ids)

# 戻り値は true か例外が返る
pp result #=> true
```

インスタンスの停止

- 起動したインスタンスを停止するには、TerminateInstances APIを使用します。TerminateInstances APIを使用してインスタンスを停止した場合、インスタンスのルートディスクに保存したデータはすべて失われてしまいます。そのため、インスタンスを停止する前に大事なデータやイメージは保存しておくべきです。TerminateInstances APIはRebootInstances APIと同様に複数個、最低1つのインスタンスIDをパラメータとして指定する必要があります。
- RightAwsでTerminateInstancesを実行

```
# 引数は、インスタンスID を渡す。複数渡す場合は Array でも可
instance_ids = ['i-12345678', 'i-87654321']
result = @ec2.terminate_instances(instance_ids)

# 戻り値は true か例外が返る
pp result #=> true
```

ネットワークの操作 -1-

- ElasticIPを操作するAPI
  - ElasticIPを操作するAPIには下記の5つのAPIがあり、ElasticIPを確保・解放するAPIと、現在確保しているElasticIPの一覧を取得するAPIと、ElasticIPをインスタンスに取り付け・取り外すAPIがあります。

API	説明
AllocateAddress	ElasticIPを1つ確保するAPI
ReleaseAddress	確保しているElasticIPを1つ解放するAPI
DescribeAddresses	確保しているElasticIPの一覧を取得するAPI
AssociateAddress	確保しているElasticIPをインスタンスに取り付けるAPI
DisassociateAddress	インスタンスに取り付けているElasticIPを取り外すAPI

ネットワークの操作 -2-

- セキュリティグループを操作するAPI
  - セキュリティグループを操作するAPIには下記の5つのAPIがあり、セキュリティグループを作成・削除するAPIと、作成済みのセキュリティグループの一覧を取得するAPIと、セキュリティグループにルールを追加・削除するAPIがあります。

API	説明
CreateSecurityGroup	セキュリティグループを作成するAPI
DeleteSecurityGroup	セキュリティグループを削除するAPI
DescribeSecurityGroups	作成済みのセキュリティグループの一覧を取得するAPI
AuthorizeSecurityGroupIngress	セキュリティグループにルールを追加するAPI
RevokeSecurityGroupIngress	セキュリティグループからルールを削除するAPI

ElasticIPを確保する

- ElasticIPを確保するためには、AllocateAddressを使います。AllocateAddress APIを呼び出す際のパラメータはありません。

```
# 引数はない
elastic_ip = @ec2.allocate_address

# 戻り値は取得できたElasticIPの文字列が返る
pp elastic_ip #=> '192.0.2.100'
```

ElasticIPを解放する

- 確保したElasticIPを解放するためには、ReleaseAddressを使います。ReleaseAddress APIを呼び出す際のパラメータとして、確保しているElasticIP群に含まれているアドレスを指定しなければなりません。

```
# 引数は、確保している ElasticIP
result = @ec2.release_address('192.0.2.100')

# 戻り値は true か例外が返る
pp result #=> true
```

ElasticIPの一覧を取得する

- 確保したElasticIPの一覧を取得するためには、DescribeAddressesを使います。DescribeAddresses APIを呼び出す際のパラメータとして、確保しているElasticIPを指定することができますが、Eucalyptusの場合はパラメータを指定していても指定していなくとも、現在そのユーザが確保しているElasticIPの一覧が返ります。

```
# 引数はない
elastic_ips = @ec2.describe_addresses
# インスタンスへ取り付けられている場合はインスタンスIDの情報も返る
pp elastic_ips
# => [{:public_ip=>"192.0.2.124"},
#     {:public_ip=>"192.0.2.125", :instance_id=>"i-38720688"},
#     {:public_ip=>"192.0.2.126"},
#     {:public_ip=>"192.0.2.127"}]
```

ElasticIPをインスタンスに取り付ける

- 確保したElasticIPをインスタンスに取り付けるためには、AssociateAddressを使います。AssociateAddress APIを呼び出す際のパラメータには確保したElasticIPのアドレスとインスタンスIDを指定します。

```
# 引数は、インスタンスID, ElasticIP
result = @ec2.associate_address('i-38720688', '192.0.2.125')

# 戻り値は true か例外が返る
pp result #=> true
```

ElasticIPをインスタンスから取り外す

- 取り付けたElasticIPをインスタンスから取り外すためには、DisassociateAddressを使います。DisassociateAddress APIを呼び出す際のパラメータにはElasticIPのアドレスを指定します。

```
# 引数は、ElasticIP
result = @ec2.disassociate_address('192.0.2.125')

# 戻り値は true か例外が返る
pp result #=> true
```

セキュリティグループを作成する

- セキュリティグループを作成するためには、CreateSecurityGroupを使います。CreateSecurityGroup APIを呼び出す際のパラメータにはセキュリティグループ名とセキュリティグループの説明を指定します。なおセキュリティグループ名は必須ですが、RightAwsではセキュリティグループの説明を省略することが可能です。

```
# 引数は、セキュリティグループ名, セキュリティグループの説明
result = @ec2.create_security_group('webapps', 'Allowing SSH, HTTP and HTTPS ingress')

# 戻り値は true か例外が返る
pp result #=> true
```

セキュリティグループを削除する

- セキュリティグループを削除するためには、DeleteSecurityGroupを使います。DeleteSecurityGroup APIを呼び出す際のパラメータにはセキュリティグループ名を指定します。

```
# 引数は、セキュリティグループ名
result = @ec2.delete_security_group('webapps')

# 戻り値は true か例外が返る
pp result #=> true
```

セキュリティグループの一覧を取得する

- セキュリティグループの一覧を取得するためには、DescribeSecurityGroupsを使います。DescribeSecurityGroups APIを呼び出す際のパラメータにはセキュリティグループ名を指定することができます。なお、パラメータを指定しない場合は、そのユーザのセキュリティグループの一覧が返ります。

```
# 引数を省略した場合は所有する全てのセキュリティグループ情報が返る
# 引数は、セキュリティグループ名を渡す。複数渡す場合は Array でも可
security_groups = @ec2.describe_security_groups(['default', 'webapps'])

# 戻り値は以下のような情報が返る
pp security_groups
#=> [{:aws_owner=>"vtaro",
#     :aws_group_name=>"default",
#     :aws_description=>"default group",
#     :aws_perms=>[]},
#    {:aws_owner=>"vtaro",
#     :aws_group_name=>"webapps",
#     :aws_description=>"Allowing SSH, HTTP and HTTPS ingress",
#     :aws_perms=>
#      [{:from_port=>"22",
#        :to_port=>"22",
#        :protocol=>"tcp",
#        :cidr_ips=>"192.0.2.0/24"}]}]
```

セキュリティグループにルールを追加する -1-

- セキュリティグループにルールを追加するためには、AuthorizeSecurityGroupIngressを使います。AuthorizeSecurityGroupIngressを呼び出す際のパラメータには以下のものがあります。

ターメラパ	説明	例	必須/任意
GroupName	をルール追加プールグィテリュキセるす名を指定。	webapps	必須
IpProtocol	でルール指定。ルコトロプるすtcpかudpかicmpを指定。	tcp	必須
FromPort	でルール指定トーポるす番号。トーポ番号を範囲で指定るす場合は最小値を指定。tcp,udpの場合は1～65535でま指定、きでicmpの場合プイタは番号を指定。icmpの場合は-1を指定とるす全プイタのて番号を指定。	22	必須
ToPort	でルール指定トーポるす番号。トーポ番号を範囲で指定るす場合は最大値を指定。tcp,udpの場合は1～65535でま指定、きでicmpの場合プイタは番号に応ドーコたじ番号を指定。icmpの場合は-1を指定とるす全ドーコのて番号を指定。	22	必須
CidrIp	でルール接続を許可のクーワトッネるすCIDRを指定。以下のUserIdとGroupNameを指定いなし場合はターメラパのこは必須。	192.0.2.0/24	場合りよに必須
UserId	でルール接続を許可ザーユのプールグィテリュキセるすID。CidrIpを指定しいな場合はターメラパのこは必須。	012345678901	場合りよに必須
GroupName	でルール接続を許可のプールグィテリュキセるす名前。CidrIpを指定いなし場合はターメラパのこは必須。	default	場合りよに必須

セキュリティグループにルールを追加する -2-

- RightAwsでAuthorizeSecurityGroupIngressを実行

```
# CIDRを指定する場合はauthorize_security_group_IP_ingress()を使用
# 引数は、セキュリティグループ名，最小ポート番号，最大ポート番号，プロトコル，CIDR
result = @ec2.authorize_security_group_IP_ingress('webapps', 22, 22, 'tcp', '192.0.2.0/24')

# セキュリティグループを指定する場合はauthorize_security_group_named_ingress()を使用
# 引数は、セキュリティグループ名，ユーザID，セキュリティグループ名
result = @ec2.authorize_security_group_named_ingress('webapps', '0000-0000-0000', 'default')

# 戻り値は true か例外が返る
pp result #=> true
```

セキュリティグループからルールを削除する

- セキュリティグループからルールを削除するためには、RevokeSecurityGroupIngressを使います。RevokeSecurityGroupIngressを呼び出す際のパラメータはAuthorizeSecurityGroupIngressと同じです。

```
# CIDRを指定する場合はrevoke_security_group_IP_ingress()を使用
# 引数は、セキュリティグループ名，最小ポート番号，最大ポート番号，プロトコル，CIDR
result = @ec2.revoke_security_group_IP_ingress('webapps', 22, 22, 'tcp', '192.0.2.0/24')

# セキュリティグループを指定する場合はrevoke_security_group_named_ingress()を使用
# 引数は、セキュリティグループ名，ユーザID，セキュリティグループ名
result = @ec2.revoke_security_group_named_ingress('webapps', '0000-0000-0000', 'default')

# 戻り値は true か例外が返る
pp result #=> true
```

EBSの操作

- ファイルサーバやデータベースサーバなどのデータの永続化が必要な用途にインスタンスを使用する場合、EBSを利用する必要があります。EBSは、ボリュームのその時点でのスナップショットをとる機能も備わっており、非常に便利な機能です。EBSを操作するAPIには、下記のAPIがあります。

API	説明
CreateVolume	ボリュームを作成します。
DeleteVolume	ボリュームを削除します。
AttachVolume	ボリュームをインスタンスに取り付ける。
DetachVolume	ボリュームをインスタンスから取り外す。
DescribeVolumes	ボリュームの一覧を取得します。
CreateSnapshot	ボリュームのスナップショットを作成します。
DeleteSnapshot	スナップショットを削除します。
DescribeSnapshots	スナップショットの一覧を取得します。

ボリュームを作成する -1-

- ボリュームの作成には、CreateVolume APIを使用します。CreateVolume APIを呼び出す際のパラメータと説明は下記の通りです。

パラメータ	説明	例	必須/任意
Size	ボリュームのサイズを指定します。	1	SnapshotIdと本パラメータのどちらかが必須です。
SnapshotId	ボリュームの元となるスナップショットのIDを指定します。	Snap-12345678	Sizeと本パラメータのどちらかが必須です。
AvailabilityZone	ボリュームを利用するゾーンを指定します。Eucalyptusでは、クラスタ名をしてします。	cluster01	必須

ボリュームを作成する -2-

- RightAwsでCreateVolumeを実行します

```
# 引数は、スナップショットID, サイズ(GB単位), ゾーン名
# 新規のボリュームを作る場合
result = @ec2.create_volume(nil, 10, 'cluster0')

# 戻り値は以下のような情報が返る
pp result
#=> {:aws_status=>"creating",
#    :zone=>"cluster0",
#    :aws_created_at=>"2011-01-08T11:38:54.83Z",
#    :aws_size=>1,
#    :aws_id=>"vol-12345678"}

# 既存のスナップショットからボリュームを作る場合
# サイズに 0 を指定した場合はスナップショットと
# 同じサイズのボリュームを作成する
result = @ec2.create_volume('snap-12345678', 0, 'cluster0')

# 戻り値は以下のような情報が返る
pp result
#=> {:aws_status=>"creating",
#    :zone=>"cluster0",
#    :snapshot_id=>"snap-12345678",
#    :aws_created_at=>"2011-01-08T11:48:09.635Z",
#    :aws_size=>0,
#    :aws_id=>"vol-12345678"}
```

ボリュームを削除する

- ボリュームの削除には、DeleteVolume APIを使用します。DeleteVolume APIを呼び出す際のパラメータには削除対象のボリュームのIDを指定します。

```
# 引数は、ボリュームID
result = @ec2.delete_volume('vol-12345678')

# 戻り値は true か例外が返る
pp result #=> true
```

ボリュームの一覧を取得する

- ボリュームの一覧を取得するには、DescribeVolumes APIを使用します。DescribeVolumes APIを呼び出す際のパラメータにはボリュームのIDを複数個指定することができます。なお指定しない場合はそのユーザのすべてのボリュームが返却されます。

```
# 引数を省略した場合は所有する全てのボリューム情報が返る
# 引数は、ボリュームID を渡す。複数渡す場合は Array でも可
result = @ec2.describe_volumes(['vol-12345678', 'vol-87654321'])

# 戻り値は以下のような情報が返る
pp result
#=> [{:aws_attached_at=>"2011-01-08T12:21:46.148Z",
#     :aws_status=>"in-use",
#     :zone=>"cluster0",
#     :aws_instance_id=>"i-12345678",
#     :aws_created_at=>"2011-01-08T11:38:54.83Z",
#     :aws_size=>1,
#     :aws_id=>"vol-12345678",
#     :aws_device=>"/dev/sdb",
#     :aws_attachment_status=>"attached"},
#    {:aws_status=>"available",
#     :zone=>"cluster0",
#     :snapshot_id=>"snap-12345678",
#     :aws_created_at=>"2011-01-08T11:48:09.635Z",
#     :aws_size=>1,
#     :aws_id=>"vol-87654321"}]
```

ボリュームを取り付ける -1-

- ボリュームをインスタンスに取り付けるには、AttachVolume APIを使用します。AttachVolumeAPIを呼び出す際のパラメータには以下のものがあります。

パラメータ	説明	例	必須/任意
VolumeId	アタッチするボリュームのIDを指定します。	vol-12345678	必須
InstanceId	アタッチするインスタンスのIDを指定します。	i-12345678	必須
Device	アタッチするインスタンスのデバイスを指定します。	/dev/sdb	必須

ボリュームを取り付ける -2-

- RightAwsでAttachVolumeを実行する

```
# 引数は、ボリュームID, インスタンスID, デバイス名
result = @ec2.attach_volume('vol-12345678', 'i-12345678', '/dev/sdb')

# 戻り値は以下のような情報が返る
pp result
#=> {:aws_attached_at=>"2011-01-08T12:27:23.599Z",
#    :aws_instance_id=>"i-4E6A0821",
#    :aws_id=>"vol-5C210634",
#    :aws_device=>"/dev/sdb",
#    :aws_attachment_status=>"attaching"}
```

ボリュームを取り外す

- ボリュームをインスタンスから取り外すには、DetachVolume APIを使用します。DetachVolume APIを呼び出す際のパラメータには以下のものがあります。

パラメータ	説明	例	必須/任意
VolumeId	取り外すボリュームのIDを指定します。	vol-12345678	必須
InstanceId	ボリュームを取り外すインスタンスのIDを指定します。	i-12345678	任意
Device	ボリュームを取り外すインスタンスのデバイスを指定します。	/dev/sdb	任意

- RightAwsでDetachVolumeを実行する

```
# 引数は、ボリュームID
result = @ec2.detach_volume('vol-12345678')

# 戻り値は以下のような情報が返る
pp result
#=> {:aws_attached_at=>"2011-01-08T12:21:46.148Z",
#    :aws_instance_id=>"i-12345678",
#    :aws_id=>"vol-12345678",
#    :aws_device=>"/dev/sdb",
#    :aws_attachment_status=>"detaching"}
```

スナップショットを作成する

- ボリュームのスナップショットを作成するには、CreateSnapshot APIを使用します。CreateSnapshot APIを呼び出す際には、パラメータとしてボリュームのIDを指定します。

```
# 引数は、ボリュームID
result = @ec2.create_snapshot('vol-12345678')

# 戻り値は以下のような情報が返る
pp result
#=> {:aws_progress=>"0%",
#    :aws_owner=>"vtaro",
#    :aws_started_at=>"2011-01-08T11:42:18.826Z",
#    :aws_status=>"pending",
#    :aws_volume_size=>10,
#    :aws_id=>"snap-12345678",
#    :aws_volume_id=>"vol-12345678"}
```

スナップショットを削除する

- スナップショットを削除するには、DeleteSnapshot APIを使用します。DeleteSnapshot APIを呼び出すには、パラメータとして削除したいスナップショットのIDを指定します。

```
# 引数は、スナップショットID を渡す
result = @ec2.delete_snapshot('snap-87654321')

# 戻り値は true か例外が返る
pp result #=> true
```

スナップショットの一覧を取得する

- スナップショットの一覧を取得するには、DescribeSnapshots APIを使用します。DescribeSnapshots APIを呼び出す際には、パラメータとして情報を取得したいスナップショットのIDを複数個指定することが出来ます。なお、指定しない場合はユーザのすべてのスナップショットの情報が取得できます。

```
#= DescribeSnapshots
# 引数を省略した場合は所有する全てのスナップショット情報が返る
# 引数は、スナップショットID を渡す。複数渡す場合は Array でも可
result = @ec2.describe_snapshots(['snap-12345678', 'snap-87654321'])

# 戻り値は以下のような情報が返る
pp result
#=> [{:aws_progress=>"100%",
#     :aws_owner=>"vtaro",
#     :aws_started_at=>"2011-01-08T11:42:18.826Z",
#     :aws_status=>"completed",
#     :aws_volume_size=>1,
#     :aws_id=>"snap-12345678",
#     :aws_volume_id=>"vol-12345678"},
#    {:aws_progress=>"12%",
#     :aws_owner=>"vtaro",
#     :aws_started_at=>"2011-01-08T12:29:54.328Z",
#     :aws_status=>"pending",
#     :aws_volume_size=>-1,
#     :aws_id=>"snap-87654321",
#     :aws_volume_id=>"vol-12345678"}]
```

イメージの操作

- 仮想マシンイメージの登録や一覧の取得も重要なAPIです。仮想マシンイメージには、他のユーザも利用可能なパブリックなイメージにするかなどの属性も設定することができます。イメージを操作するAPIには、下記のAPIがあります。

API	説明
DescribeImages	仮想マシンイメージの情報を取得
RegisterImage	仮想マシンイメージを登録
DeregisterImage	仮想マシンイメージの登録を解除
DescribeImageAttribute	仮想マシンイメージの属性を取得
ModifyImageAttribute	仮想マシンイメージの属性を変更
ResetImageAttribute	仮想マシンイメージの属性をリセットする

イメージの一覧を取得する -1-

- 仮想マシンイメージの一覧を取得するには、DescribeImages APIを使用します。DescribeImages APIを呼び出す際のパラメータには以下のものがあります。

パラメータ	説明	例	必須/任意
ImageId	イメージのIDを指定します。複数個指定することができます。	emi-12345678	任意
Owner	イメージの所有者を指定します。複数個指定することができます。	user001	任意
ExecutableBy	イメージを起動可能なユーザを指定します。複数個指定することができます。	user001	任意

イメージの一覧を取得する -2-

- RightAwsでDescribeImagesを実行

```
# 引数を省略した場合は、当該ユーザが利用可能なイメージが返る
# 引数は、イメージID。複数指定する場合は Array で渡す。
result = @ec2.describe_images('emi-12345678')
pp result
#=> [{:aws_ramdisk_id=>"eri-12345678",
#     :aws_is_public=>true,
#     :aws_id=>"emi-12345678",
#     :aws_architecture=>"x86_64",
#     :root_device_name=>"/dev/sda1",
#     :root_device_type=>"instance-store",
#     :aws_location=>"centos.emi.006/centos55.img.manifest.xml",
#     :aws_image_type=>"machine",
#     :aws_state=>"available",
#     :aws_kernel_id=>"eki-12345678",
#     :aws_owner=>"admin"}]

# 引数に与えられたユーザ名などが起動できるイメージの一覧を取得
# 以下は自身が起動できるイメージの一覧を取得する例
result = @ec2.describe_images_by_executable_by('self')

# 引数に与えられたユーザが所有するイメージの一覧を取得
# 以下は自身が所有するイメージの一覧を取得する例
result = @ec2.describe_images_by_owner('self')
```

イメージを登録する

- 仮想マシンイメージを登録するには、RegisterImage API を使用します。RegisterImageを呼び出す際には、パラメータとしてWalrus上にアップロードしたイメージの情報を書いたマニフェストファイルを指定する必要があります。また、マニフェストファイルとバンドル済みのイメージはWalrus上にアップロードしておく必要があります。これらの必要なファイルの作成とアップロードはeuca2oolsで提供されている「euca-bundle-image」コマンドと「euca-upload-bundle」コマンドを使用して行うことができます。

```
# 引数は、イメージロケーション(バケット名/マニフェスト)
result = @ec2.register_image('vtaro.emi.001/centos.5-3.x86-64.img.manifest.xml')

# 戻り値はイメージID
pp result #=> emi-12345678
```

イメージを登録を解除する

- 仮想マシンイメージの登録を解除するには、DeregisterImage APIを使用します。
  DeregisterImage APIを呼び出す際には、パラメータとして登録を解除したいイメージのIDを指定します。

```
# 引数は、イメージID
result = @ec2.deregister_image('emi-12345678')

# 戻り値は true/false が返る
pp result #=> true
```

イメージの属性を取得する

- 仮想マシンイメージにはパブリックなイメージかどうかや誰が起動可能かなどの属性を持っています。これらの仮想マシンイメージの属性を取得するには、DescribeImageAttribute APIを使用します。DescribeImageAttribute APIを呼び出す際のパラメータは以下のものがあります。

パラメータ	説明	例	必須/任意
ImageId	属性を取得する仮想マシンイメージのIDを指定します。	emi-12345678	必須
Attribute	取得する属性を指定します。	launchPermission	必須

- Eucalyptusで指定可能なAttributeには以下のものがあります。

属性	説明
launchPermission	起動可能なユーザの設定です。グループ「all」が設定されている場合は、すべてのユーザが利用できます。プライベートなイメージにしたい場合は、このパーミッションを削除します。また、ユーザ単位で指定することができます。デフォルトでは自ユーザのみ設定されます。
kernel	仮想マシンイメージを起動する際に使用するカーネルイメージのIDです。
ramdisk	仮想マシンイメージを起動する際に使用するラムディスクイメージのIDです。

- RightAwsでDescribeImageAttributeを実行

```
# 引数は、イメージID, アトリビュート
result = @ec2.describe_image_attribute('emi-12345678', 'launchPermission')

# 戻り値は以下の情報が返る
pp result
#=> {:users=>["vtaro"], :aws_id=>"emi-12345678", :groups=>["all"]}
```

イメージの属性を変更する -1-

- 仮想マシンイメージの属性を変更するには、ModifyImageAttribute APIを使用します。ただし、ModifyImageAttribute APIを使用して変更可能な属性は「launchPermission」のみとなります。ModifyImageAttribute APIを呼び出す際のパラメータ以下のものがあります。

パラメータ	説明	例	必須/任意
ImageId	属性を変更するイメージIDを指定します。	emi-12345678	必須
Attribute	変更する属性を指定します。現在のところ、「launchPermission」のみしか指定できません。	launchPermission	必須
OperationType	属性を追加するか削除するかを指定します。属性の追加を場合は「add」、削除する場合は「remove」を指定します。	add	必須
UserId	launchPermissionに追加もしくは削除するユーザのIDを指定します。複数個指定することができます。本項目かGroupのいずれかを指定する必要があります。	user001	場合により必須
Group	launchPermissionに追加もしくは削除するグループのIDを指定します。現在のところ、「all」以外は指定できません。本項目かUserIdのいずれかを指定する必要があります。	all	場合により必須

イメージの属性を変更する -2-

- RightAwsでModifyImageAttributeを実行

```
# 引数は、イメージID，アトリビュート，オペレーション，アトリビュートの値
# サポートしているアトリビュートは 'launchPermission' のみ
# サポートしているオペレーションは 'add' と 'remove' のみ
# サポートしているアトリビュートの値は、以下。
# :user_group => 'all' のみ
# :user_id => ユーザ名
# :product_code => プロダクトコード
result = @ec2.modify_image_attribute('emi-12345678', 'launchPermission', 'add', :user_id => 'vjiro')

# 戻り値は ture か例外が返る
pp result #=> true

# RightAws では modify_image_attribute() より以下のメソッドの使用を推奨している
# それぞれのメソッドの戻り値は true か例外が返る

# 指定したイメージの起動権限にグループを追加 (ただしグループ all のみサポート)
@ec2.modify_image_launch_perm_add_groups('emi-12345678', ['all'])

# 指定したイメージの起動権限にユーザを追加
@ec2.modify_image_launch_perm_add_users('emi-12345678', ['vtaro', 'vjiro'])

# 指定したイメージの起動権限からグループを削除 (ただしグループ all のみサポート)
@ec2.modify_image_launch_perm_remove_groups('emi-12345678', ['all'])

# 指定したイメージの起動権限からユーザを削除
@ec2.modify_image_launch_perm_remove_users('emi-12345678', ['vtaro', 'vjiro'])
```

イメージの属性をリセットする

- ModifyImageAttribute APIを使用して変更した仮想マシンイメージの属性値をデフォルト値に戻したい場合には、ResetImageAttribute APIを使用します。ResetImageAttribute APIを呼び出す際には、パラメータとして仮想マシンイメージのIDとデフォルト値に戻したい属性を指定します。ただし、こちらもModifyImageAttribute APIと同様に指定可能な属性は「launchPermission」のみとなります。

```
# 引数は、イメージID, アトリビュート
result = @ec2.reset_image_attribute('emi-12345678', 'launchPermission')

# 戻り値は ture か例外が返る
pp result #=> true
```

S3 APIの演習

S3 APIを利用できるライブラリ

- EC2 APIを呼び出すライブラリと同様に、S3のAPIを呼び出すためのライブラリはPython、Ruby、Javaなどの各言語向けに多様な種類のライブラリが開発および公開されています。それらのうち以下の3つのライブラリについて紹介します。

ライブラリ	言語	ライセンス	公式サイト
boto	Python	MIT License	http://code.google.com/p/boto/
RightAws	Ruby	MIT License	http://rubyforge.org/projects/rightscale
JetS3t	Java	Apache License 2.0	http://jets3t.s3.amazonaws.com/index.html

主なライブラリの紹介

- boto
  - EC2 APIの項を参照
- RightAws
  - EC2 APIの項を参照
- JetS3t
  - JetS3tはJava言語で記述されたAPIライブラリで、S3の他にCloud FrontやGoogle Storage for Developersにも対応しています。Typicaの公式サイトでは、S3 APIを使用したい場合はJetS3tを使うようことを薦めており、Typicaと同じくApache License 2.0で公開されています。

RightAwsのセットアップ

- RightAwsもEC2 APIの場合と同様にインストールする方法は同じため省略します。RightAwsを使用してWalrusに接続するには下記のように記述します。

```
#!/usr/bin/ruby -s

# RightAwsのライブラリをロード
require 'right_aws'

# このスクリプトの実行時に -ec2_access_key, -ec2_secret_key,
# -ec2_url, -s3_url のオプションが設定されていない場合は、
# 環境変数から情報を取得する
$ec2_access_key ||= ENV['EC2_ACCESS_KEY'] if ENV.has_key?('EC2_ACCESS_KEY')
$ec2_secret_key ||= ENV['EC2_SECRET_KEY'] if ENV.has_key?('EC2_SECRET_KEY')
$ec2_url ||= ENV['EC2_URL'] if ENV.has_key?('EC2_URL')
$s3_url ||= ENV['S3_URL'] if ENV.has_key?('S3_URL')

# 上記の値を使用して、Walrus に接続
@s3 = RightAws::S3Interface.new(
   $ec2_access_key, $ec2_secret_key, {
      :endpoint_url => s3_url,
   }
)
```

バケットの操作

- バケットの操作のうち本講では以下について説明します
  - バケットの作成
  - バケットの削除
  - バケットの一覧取得

バケットを作成する

- バケットの作成は単純にバケット名を指定するだけです。ただし、バケット名はシステム全体で一意となります。ユーザ毎ではないため、他のユーザが作ったバケットと同名のバケットを作成することはできません。また、バケットの中にバケットを作成することはできません。

```
# バケットを作成する
# 引数は、バケット名を渡す。
# バケット名の頭文字が小文字の場合は VirtualHostingOfBuckets の仕様に
# 従ったバケット名として扱おうとするため、Walrus ではエラーになる。
# もし RightAws で頭文字が小文字のバケット名を利用する場合は、
# eucalyptus.conf にある DISABLE_DNS を "N" とする必要がある
# もしくは right_s3_interface.rb を修正する必要がある(詳しくはコラムを参照)
result = @s3.create_bucket('Test001')

# 戻り値は true か例外が返る
pp result #=> true
```

バケットを削除する

- バケットの削除方法を説明します。バケットの削除には注意すべき点が1つあります。それはバケットの中にオブジェクトが存在する場合はバケットの削除がエラーになることです。後述の方法でオブジェクトを削除してからバケットを削除する必要があります。

```
# バケットを削除する
# 引数は、バケット名を渡す。
result = @s3.delete_bucket('Test001')

# 戻り値は true か例外が返る
pp result #=> true
```

バケットの一覧を取得する

- バケットの一覧を取得する方法を説明します。バケットの一覧取得には特に指定すべきパラメータはありません。

```
# バケットの一覧を取得する
# 引数はない
result = @s3.list_all_my_buckets

# 戻り値は以下の情報が返る
pp result
#=> [{:creation_date=>"2011-01-08T15:16:13.000Z",
#     :owner_id=>"g8cNahGO2S16Q8h0idb46oWXkWV9dd8qnKNDA",
#     :name=>"vtaro.emi.001",
#     :owner_display_name=>"vtaro"}]
```

RightAwsでEucalyptusのバケットを使用するには

- 前述したようにRightAwsを使用する場合はVirtual Hosting of Bucketsの仕様に気をつける必要があります。EucalyptusでDISABLE_DNS="N"の設定をするか、もしくはバケット名の頭文字は大文字しか使用しないというのであれば、このRightAwsの仕様は気にしなくても良いのですが、それ以外の場合においては注意してRightAwsを使用する必要があります。もしDISABLE_DNS="Y"の設定を変更せずにRightAwsで頭文字が小文字のバケット名を使用する場合は以下のようにright_s3_interface.rbのfetch_request_paramsを修正する必要があります。

```
def fetch_request_params(headers) #:nodoc:
(中略)
path_to_sign = "#{service}/#{bucket_name}#{key_path}#{params_list}"
#path_to_sign = "/#{bucket_name}#{key_path}#{params_list}"
[ server, path, path_to_sign ]
end
```

オブジェクトの操作

- オブジェクトの操作のうち本講では以下について説明します
  - オブジェクトのアップロード
  - オブジェクトのダウンロード
  - オブジェクトの削除
  - オブジェクトの一覧取得

オブジェクトをアップロードする

- オブジェクトのアップロードを説明します。オブジェクトの名前をキー(key)と呼びます。同一バケットの中ではキーは一意にする必要があります。また、オブジェクトのキーとバケット名でオブジェクトを一意に特定することができます。

```
# オブジェクトをアップロードする
# 引数は、バケット名, キー名, データ
result = @s3.put('Test001', 'upload.dat', File.open('sample001.dat'))

# 戻り値は true か例外が返る
pp result #=> true
```

オブジェクトをダウンロードする

- アップロードしたオブジェクトをダウンロードします。オブジェクトのダウンロードは、バケット名とオブジェクトのキーを指定します。

```
# オブジェクトをダウンロードする
file = File.new('sample002.dat', File::CREAT|File::RDWR)
# 引数は、バケット名, キー名
result = @s3.get('Test001', 'upload.dat') do |chunk|
   file.write(chunk)
end
file.close

# 戻り値は以下の情報が返る
pp result
#=> {:headers=>
#     {"etag"=>"d6ab82f23a8861c3c5439d64e496f5e7",
#      "last-modified"=>"2011-01-08T18:41:29.000Z",
#      "content-type"=>"",
#      "content-length"=>"750"},
#    :object=>#<Net::ReadAdapter>}
```

オブジェクトを削除する

- オブジェクトの削除方法について説明します。オブジェクトの削除をするには、削除したいオブジェクトのキーとバケット名を指定します。

```
# オブジェクトを削除する
# 引数は、バケット名, オブジェクト名
result = @s3.delete('Test001', 'upload.dat')

# 戻り値は true か例外が返る
pp result #=> true
```

オブジェクトの一覧を取得する

- オブジェクトの一覧を取得します。オブジェクトの一覧を取得するには、バケット名を指定します。

```
# オブジェクトの一覧を取得する
# 引数は、バケット名
result = @s3.list_bucket('Test001')

# 戻り値は以下の情報が返る
pp result
#=> [{:key=>"upload.dat",
#     :owner_id=>"g8cNahGO2S16Q8h0idb46oWXkWV9dd8qnKNDA",
#     :service=>
#      {"name"=>"Test001",
#       "prefix"=>"",
#       "is_truncated"=>false,
#       "max-keys"=>"1000"},
#     :last_modified=>"2011-01-08T18:41:29.000Z",
#     :e_tag=>"d6ab82f23a8861c3c5439d64e496f5e7",
#     :size=>750,
#     :storage_class=>"STANDARD",
#     :owner_display_name=>"vtaro"}]
```

メタデータに関する演習

メタデータの概要

- メタデータとはインスタンスから特定のURLに対してリクエストを送ると、それに応じたインスタンス固有の値が返却される機能です。
- メタデータはCLCによって提供されますが、インスタンス外部からメタデータを要求するリクエストを送っても何ら情報を返しません。インスタンスから以下のようなURIに対して要求を送信することにより、そのインスタンスに対して情報を返します。

```
http://169.254.169.254/latest/meta-data/instance-id
       (1)          (2)       (3)          (4)
```

(1) メタデータ取得用IPアドレス
(2) メタデータのAPIバージョン
(3) メタデータを取得するのかユーザデータを取得のかを指定
(4) 取得するメタデータの名前

- (1)のメタデータ取得用IPアドレスは、どのEucalyptusクラウドでもどのインスタンスでも常に169.254.169.254になります。
- (2)のlatestはメタデータAPIのバージョンを指定します。なお、Amazon EC2互換としてyyyy-mm-dd形式も受け付けますが、latest以外のバージョンを指定しても常にlatestの内容が返ります。
- (3)はuser-dataとmeta-dataがあり 、user-dataはインスタンス起動時に指定したデータが取得でき、meta-dataは(4)のメタデータの名前と組合せて使用することにより、インスタンスに関する情報が取得できます。

メタデータの一覧

- Eucalyptus 2系で実装されている取得可能なメタデータは以下になります。

メタデータ名	説明	例
ami-id	インスタンスが起動しているイメージのイメージID	emi-4ABBHG6
ami-launch-index	インスタンスの起動番号。インスタンス起動時の同時起動数によって変わる	1
ami-manifest-path	起動イメージのWalrus上のでの保存先	/bucket/image.manifest.xml
hostname	Eucalyptusの場合はパブリックIPが返却される	192.0.2.100
instance-id	インスタンスのインスタンスID	i-88EADG45
instance-type	インスタンス起動時に指定したインスタンスタイプ	m1.large
local-hostname	EucalyptusでDISABLE_DNS="N"を設定した場合はプライベートIPに対応したFQDNが返り、それ以外は下記のlocal-ipv4と同じ値が返ります。	euca-10-1-1-10.eucalyptus.internal
local-ipv4	インスタンスのプライベートIP	10.1.1.10
public-hostname	EucalyptusでDISABLE_DNS="N"を設定した場合はパブリックIPに対応したFQDNが返り、それ以外は下記のpublic-ipv4と同じ値が返ります。	euca-192-0-2-100.eucalyptus.localhost
public-ipv4	インスタンスのパブリックIP	192.0.2.100
public-keys/0/openssh-key	インスタンスの起動時に指定したキーペアの公開鍵	ssh-rsa AAAAB...9l5vR
reservation-id	インスタンス起動時に生成されるリザベーションID	r-A67FG231
kernel-id	インスタンスが利用しているカーネルイメージのイメージID	eki-9345FEC
ramdisk-id	インスタンスが利用しているラムディスクイメージのイメージID	eri-872EEFC
security-groups	インスタンスが所属するセキュリティグループ名	default
placement/availability-zone	インスタンスが所属するクラスタ名	cluster1

メタデータの利用について

- メタデータはHTTPでアクセスすることができれば取得することが可能なため、特殊な方法を用いる必要がありません。つまり大抵のLinux環境にインストールされているwgetコマンドやcurlコマンドで簡単に取得できます。
- 以降にメタデータを利用するサンプルを紹介します。

SSH用の公開鍵をインスタンスに設定 -1-

- Eucalyptusでは、インスタンスを起動する前にNC上にてインスタンスのディスクに対し直接SSH用の公開鍵を追加するという方法 を実装しています。しかしこの実装は/root/.ssh/authorized_keysにしか追加しないため、インスタンス上に作成した一般ユーザのauthorized_keysに設定することができません。そこで以下のようなスクリプトを作成し、起動時に呼び出すようにすることでrootユーザ以外のユーザにもSSH用の公開鍵を設定することができます(以下のサンプルコードではインスタンスを起動したEucalyptusのユーザ名と同じユーザをインスタンスに作成しています)。

SSH用の公開鍵をインスタンスに設定 -2-

- 以下のスクリプトは/etc/rc.localから呼び出されるように設定すると起動時に自動的に実行されます。

```
#!/bin/bash

url1='http://169.254.169.254'
url2='/latest/meta-data/public-keys/0/openssh-key'
WGET=`which wget`
CURL=`which curl`
NC=`which nc`
if [ -n "${WGET}" ]; then
   public_key=`${WGET} -O - ${url1}${url2} 2>/dev/null`
elif [ -n "${CURL}" ]; then
   public_key=`${CURL} ${url1}${url2} 2>/dev/null`
elif [ -n "${NC}" ]; then
   public_key=`echo -e "GET ${url2} HTTP/1.0¥n" ¥
               | nc 169.254.169.254 80 2>/dev/null | tail -n 1`
fi

if [ -n "${public_key}" ]; then
   user_name=`echo ${public_key} | cut -d' ' -f3 | cut -d'@' -f1`
   [[ -z `id ${user_name} 2>/dev/null` ]] && useradd -m ${user_name}
   mkdir -p /home/${user_name}/.ssh
   echo ${public_key} > /home/${user_name}/.ssh/authorized_keys
   chmod 400 /home/${user_name}/.ssh/authorized_keys
fi
```

インスタンスのホスト名を設定 -1-

- Eucalyptus社が提供している仮想マシンイメージではインスタンスのホスト名は常に固定(CentOSのイメージではlocalhost、Ubuntuのイメージではubuntu)となります。そこで以下のようなスクリプトを作成し、起動時に呼び出すようにすることでAmazon EC2のインスタンスのような感じのホスト名を設定することができます。

インスタンスのホスト名を設定 -2-

- 以下のスクリプトは/etc/rc.localから呼び出されるように設定すると起動時に自動的に実行されます。

```
#!/bin/bash

ip2fqdn () {
   pattern1='¥([0-9]*¥)¥.¥([0-9]*¥)¥.¥([0-9]*¥)¥.¥([0-9]*¥)'
   pattern2='euca-¥1-¥2-¥3-¥4.eucalyptus.internal'
   echo $1 | sed -e "s/${pattern1}/${pattern2}/"
}

url='http://169.254.169.254/latest/meta-data/'
WGET=`which wget`
CURL=`which curl`
NC=`which nc`
if [ -n "${WGET}" ]; then
   local_hostname=`${WGET} -O - ${url}local-hostname 2>/dev/null`
   local_ipv4=`${WGET} -O - ${url}local-ipv4 2>/dev/null`
elif [ -n "${CURL}" ]; then
   local_hostname=`${CURL} ${url}local-hostname 2>/dev/null`
   local_ipv4=`${CURL} ${url}local-ipv4 2>/dev/null`
elif [ -n "${NC}" ]; then
   local_hostname=`echo -e "GET /latest/meta-data/local-hostname HTTP/1.0¥n"¥
                   | nc 169.254.169.254 80 2>/dev/null | tail -n 1`
   local_ipv4=`echo -e "GET /latest/meta-data/local-ipv4 HTTP/1.0¥n"¥
                   | nc 169.254.169.254 80 2>/dev/null | tail -n 1`
fi
```

インスタンスのホスト名を設定 -3-

- （前ページの続き）

```
if [ -n "${local_hostname}" ]; then
   if [ "${local_hostname}" == "${local_ipv4}" ]; then
      local_hostname=`ip2fqdn "${local_ipv4}"`
   fi
else
   local_hostname=`ip2fqdn "${local_ipv4}"`
fi

hostname ${local_hostname}
echo ${local_hostname} > /etc/hostname
echo -e "${local_ipv4}¥t¥t${local_hostname}" >> /etc/hosts
```