Tutorial

AWS and Cisco Metacloud™ API Compatibility

API compatibility is important to customers who are currently using AWS but are interested in either interoperating with private cloud resources or repatriating AWS application stacks back to the private cloud. For example, building dev/test for code development in AWS, but running production in an on-premises private cloud. AWS-compatible APIs allow for lifecycle management of both types of clouds using familiar AWS tools and processes.

After reading this tutorial, the reader will have a clear understanding of OpenStack's API compatibility with AWS, as well as insight as to how Metacloud excels in this regard compared to VMware and other private cloud solutions with incompatible APIs.

Specifically, we will explore the following topics:

- Introduction to Cisco Metacloud
- Overview of OpenStack's API compatibility with AWS
- Supported EC2 and EBS compatible features
- Overview of API tool installation and configuration examples

Introduction to Cisco Metacloud

Cisco Metacloud delivers a true public cloud experience for customers on their premises and behind their firewalls. It offers full administrative control and multi-tenancy. It's a production-ready, OpenStack-based solution that Cisco engineers, deploys, upgrades, and remotely operates on the customer's behalf, 24 hours a day, 365 days a year. This managed OpenStack distribution allows our customers to deliver IT as a service to their own lines of business as if the IT department were their own public cloud provider. Metacloud is truly the OpenStack easy button!

The remainder of this tutorial will focus on Metacloud's support of the AWS APIs.

Overview of OpenStack's API compatibility with AWS

At a high level there are two levels of API compatibility between AWS and OpenStack.

The first is API syntax. This refers to the actual API calls being available at the same location with the same parameters. If two APIs are syntactically compatible, you can use a client that was designed for one with the other. For example, the EC2 and Euca2ool APIs are syntactically compatible, whereas the EC2 and VMware's APIs are not. Euca2ool will be demonstrated later in the tutorial.

The second and subtler method is API semantics. Two semantically compatible APIs allow you to manipulate equivalent objects and relationships at a commensurate level of abstraction. For example, the EC2 and OpenStack native APIs are (mostly) semantically compatible, but the EC2 and CloudFormation APIs aren't.

In fact, orchestration is an area where semantically compatible tools are quite important. Currently compatibility exists at several levels in OpenStack in general and Neutron specifically. Some of these capabilities exist in Heat as well, an OpenStack service that uses templates to orchestrate multiple composite cloud applications. Heat also has an OpenStack-native REST API as well as a query API compatible with Amazon Web Services' (AWS).

Heat and AWS CloudFormation are an example of semantically compatible tools, as both tools can support the OpenStack cloud operating system. Orchestration templates can function as a layer of abstraction which Heat can use the native Yaml format or AWS CFN to build and manipulate application stacks to native OpenStack or AWS compatible APIs in Metacloud.

The main benefit of API compatibility is choice. The customer can use their preferred syntax and/or semantically compatible tools when building dual AWS and OpenStack environments.

EC2-Compatible APIs

In addition to the native compute API, OpenStack provides an EC2-compatible API. This API allows EC2 legacy workflows to work with OpenStack. As mentioned earlier this is a syntax approach to compatibility.

Numerous third-party tools and language-specific SDKs can be used to interact with OpenStack clouds, using both native and compatible APIs. The following table lists the more popular third-party tools with syntax compatibility:

Table 1. EC2-compatible API syntax tools

Tool	Description
Euca2ool	A popular open source command-line tool for interacting with the EC2 API. This is convenient for multi-cloud environments where EC2 is the common API, or for transitioning from EC2-based clouds to OpenStack. For more information, see the euca2ools site.
Hybridfox	A Firefox browser add-on that provides a graphical interface to many popular public and private cloud technologies, including OpenStack. For more information, see the hybridfox site.
boto	A Python library for interacting with Amazon Web Services. It can be used to access OpenStack through the EC2 compatibility API. For more information, see the boto project page on GitHub.
fog	A Ruby cloud services library. It provides methods for interacting with a large number of cloud and virtualization platforms, including OpenStack. For more information, see the fog site.
php-opencloud	A PHP SDK designed to work with most OpenStack-based cloud deployments, as well as Rackspace public cloud. For more information, see the php-opencloud site.

Table 2. EC2-compatible API semantic tools

Tool	Description
Heat	The OpenStack orchestration service that allows you to spin up multiple instances, logical networks, and other cloud services in an automated fashion. CFN: Short for AWS CloudFormation, this is the second template format that is supported by Heat. CFN-formatted templates are typically expressed in JSON.
Packer by HashiCorp	Packer supports creating images for several platforms or targets. Amazon EC2 (AMI): Both EBS-backed and instance-store AMIs within EC2, optionally distributed to multiple regions. OpenStack: Images for OpenStack that can be used to start pre-configured OpenStack servers.

Supported EC2 and EBS compatible features

Currently the AWS compatible APIs support approximately 80% of the main EC2 and EBS functionality when running in OpenStack. The following tables will provide the matrix of OpenStack support and lack thereof in several categories. For a detailed list of supported and non-supported features, please see: https://wiki.openstack.org/wiki/Nova/APIFeatureComparison

The following tables include only the supported features for using the AWS compatible APIs in OpenStack.

Table 3. General API support

Feature	OpenStack
EC2 Query API	Y
OpenStack API / Rackspace API	Y
Horizontal Component Scalability	Y
Web-based UI	Y
Command line interface	Y

Table 4. Amazon EC2 High Level Feature Support

EC2 feature	OpenStack
Shared AMIs	Y
Parameterized launch (user-data)	Y
Instance metadata	Y
Public AMIs	Y
Launch/Terminate Instance	Y

Reboot Instance	Y
Start/Stop Persisted Instance	Y
Retrieve Console Output	Y
Multiple Instance Types	Y
Instance Launch Time	Y
Elastic IPs	Y
Availability Zones	Y
Region Support	Y
User selectable kernels	Y
Elastic Block Store	Y
Booting without a ramdisk	Y
Windows Support	Y
AMIs backed by EBS	Y
Import keypair	Y

Table 5. Amazon EC2 API Compatibility

EC2 API method	OpenStack
AllocateAddress	Y
AssociateAddress	Y
AttachVolume	Y
AuthorizeSecurityGroupIngress	Y
CreateKeyPair	Y
CreateSecurityGroup	Y
CreateSnapshot	Y
CreateVolume	Y
DeleteKeyPair	Y
DeleteSecurityGroup	Y
DeleteSnapshot	Y
DeleteVolume	Y
DeregisterImage	Y
DescribeAddresses	Y
DescribeAvailabilityZones	Y
DescribeImageAttribute	Y
DescribeImages	Y
DescribeInstances	Y
DescribeKeyPairs	Y
DescribeRegions	Y
DescribeSecurityGroups	Y
DescribeSnapshots	Y
DescribeVolumes	Y
DetachVolume	Y
DisassociateAddress	Y
GetConsoleOutput	Y
ImportKeyPair	Y

ModifyImageAttribute	Y
RebootInstances	Y
RegisterImage	Y
ReleaseAddress	Y
RevokeSecurityGroupIngress	Y
RunInstances	Y
StartInstances	Y
StopInstances	Y
TerminateInstances	Y

Overview of API tool installation and configuration examples

In this section we will install the native OpenStack API tool and Euca2ool and configure IaaS resources on both AWS and Metacloud. The native OpenStack API tool is not AWS compatible but used as a point of reference. The Euca2ool is a standalone open source API tool based on the commercially available tool Eucalyptus, which was acquired by HP. Euca2ool is one of many available syntax compatible options and was chosen because it is simple to install.

OpenStack API tool installation

In a standard OpenStack installation, there are two different interfaces for managing cloud resources. One is via Horizon, the web-based OpenStack dashboard, and the other is via the OpenStack command line interface (CLI).

Before using the command line to communicate with your OpenStack environment you will need to download and install the CLI tools for your particular OS. Below are the instructions that explain how to install the prerequisite software and the Python package for each OpenStack client.

http://docs.openstack.org/user-guide/common/cli_install_openstack_command_line_clients.html
https://support.metacloud.com/entries/100998896-Installing-Command-line-clients
https://support.metacloud.com/entries/100996126-Available-OpenStack-Services-Clients

Euca2ool API tool installation

Upon installing OpenStack CLI on the platform of choice (Linux, MAC, etc.) we can subsequently install the Euca2ool. Euca2ools is the Eucalyptus command line interface for interacting with web services. This set of tools was written in Python.

Most Euca2ools commands are compatible with Amazon's EC2, S3, IAM, Auto Scaling, Elastic Load Balancing, CloudFormation, Virtual Private Cloud (VPC), and CloudWatch services and generally accept the same options and honor the same environment variables. For the purposes of this tutorial we are only focusing on EC2.

This section covers how to install Euca2ools.

- Installing Euca2ools on RHEL / CentOS
- Installing Euca2ools on Mac OS X

Using Euca2ool API tool in AWS

After Euca2ools is installed we can create a bash script to log into EC2 using our AWS credentials. The AWS credentials are obtained using the AWS console.

```
mkdir ~/.euca; chmod 0700 ~/.euca; cd ~/.euca

vi aws.sh
```

Add the following AWS credentials and information to the file, and save.

```
#!/bin/bash

export EC2_ACCESS_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

export EC2_SECRET_KEY=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

export EC2_URL=https://ec2.us-east-1.amazonaws.com

esc

:wq
```

```
source ~/.euca/aws.sh   This will load the credentials.

euca-describe-instances
```

```
tdubiel@DevOps:~/.euca$ euca-describe-instances
RESERVATION     r-58db89a8      255712896315
INSTANCE        i-e2e5d84a      ami-45ae712e    ec2-54-172-157-121.compute-1.amazonaws.com      ip-172-31-
internal        running API-key 0               t1.micro        2015-07-30T23:14:08.000Z        us-east-1c
499ccb20                        monitoring-disabled     54.172.157.121  172.31.61.17    vpc-dfeba1ba    su
c92     ebs                                     paravirtual     xen             sg-96a835f2     default fa
BLOCKDEVICE     /dev/sda        vol-f57f341f    2015-07-30T23:14:11.000Z        true
NIC     eni-eee6f9c1    subnet-b940ec92 vpc-dfeba1ba    255712896315    in-use  172.31.61.17    ip-172-31-
internal        true
NICATTACHMENT           0       attached        2015-07-30T23:14:08.000Z        true
NICASSOCIATION  54.172.157.121  amazon  {'privateDnsName': 'ip-172-31-61-17.ec2.internal', 'association':
': '54.172.157.121', 'publicDnsName': 'ec2-54-172-157-121.compute-1.amazonaws.com', 'ipOwnerId': 'amazon'}
': 'true', 'privateIpAddress': '172.31.61.17'}
```

Describe a specific instance ID

```
euca-describe-instances i-708640d8
```

```
tdubiel@DevOps:~/.euca$ euca-describe-instances i-708640d8
RESERVATION     r-266ccad6      255712896315
INSTANCE        i-708640d8      ami-1100cf7a    ec2-54-165-194-131.compute-1.amazonaws.com      ip
.internal       running jclouds#:192-168-20-7-WebSrv1   0               t1.micro        2015-07-07
s-east-1d       aki-499ccb20                    monitoring-disabled     54.165.194.131  172.31.49.
subnet-b940ec92 ebs                                     paravirtual     xen             sg-940047f
e
BLOCKDEVICE     /dev/sda        vol-ed4ded07    2015-07-07T17:49:52.000Z        true
NIC     eni-f14435de    subnet-b940ec92 vpc-dfeba1ba    255712896315    in-use  172.31.49.149   ip
.internal       true
NICATTACHMENT           0       attached        2015-07-07T17:49:49.000Z        true
NICASSOCIATION  54.165.194.131  amazon  {'privateDnsName': 'ip-172-31-49-149.ec2.internal', 'assoc
p': '54.165.194.131', 'publicDnsName': 'ec2-54-165-194-131.compute-1.amazonaws.com', 'ipOwnerId':
y': 'true', 'privateIpAddress': '172.31.49.149'}
GROUP   sg-940047f0     secgrp-AWSDubiel-VPC-1429189981
PRIVATEIPADDRESS        172.31.49.149
TAG     instance        i-708640d8      Name    WebSrv1
TAG     instance        i-708640d8      VNMC_RES_ID     00051a4c-6532-ff11-0005-1a4c6532ff11
```

Now let's list the available AMI images and launch one that was created from a smaller micro-size instance type.

```
euca-describe-images
```

```
IMAGE    ami-45ae712e    255712896315/WebSrvMicro        255712896315    available        p
499ccb20                         ebs     paravirtual     xen
BLOCKDEVICEMAPPING      /dev/sda        snap-f2c13288   2       true
```

```
euca-run-instances ami-45ae712e -n 1 -t t1.micro -k API-key
```

```
tdubiel@DevOps:~/.euca$ euca-run-instances ami-45ae712e -n 1 -t t1.micro -k API-key
RESERVATION     r-58db89a8      255712896315
INSTANCE        i-e2e5d84a      ami-45ae712e            ip-172-31-61-17.ec2.internal    pending API-key 0          t
1.micro 2015-07-30T23:14:08.000Z        us-east-1d      aki-499ccb20                    monitoring-disabled        1
72.31.61.17     vpc-dfeba1ba    subnet-b940ec92 ebs                                     paravirtual     xen        s
g-96a835f2      default false
NIC     eni-eee6f9c1    subnet-b940ec92 vpc-dfeba1ba    255712896315    in-use  172.31.61.17    ip-172-31-61-17.ec2.
internal        true
NICATTACHMENT           0       attaching       2015-07-30T23:14:08.000Z        true
GROUP   sg-96a835f2     default
```

Parse through the above output and locate the instance ID created from the above AMI image file.

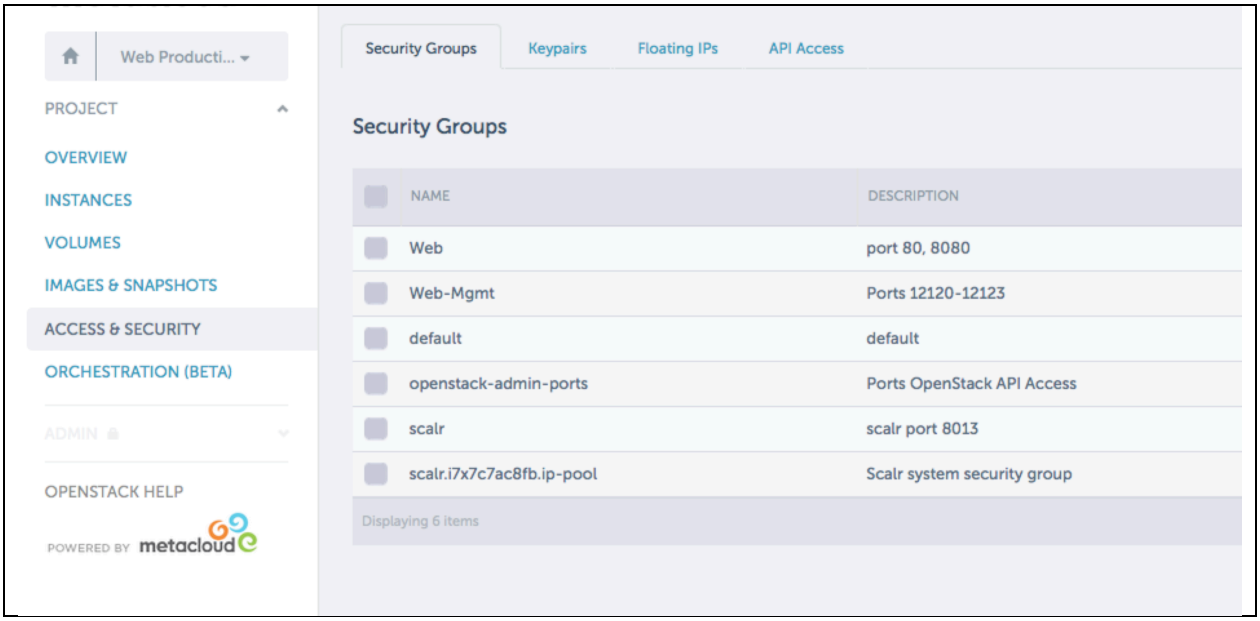
```
euca-describe-instances i-e2e5d84a
```

```
tdubiel@DevOps:~/.euca$ euca-describe-instances i-e2e5d84a
RESERVATION     r-58db89a8      255712896315
INSTANCE        i-e2e5d84a      ami-45ae712e
015-07-30T23:14:08.000Z us-east-1d      aki-499ccb20
bs                                      paravirtual     xen
```

Cisco Metacloud and API tool configuration

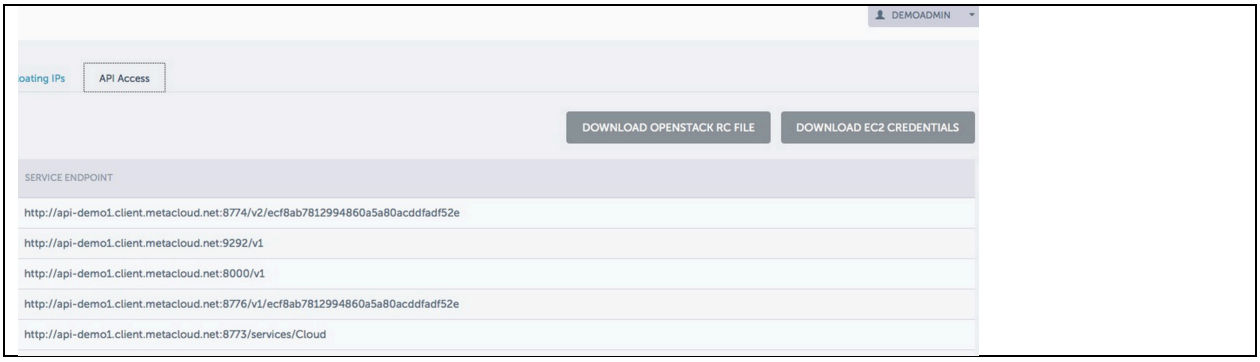
After you have the core CLI utilities installed you next need to get the login script from the OpenStack instance you are talking to, (or you can build one from scratch). To get the login CLI script from the OpenStack instance you first need to login, then you will need to navigate to the Access and Security Tab on the left navigation menu.

Figure 1. Access & Security [API Access]



Next, navigate to the API tab on the top of the panel and notice the two buttons at the top for RC file downloads. One is for the OpenStack CLI and the other is to use the EC2 CLI tools.

Figure 2. Download Credentials



Select the DOWNLOAD OPENSTACK RC file, which will download the RC file to your local computer. Also download EC2 credentials. Next you need to unzip and run those two batch files. The OpenStack RC requires the tenant to authenticate into the project you downloaded the RC file from. We will revisit this step below. Please note if using a remote instance instead of your local laptop to run the CLI tools than use SCP to copy the RC files.

Stash the EC2 credentials in ~/.euca:

```
cp ~/ yourproject-API-x509.zip ~/.euca; unzip *.zip

Finally let's source the rc scripts:

source ~/.euca/ec2rc.sh

source ~/ yourproject-openrc.sh
```

You'll see the openrc.sh script asks you for a password.

AWS Compatibility API examples

First step is to test that the Euca2ool and OpenStack CLI are working.

```
euca-describe-instances
```

```
tdubiel@DevOps:~/.euca$ euca-describe-instances
RESERVATION     r-0708417f      3e29ed249d0c47d3808902698f75b35d        default
INSTANCE        i-00000577      ami-0000002b    38.84.67.182    image-upload    running ansible (3e29ed249d(
902698f75b35d, mhv2.trial2.mc.metacloud.in)     0               m1.medium       2015-03-08T22:24:37.000Z
12      aki-0000002e    ari-00000031                    38.84.67.182    10.2.11.3                       inst
re
TAG     instance        i-00000577      latest_hv_ver   1005004
TAG     instance        i-00000577      domain_hv_ver   1005003
RESERVATION     r-584iu8qd      0031936806a546bdb56bab676038cbf2        SSH
INSTANCE        i-00000c35      ami-0000008b    38.84.67.172    screech-test    running ND-POC (0031936806a5
ab676038cbf2, mhv2.trial2.mc.metacloud.in)      0               m1.tiny 2015-05-29T12:57:40.000Z        tria
8.84.67.172     10.2.8.2                        instance-store
TAG     instance        i-00000c35      latest_hv_ver   1005004
TAG     instance        i-00000c35      domain_hv_ver   1005003
RESERVATION     r-7ne6802e      0e5f5920402d4017833fbaeb43951413        default
INSTANCE        i-000005a4      ami-00000001            abdi-server1    running None (0e5f5920402d4017833fba
13, mhv3.trial2.mc.metacloud.in)        0               m1.tiny 2015-03-12T21:00:57.000Z        trial2
```

Considering this is quite a bit of output, lets look at a specific instance ID.

euca-describe-instances i-00000eea

```
tdubiel@DevOps:~/.euca$ euca-describe-instances i-00000eea
RESERVATION     r-sf3wutsc      ee006b4d4ca9461db152792a77d75046        Puppet
INSTANCE        i-00000eea      ami-000000a1    38.84.67.184    puppetagent1    running None (ee006b4d4ca
2a77d75046, mhv1.trial2.mc.metacloud.in)        0               m1.small        2015-07-21T17:10:31.000Z
12                                      38.84.67.184    10.2.7.2                        instance-store
TAG     instance        i-00000eea      latest_hv_ver   1005004
TAG     instance        i-00000eea      domain_hv_ver   1005004
tdubiel@DevOps:~/.euca$
```

In order to test the OpenStack CLI, enter:

Nova list

```
+--------------------------------------+-----------------------------------------------------+--------+------------+-------
-----+--------------------------------------+
| ID                                   | Name                                                | Status | Task State | Power
tate | Networks                             |
+--------------------------------------+-----------------------------------------------------+--------+------------+-------
-----+--------------------------------------+
| 4b17bca8-2421-4de4-b021-474adba0a948 | API-CLI                                             | ACTIVE | -          | Runnir
     | trial2-3007=10.2.7.4, 38.84.67.181 |
| c870785a-452e-43a1-9240-f4f8fac3f94e | Puppetagent1                                        | ACTIVE | -          | Runnir
     | trial2-3007=10.2.7.2, 38.84.67.184 |
```

Launch a new instance in Metacloud using the AWS API from an existing AMI image.

euca-describe-images

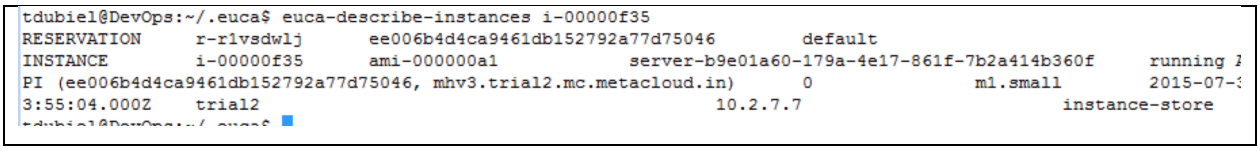
```
IMAGE   ami-000000a1    None (puppetagent1)     ee006b4d4ca9461db152792a77d75046        available       public
achine                          instance-store
IMAGE   ami-000000aa    None (Lampstack1)       8e2004521bf944e09d0e2f8f31ceb4d0        available       public
```

euca-run-instances ami-000000a1 -n 1 -t m1.small -k AWSAPI

```
tdubiel@DevOps:~/.euca$ euca-run-instances ami-000000a1 -n 1 -t m1.small -k AWSAPI
RESERVATION     r-r1vsdwlj      ee006b4d4ca9461db152792a77d75046        default
INSTANCE        i-00000f35      ami-000000a1            server-b9e01a60-179a-4e17-861f-7b2a414b360f     p
PI (ee006b4d4ca9461db152792a77d75046, None)     0               m1.small        2015-07-30T23:55:04.000Z
12                                                              instance-store
```

Afterwards, we can check the status of the new instance we launched.

euca-describe-instances i-00000f35



Summary

AWS-compatible APIs for OpenStack are valuable for customers with permanent or incremental requirements to run dual clouds, such as AWS in the public cloud and OpenStack in the private cloud. There are several use cases for running dual clouds, however the two main use cases are dev/test and repatriation. The former is when code is developed in AWS but production is deployed in the on-premises OpenStack cloud; the latter is for permanently migrating production stacks out of AWS to the on-premises OpenStack cloud. Whether your DevOps teams are more familiar with the AWS API syntax or simply prefer using AWS syntax versus the OpenStack APIs, AWS-compatible APIs support this option. It allows them to benefit from syntax-based tools, such as Euc2ool to perform VM lifecycle management tasks (launch, power-on, terminate, and import, etc.) on both clouds. For more complex tasks, such as deploying purpose-built stacks from templates to include network tiers, instances, and L4-7 services, CloudFormation templates can be used for orchestration in both AWS and Metacloud. Lastly, to avoid code rewrite, applications that were written originally to communicate with AWS APIs can leverage the AWS-compatible APIs for OpenStack as well.

For More Information

Visit our website to read more about Cisco Metacloud features and benefits.
To access technical tutorials about this product, visit our Community page.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**