cloudbridge Documentation

Release 0.1

GVL and Galaxy Projects

Sep 28, 2017

Contents

CloudBridge aims to provide a simple layer of abstraction over different cloud providers, reducing or eliminating the need to write conditional code for each cloud.

CHAPTER 1

Usage example

The simplest possible example for doing something useful with CloudBridge would look like the following.

```
from cloudbridge.cloud.factory import CloudProviderFactory, ProviderList

provider = CloudProviderFactory().create_provider(ProviderList.AWS, {})
print(provider.compute.instances.list())
```

In the example above, the AWS_ACCESS_KEY and AWS_SECRET_KEY environment variables must be set to your cloud credentials.

CHAPTER 2

Quick Reference

The following object graph shows how to access various provider services, and the resource that they return. Click on any object to drill down into its details.

CHAPTER 3

Installation

The latest release can always be installed form PyPI. For other installation options, see the installation page:

```
pip install cloudbridge
```

CHAPTER 4

Documentation

Concepts and Organisation

Conceptually, CloudBridge consists of the following types of objects.

1. Providers - Represents a connection to a cloud provider, and is the gateway to using its services.

2. Services - Represents a service provided by a cloud provider, such as its compute service, block storage service, object storage etc. Services may in turn be divided into smaller services. Smaller services tend to have uniform methods, such as create, find and list. For example, InstanceService.list(), InstanceService.find() etc. which can be used to access cloud resources. Larger services tend to provide organisational structure only. For example, the block store service provides access to the VolumeService and SnapshotService.

3. Resources - resources are objects returned by a service, and represent a remote resource. For example, InstanceService.list() will return a list of Instance objects, which can be used to manipulate an instance. Similarly, VolumeService.create() will return a Volume object.

The actual source code structure of CloudBridge also mirrors this organisation.

Detailed class relationships

The following diagram shows a typical provider object graph and the relationship between services.

Some services are nested. For example, to access the instance service, you can use *provider.compute.instances*. Similarly, to get a list of all instances, you can use the following code.

```
instances = provider.compute.instances.list()
print(instances[0].name)
```

Getting Started

This getting started guide will provide a quick tour of some CloudBridge features. For more details on individual features, see the Using CloudBridge section or the API reference.

Installation

CloudBridge is available on PyPI so to install the latest available version, run:

```
pip install --upgrade cloudbridge
```

Create a provider

To start, you will need to create a reference to a provider object. The provider object identifies the cloud you want to work with and supplies your credentials. The following two code snippets setup a necessary provider object, for AWS and OpenStack. For the details on other providers, take a look at the Setup page. The remainder of the code is the same for either provider.

AWS:

```
from cloudbridge.cloud.factory import CloudProviderFactory, ProviderList

config = {'aws_access_key': 'AKIAJW2XCYO4AF55XFEQ',
          'aws_secret_key': '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'}
provider = CloudProviderFactory().create_provider(ProviderList.AWS, config)
image_id = 'ami-2d39803a'  # Ubuntu 14.04 (HVM)
```

OpenStack (with Keystone authentication v2):

```
from cloudbridge.cloud.factory import CloudProviderFactory, ProviderList

config = {'os_username': 'username',
          'os_password': 'password',
          'os_auth_url': 'authentication URL',
          'os_region_name': 'region name',
          'os_project_name': 'project name'}
provider = CloudProviderFactory().create_provider(ProviderList.OPENSTACK,
                                                  config)
image_id = 'c1f4b7bc-a563-4feb-b439-a2e071d861aa'  # Ubuntu 14.04 @ NeCTAR
```

OpenStack (with Keystone authentication v3):

```
from cloudbridge.cloud.factory import CloudProviderFactory, ProviderList

config = {'os_username': 'username',
          'os_password': 'password',
          'os_auth_url': 'authentication URL',
          'os_project_name': 'project name',
          'os_project_domain_name': 'project domain name',
          'os_user_domain_name': 'domain name'}
provider = CloudProviderFactory().create_provider(ProviderList.OPENSTACK,
                                                  config)
image_id = '97755049-ee4f-4515-b92f-ca00991ee99a'  # Ubuntu 14.04 @ Jetstream
```

List some resources

Once you have a reference to a provider, explore the cloud platform:

```
provider.compute.images.list()
provider.security.security_groups.list()
provider.block_store.snapshots.list()
provider.object_store.list()
```

This will demonstrate the fact that the library was properly installed and your provider object is setup correctly but it is not very interesting. Therefore, let's create a new instance we can ssh into using a key pair.

Create a key pair

We'll create a new key pair and save the private portion of the key to a file on disk as a read-only file.

```
kp = provider.security.key_pairs.create('cloudbridge_intro')
with open('cloudbridge_intro.pem', 'w') as f:
    f.write(kp.material)
import os
os.chmod('cloudbridge_intro.pem', 0400)
```

Create a security group

Next, we need to create a security group and add a rule to allow ssh access. A security group needs to be associated with a private network, so we'll also need to fetch it.

```
provider.network.list()  # Find a desired network ID
net = provider.network.get('desired network ID')
sg = provider.security.security_groups.create(
    'cloudbridge_intro', 'A security group used by CloudBridge', net.id)
sg.add_rule('tcp', 22, 22, '0.0.0.0/0')
```

Launch an instance

We can now launch an instance using the created key pair and security group. We will launch an instance type that has at least 2 CPUs and 4GB RAM. We will also add the network interface as a launch argument.

```
img = provider.compute.images.get(image_id)
inst_type = sorted([t for t in provider.compute.instance_types.list()
                    if t.vcpus >= 2 and t.ram >= 4],
                   key=lambda x: x.vcpus*x.ram)[0]
inst = provider.compute.instances.create(
    name='CloudBridge-intro', image=img, instance_type=inst_type,
    network=net, key_pair=kp, security_groups=[sg])
# Wait until ready
inst.wait_till_ready()  # This is a blocking call
# Show instance state
inst.state
# 'running'
```

Assign a public IP address

To access the instance, let's assign a public IP address to the instance. For this step, we'll first need to allocate a floating IP address for our account and then associate it with the instance.

```
fip = provider.network.create_floating_ip()
inst.add_floating_ip(fip.public_ip)
inst.refresh()
inst.public_ips
# [u'54.166.125.219']
```

From the command prompt, you can now ssh into the instance `ssh -i cloudbridge_intro.pem ubuntu@54.166.125.219`.

Cleanup

To wrap things up, let's clean up all the resources we have created

```
inst.terminate()
from cloudbridge.cloud.interfaces import InstanceState
inst.wait_for([InstanceState.TERMINATED, InstanceState.UNKNOWN],
               terminal_states=[InstanceState.ERROR])  # Blocking call
fip.delete()
sg.delete()
kp.delete()
os.remove('cloudbridge_intro.pem')
router.remove_route(sn.id)
router.detach_network()
router.delete()
sn.delete()
net.delete()
```

And that's it - a full circle in a few lines of code. You can now try the same with a different provider. All you will need to change is the cloud-specific data, namely the provider setup and the image ID.

Using CloudBridge

Introductions to all the key parts of CloudBridge you'll need to know:

Installation

Prerequisites: CloudBridge runs on Python 2.7 and higher. Python 3 is recommended.

We highly recommend installing CloudBridge in a virtualenv. Creating a new virtualenv is simple:

```
pip install virtualenv
virtualenv .venv
source .venv/bin/activate
```

Latest release

The latest release of cloudbridge can be installed from PyPI:

```
pip install cloudbridge
```

Latest unreleased dev version

The development version of the library can be installed from the Github repo:

```
$ git clone https://github.com/gvlproject/cloudbridge.git
$ cd cloudbridge
$ python setup.py install
```

Developer installation

To install additional libraries required by CloudBridge contributors, such as tox, clone the source code repository and run the following command from the repository root directory:

```
pip install -e .[dev]
```

To check what version of the library you have installed, do the following:

```
import cloudbridge
cloudbridge.get_version()
```

Setup

To initialize a connection to a cloud and get a provider object, you will need to provide the cloud's access credentials to CloudBridge. These may be provided in one of following ways:

1. Environment variables
2. A dictionary
3. Configuration file

Providing access credentials through environment variables

The following environment variables must be set, depending on the provider in use.

Amazon

Mandatory variables	Optional Variables
AWS_ACCESS_KEY	
AWS_SECRET_KEY	

Openstack

Mandatory variables	Optional Variables
OS_AUTH_URL	NOVA_SERVICE_NAME
OS_USERNAME	OS_COMPUTE_API_VERSION
OS_PASSWORD	OS_VOLUME_API_VERSION
OS_PROJECT_NAME	OS_STORAGE_URL
OS_REGION_NAME	OS_AUTH_TOKEN

Once the environment variables are set, you can create a connection as follows:

```
from cloudbridge.cloud.factory import CloudProviderFactory, ProviderList

provider = CloudProviderFactory().create_provider(ProviderList.OPENSTACK, {})
```

Providing access credentials through a dictionary

You can initialize a simple config as follows. The key names are the same as the environment variables, in lower case. Note that the config dictionary will override environment values.

```
from cloudbridge.cloud.factory import CloudProviderFactory, ProviderList

config = {'aws_access_key' : '<your_access_key>',
          'aws_secret_key' : '<your_secret_key>'}
provider = CloudProviderFactory().create_provider(ProviderList.AWS, config)
```

Some optional configuration values can only be provided through the config dictionary. These are listed below for each provider.

CloudBridge

Variable	Description
default_result_limit	Number of results that a `.list()` method should return. Defaults to 50.

Amazon

Variable	Description
aws_session_token	Session key for your AWS account (if using temporary credentials).
ec2_is_secure	True to use an SSL connection. Default is `True`.
ec2_region_name	Default region name. Defaults to `us-east-1`.
ec2_region_endpoint	Endpoint to use. Default is `ec2.us-east-1.amazonaws.com`.
ec2_port	EC2 connection port. Does not need to be specified unless EC2 service is running on an alternative port.
ec2_conn_path	Connection path. Defaults to `/`.
ec2_validate_certs	Whether to use SSL certificate verification. Default is `False`.
s3_is_secure	True to use an SSL connection. Default is `True`.
s3_host	Host connection endpoint. Default is `s3.amazonaws.com`.
s3_port	Host connection port. Does not need to be specified unless S3 service is running on an alternative port.
s3_conn_path	Connection path. Defaults to `/`.
s3_validate_certs	Whether to use SSL certificate verification. Default is `False`.

Providing access credentials in a file

CloudBridge can also read credentials from a file on your local file system. The file should be placed in one of two locations: `/etc/cloudbridge.ini` or `~/.cloudbridge`. Each set of credentials should be delineated with the provider ID (e.g., `openstack`, `aws`) with the necessary credentials being supplied in YAML format. Note that only one set of credentials per cloud provider type can be supplied (i.e., via this method, it is not possible to provide credentials for two different OpenStack clouds).

```
[openstack]
os_username: username
os_password: password
os_auth_url: auth url
os_user_domain_name: user domain name
```

```
os_project_domain_name: project domain name
os_project_name: project name

[aws]
aws_access_key: access key
aws_secret_key: secret key
```

Other configuration variables

In addition to the provider specific configuration variables above, there are some general configuration environment variables that apply to CloudBridge as a whole

Launching instances

Before being able to run below commands, you will need a `provider` object (see this page).

Common launch data

Before launching an instance, you need to decide what image to launch as well as what type of instance. We will create those objects here. The specified image ID is a base Ubuntu image on AWS so feel free to change it as desired. For instance type, we're going to let CloudBridge figure out what's the appropriate name on a given provider for an instance with at least 2 CPUs and 4 GB RAM.

```
img = provider.compute.images.get('ami-f4cc1de2')  # Ubuntu 16.04 on AWS
inst_type = sorted([t for t in provider.compute.instance_types.list()
                    if t.vcpus >= 2 and t.ram >= 4],
                   key=lambda x: x.vcpus*x.ram)[0]
```

When launching an instance, you can also specify several optional arguments such as the security group, a key pair, or instance user data. To allow you to connect to the launched instances, we will also supply those parameters (note that we're making an assumption here these resources exist; if you don't have those resources under your account, take a look at the Getting Started guide).

```
kp = provider.security.key_pairs.find(name='cloudbridge_intro')[0]
sg = provider.security.security_groups.list()[0]
```

Launch an instance

Once we have all the desired pieces, we'll use them to launch an instance:

```
inst = provider.compute.instances.create(
    name='CloudBridge-VPC', image=img, instance_type=inst_type,
    key_pair=kp, security_groups=[sg])
```

Private networking

Private networking gives you control over the networking setup for your instance(s) and is considered the preferred method for launching instances. To launch an instance with an explicit private network, supply a subnet within a network as an additional argument to the `create` method:

```
provider.network.list()  # Find a desired network ID
net = provider.network.get('desired network ID')
sn = net.subnets()[0]  # Get a handle on the desired subnet to launch with
inst = provider.compute.instances.create(
    name='CloudBridge-VPC', image=img, instance_type=inst_type,
    subnet=sn, key_pair=kp, security_groups=[sg])
```

For more information on how to create and setup a private network, take a look at Networking.

Block device mapping

Optionally, you may want to provide a block device mapping at launch, specifying volume or ephemeral storage mappings for the instance. While volumes can also be attached and mapped after instance boot using the volume service, specifying block device mappings at launch time is especially useful when it is necessary to resize the root volume.

The code below demonstrates how to resize the root volume. For more information, refer to *LaunchConfig*.

```
lc = provider.compute.instances.create_launch_config()
lc.add_volume_device(source=img, size=11, is_root=True)
inst = provider.compute.instances.create(
    name='CloudBridge-BDM', image=img,  instance_type=inst_type,
    launch_config=lc, key_pair=kp, security_groups=[sg])
```

where `img` is the `Image` object to use for the root volume.

After launch

After an instance has launched, you can access its properties:

```
# Wait until ready
inst.wait_till_ready()  # This is a blocking call
inst.state
# 'running'
```

Depending on the provider's networking setup, it may be necessary to explicitly assign a floating IP address to your instance. This can be done as follows:

```
# List all the IP addresses and find the desired one
provider.network.floating_ips()
# Assign the desired IP to the instance
inst.add_floating_ip('149.165.168.143')
inst.refresh()
inst.public_ips
# [u'149.165.168.143']
```

Private networking

Private networking gives you control over the networking setup for your instance(s) and is considered the preferred method for launching instances. Also, providers these days are increasingly requiring use of private networks.

If you do not explicitly specify a private network to use when launching an instance, CloudBridge will attempt to use a default one. A 'default' network is one tagged as such by the native API. If such tag or functionality does not exist, CloudBridge will look for one with a predefined name (by default, called 'CloudBridgeNet', which can be overridden with environment variable `CB_DEFAULT_NETWORK_NAME`).

Create a new private network

Creating a private network is a simple, one-line command but appropriately connecting it so it has Internet access is a multi-step process: (1) create a network; (2) create a subnet within this network; (3) create a router; (4) attach the router to an external network; and (5) add a route to the router that links with a subnet. For some providers, any network can be external (ie, connected to the Internet) while for others it's a specific, pre-defined one that exists in the an account by default. In order to properly connect the router, we need to ensure we're using an external network.

When creating the subnet, we need to set an address pool. We can obtain the private network address space via network object's `cidr_block` field (e.g., `10.0.0.0/16`). Below, we'll create a subnet starting from the beginning of the block and allow up to 16 IP addresses into the subnet (`/28`).

```
net = provider.network.create('cloudbridge_intro')
sn = net.create_subnet('10.0.0.0/28', 'cloudbridge-intro')
router = provider.network.create_router('cloudbridge-intro')
if not net.external:
    for n in self.provider.network.list():
        if n.external:
            net = n
            break
router.attach_network(net.id)
router.add_route(sn.id)
```

Retrieve an existing private network

If you already have existing networks, we can query for those:

```
provider.network.list()  # Find a desired network ID
net = provider.network.get('desired network ID')
```

Object states and lifecycles

Overview

Some objects, namely, Instances, Volumes, Snapshots and Images, have a specific life-cycle and a set of states they can be in. For example, an Instance may be in state RUNNING and a volume maybe in state AVAILABLE. These provider specific states are mapped to a common vocabulary of states in CloudBridge.

In addition, it is common to want to wait for objects to reach a particular state. For example, wait for an instance to transition from PENDING->RUNNING. To facilitate this, all such objects in CloudBridge implement the `ObjectLifeCycleMixin`. Each object with a lifecycle has a state property, a refresh method and a wait_for method.

The state property will return the object's current state, and the refresh() method can be used to requery the server and update the object's state. The wait_for method can be used to wait for the object to transition to a desired state or set of states.

There is also a convenience method named wait_till_ready(), which will wait for the object to reach a ready-to-use state. A ready-to-use state would mean that the object has been successfully created and can be interacted with, and is not in an error state. Since this can be tedious to get right, the wait_till_ready() method encapsulates this behaviour. For example - in the case of an Instance, wait_till_ready() will internally call the wait_for method as follows:

```
self.wait_for(
    [InstanceState.RUNNING],
    terminal_states=[InstanceState.TERMINATED, InstanceState.ERROR],
    timeout=timeout,
    interval=interval)
```

This would cause the wait_for method to repeatedly call refresh() till the object's state reaches RUNNING. It will raise a `WaitStateException` if the timeout expires, or the object reaches a terminal state, such as TERMINATED or ERROR, in which case it is no longer reasonable to wait for the object to reach a running state.

Informational states and actionable states

As in the wait_for example above, some states are purely informational, and some states are actionable. Informational states are meant to provide information to the end-user, and should generally not be used by program logic to take actions. For example, it would be incorrect to write a wait_for function as follows:

```
wait_for([InstanceState.PENDING], ...)
```

This is because the PENDING state is fleeting and may occur too fast to be reliably detected by the client. In contrast, a state such as RUNNING is more stable and program logic can reasonably take actions based on such states. In the discussion that follows, we will specifically differentiate between informational and actionable states.

Instance states and lifecycle

The following states are defined for a CloudBridge Instance.

State	Category	Description
UNKNOWN	actionable	Instance state is unknown. This means that the instance does not exist or CloudBridge does not know how to map this state.
PENDING	informational	Instance is pending
CONFIGURING	informational	Instance is being reconfigured in some way and may not be usable.
RUNNING	actionable	Instance is running.
REBOOTING	informational	Instance is rebooting.
TERMINATED	actionable	Instance is terminated. No further operations possible.
STOPPED	actionable	Instance is stopped. Instance can be resumed.
ERROR	actionable	Instance is in an error state. No further operations possible.

The lifecycle diagram is as follows:

Actionable states are shown in blue and informational states in cyan. Note that any state could potentially transition into an ERROR state. An Instance may initially be in an UNKNOWN state and transition into a PENDING state on launch. Similarly, it may transition into an UNKNOWN state after TERMINATION and the object no longer exists. More rarely, an instance may transition into an UNKNOWN state if CloudBridge does not know how to map the state reported by the cloud provider. Therefore, when writing a wait_for method, these potential transitions should be taken into account.

Volume states and lifecycle

The following states are defined for a CloudBridge Volume.

State	Category	Description
UNKNOWN	actionable	Volume state is unknown. This means that the volume does not exist or CloudBridge does not know how to map this state.
CREATING	informational	Volume is pending
CONFIGURING	informational	Volume is being reconfigured in some way and may not be usable.
AVAILABLE	actionable	Volume is unattached and available for use.
IN_USE	informational	Volume is attached to an instance and in-use.
DELETED	actionable	Volume has been deleted. No further operations possible.
ERROR	actionable	Volume is in an error state. No further operations possible.

The lifecycle diagram is as follows:

Actionable states are shown in blue and informational states in cyan. Note that any state could potentially transition into an ERROR state. A Volume may initially be in an UNKNOWN state and transition into a CREATING state when created anew. Similarly, it may transition into an UNKNOWN state after DELETED and the object no longer exists. More rarely, a volume may transition into an UNKNOWN state if CloudBridge does not know how to map the state reported by the cloud provider. A Volume will typically transition through a CONFIGURING stage before going to an IN_USE stage and vice versa.

Snapshot states and lifecycle

The following states are defined for a CloudBridge Snapshot.

State	Category	Description
UNKNOWN	actionable	Snapshot state is unknown. This means that the snapshot does not exist or CloudBridge does not know how to map this state.
PENDING	informational	Snapshot is pending
CONFIGURING	informational	Snapshot is being reconfigured in some way and may not be usable.
AVAILABLE	actionable	Snapshot is ready.
ERROR	actionable	Snapshot is in an error state. No further operations possible.

The lifecycle diagram is as follows:

Actionable states are shown in blue and informational states in cyan. Note that any state could potentially transition into an ERROR state. A Snapshot may initially be in an UNKNOWN state and transition into a PENDING state when created anew. Similarly, it may transition into an UNKNOWN state after deleted and the object no longer exists.

More rarely, a snapshot may transition into an UNKNOWN state if CloudBridge does not know how to map the state reported by the cloud provider.

Image states and lifecycle

The following states are defined for a CloudBridge Image.

State	Category	Description
UNKNOWN	actionable	Image state is unknown. This means that the Image does not exist or CloudBridge does not know how to map this state.
PENDING	informational	Image is pending
AVAILABLE	actionable	Image is ready.
ERROR	actionable	Image is in an error state. No further operations possible.

The lifecycle diagram is as follows:

Actionable states are shown in blue and informational states in cyan. Note that any state could potentially transition into an ERROR state. An Image may initially be in an UNKNOWN state and transition into a PENDING state when created anew. Similarly, it may transition into an UNKNOWN state after deleted and the image no longer exists. More rarely, an Image may transition into an UNKNOWN state if CloudBridge does not know how to map the state reported by the cloud provider.

Paging and iteration

Overview

Most provider services have list() methods, and all list methods accept a limit parameter which specifies the maximum number of results to return. If a limit is not specified, CloudBridge will default to the global configuration variable *default_result_limit*, which can be modified through the provider config.

Since the returned result list may have more records available, CloudBridge will always return a `ResultList` object to assist with paging through additional results. A ResultList extends the standard `list` and the following example illustrates how to fetch additional records.

Example:

```
# get first page of results
rl = provider.compute.instances.list(limit=50)
for result in rl:
    print("Instance Data: {0}", result)
if rl.supports_total:
    print("Total results: {0}".format(rl.total_results))
else:
    print("Total records unknown,"
          "but has more data?: {0}."format(rl.is_truncated))

# Page to next set of results
if (rl.is_truncated)
    rl = provider.compute.instances.list(limit=100,
                                         marker=rl.marker)
```

To ease development, CloudBridge also provides standard Python iterators that will page the results in for you automatically. Therefore, when you need to iterate through all available objects, the following shorthand is recommended:

Example:

```
# Iterate through all results
for instance in provider.compute.instances:
    print("Instance Data: {0}", instance)
```

Working with block storage

To add persistent storage to your cloud environments, you would use block storage devices, namely volumes and volume snapshots. A volume is attached to an instance and mounted as a file system for use by an application. A volume snapshot is a point-in-time snapshot of a volume that can be shared with other cloud users. Before a snapshot can be used, it is necessary to create a volume from it.

Volume storage

Operations, such as creating a new volume and listing the existing ones, are performed via the *VolumeService*. To start, let's create a 1GB volume.

```
vol = provider.block_store.volumes.create('CloudBridge-vol', 1, 'us-east-1e')
vol.wait_till_ready()
provider.block_store.volumes.list()
```

Next, let's attach the volume to a running instance as device `/dev/sdh`:

```
vol.attach('i-dbf37022', '/dev/sdh')
vol.refresh()
vol.state
# 'in-use'
```

Once attached, from within the instance, it is necessary to create a file system on the new volume and mount it.

Once you wish to detach a volume from an instance, it is necessary to unmount the file system from within the instance and detach it. The volume can then be attached to a different instance with all the data on it preserved.

```
vol.detach()
vol.refresh()
vol.state
# 'available'
```

Snapshot storage

A volume snapshot it created from an existing volume. Note that it may take a long time for a snapshot to become ready, particularly on AWS.

```
snap = vol.create_snapshot('cloudbridge-snap',
                           'A demo snapshot created via CloudBridge.')
snap.wait_till_ready()
snap.state
# 'available'
```

In order to make use of a snapshot, it is necessary to create a volume from it:

```
vol = provider.block_store.volumes.create(
    'CloudBridge-snap-vol', 1, 'us-east-1e', snapshot=snap)
```

The newly created volume behaves just like any other volume and can be attached to an instance for use.

Contributor Guide

This section has information on how to contribute to CloudBridge development, and a walkthrough of the process of getting started on developing a new CloudBridge Provider.

Design Goals

1. Create a cloud abstraction layer which minimises or eliminates the need for cloud specific special casing (i.e., Not require clients to write `if EC2 do x else if OPENSTACK do y`.)
2. Have a suite of conformance tests which are comprehensive enough that goal 1 can be achieved. This would also mean that clients need not manually test against each provider to make sure their application is compatible.
3. Opt for a minimum set of features that a cloud provider will support, instead of a lowest common denominator approach. This means that reasonably mature clouds like Amazon and OpenStack are used as the benchmark against which functionality & features are determined. Therefore, there is a definite expectation that the cloud infrastructure will support a compute service with support for images and snapshots and various machine sizes. The cloud infrastructure will very likely support block storage, although this is currently optional. It may optionally support object storage.
4. Make the CloudBridge layer as thin as possible without compromising goal 1. By wrapping the cloud provider's native SDK and doing the minimal work necessary to adapt the interface, we can achieve greater development speed and reliability since the native provider SDK is most likely to have both properties.

Running tests

In the spirit of the library's *Design Goals*, the aim is to have thorough tests for the entire library. This page explains the testing philosophy and shows how to run the tests locally.

Testing philosophy

Our testing goals are to:

1. Write one set of tests that all provider implementations must pass.
2. Make that set of tests a 'conformance' test suite, which validates that each implementation correctly implements the CloudBridge specification.
3. Make the test suite comprehensive enough that a provider which passes all the tests can be used safely by an application with no additional testing. In other words, the CloudBridge specification and accompanying test suite must be comprehensive enough that no provider specific workarounds, code or testing is required.
4. For development, mock providers may be used to speed up the feedback cycle, but providers must also pass the full suite of tests when run against actual cloud infrastructure to ensure that we are not testing against an idealised or imagined environment.
5. Aim for 100% code coverage.

Running tests

To run the test suite locally:

1. Install tox with `pip install tox`
2. Export all environment variables listed in `tox.ini` (under `passenv`)
3. Run `tox` command

This will run all the tests for all the environments defined in file `tox.ini`.

Specific environment and infrastructure

If you'd like to run the tests on a specific environment only, say Python 2.7, against a specific infrastructure, say aws, use a command like this: `tox -e py27-aws`. The available provider names are listed in the ProviderList class (e.g., `aws` or `openstack`).

Specific test cases

You can run a specific test case, as follows: `tox -- -s test.test_cloud_factory.CloudFactoryTestCase`

It can also be restricted to a particular environment as follows: `tox -e "py27-aws" -- -s test.test_cloud_factory.CloudFactoryTestCase`

Using unittest directly

You can also run the tests against your active virtual environment directly with `python setup.py test`. You will need to set the `CB_TEST_PROVIDER` and `CB_USE_MOCK_PROVIDERS` environment variables prior to running the tests, or they will default to `CB_TEST_PROVIDER=aws` and `CB_USE_MOCK_PROVIDERS=True`.

You can also run a specific test case, as follows: `python setup.py test -s test.test_cloud_factory.CloudFactoryTestCase`

Using a mock provider

Note that running the tests may create various cloud resources, for which you may incur costs. For the AWS cloud, there is also a mock provider (moto) that will simulate AWS resources. It is used by default when running the test suite. You can toggle the use of mock providers by setting an environment variable: `CB_USE_MOCK_PROVIDERS` to `Yes` or `No`.

Provider Development Walkthrough

This guide will walk you through the basic process of developing a new provider for CloudBridge.

1. We start off by creating a new folder for the provider within the `cloudbridge/cloud/providers` folder. In this case: `gce`. Further, install the native cloud provider Python library, here `pip install google-api-python-client==1.4.2` and a couple of its requirements `oauth2client==1.5.2` and `pycrypto==2.6.1`.

2. Add a `provider.py` file. This file will contain the main implementation of the cloud provider and will be the entry point that CloudBridge uses for all provider related services. You will need to subclass `BaseCloudProvider` and add a class variable named `PROVIDER_ID`.

```
from cloudbridge.cloud.base import BaseCloudProvider


class GCECloudProvider(BaseCloudProvider):

    PROVIDER_ID = 'gce'

    def __init__(self, config):
        super(GCECloudProvider, self).__init__(config)
```

3. Add an `__init__.py` to the `cloudbridge/cloud/providers/gce` folder and export the provider.

```
from .provider import GCECloudProvider  # noqa
```

Tip: You can view the code so far here: commit 1

4. Next, we need to register the provider with the factory. This only requires that you register the provider's ID in the `ProviderList`. Add GCE to the `ProviderList` class in `cloudbridge/cloud/factory.py`.

5. Run the test suite. We will get the tests passing on py27 first.

```
export CB_TEST_PROVIDER=gce
tox -e py27
```

You should see the tests fail with the following message:

```
TypeError: Can't instantiate abstract class GCECloudProvider with abstract
methods block_store, compute, object_store, security, network
```

6. Therefore, our next step is to implement these methods. We can start off by implementing these methods in `provider.py` and raising a `NotImplementedError`.

```
@property
def compute(self):
    raise NotImplementedError(
        "GCECloudProvider does not implement this service")

@property
def network(self):
    raise NotImplementedError(
        "GCECloudProvider does not implement this service")

@property
def security(self):
    raise NotImplementedError(
        "GCECloudProvider does not implement this service")

@property
def block_store(self):
    raise NotImplementedError(
        "GCECloudProvider does not implement this service")

@property
```

```
def object_store(self):
    raise NotImplementedError(
        "GCECloudProvider does not implement this service")
```

Running the tests now will complain as much. We will next implement each Service in turn.

7. We will start with the compute service. Add a `services.py` file.

```
from cloudbridge.cloud.base.services import BaseSecurityService


class GCESecurityService(BaseSecurityService):

    def __init__(self, provider):
        super(GCESecurityService, self).__init__(provider)
```

8. We can now return this new service from the security property in `provider.py` as follows:

```
def __init__(self, config):
    super(GCECloudProvider, self).__init__(config)
    self._security = GCESecurityService(self)

@property
def security(self):
    return self._security
```

Tip: You can view the code so far here: commit 2

9. Run the tests, and the following message will cause all security service tests to fail:

```
TypeError: Can't instantiate abstract class GCESecurityService with abstract
methods key_pairs, security_groups
```

The Abstract Base Classes are doing their job and flagging all methods that need to be implemented.

10. Since the security service simply provides organisational structure, and is a container for the `key_pairs` and `security_groups` services, we must next implement these services.

```
from cloudbridge.cloud.base.services import BaseKeyPairService
from cloudbridge.cloud.base.services import BaseSecurityGroupService
from cloudbridge.cloud.base.services import BaseSecurityService


class GCESecurityService(BaseSecurityService):

    def __init__(self, provider):
        super(GCESecurityService, self).__init__(provider)

        # Initialize provider services
        self._key_pairs = GCEKeyPairService(provider)
        self._security_groups = GCESecurityGroupService(provider)

    @property
    def key_pairs(self):
        return self._key_pairs

    @property
    def security_groups(self):
        return self._security_groups


class GCEKeyPairService(BaseKeyPairService):

    def __init__(self, provider):
        super(GCEKeyPairService, self).__init__(provider)


class GCESecurityGroupService(BaseSecurityGroupService):

    def __init__(self, provider):
        super(GCESecurityGroupService, self).__init__(provider)
```

Tip: You can view the code so far here: commit 3

Once again, running the tests will complain of missing methods:

```
TypeError: Can't instantiate abstract class GCEKeyPairService with abstract
methods create, find, get, list
```

11. Keep implementing the methods till the security service works, and the tests pass.

Note: We start off by implementing the list keypairs method. Therefore, to obtain the keypair, we need to have a connection to the cloud provider. For this, we need to install the Google sdk, and thereafter, to obtain the desired connection via the sdk. While the design and structure of that connection is up to the implementor, a general design we have followed is to have the cloud connection globally available within the provider.

To add the sdk, we edit CloudBridge's main `setup.py` and list the dependencies.

```
gce_reqs = ['google-api-python-client==1.4.2']
full_reqs = base_reqs + aws_reqs + openstack_reqs + gce_reqs
```

We will also register the provider in `cloudbridge/cloud/factory.py`'s provider list.

```
class ProviderList(object):
    AWS = 'aws'
    OPENSTACK = 'openstack'
    ...
    GCE = 'gce'
```

Tip: You can view the code so far here: commit 4

12. Thereafter, we create the actual connection through the sdk. In the case of GCE, we need a Compute API client object. We will make this connection available as a public property named `gce_compute` in the provider. We will then lazily initialize this connection.

A full implementation of the KeyPair service can now be made in a provider specific manner.

Tip: You can view the code so far here: commit 5

Release Process

1. Increment version number in cloudbridge/__init__.py as per semver rules.
2. Freeze all library dependencies in setup.py. The version numbers can be a range with the upper limit being the latest known working version, and the lowest being the last known working version.
3. Run all tox tests.
4. Add release notes to CHANGELOG.rst. Also add last commit hash to changelog.
5. Release to PyPi

```
python setup.py sdist upload
python setup.py bdist_wheel upload
```

6. Tag release and make github release.

API reference

This section includes the API documentation for the reference interface.

Providers

CloudProvider

class `cloudbridge.cloud.interfaces.provider.`**`CloudProvider`**(*config*)
Base interface for a cloud provider

`__init__`(*config*)
Create a new provider instance given a dictionary of configuration attributes.

Parameters **`config`** (`dict`) – A dictionary object containing provider initialization values. Alternatively, this can be a Bunch or any other object whose fields can be accessed as members. See specific provider implementation for the required fields.

Return type *`CloudProvider`*

Returns a concrete provider instance

`authenticate`()
Checks whether a provider can be successfully authenticated with the configured settings. Clients are *not* required to call this method prior to accessing provider services, as most cloud connections are initialized lazily. The authenticate() method will return True if cloudbridge can establish a successful connection to the provider. It will raise an exception with the appropriate error details otherwise.

Example:

```
try:
    if provider.authenticate():
        print("Provider connection successful")
except ProviderConnectionException as e:
    print("Could not authenticate with provider: %s" % (e, ))
```

Return type `bool`

Returns `True` if authentication is successful.

block_store

Provides access to the volume and snapshot services in this provider.

Example:

```
volumes = provider.block_store.volumes.list()
snapshots = provider.block_store.snapshots.list()
```

Return type *BlockStoreService*

Returns a BlockStoreService object

compute

Provides access to all compute related services in this provider.

Example:

```
regions = provider.compute.regions.list()
instance_types = provider.compute.instance_types.list()
instances = provider.compute.instances.list()
images = provider.compute.images.list()

# Alternatively
for instance in provider.compute.instances:
   print(instance.name)
```

Return type *ComputeService*

Returns a ComputeService object

config

Returns the config object associated with this provider. This object is a subclass of `dict` and will contain the properties provided at initialization time. In addition, it also contains extra provider-wide properties such as the default result limit for list() queries.

Example:

```
config = { 'aws_access_key' : '<my_key>' }
provider = factory.create_provider(ProviderList.AWS, config)
print(provider.config.get('aws_access_key'))
print(provider.config.default_result_limit))
# change provider result limit
provider.config.default_result_limit = 100
```

Return type `Configuration`

Returns An object of class Configuration, which contains the values used to initialize the provider, as well as other global configuration properties.

has_service(*service_type*)

Checks whether this provider supports a given service.

Example:

```
if provider.has_service(CloudServiceType.OBJECT_STORE):
   print("Provider supports object store services")
   provider.object_store.list()
```

Parameters **service_type** (*CloudServiceType*) – Type of service to check support for.

Return type `bool`

Returns `True` if the service type is supported.

network

Provide access to all network related services in this provider.

Example:

```
networks = provider.network.list()
network = provider.network.create(name="DevNet")
```

Return type *NetworkService*

Returns a NetworkService object

object_store

Provides access to object storage services in this provider.

Example:

```
if provider.has_service(CloudServiceType.OBJECT_STORE):
   print("Provider supports object store services")
   print(provider.object_store.list())
```

Return type `object` of *ObjectStoreService*

Returns an ObjectStoreService object

security

Provides access to key pair management and firewall control

Example:

```
keypairs = provider.security.keypairs.list()
security_groups = provider.security.security_groups.list()
```

Return type `object` of *SecurityService*

Returns a SecurityService object

ContainerProvider

class `cloudbridge.cloud.interfaces.provider.`**ContainerProvider**

Represents a container instance, such as Docker or LXC

Services

CloudService

class `cloudbridge.cloud.interfaces.services.`**`CloudService`**

Base interface for any service supported by a provider. This interface has a provider property that can be used to access the provider associated with this service.

`provider`

Returns the provider instance associated with this service.

Return type *`CloudProvider`*

Returns a CloudProvider object

ComputeService

class `cloudbridge.cloud.interfaces.services.`**`ComputeService`**

The compute service interface is a collection of services that provides access to the underlying compute related services in a provider. For example, the compute.instances service can be used to launch a new instance, and the compute.images service can be used to list available machine images.

`images`

Provides access to all Image related services in this provider. (e.g. Glance in OpenStack)

Example:

```
# print all images
for image in provider.compute.images:
    print(image.id, image.name)

# print only first 50 images
for image in provider.compute.images.list(limit=50):
    print(image.id, image.name)

# find image by name
image = provider.compute.images.find(name='Ubuntu 14.04')
print(image.id, image.name)
```

Return type *`ImageService`*

Returns an ImageService object

instance_types

Provides access to all Instance type related services in this provider.

Example:

```
# list all instance sizes
for inst_type in provider.compute.instance_types:
    print(inst_type.id, inst_type.name)

# find a specific size by name
inst_type = provider.compute.instance_types.find(name='m1.small')
print(inst_type.vcpus)
```

Return type `InstanceTypeService`

Returns an InstanceTypeService object

instances

Provides access to all Instance related services in this provider.

Example:

```
# launch a new instance
image = provider.compute.images.find(name='Ubuntu 14.04')[0]
size = provider.compute.instance_types.find(name='m1.small')
instance = provider.compute.instances.create('Hello', image, size)
print(instance.id, instance.name)
```

Return type *InstanceService*

Returns an InstanceService object

regions

Provides access to all Region related services in this provider.

Example:

```
for region in provider.compute.regions:
    print("Region: ", region.name)
    for zone in region.zones:
       print("\tZone: ", zone.name)
```

Return type *RegionService*

Returns a RegionService object

InstanceService

class `cloudbridge.cloud.interfaces.services.`**`InstanceService`**

Provides access to instances in a provider, including creating, listing and deleting instances.

create(*name, image, instance_type, subnet, zone=None, key_pair=None, security_groups=None, user_data=None, launch_config=None, **kwargs*)

Creates a new virtual machine instance.

Parameters

- **name** (`str`) – The name of the virtual machine instance
- **image** (`MachineImage` or `str`) – The MachineImage object or id to boot the virtual machine with
- **instance_type** (`InstanceType` or `str`) – The InstanceType or name, specifying the size of the instance to boot into
- **subnet** (`Subnet` or `str`) – The subnet object or a subnet string ID with which the instance should be associated. The subnet is a mandatory parameter, and must be provided when launching an instance.

  Note: Older clouds (with classic networking), may not have proper subnet support and are not guaranteed to work. Some providers (e.g. OpenStack) support a null value but the behaviour is implementation specific.
- **zone** (`Zone` or `str`) – The Zone or its name, where the instance should be placed. This parameter is provided for legacy compatibility (with classic networks).

  The subnet's placement zone will take precedence over this parameter, but in its absence, this value will be used.
- **key_pair** (`KeyPair` or `str`) – The KeyPair object or its name, to set for the instance.
- **security_groups** (A `list` of `SecurityGroup` objects or a list of `str` object IDs) – A list of `SecurityGroup` objects or a list of `SecurityGroup` IDs, which should be assigned to this instance.

  The security groups must be associated with the same network as the supplied subnet. Use `network.security_groups` to retrieve a list of security groups belonging to a network.
- **user_data** (`str`) – An extra userdata object which is compatible with the provider.
- **launch_config** (`LaunchConfig` object) – A `LaunchConfig` object which describes advanced launch configuration options for an instance. Currently, this includes only block_device_mappings. To construct a launch configuration object, call provider.compute.instances.create_launch_config()

Return type `object` of *`Instance`*

Returns an instance of Instance class

`create_launch_config`()

Creates a `LaunchConfig` object which can be used to set additional options when launching an instance, such as block device mappings and network interfaces.

Return type `object` of *`LaunchConfig`*

Returns an instance of a LaunchConfig class

`find`(*name*)

Searches for an instance by a given list of attributes.

Return type List of `object` of *`Instance`*

Returns A list of Instance objects matching the supplied attributes.

`get`(*instance_id*)

Returns an instance given its id. Returns None if the object does not exist.

Return type `object` of *`Instance`*

Returns an Instance object

list (*limit=None*, *marker=None*)
List available instances.

The returned results can be limited with limit and marker. If not specified, the limit defaults to a global default. See `list()` for more information on how to page through returned results.

example:

```
# List instances
instlist = provider.compute.instances.list()
for instance in instlist:
    print("Instance Data: {0}", instance)
```

Parameters

- **limit** (`int`) – The maximum number of objects to return. Note that the maximum is not guaranteed to be honoured, and a lower maximum may be enforced depending on the provider. In such a case, the returned ResultList's is_truncated property can be used to determine whether more records are available.
- **marker** (`str`) – The marker is an opaque identifier used to assist in paging through very long lists of objects. It is returned on each invocation of the list method.

Return type `ResultList` of *`Instance`*

Returns A ResultList object containing a list of Instances

VolumeService

class `cloudbridge.cloud.interfaces.services.`**VolumeService**
Base interface for a Volume Service.

create (*name*, *size*, *zone*, *snapshot=None*, *description=None*)
Creates a new volume.

Parameters

- **name** (`str`) – The name of the volume.
- **size** (`int`) – The size of the volume (in GB).
- **zone** (`str` or *`PlacementZone`* object) – The availability zone in which the Volume will be created.
- **snapshot** (`str` or *`Snapshot`* object) – An optional reference to a snapshot from which this volume should be created.
- **description** (`str`) – An optional description that may be supported by some providers. Providers that do not support this property will return `None`.

Return type `object` of *`Volume`*

Returns a newly created Volume object.

find (*name*, *limit=None*, *marker=None*)
Searches for a volume by a given list of attributes.

Return type `object` of *`Volume`*

Returns a Volume object or `None` if not found.

get (*volume_id*)
Returns a volume given its id.

Return type `object` of *Volume*

Returns a Volume object or `None` if the volume does not exist.

list (*limit=None*, *marker=None*)
List all volumes.

Return type `list` of *Volume*

Returns a list of Volume objects.

SnapshotService

class `cloudbridge.cloud.interfaces.services.`**SnapshotService**
Base interface for a Snapshot Service.

create (*name*, *volume*, *description=None*)
Creates a new snapshot off a volume.

Parameters

- **name** (`str`) – The name of the snapshot
- **volume** (`str` or `Volume`) – The volume to create a snapshot of.
- **description** (`str`) – An optional description that may be supported by some providers. Providers that do not support this property will return None.

Return type `object` of *Snapshot*

Returns a newly created Snapshot object.

find (*name*, *limit=None*, *marker=None*)
Searches for a snapshot by a given list of attributes.

Return type list of *Snapshot*

Returns a Snapshot object or an empty list if none found.

get (*volume_id*)
Returns a snapshot given its id.

Return type `object` of *Snapshot*

Returns a Snapshot object or `None` if the snapshot does not exist.

list (*limit=None*, *marker=None*)
List all snapshots.

Return type `list` of *Snapshot*

Returns a list of Snapshot objects.

BlockStoreService

class `cloudbridge.cloud.interfaces.services.`**BlockStoreService**
The Block Store Service interface provides access to block device services, such as volume and snapshot services in the provider.

snapshots
Provides access to volume snapshots for this provider.

Example:

```
# print all snapshots
for snap in provider.block_store.snapshots:
    print(snap.id, snap.name)

# find snapshot by name
snap = provider.block_store.snapshots.find(name='my_snap')[0]
print(snap.id, snap.name)
```

Return type `SnapshotService`

Returns an SnapshotService object

volumes
Provides access to volumes (i.e., block storage) for this provider.

Example:

```
# print all volumes
for vol in provider.block_store.volumes:
    print(vol.id, vol.name)

# find volume by name
vol = provider.block_store.volumes.find(name='my_vol')[0]
print(vol.id, vol.name)
```

Return type `VolumeService`

Returns a VolumeService object

ImageService

class `cloudbridge.cloud.interfaces.services.`**`ImageService`**
Base interface for an Image Service

find(*name*, *limit=None*, *marker=None*)
Searches for an image by a given list of attributes

Return type `object` of `Image`

Returns an Image instance

get(*image_id*)
Returns an Image given its id. Returns None if the Image does not exist.

Return type `object` of `Image`

Returns an Image instance

list(*limit=None*, *marker=None*)
List all images.

Return type `list` of `Image`

Returns list of image objects

NetworkService

class `cloudbridge.cloud.interfaces.services.`**`NetworkService`**

Base interface for a Network Service.

`create`(*name=None*)

Create a new network.

Parameters **`name`** (`str`) – An optional network name. The name will be set if the provider supports it.

Return type `object` of *`Network`*

Returns A Network object

`create_floating_ip`()

Allocate a new floating (i.e., static) IP address.

Parameters **`network_id`** (`str`) – The ID of the network with which to associate the new IP address.

Return type `FloatingIP`

Returns floating IP object

`create_router`(*name=None*)

Create a new router/gateway.

Parameters **`name`** (`str`) – An optional router name. The name will be set if the provider supports it.

Return type `Router`

Returns a newly created router object

`delete`(*network_id*)

Delete an existing Network.

Parameters **`network_id`** (`str`) – The ID of the network to be deleted.

Return type `bool`

Returns `True` if the network does not exist, `False` otherwise. Note that this implies that the network may not have been deleted by this method but instead has not existed at all.

`floating_ips`(*network_id=None*)

List floating (i.e., static) IP addresses.

Parameters **`network_id`** (`str`) – The ID of the network by which to filter the IPs.

Return type `list` of `FloatingIP`

Returns list of floating IP objects

`get`(*network_id*)

Returns a Network given its ID or `None` if not found.

Parameters **`network_id`** (`str`) – The ID of the network to retrieve.

Return type `object` of *`Network`*

Returns a Network object

`list`(*limit=None*, *marker=None*)

List all networks.

Return type `list` of *`Network`*

Returns list of Network objects

routers()
Get a list of available routers.

Return type `list` of :class: *Router*

Returns list of routers

subnets
Provides access to subnets.

Example:

```
# Print all subnets
for s in provider.network.subnets:
    print(s.id, s.name)

# Get subnet by ID
s = provider.network.subnets.get('subnet-id')
print(s.id, s.name)
```

Return type `SubnetService`

Returns a SubnetService object

ObjectStoreService

class `cloudbridge.cloud.interfaces.services.`**ObjectStoreService**
The Object Storage Service interface provides access to the underlying object store capabilities of this provider. This service is optional and the `CloudProvider.has_service()` method should be used to verify its availability before using the service.

create(*name*, *location=None*)
Create a new bucket.

If a bucket with the specified name already exists, return a reference to that bucket.

Example:

```
bucket = provider.object_store.create('my_bucket_name')
print(bucket.name)
```

Parameters

- **name** (*str*) – The name of this bucket.
- **location** (`object` of *Region*) – The region in which to place this bucket.

Returns a Bucket object

Return type `object` of *Bucket*

find(*name*)
Searches for a bucket by a given list of attributes.

Example:

```
buckets = provider.object_store.find(name='my_bucket_name')
for bucket in buckets:
    print(bucket.id, bucket.name)
```

Return type *Bucket*

Returns a Bucket instance

get (*bucket_id*)

Returns a bucket given its ID. Returns `None` if the bucket does not exist. On some providers, such as AWS and OpenStack, the bucket id is the same as its name.

Example:

```
bucket = provider.object_store.get('my_bucket_id')
print(bucket.id, bucket.name)
```

Return type *Bucket*

Returns a Bucket instance

list (*limit=None*, *marker=None*)

List all buckets.

Example:

```
buckets = provider.object_store.find(name='my_bucket_name')
for bucket in buckets:
    print(bucket.id, bucket.name)
```

Return type *Bucket*

Returns list of bucket objects

SecurityService

class `cloudbridge.cloud.interfaces.services.`**SecurityService**

The security service interface can be used to access security related functions in the provider, such as firewall control and keypairs.

key_pairs

Provides access to key pairs for this provider.

Example:

```
# print all keypairs
for kp in provider.security.keypairs:
    print(kp.id, kp.name)

# find keypair by name
kp = provider.security.keypairs.find(name='my_key_pair')[0]
print(kp.id, kp.name)
```

Return type *KeyPairService*

Returns a KeyPairService object

security_groups
Provides access to security groups for this provider.

Example:

```
# print all security groups
for sg in provider.security.security_groups:
    print(sg.id, sg.name)

# find security group by name
sg = provider.security.security_groups.find(name='my_sg')[0]
print(sg.id, sg.name)
```

Return type *SecurityGroupService*

Returns a SecurityGroupService object

KeyPairService

class `cloudbridge.cloud.interfaces.services.`**`KeyPairService`**
Base interface for key pairs.

create(*name*)
Create a new key pair or raise an exception if one already exists.

Parameters **name** (*str*) – The name of the key pair to be created.

Return type `object` of *KeyPair*

Returns A keypair instance or `None`.

delete(*key_pair_id*)
Delete an existing SecurityGroup.

Parameters **key_pair_id** (*str*) – The id of the key pair to be deleted.

Return type `bool`

Returns `True` if the key does not exist, `False` otherwise. Note that this implies that the key may not have been deleted by this method but instead has not existed at all.

find(*name*, *limit=None*, *marker=None*)
Searches for a key pair by a given list of attributes.

Return type `object` of *KeyPair*

Returns a KeyPair object

get(*key_pair_id*)
Return a KeyPair given its ID or `None` if not found.

On some providers, such as AWS and OpenStack, the KeyPair ID is the same as its name.

Example:

```
key_pair = provider.security.keypairs.get('my_key_pair_id')
print(key_pair.id, key_pair.name)
```

Return type *KeyPair*

Returns a KeyPair instance

list (*limit=None*, *marker=None*)
List all key pairs associated with this account.

Return type `list` of *KeyPair*

Returns list of KeyPair objects

SecurityGroupService

class `cloudbridge.cloud.interfaces.services.`**SecurityGroupService**
Base interface for security groups.

create (*name*, *description*, *network_id*)
Create a new SecurityGroup.

Parameters

- **name** (*str*) – The name of the new security group.
- **description** (*str*) – The description of the new security group.
- **network_id** (`str`) – Network ID under which to create the security group.

Return type `object` of *SecurityGroup*

Returns A SecurityGroup instance or `None` if one was not created.

delete (*group_id*)
Delete an existing SecurityGroup.

Parameters **group_id** (*str*) – The security group ID to be deleted.

Return type `bool`

Returns `True` if the security group does not exist, `False` otherwise. Note that this implies that the group may not have been deleted by this method but instead has not existed in the first place.

find (*name*, *limit=None*, *marker=None*)
Get security groups associated with your account filtered by name.

Parameters **name** (*str*) – The name of the security group to retrieve.

Return type list of `SecurityGroup`

Returns A list of SecurityGroup objects or an empty list if none found.

get (*security_group_id*)
Returns a SecurityGroup given its ID. Returns `None` if the SecurityGroup does not exist.

Example:

```
sg = provider.security.security_groups.get('my_sg_id')
print(sg.id, sg.name)
```

Return type *SecurityGroup*

Returns a SecurityGroup instance

list (*limit=None*, *marker=None*)
List all security groups associated with this account.

Return type `list` of *SecurityGroup*

Returns list of SecurityGroup objects

InstanceTypesService

class `cloudbridge.cloud.interfaces.services.`**`InstanceTypesService`**

`find`(***kwargs*)
Searches for an instance by a given list of attributes.

Return type `object` of *`InstanceType`*

Returns an Instance object

`get`(*instance_type_id*)
Returns an InstanceType given its ID. Returns `None` if the InstanceType does not exist.

Example:

```
itype = provider.compute.instance_types.get('my_itype_id')
print(itype.id, itype.name)
```

Return type *`InstanceType`*

Returns an InstanceType instance

`list`(*limit=None*, *marker=None*)
List all instance types.

Return type `list` of *`InstanceType`*

Returns list of InstanceType objects

RegionService

class `cloudbridge.cloud.interfaces.services.`**`RegionService`**
Base interface for a Region service

`current`
Returns the current region that this provider is connected to.

If the current region cannot be discovered, return `None`.

Return type `object` of *`Region`*

Returns a Region instance or `None`

`get`(*region_id*)
Returns a region given its id. Returns None if the region does not exist.

Return type `object` of *`Region`*

Returns a Region instance

`list`(*limit=None*, *marker=None*)
List all regions.

Return type `list` of *`Region`*

Returns list of region objects

Resources

CloudServiceType

class `cloudbridge.cloud.interfaces.resources.`**`CloudServiceType`**

Defines possible service types that are offered by providers.

Providers can implement the `has_service` method and clients can check for the availability of a service with:

```
if (provider.has_service(CloudServiceTypes.OBJECTSTORE))
    ...
```

ObjectLifeCycleMixin

class `cloudbridge.cloud.interfaces.resources.`**`ObjectLifeCycleMixin`**

A mixin for an object with a defined life-cycle, such as an Instance, Volume, Image or Snapshot. An object that supports ObjectLifeCycleMixin will always have a state, defining which point in its life cycle it is currently at.

It also defines a wait_till_ready operation, which indicates that the object is in a state in its life cycle where it is ready to be used by an end-user.

A refresh operation allows the object to synchronise its state with the service provider.

refresh()
Refreshs this object's state and synchronize it with the underlying service provider.

state
Get the current state of this object.

Return type `str`

Returns The current state as a string.

wait_for(*target_states*, *terminal_states=None*, *timeout=None*, *interval=None*)
Wait for a specified timeout for an object to reach a set of desired target states. If the object does not reach the desired state within the specified timeout, a `WaitStateException` will be raised. The optional terminal_states property can be used to specify an additional set of states which, should the object reach one, the object thereafter will not transition into the desired target state. Should this happen, a `WaitStateException` will be raised.

Example:

```
instance.wait_for(
    [InstanceState.TERMINATED, InstanceState.UNKNOWN],
    terminal_states=[InstanceState.ERROR])
```

Parameters

- **target_states** (`list` of states) – The list of target states to wait for.
- **terminal_states** (`list` of states) – A list of terminal states after which the object will not transition into a target state. A WaitStateException will be raised if the object transition into a terminal state.
- **timeout** (`int`) – The maximum length of time (in seconds) to wait for the object to changed to desired state. If no timeout is specified, the global default_wait_timeout defined in the provider config will apply.
- **interval** (`int`) – How frequently to poll the object's state (in seconds). If no interval is specified, the global default_wait_interval defined in the provider config will apply.

Return type `True`

Returns Returns `True` if successful. A `WaitStateException` exception may be thrown by the underlying service if the object cannot get into a ready state (e.g. if the object is in an error state).

wait_till_ready(*timeout*, *interval*)
A convenience method to wait till the current object reaches a state which is ready for use, which is any state where the end-user can successfully interact with the object. Will throw a `WaitStateException` if the object is not ready within the specified timeout.

Parameters

- **timeout** (`int`) – The maximum length of time (in seconds) to wait for the object to become ready.
- **interval** (`int`) – How frequently to poll the object's ready state (in seconds).

Return type `True`

Returns Returns `True` if successful. A `WaitStateException` exception may be thrown by the underlying service if the object cannot get into a ready state (e.g. if the object is in an error state).

ResultList

class `cloudbridge.cloud.interfaces.resources.`**`ResultList`**

This is a wrapper class around a standard Python `list` class and provides some extra properties to aid with paging through a large number of results.

Example:

```
# get first page of results
rl = provider.compute.instances.list(limit=50)
for result in rl:
    print("Instance Data: {0}", result)
if rl.supports_total:
    print("Total results: {0}".format(rl.total_results))
else:
    print("Total records unknown,"
          "but has more data?: {0}."format(rl.is_truncated))

# Page to next set of results
if (rl.is_truncated)
    rl = provider.compute.instances.list(limit=100,
                                         marker=rl.marker)
```

`is_truncated`

Indicates whether this result list has more results that can be paged in.

`marker`

This is an opaque identifier used to assist in paging through very long lists of objects. This marker can be provided to the list method to get the next set of results.

`supports_server_paging`

Indicates whether this ResultList supports client side paging or server side paging. If server side paging is not supported, the data property provides direct access to all available data.

`supports_total`

Indicates whether the provider supports returning the total number of available results. The supports_total property should be checked before accessing the total_results property.

`total_results`

Indicates the total number of results for a particular query. The supports_total property should be used to check whether the provider supports returning the total number of results, before accessing this property, or the behaviour is indeterminate.

InstanceState

class `cloudbridge.cloud.interfaces.resources.`**`InstanceState`**

Standard states for a node

Variables

- **`UNKNOWN`** – Instance state unknown.
- **`PENDING`** – Instance is pending

- **CONFIGURING** – Instance is being reconfigured in some way.
- **RUNNING** – Instance is running.
- **REBOOTING** – Instance is rebooting.
- **TERMINATED** – Instance is terminated. No further operations possible.
- **STOPPED** – Instance is stopped. Instance can be resumed.
- **ERROR** – Instance is in an error state. No further operations possible.

Instance

class `cloudbridge.cloud.interfaces.resources.`**`Instance`**

`add_floating_ip`(*ip_address*)
Add a public IP address to this instance.

Parameters **`ip_address`** (`str`) – The IP address to associate with the instance.

`add_security_group`(*sg*)
Add a security group to this instance

Parameters **`sg`** (`SecurityGroup`) – The SecurityGroup to associate with the instance.

`create_image`(*name*)
Create a new image based on this instance.

Return type :class:`.Image`

Returns an Image object

`id`
Get the instance identifier.

Return type `str`

Returns ID for this instance as returned by the cloud middleware.

`image_id`
Get the image ID for this instance.

Return type `str`

Returns Image ID (i.e., AMI) this instance is using.

`instance_type`
Retrieve full instance type information for this instance.

Return type *`InstanceType`*

Returns Instance type for this instance

`instance_type_id`
Get the instance type id for this instance. This will typically be a string value like 'm1.large'. On OpenStack, this may be a number or UUID. To get the full :class:`.InstanceType` object, you can use the instance.instance_type property instead.

Return type `str`

Returns Instance type name for this instance (e.g., `m1.large`)

`key_pair_name`
Get the name of the key pair associated with this instance.

Return type `str`

Returns Name of the ssh key pair associated with this instance.

`name`
Get the instance name.

Return type `str`

Returns Name for this instance as returned by the cloud middleware.

`private_ips`
Get all the private IP addresses for this instance.

Return type `list`

Returns A list of private IP addresses associated with this instance.

`public_ips`
Get all the public IP addresses for this instance.

Return type `list`

Returns A list of public IP addresses associated with this instance.

`reboot``()`
Reboot this instance (using the cloud middleware API).

Return type `bool`

Returns `True` if the reboot was successful; `False` otherwise.

`remove_floating_ip``(`*ip_address*`)`
Remove a public IP address from this instance.

Parameters **`ip_address`** (`str`) – The IP address to remove from the instance.

`remove_security_group``(`*sg*`)`
Remove a security group from this instance

Parameters **`sg`** (`SecurityGroup`) – The SecurityGroup to associate with the instance.

`security_group_ids`
Get the IDs of the security groups associated with this instance.

Return type list or :class:`str`

Returns A list of the SecurityGroup IDs associated with this instance.

`security_groups`
Get the security groups associated with this instance.

Return type list or *`SecurityGroup`* objects

Returns A list of SecurityGroup objects associated with this instance.

`terminate``()`
Permanently terminate this instance.

`zone_id`
Get the placement zone ID where this instance is running.

Return type `str`

Returns Region/zone/placement where this instance is running.

MachineImageState

class `cloudbridge.cloud.interfaces.resources.`**`MachineImageState`**

Standard states for a machine image

Variables

- **`UNKNOWN`** – Image state unknown.
- **`PENDING`** – Image is pending
- **`AVAILABLE`** – Image is available
- **`ERROR`** – Image is in an error state. Not recoverable.

LaunchConfig

class `cloudbridge.cloud.interfaces.resources.`**`LaunchConfig`**

Represents an advanced launch configuration object.

Theis object can contain information such as BlockDeviceMappings configurations, and other advanced options which may be useful when launching an instance.

Example:

```
lc = provider.compute.instances.create_launch_config()
lc.add_block_device(...)

inst = provider.compute.instances.create(name, image, instance_type,
                                         network, launch_config=lc)
```

`add_ephemeral_device``()`

Adds a new ephemeral block device mapping to the boot configuration. This can be used to add existing ephemeral devices to the instance. (The total number of ephemeral devices available for a particular InstanceType can be determined by querying the InstanceTypes service). Note that on some services, such as AWS, ephemeral devices must be added in as a device mapping at instance creation time, and cannot be added afterwards.

Note that the device name, such as /dev/sda1, cannot be selected at present, since this tends to be provider and instance type specific. However, the order of device addition coupled with device type will generally determine naming order, with devices added first getting lower letters than instances added later.

Example:

```
lc = provider.compute.instances.create_launch_config()

# 1. Add all available ephemeral devices
inst_type = provider.compute.instance_types.find(name='m1.tiny')[0]
for i in range(inst_type.num_ephemeral_disks):
    lc.add_ephemeral_device()
```

`add_volume_device``(`*source=None*, *is_root=None*, *size=None*, *delete_on_terminate=None*`)`

Adds a new volume based block device mapping to the boot configuration. The volume can be based on a snapshot, image, existing volume or be a blank new volume, and is specified by the source parameter.

The property is_root can be set to True to override any existing root device mappings. Otherwise, the default behaviour is to add new block devices to the instance.

Note that the device name, such as /dev/sda1, cannot be selected at present, since this tends to be provider and instance type specific. However, the order of device addition coupled with device type will generally determine naming order, with devices added first getting lower letters than instances added later (except when is_root is set).

Example:

```
lc = provider.compute.instances.create_launch_config()

# 1. Create and attach an empty volume of size 100GB
lc.add_volume_device(size=100, delete_on_terminate=True)

# 2. Create and attach a volume based on a snapshot
snap = provider.block_store.snapshots.get('<my_snapshot_id>')
lc.add_volume_device(source=snap)

# 3. Create+attach a volume based on an image and set it as root
img = provider.compute.images.get('<my_image_id>')
lc.add_volume_device(source=img, size=100, is_root=True)
```

Parameters

- **`source`** (`Volume`, `Snapshot`, `Image` or None.) – The source `block_device` to add. If `Volume`, the volume will be attached directly to the instance. If `Snapshot`, a volume will be created based on the Snapshot and attached to the instance. If `Image`, a volume based on the Image will be attached to the instance. If `None`, the source is assumed to be a blank volume.
- **`is_root`** (`bool`) – Determines which device will serve as the root device. If more than one device is defined as root, an `InvalidConfigurationException` will be thrown.
- **`size`** (`int`) – The size of the volume to create. An implementation may ignore this parameter for certain sources like 'Volume'.
- **`delete_on_terminate`** (`bool`) – Determines whether to delete or keep the volume on instance termination.

MachineImage

class `cloudbridge.cloud.interfaces.resources.`**`MachineImage`**

`delete``()`

Delete this image

Return type `bool`

Returns `True` if the operation succeeded.

`description`

Get the image description.

Return type `str`

Returns Description for this image as returned by the cloud middleware.

`id`

Get the image identifier.

Return type `str`

Returns ID for this instance as returned by the cloud middleware.

min_disk
Returns the minimum size of the disk that's required to boot this image (in GB)

Return type `int`

Returns The minimum disk size needed by this image

name
Get the image name.

Return type `str`

Returns Name for this image as returned by the cloud middleware.

Network

class `cloudbridge.cloud.interfaces.resources.`**`Network`**
Represents a software-defined network, like the Virtual Private Cloud.

cidr_block
A CIDR block for this network.

Note: OpenStack does not define a CIDR block for networks.

Return type `str`

Returns A CIDR block string.

create_subnet(*cidr_block*, *name=None*, *zone=None*)
Create a new network subnet and associate it with this Network.

Parameters

- **cidr_block** (`str`) – CIDR block within this Network to assign to the subnet.
- **name** (`str`) – An optional subnet name. The name will be set if the provider supports it.
- **zone** (`str`) – Placement zone where to create the subnet. Some providers may not support subnet zones, in which case the value is ignored.

Return type `object` of *`Subnet`*

Returns A Subnet object

delete()
Delete this network.

Return type `bool`

Returns `True` if successful.

external
A flag to indicate if this network is capable of Internet-connectivity.

Return type `bool`

Returns `True` if the network can be connected to the Internet.

id
Get the network identifier.

Return type `str`

Returns ID for this network. Will generally correspond to the cloud middleware's ID, but should be treated as an opaque value.

name

Get the network name.

Return type `str`

Returns Name for this network as returned by the cloud middleware.

state

The state of the network.

Return type `str`

Returns One of `unknown`, `pending`, `available`, `down` or `error`.

subnets()

The associated subnets.

Return type `list` of *Subnet*

Returns List of subnets associated with this network.

Subnet

class `cloudbridge.cloud.interfaces.resources.`**Subnet**

Represents a subnet, as part of a Network.

cidr_block

A CIDR block for this subnet.

Return type `str`

Returns A CIDR block string.

delete()

Delete this subnet.

Return type `bool`

Returns `True` if successful.

id

Get the subnet identifier.

Return type `str`

Returns ID for this network. Will generally correspond to the cloud middleware's ID, but should be treated as an opaque value.

name

Get the subnet name.

Return type `str`

Returns Name for this subnet as returned by the cloud middleware.

network_id

ID of the network associated with this this subnet.

Return type `str`

Returns Network ID.

zone
Placement zone of the subnet.

If the provider does not support subnet placement, return `None`.

Return type *PlacementZone* object

Returns Placement zone of the subnet, or `None` if not defined.

VolumeState

class `cloudbridge.cloud.interfaces.resources.`**VolumeState**
Standard states for a volume

Variables

- **UNKNOWN** – Volume state unknown.
- **CREATING** – Volume is being created.
- **CONFIGURING** – Volume is being configured in some way.
- **AVAILABLE** – Volume is available and can be attached to an instance.
- **IN_USE** – Volume is attached and in-use.
- **DELETED** – Volume has been deleted. No further operations possible.
- **ERROR** – Volume is in an error state. No further operations possible.

Volume

class `cloudbridge.cloud.interfaces.resources.`**Volume**

attach(*instance*, *device*)
Attach this volume to an instance.

Parameters

- **instance** (`str` or *Instance* object) – The ID of an instance or an `Instance` object to which this volume will be attached.
- **device** (`str`) – The device on the instance through which the volume will be exposed (e.g. /dev/sdh).

Return type `bool`

Returns `True` if successful.

attachments
Get attachment information for this volume.

Return type `AttachmentInfo`

Returns Returns an AttachmentInfo object.

create_snapshot(*name*, *description=None*)
Create a snapshot of this Volume.

Parameters

- **name** (`str`) – The name of this snapshot.
- **description** (`str`) – A description of the snapshot. Limited to 256 characters.

Return type *Snapshot*

Returns The created Snapshot object.

create_time
Get the creation data and time for this volume.

Return type `DateTime`

Returns Creation time for this volume as returned by the cloud middleware.

delete()
Delete this volume.

Return type `bool`

Returns `True` if successful.

description
Get the volume description. Some cloud providers may not support this property, and will return the volume name instead.

Return type `str`

Returns Description for this volume as returned by the cloud middleware.

detach(*force=False*)
Detach this volume from an instance.

Parameters **force** (`bool`) – Forces detachment if the previous detachment attempt did not occur cleanly. This option is supported on select clouds only. This option can lead to data loss or a corrupted file system. Use this option only as a last resort to detach a volume from a failed instance. The instance will not have an opportunity to flush file system caches nor file system meta data. If you use this option, you must perform file system check and repair procedures.

Return type `bool`

Returns `True` if successful.

id
Get the volume identifier.

Return type `str`

Returns ID for this volume. Will generally correspond to the cloud middleware's ID, but should be treated as an opaque value.

name
Get the volume name.

Return type `str`

Returns Name for this volume as returned by the cloud middleware.

size
Get the volume size (in GB).

Return type `int`

Returns Size for this volume as returned by the cloud middleware.

source
If available, get the source that this volume is based on (can be a Snapshot or an Image). Returns None if no source.

Return type `Snapshot` or ``Image`

Returns Snapshot or Image source for this volume as returned by the cloud middleware.

`zone_id`

Get the placement zone id that this volume belongs to.

Return type `str`

Returns PlacementZone for this volume as returned by the cloud middleware.

SnapshotState

class `cloudbridge.cloud.interfaces.resources.`**`SnapshotState`**

Standard states for a snapshot

Variables

- **`UNKNOWN`** – Snapshot state unknown.
- **`PENDING`** – Snapshot is pending.
- **`CONFIGURING`** – Snapshot is being configured in some way.
- **`AVAILABLE`** – Snapshot has been completed and is ready for use.
- **`ERROR`** – Snapshot is in an error state. No further operations possible.

Snapshot

class `cloudbridge.cloud.interfaces.resources.`**`Snapshot`**

`create_time`

Get the creation data and time for this snapshot.

Return type `DateTime`

Returns Creation time for this snapshot as returned by the cloud middleware.

`create_volume`(*placement*, *size=None*, *volume_type=None*, *iops=None*)

Create a new Volume from this Snapshot.

Parameters

- **`placement`** (`str`) – The availability zone where to create the Volume.
- **`size`** (`int`) – The size of the new volume, in GiB (optional). Defaults to the size of the snapshot.
- **`volume_type`** (`str`) – The type of the volume (optional). Availability and valid values depend on the provider.
- **`iops`** (`int`) – The provisioned IOPs you want to associate with this volume (optional). Availability depends on the provider.

Return type *`Volume`*

Returns An instance of the created Volume.

`delete`()

Delete this snapshot.

Return type `bool`

Returns `True` if successful.

description
Get the snapshot description. Some cloud providers may not support this property, and will return the snapshot name instead.

Return type `str`

Returns Description for this snapshot as returned by the cloud middleware.

id
Get the snapshot identifier.

Return type `str`

Returns ID for this snapshot. Will generally correspond to the cloud middleware's ID, but should be treated as an opaque value.

name
Get the snapshot name.

size
Get the snapshot size (in GB).

Return type `int`

Returns Size for this snapshot as returned by the cloud middleware.

volume_id
Get the id of the volume that this snapshot is based on. May return None if the source volume no longer exists.

Return type `int`

Returns Id of the volume that this snapshot is based on

KeyPair

class `cloudbridge.cloud.interfaces.resources.`**`KeyPair`**

delete()
Delete this key pair.

Return type `bool`

Returns `True` if successful.

id
Return the id of this key pair.

Return type `str`

Returns ID for this snapshot. Will generally correspond to the cloud middleware's name, but should be treated as an opaque value.

material
Unencrypted private key.

Return type `str`

Returns Unencrypted private key or `None` if not available.

name
Return the name of this key pair.

Return type `str`

Returns A name of this ssh key pair.

Region

class `cloudbridge.cloud.interfaces.resources.`**`Region`**

Represents a cloud region, typically a separate geographic area and will contain at least one placement zone.

`id`

The id for this region

Return type `str`

Returns ID of the region.

`name`

Name of the region.

Return type `str`

Returns Name of the region.

`zones`

Access information about placement zones within this region.

Return type Iterable

Returns Iterable of available placement zones in this region.

PlacementZone

class `cloudbridge.cloud.interfaces.resources.`**`PlacementZone`**

Represents a placement zone. A placement zone is contained within a Region.

`id`

Name of the placement zone.

Return type `str`

Returns Name of the placement zone.

`name`

Name of the placement zone.

Return type `str`

Returns Name of the placement zone.

`region_name`

A region this placement zone is associated with.

Return type `str`

Returns The name of the region the zone is associated with.

InstanceType

class `cloudbridge.cloud.interfaces.resources.`**`InstanceType`**

An instance type object.

`extra_data`

A dictionary of extra data about this instance. May contain nested dictionaries, but all key value pairs are strings or integers.

Return type `dict`

Returns Extra attributes for this instance type.

`family`

The family/group that this instance type belongs to.

For example, General Purpose Instances or High-Memory Instances. If the provider does not support such a grouping, it may return `None`.

Return type `str`

Returns Name of the instance family or `None`.

`num_ephemeral_disks`

The total number of ephemeral disks on this instance type.

Return type `int`

Returns Number of ephemeral disks available.

`ram`

The amount of RAM (in MB) supported by this instance type.

Return type `int`

Returns Total RAM (in MB).

`size_ephemeral_disks`

The size of this instance types's total ephemeral storage (in GB).

Return type `int`

Returns Size of ephemeral disks (in GB).

`size_root_disk`

The size of this instance types's root disk (in GB).

Return type `int`

Returns Size of root disk (in GB).

`size_total_disk`

The total disk space available on this instance type (root_disk + ephemeral).

Return type `int`

Returns Size of total disk space (in GB).

`vcpus`

The number of VCPUs supported by this instance type.

Return type `int`

Returns Number of VCPUs.

SecurityGroup

class `cloudbridge.cloud.interfaces.resources.`**`SecurityGroup`**

add_rule(*ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*, *src_group=None*)
Create a security group rule. If the rule already exists, simply returns it.

You need to pass in either `src_group` OR `ip_protocol` AND `from_port`, `to_port`, `cidr_ip`. In other words, either you are authorizing another group or you are authorizing some ip-based rule.

Parameters

- **ip_protocol** (`str`) – Either `tcp` | `udp` | `icmp`.
- **from_port** (`int`) – The beginning port number you are enabling.
- **to_port** (`int`) – The ending port number you are enabling.
- **cidr_ip** (`str` or list of `str`) – The CIDR block you are providing access to.
- **src_group** (*SecurityGroup*) – The Security Group object you are granting access to.

Return type *SecurityGroupRule*

Returns Rule object if successful or `None`.

delete()
Delete this security group.

Return type `bool`

Returns `True` if successful.

description
Return the description of this security group.

Return type `str`

Returns A description of this security group.

get_rule(*ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*, *src_group=None*)
Get a security group rule with the specified parameters.

You need to pass in either `src_group` OR `ip_protocol` AND `from_port`, `to_port`, and `cidr_ip`. Note that when retrieving a group rule, this method will return only one rule although possibly several rules exist for the group rule. In that case, use the `.rules` property and filter the results as desired.

Parameters

- **ip_protocol** (`str`) – Either `tcp` | `udp` | `icmp`.
- **from_port** (`int`) – The beginning port number you are enabling.
- **to_port** (`int`) – The ending port number you are enabling.
- **cidr_ip** (`str` or list of `str`) – The CIDR block you are providing access to.
- **src_group** (*SecurityGroup*) – The Security Group object you are granting access to.

Return type *SecurityGroupRule*

Returns Role object if one can be found or `None`.

id
Get the ID of this security group.

Return type `str`

Returns Security group ID.

name
Return the name of this security group.

Return type `str`

Returns A name of this security group.

network_id
Network ID with which this security group is associated.

Return type `str`

Returns Provider-supplied network ID or `None` is not available.

rules
Get the list of rules for this security group.

Return type list of *`SecurityGroupRule`*

Returns A list of security group rule objects.

SecurityGroupRule

class `cloudbridge.cloud.interfaces.resources.`**`SecurityGroupRule`**
Represents a security group rule.

cidr_ip
CIDR block this security group is providing access to.

Return type `str`

Returns CIDR block.

delete()
Delete this rule.

from_port
Lowest port number opened as part of this rule.

Return type `int`

Returns Lowest port number or 0 if not set.

group
Security group given access permissions by this rule.

Return type :class:.`SecurityGroup`

Returns The Security Group with granting access.

id
ID for this rule.

Note that this may be a CloudBridge-specific ID if the underlying provider does not support rule IDs.

Return type `str`

Returns Role ID.

ip_protocol
IP protocol used. Either `tcp` | `udp` | `icmp`.

Return type `str`

Returns Active protocol.

to_port
Highest port number opened as part of this rule.

Return type `int`

Returns Highest port number or 0 if not set.

BucketObject

class `cloudbridge.cloud.interfaces.resources.`**BucketObject**
Represents an object stored within a bucket.

delete()
Delete this object.

Return type `bool`

Returns `True` if successful.

generate_url(*expires_in=0*)
Generate a URL to this object.

If the object is public, *expires_in* argument is not necessary, but if the object is private, the lifetime of URL is set using *expires_in* argument.

Parameters **expires_in** (`int`) – Time to live of the generated URL in seconds.

Return type `str`

Returns A URL to access the object.

id
Get this object's id.

Return type `str`

Returns id of this object as returned by the cloud middleware.

iter_content()
Returns this object's content as an iterable.

Return type Iterable

Returns An iterable of the file contents

last_modified
Get the date and time this object was last modified.

Return type `str`

Returns Date and time formatted string %Y-%m-%dT%H:%M:%S.%f

name
Get this object's name.

Return type `str`

Returns Name of this object as returned by the cloud middleware.

save_content(*target_stream*)
Save this object and write its contents to the `target_stream`.

size
Get this object's size.

Return type `int`

Returns Size of this object in bytes.

upload(*source_stream*)
Set the contents of this object to the data read from the source stream.

Return type `bool`

Returns `True` if successful.

upload_from_file(*path*)
Store the contents of the file pointed by the "path" variable.

Parameters **path** (`str`) – Absolute path to the file to be uploaded to S3.

Bucket

class `cloudbridge.cloud.interfaces.resources.`**Bucket**

create_object(*name*)
Create a new object within this bucket.

Return type :class:`.BucketObject`

Returns The newly created bucket object

delete(*delete_contents=False*)
Delete this bucket.

Parameters **delete_contents** (`bool`) – If `True`, all objects within the bucket will be deleted.

Return type `bool`

Returns `True` if successful.

get(*name*)
Retrieve a given object from this bucket.

Parameters **name** (`str`) – The identifier of the object to retrieve

Return type :class:`.BucketObject`

Returns The BucketObject or `None` if it cannot be found.

id
Get this bucket's id.

Return type `str`

Returns ID of this bucket as returned by the cloud middleware.

list(*limit=None*, *marker=None*, *prefix=None*)
List objects in this bucket.

Parameters

- **limit** (`int`) – Maximum number of elements to return.
- **marker** (`int`) – Fetch results after this offset.
- **prefix** (`str`) – Prefix criteria by which to filter listed objects.

Return type :class:`.BucketObject`

Returns List of all available BucketObjects within this bucket.

name
Get this bucket's name.

Return type `str`

Returns Name of this bucket as returned by the cloud middleware.

Exceptions

CloudBridgeBaseException

class `cloudbridge.cloud.interfaces.exceptions.`**`CloudBridgeBaseException`**
Base class for all CloudBridge exceptions

WaitStateException

class `cloudbridge.cloud.interfaces.exceptions.`**`WaitStateException`**
Marker interface for object wait exceptions. Thrown when a timeout or errors occurs waiting for an object does not reach the expected state within a specified time limit.

InvalidConfigurationException

class `cloudbridge.cloud.interfaces.exceptions.`**`InvalidConfigurationException`**
Marker interface for invalid launch configurations. Thrown when a combination of parameters in a LaunchConfig object results in an illegal state.

CHAPTER 5

Page index

- genindex

Index