Tobiko

Release 0.4.1

Tobiko's Team

Mar 19, 2021

CONTENTS

CHAPTER

ONE

TEST BIG CLOUD OPERATIONS

Tobiko is an OpenStack testing framework focusing on areas mostly complementary to Tempest. While tempest main focus has been testing OpenStack rest APIs, the main Tobiko focus is to test OpenStack system operations while "simulating" the use of the cloud as the final user would.

Tobiko's test cases populate the cloud with workloads such as Nova instances; they execute disruption operations such as services/nodes restart; finally they run test cases to validate that the cloud workloads are still functional.

Tobiko's test cases can also be used, for example, for testing that previously created workloads are working right after OpenStack services update/upgrade operation.

CHAPTER

TWO

PROJECT REQUIREMENTS

Tobiko Python framework is being automatically tested with below Python versions:

- Python 3.6
- Python 3.8
- Python 3.9 (new)

and below Linux distributions:

- CentOS 7 / RHEL 7 (with Python 3.6)
- CentOS 8 / RHEL 8 (with Python 3.6)
- Ubuntu Focal (with Python 3.8)

Tobiko has also been tested for development purposes with below OSes:

- Fedora 31 (with Python 3.7)
- Fedora 32 (with Python 3.8)
- Fedora 33 (with Python 3.9)
- OSX (with Python 3.6 and Python 3.8)
- Ubuntu Bionic (with Python 3.6)
- Ubuntu Focal (with Python 3.9)

The Tobiko Python framework is being used to implement test cases. As Tobiko can be executed on nodes that are not part of the cloud to test against, this doesn't mean Tobiko requires cloud nodes have to run with one of above Python versions or Linux distributions.

2.1 Main Project Goals

- To test OpenStack and Red Hat OpenStack Platform projects before they are released.
- To provide a Python framework to write system scenario test cases (create and test workloads), to write white boxing test cases (to log to cloud nodes for internal inspection purpose), to write disruptive test cases (to simulate service disruptions like for example rebooting/interrupting a service to verify cloud reliability).
- To provide Ansible roles to implement a work-flow designed to run an ordered sequence of test cases groups (like for example tests that creates resources and verify they are working, tests that execute cloud disruptions, and finally tests that verify if resources initially created are still working). The main use of these roles is writing continuous integration jobs for Zuul (via bare Ansible roles) or other services like Jenkins (via the InfraRed plug-in).

- To verify previously created workloads are working fine after executing OpenStack nodes update/upgrade.
- To provide tools to monitor and recollect the healthy status of the cloud as seen from user perspective (black-box testing) or from inside (white-box testing).

CHAPTER

THREE

REFERENCES

- Free software: Apache License, Version 2.0
- Documentation: https://tobiko.readthedocs.io/en/latest/
- Release notes: https://docs.openstack.org/releasenotes/tobiko/
- Source code: https://opendev.org/x/tobiko
- Bugs: https://storyboard.openstack.org/#!/project/x/tobiko

3.1 Related projects

- OpenStack: https://www.openstack.org/
- Red Hat OpenStack Platform: https://www.redhat.com/en/technologies/linux-platforms/openstack-platform
- Python: https://www.python.org/
- Testtools: https://github.com/testing-cabal/testtools
- Ansible: https://www.ansible.com/
- InfraRed: https://infrared.readthedocs.io/en/latest/
- DevStack: https://docs.openstack.org/devstack/latest/
- Zuul: https://docs.openstack.org/infra/system-config/zuul.html
- Jenkins: https://www.jenkins.io/

CHAPTER

FOUR

DOCUMENT OVERVIEW

This document describes the tools for final user and contributors of the project, and assumes that you are already familiar with OpenStack from an end-user perspective. If not, hop over to the OpenStack doc site.

You can look for additional documentation also in the OpenStack wiki.

This documentation is generated by the Sphinx toolkit and lives in the source tree.

Enjoy!

CHAPTER

FIVE

TOBIKO DOCUMENTATION CONTENTS

5.1 Tobiko User Guide

5.1.1 Tobiko Quick Start Guide

Document Overview

This document describes how to install execute Tobiko test cases using Tox.

See also

To install Tobiko inside a virutalenv please read *Tobiko Installation Guide*.

To configure Tobiko please read *Tobiko Configuration Guide*.

To run Tobiko scenario test cases please look at *Tobiko Test Cases Execution Guide*.

Install Dependencies

Install Basic Python Packages

Make sure Git and Python 3 are installed on your system.

For instance on RedHat Linux / Fedora:

```
sudo dnf install -y git python3 which
```

Check your Python 3 version is 3.6 or greater:

```
python3 --version
```

Make sure pip is installed and up-to date:

```
curl https://bootstrap.pypa.io/get-pip.py | sudo python3
```

Check installed Pip version:

```
python3 -m pip --version
```

Make sure basic Python packages are installed and up-to-date:

```
sudo python3 -m pip install --upgrade setuptools wheel virtualenv tox six
```

Check installed Tox version:

```
tox --version
```

Clone the Tobiko repository

Clone the Tobiko repository using Git:

```
git clone https://opendev.org/x/tobiko.git
cd tobiko
```

Install Missing Binary Packages

Install required binary packages:

```
tools/install-bindeps.sh
```

Configure Logging Options

Test cases load most of the configuration parameters from an INI configuration file, typically found at one of the following locations:

- ./tobiko.conf (Tobiko source files directory)
- ~/.tobiko/tobiko.conf
- /etc/tobiko/tobiko.conf

Create it in the Tobiko source directory with the following (or as your preferences). Example:

```
[DEFAULT]
debug = True
log_file = tobiko.log
```

The file 'tobiko.log' is the default file where test cases and the Python framework are going to write their logging messages. By setting debug as 'true' you ensure that messages with the lowest logging level are written there (DEBUG level). The log_file location specified above is relative to the tobiko.conf file location. In this example it is the Tobiko source files' directory itself.

Configure Tobiko Credentials

In order to run the OpenStack test cases you'll need to set up Keystone credentials. You can do it in one of following ways:

- *Set Tobiko Credentials from clouds.yaml file*
- *Set Tobiko Credentials Via Environment Variables*
- *Set Tobiko Credentials Via tobiko-conf File*

Set Tobiko Credentials from clouds.yaml file

Make sure that in any one of below locations there is a valid OpenStack clouds file containing valid Keystone credentials:

- Tobiko source files directory
- ~/.config/openstack
- /etc/openstack

Finally, you will need to specify which credentials Tobiko should pick up via 'OS_CLOUD' environment variable or by specifying the cloud_name in tobiko.conf file (section 'keystone', option 'cloud_name').

Specify 'OS_CLOUD' environment variable

Ensure *OS_CLOUD* environment variable is defined before executing Tobiko test cases:

```
export OS_CLOUD=<cloud_name>
```

Please choose a valid cloud_name from your clouds.yaml file.

Specify cloud_name in tobiko.conf file

Create file *tobiko.conf* in Tobiko sources folder adding a section like below:

```
[keystone]
cloud_name = <cloud_name>
```

Please choose a valid cloud_name from your clouds.yaml file.

Set Tobiko Credentials Via Environment Variables

See also

For more details about supported environment variables please read *Authentication Environment Variables* section.

You can use an existing shell RC file that is valid for Python OpenStack client

```
source openstackrc
```

An example of 'openstackrc' file could looks like below:

```
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_URL=https://my_cloud:13000/v3
export OS_USERNAME=admin
export OS_PASSWORD=secret
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
```

Set Tobiko Credentials Via tobiko.conf File

See also

For more details about supported configuration options please read *Authentication Configuration* section.

Create a file at *~/.tobiko/tobiko.conf* and add a section as in the example below (Or add to your existing file):

```
[keystone]
api_version = 3
auth_url = http://my_cloud:13000/v3
username = admin
password = secret
project_name = admin
user_domain_name = Default
project_domain_name = Default
```

Setup Required Resources

A public Neutron network is required to be able to execute Tobiko scenario test cases by creating a floating IP port on it.

To execute commands from a virtualenv created by Tox you can type as below:

```
tox -e venv -- <your-commands>
```

You need to make sure *Authentication Environment Variables* are properly set so you can list available public networks:

```
tox -e venv -- openstack network list
```

If there isn't any valid public network, you need to create one before running Tobiko OpenStack test cases. Please refer to the Neutron documentation for additional information.

If there is a valid public network for creating floating-IP ports on, Tobiko tests cases will automatically use it. To explicitly select a network, please add a reference to the network in *tobiko.conf* file:

```
[neutron]
floating_network = public
```

Running Test Cases

Running Scenario Test Cases

Scenario test cases are used to create workloads that simulate real-world use of OpenStack. They create networks, virtual machines, ports, routers, etc. They also test validate that these workloads functioning.

Running Tobiko scenario test cases using Tox (may take some time to complete (minutes)):

```
tox -e scenario
```

To list Heat stacks and Glance images created by test cases:

```
tox -e venv -- openstack image list
tox -e venv -- openstack stack list
```

Scenario test cases are also used to check that previously created resources are still up and working as expected. To ensure test cases will not create those resources again we can set *TOBIKO_PREVENT_CREATE* environment variable before re-running test cases:

```
TOBIKO_PREVENT_CREATE=yes tox -e scenario
```

Cleaning Up Tobiko Workloads

Once Tobiko test cases have been executed, we may want to clean up all workloads remaining on the cloud so that we restore it to its original state.

Cleaning Up Heat Stacks

Because Tobiko is using Heat stacks for orchestrating the creation of most of the resources, deleting all stacks created with Tobiko will clean up almost all resources:

```
tox -e venv -- bash -c 'openstack stack list -f value -c ID | xargs openstack stack␣
↪delete'
```

Cleaning Up Glance Images

Because Heat doesn't support creation of Glance images, Tobiko implements some specific fixtures to download images from the Web and upload them to the Glance service:

```
tox -e venv -- bash -c 'openstack image list -f value -c ID | xargs openstack image␣
↪delete'
```

Running Disruptive Test Cases

Disruptive test cases are used for testing that after inducing some critical disruption to the operation of the cloud, the services return working as expected after a while. To execute them you can type:

```
tox -e faults
```

The faults induced by these test cases could be cloud nodes reboot, OpenStack services restart, virtual machines migrations, etc.

Please note that while scenario test cases are being executed in parallel (to speed up test case execution), disruptive test case are only executed sequentially. This is because the operations executed by such cases could break some functionality for a short time and alter the regular state of the system which may be assumed by other test cases to be executed.

Running the Tobiko Workflow

Scenario and disruptive test cases, being executed in a specific sequence could be used to uncover more issues with the cloud than disruptive test cases alone.

- First ensure there are workloads properly running by running scenario test cases:

```
tox -e scenario
```

Note

As second step we may, instead, update or upgrade OpenStack nodes.

- Next we could execute disruptive test cases to "stress" the cloud:

```
tox -e faults
```

- Finally we might re-run scenario test cases to check that everything is still running as expected:

```
TOBIKO_PREVENT_CREATE=yes tox -e scenario
```

Test Cases Report Files

After executing test cases we can view the results in greater detail via a small set of files:

- **test_results.html**: A user-browseable HTML view of test case results
- **test_results.log**: a log file with logging traces collected from every individual test case
- **test_results.subunit**: the original subunit binary file generated by test runner
- **test_results.xml**: an XML Junit file to be used, for example, to show test cases result by Jenkins CI server

The names of the above files can be changed from the default value (*test_results*) to a custom one by setting the *TOX_REPORT_NAME* environment variable.

Legend

{toxinidir} stand for the Tobiko source files directory.

{envname} is the name of the Tox enviroment to be executed (IE scenario, faults, etc.)

The above files are saved into a folder that can be specified with *TOX_REPORT_DIR* environment variable.

By default the full path of the report directory is made from the below:

```
{toxinidir}/report/{envname}
```

5.1.2 Tobiko Installation Guide

Document Overview

This document describes how to install Tobiko inside a Python virtualenv.

See also

For a quick and simpler start you can jump to the *Tobiko Quick Start Guide*.

To configure Tobiko please read *Tobiko Configuration Guide*.

To run Tobiko scenario test cases please look at *Tobiko Test Cases Execution Guide*.

Install Tobiko Using virtualenv

Make sure gcc, Git and base Python packages are installed on your system.

For instance on a RHEL7 or CentOS 7 machine you could type:

```
sudo yum install -y gcc git python python-devel wget
```

For instance on a RHEL8 or CentOS 8 machine you could type:

```
sudo dnf install -y gcc git python3 python3-devel wget
sudo alternatives --set python /usr/bin/python3
```

Make sure pip is installed and up-to-date:

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
PIP=$(which pip)
```

Make sure setuptools, virtualenv, and wheel are installed and up-to-date:

```
sudo $PIP install --upgrade setuptools virtualenv wheel
```

Get Tobiko source code using Git and enter into Tobiko source folder:

```
git clone https://opendev.org/x/tobiko.git
cd tobiko
```

To install Tobiko and its dependencies it's safest to create a clean virtualenv to install it. Create a virtualenv and activate it:

```
virtualenv .tobiko-env
source .tobiko-env/bin/activate
```

Install Tobiko and its requirements:

```
pip install \
    -c https://opendev.org/openstack/requirements/raw/branch/master/upper-constraints.
↪txt \
    .
```

What's Next

To know how to configure Tobiko please read *Tobiko Configuration Guide*.

5.1.3 Tobiko Configuration Guide

Document Overview

This document describes how to configure Tobiko.

See also

For a quick and simpler start you can jump to the *Tobiko Quick Start Guide*.

To install Tobiko inside a virutalenv please read *Tobiko Installation Guide*.

To run Tobiko scenario test cases please look at *Tobiko Test Cases Execution Guide*.

Configure Tobiko Framework

In order to make sure Tobiko tools can connect to OpenStack services via Rest API configuration parameters can be passed either via environment variables or via an INI configuration file (referred here as *tobiko.conf*). Please look at *Authentication Methods* for more details.

To be able to execute scenario test cases there some OpenStack resources that have to be created before running test cases. Please look at *Setup Required Resources* for more details.

tobiko.conf

Tobiko tries to load *tobiko.conf* file from one of the below locations:

- current directory:

```
./tobiko.conf
```

- user home directory:

```
~/.tobiko/tobiko.conf
```

- system directory:

```
/etc/tobiko/tobiko.conf
```

Configure Logging

Tobiko can configure a logging system to write messages to a log file. You can edit the below options in *tobiko.conf* to enable it as below:

```
[DEFAULT]
# Whenever to allow debugging messages to be written out or not
debug = true

# Name of the file where log messages will be appended.
log_file = tobiko.log

# The base directory used for relative log_file paths.
log_dir = .
```

Authentication Methods

Tobiko uses OpenStack client to connect to OpenStack services.

Authentication Environment Variables

To configure how Tobiko can connect to services you can use the same environment variables you would use for OpenStack Python client CLI.

Currently supported variables are:

```
# Identity API version
export OS_IDENTITY_API_VERSION=3

# URL to be used to connect to OpenStack Irentity Rest API service
export OS_AUTH_URL=http://10.0.0.109:5000/v3

# Authentication username (name or ID)
export OS_USERNAME=admin
export OS_USER_ID=...

# Authentication password
export OS_PASSWORD=...

# Project-level authentication scope (name or ID)
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_PROJECT_ID=...
export OS_TENANT_ID=...

# Domain-level authorization scope (name or ID)
export OS_DOMAIN_NAME=Default
export OS_DOMAIN_ID=...

# Domain name or ID containing user
export OS_USER_DOMAIN_NAME=Default
export OS_USER_DOMAIN_ID=...

# Domain name or ID containing project
```

(continues on next page)

(continued from previous page)

```
export OS_PROJECT_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_ID=...

# ID of the trust to use as a trustee user
export OS_TRUST_ID=...
```

Authentication Configuration

You can also configure the same authentication parameters by editing 'keystone' section in *tobiko.conf* file. For example:

```
[keystone]

# Identity API version
api_version = 3

# URL to be used to connect to OpenStack Identity REST API service
auth_url = http://10.0.0.109:5000/v3

# Authentication username (name or ID)
username = admin

# Authentication password
password = ...

# Project-level authentication scope (name or ID)
project_name = admin

# Domain-level authorization scope (name or ID)
domain = default

# Domain name or ID containing user
user_domain_name = default

# Domain name or ID containing project
project_domain_name = default

# ID of the trust to use as a trustee user
trust_id = ...
```

Proxy Server Configuration

The first thing to do is make sure Tobiko can reach OpenStack services. In case OpenStack is not directly accessible from where test cases or Tobiko CLI are executed, it is possible to use an HTTP proxy server running on a network that is able to reach all OpenStack REST API service. This can be performed by using below standard environment variables:

```
export http_proxy=http://<proxy-host>:<proxy-port>/
export https_proxy=http://<proxy-host>:<proxy-port>/
export no_proxy=127.0.0.1,...
```

For convenience it is also possible to specify the same parameters via *tobiko.conf*:

```
[http]
http_proxy = http://<proxy-host>:<proxy-port>/
https_proxy = http://<proxy-host>:<proxy-port>/
no_proxy = 127.0.0.1,...
```

Because Tobiko test cases might execute local commands (like for example ping) to reach network services we have to specify in tobiko.conf file a shell (like OpenSSH client) to be used instead of the default local one ('/bin/sh'):

```
[shell]
command = /usr/bin/ssh <proxy-host>
```

Please make sure it is possible to execute commands on local system without having to pass a password:

```
/usr/bin/ssh <proxy-host> echo 'Yes it works!'
```

To archive it please follow one of the many guides available on Internet .

Setup Required Resources

To be able to execute Tobiko scenario test cases there some OpenStack resources that have to be created before running test cases.

Install required Python OpenStack clients:

```
pip install --upgrade \
    -c https://opendev.org/openstack/requirements/raw/branch/master/upper-constraints.
↪txt \
    python-openstackclient \
    python-neutronclient
```

You need to make sure *Authentication Environment Variables* are properly set:

```
source openstackrc
openstack network list
```

Add reference to the network where Tobiko should create floating IP instances in *tobiko.conf* file:

```
[neutron]
floating_network = public
```

Skipping resources creation

In some cases, for example when Tobiko is run after an upgrade of a cloud, it may be expected that resources used for tests have already been created. Tobiko should not try to create resources than and just run tests using what has already been created. To configure Tobiko to not create test resources, the environment variable `TOBIKO_PREVENT_CREATE` can be used:

```
export TOBIKO_PREVENT_CREATE=True
```

If this is set to `True` or `1` then Tobiko will not try to create resources like VMs, networks, routers, or images and just run validations of what exists in the cloud already.

What's Next

To know how to run Tobiko scenario test cases you can look at *Tobiko Test Cases Execution Guide*

5.1.4 Tobiko Test Cases Execution Guide

This document describes how to execute Tobiko scenario test cases.

See also

For a quick and simpler start you can jump to the *Tobiko Quick Start Guide*.

To install Tobiko inside a virtualenv please read *Tobiko Installation Guide*.

To configure Tobiko please read *Tobiko Configuration Guide*.

Prepare Your System

Before running Tobiko test cases, you need to be sure you are doing it from Tobiko source files folder and that you have activated a virtualenv where Tobiko and its requirements are installed. Please refer to *Tobiko Installation Guide* and *Tobiko Configuration Guide* to know how to setup your system before running test cases.

Run Scenario Test Cases

To run test cases you need a test runner able to execute Python test cases. Test cases delivered with Tobiko has been tested using stestr

From the Tobiko source folder you can run scenario test cases using the below command:

```
stestr run --test-path tobiko/tests/scenario/
```

5.2 Project Contributor Guide

See also

To configure your workstation to develop tobiko test cases please read *Tobiko Contributor Guide*.

5.2.1 Tobiko Contributor Guide

Document Overview

This document describes how to configure a developer workstation to run Tobiko test cases against a remote OpenStack cloud

See also

To just execute Tobiko test cases pelase read *Tobiko Quick Start Guide*.

To install Tobiko inside a virutalenv please read *Tobiko Installation Guide*.

To configure Tobiko please read *Tobiko Configuration Guide*.

To run Tobiko scenario test cases please look at *Tobiko Test Cases Execution Guide*.

This tutorial will guide you to configure a developer workstation to be able to run Tobiko test cases locally without requiring direct network connectivity to a remote OpenStack cloud (DevStack based or TripleO based) so that the edit-try-debug cycle should get shorter and simpler than by editing test cases on one host that is not the same where test cases are being executed:

```
[ test cases host ] - SSH -> [SSH proxy host] - IP -> [OpenStack cloud]
```

Install Dependencies

Install Basic Python Packages

Make sure Git and Python 3 are installed on your system.

For instance on RedHat Linux / Fedora:

```
sudo dnf install -y git python3 which
```

Check your Python 3 version is 3.6 or greater:

```
python3 --version
```

Make sure pip is installed and up-to date:

```
curl https://bootstrap.pypa.io/get-pip.py | sudo python3
```

Check installed Pip version:

```
python3 -m pip --version
```

Make sure basic Python packages are installed and up-to-date:

```
sudo python3 -m pip install --upgrade setuptools wheel virtualenv tox six
```

Check installed Tox version:

```
tox --version
```

Clone the Tobiko repository

Clone the Tobiko repository using Git:

```
git clone https://opendev.org/x/tobiko.git
cd tobiko
```

Install Missing Binary Packages

Install required binary packages:

```
tools/install-bindeps.sh
```

Configure Logging Options

Test cases load most of the configuration parameters from an INI configuration file, typically found at one of the following locations:

- ./tobiko.conf (Tobiko source files directory)
- ~/.tobiko/tobiko.conf
- /etc/tobiko/tobiko.conf

Create it in the Tobiko source directory with the following (or as your preferences). Example:

```
[DEFAULT]
debug = true
log_file = tobiko.log
```

The file 'tobiko.log' is the default file where test cases and the Python framework are going to write their logging messages. By setting debug as 'true' you ensure that messages with the lowest logging level are written there (DEBUG level). The log_file location specified above is relative to the tobiko.conf file location. In this example it is the Tobiko source files directory itself because in case of a relative path the directory where the tobiko.conf file is used as current directory.

SetUp SSH public key to connect to remote cloud

Tobiko test cases will be able to setup some type of SSH tunneling to be able to reach the remote cloud, but for archiving it you are required to be able to connect to a remote SSH server that is able to connect to the OpenStack services and hosts. We will call that server here as the SSH proxy host.

Tobiko test cases will execute some commands on the SSH proxy host (like ping, nc, curl, etc.) as soon as these command need to have direct connectivity to target cloud.

Test case will use Python REST API clients configured to make HTTP requests coming out from such SSH server (mainly by using nc command) or SSH server direct connect feature.

Test cases will make all SSH connection to cloud nodes by using this SSH proxy host

To resume the purpose of the SSH proxy, all network packages sent by Tobiko test cases to the tested cloud will come from the SSH proxy host, while all Tobiko test cases will be executed from the developer workstation.

In order to archive it, first of all we need to make sure we can connect to the SSH proxy server without requiring any password. We therefore need to have a local SSH key pair to be used by tobiko. This key by default is the same default one used by openSSH client: - default SSH private key filename: *~/.ssh/id_rsa* - default SSH public key filename:

~/.ssh/id_rsa.pub To avoid having problems with other uses of the same file, lets instead create our SSH key pair only for Tobiko in a sub-folder near to your tobiko.conf

Ensure we do have this key pair on your workstation by typing:

```
mkdir -p .ssh
chmod 700 .ssh
ssh-keygen -v -f .ssh/id -N ''
chmod 600 .ssh/id .ssh/id.pub
```

Please note in case you already have this pair of files created before that, the key pair must have an empty passphrase. That means the SSH client will never ask you a password to connect to a remote server using that key pair.

Define below variables to later connect to your SSH server:

```
SSH_HOST=<your-ssh-proxy-address>
SSH_USERNAME=<your-ssh-proxy-user>
```

Ensure you can connect to the remote SSH server using our new key pair without a password:

```
ssh-copy-id -i .ssh/id "${SSH_USERNAME}@${SSH_HOST}"
```

Check the SSH key pair is working:

```
ssh -i .ssh/id "${SSH_USERNAME}@${SSH_HOST}" hostname
```

Create '.ssh/config' file with SSH proxy connection parameters:

```
echo "
Host ssh-proxy ${SSH_HOST}
    IdentityFile .ssh/id
    IdentitiesOnly yes
    HostName ${SSH_HOST}
    User ${SSH_USERNAME}
    StrictHostKeyChecking no
    PasswordAuthentication no
    UserKnownHostsFile /dev/null
" > .ssh/config
chmod 600 .ssh/config
```

Check the SSH config file is working:

```
ssh -F .ssh/config ssh-proxy hostname
```

Now let tell Tobiko test cases to use these SSH key pair and to connect to your SSH remote host by editing tobiko.conf file:

```
[ssh]
proxy_jump = ssh-proxy
config_files = .ssh/config
```

We also want to tell Tobiko to use the same key pair to connect to VMs created by Tobiko test cases:

```
[nova]
key_file = .ssh/id
```

Configure Tobiko Credentials

In order to run the OpenStack test cases Tobiko needs to have Keystone credentials. To make our life simpler we are going to assume you are using one of the two OpenStack distributions supported by Tobiko:

- DevStack
- TripleO

Get credentials from a DevStack host

Get the clouds.yaml file from a remote DevStack host:

```
ssh <... connection options here ...> cat /etc/openstack/clouds.yaml > clouds.yanl
```

If your SSH proxy host configured before is one of your DevStack cloud hosts then you can type:

```
ssh -F .ssh/config ssh-proxy cat /etc/openstack/clouds.yaml > clouds.yaml
```

Edit your tobiko.conf file to pick your DevStack based cloud:

```
[keystone]
cloud_name = devstack-admin
clouds_file_names = clouds.yaml
```

Get credentials from TripleO undercloud host

First of all let discover undercloud host IP by pinging it from ssh-proxy host:

```
UNDERCLOUD_IP=$(
    ssh -F .ssh/config ssh-proxy ping -c 1 undercloud-0 |
    awk '/^PING/{gsub(/\(|\)/,""); print $3}')
echo $UNDERCLOUD_IP
```

Tobiko should be able to get credentials directly from such undercloud node but it must know the address of the undercloud host, so we must edit tobiko.conf to let it know:

```
[tripleo]
undercloud_ssh_hostname=<undercloud-host-address>
```

Run Tobiko test cases

Running Scenario Test Cases

To see if we are now able to execute Tobiko test cases please keep open a new terminal where to watch tobiko.log file on the same folder of tobiko.conf file:

```
tail -F tobiko.log
```

Then in the first terminal execute some Tobiko test cases as below:

```
tox -v -e scenario -- -v tobiko/tests/scenario/neutron/test_floating_ip.
↪py::FloatingIPTest
```

Scenario test cases are used to create workloads that simulate real-world use of OpenStack. They create networks, virtual machines, ports, routers, etc. They also test validate that these workloads functioning.

Running Tobiko scenario test cases using Tox (may take several minutes to complete):

```
tox -e scenario
```

Listing Tobiko Workloads

To manage workloads created by Tobiko please log to remote cloud node

Listing Tobiko Workloads on DevStack

To list workloads generated by tobiko you can use glance and heat CLI from the SSH proxy host node:

```
ssh -i .ssh/config ssh-proxy
export OS_CLOUD=devstack-amdin
openstack image list
openstack stack list
```

Listing Tobiko Workloads on DevStack

To list workloads generated by tobiko you can use glance and heat CLI from the undercloud-0 host node:

```
ssh -F .ssh/config ssh-proxy -t ssh stack@undercloud-0
source overcloudrc
openstack image list
openstack stack list
```

Verify Tobiko Workloads

Scenario test cases are also used to check that previously created resources are still up and working as expected. To ensure test cases will not create those resources again we can set *TOBIKO_PREVENT_CREATE* environment variable before re-running test cases:

```
TOBIKO_PREVENT_CREATE=yes tox -e scenario -- -v tobiko/tests/scenario/neutron/test_
↪floating_ip.py::FloatingIPTest
```

Cleaning Up Tobiko Workloads

Once Tobiko test cases have been executed, we may want to clean up all workloads remaining on the cloud so that we restore it to its original state.

Cleaning Up Heat Stacks

Because Tobiko is using Heat stacks for orchestrating the creation of most of the resources, deleting all stacks created with Tobiko will clean up almost all resources:

```
openstack stack list -f value -c ID | xargs openstack stack delete
```

Cleaning Up Glance Images

Because Heat doesn't support creation of Glance images, Tobiko implements some specific fixtures to download images from the Web and upload them to the Glance service:

```
openstack image list -f value -c ID | xargs openstack image delete
```

Running Disruptive Test Cases

Disruptive test cases are used for testing that after inducing some critical disruption to the operation of the cloud, the services return working as expected after a while. To execute them you can type:

```
tox -e faults
```

The faults induced by these test cases could be cloud nodes reboot, OpenStack services restart, virtual machines migrations, etc.

Please note that while scenario test cases are being executed in parallel (to speed up test case execution), disruptive test case are only executed sequentially. This is because the operations executed by such cases could break some functionality for a short time and alter the regular state of the system which may be assumed by other test cases to be executed.

Running the Tobiko Workflow

Scenario and disruptive test cases, being executed in a specific sequence could be used to uncover more issues with the cloud than disruptive test cases alone.

- First ensure there are workloads properly running by running scenario test cases:

  ```
  tox -e scenario
  ```

Note

As second step we may, instead, update or upgrade OpenStack nodes.

- Next we could execute disruptive test cases to "stress" the cloud:

  ```
  tox -e faults
  ```

- Finally we might re-run scenario test cases to check that everything is still running as expected:

  ```
  TOBIKO_PREVENT_CREATE=yes tox -e scenario
  ```

5.3 Tobiko Framework Reference Guide

- genindex
- search

CHAPTER

SIX

CONFIGURATION REFERENCE

6.1 Tobiko Configuration Options

This section provides a list of all configuration options for Tobiko. These are auto-generated from Tobiko code when this documentation is built.

6.1.1 Configuration Reference

tobiko.conf

6.1.2 Sample Configuration Files

Sample tobiko.conf

This sample configuration can also be viewed in the raw format.

```
[DEFAULT]


[centos]

#
# From tobiko
#

# Default centos image name (string value)
#image_name = <None>

# Default centos image URL (string value)
#image_url = <None>

# Default centos image filename (string value)
#image_file = <None>

# Default centos container format (string value)
#container_format = <None>

# Default centos disk format (string value)
#disk_format = <None>

# Default centos username (string value)
```

(continues on next page)

(continued from previous page)

```
#username = <None>

# Default centos password (string value)
#password = <None>

# Default centos SSH connection timeout (seconds) (floating point value)
#connection_timeout = <None>


[cirros]

#
# From tobiko
#

# Default cirros image name (string value)
#image_name = <None>

# Default cirros image URL (string value)
#image_url = <None>

# Default cirros image filename (string value)
#image_file = <None>

# Default cirros container format (string value)
#container_format = <None>

# Default cirros disk format (string value)
#disk_format = <None>

# Default cirros username (string value)
#username = <None>

# Default cirros password (string value)
#password = <None>

# Default cirros SSH connection timeout (seconds) (floating point value)
#connection_timeout = <None>


[glance]

#
# From tobiko
#

# Default directory where to look for image files (string value)
#image_dir = ~/.tobiko/cache/glance/images


[http]

#
# From tobiko
#

# HTTP proxy URL for Rest APIs (string value)
```

(continues on next page)

(continued from previous page)

```
#http_proxy = <None>

# HTTPS proxy URL for Rest APIs (string value)
#https_proxy = <None>

# Don't use proxy server to connect to listed hosts (string value)
#no_proxy = <None>


[keystone]

#
# From tobiko
#

# Identity API version (integer value)
#api_version = <None>

# Identity service URL (string value)
#auth_url = <None>

# Username (string value)
#username = <None>

# Project name (string value)
#project_name = <None>

# Password (string value)
#password = <None>

# Domain name (string value)
#domain_name = <None>

# User domain name (string value)
#user_domain_name = <None>

# Project domain name (string value)
#project_domain_name = <None>

# Project domain ID (string value)
#project_domain_id = <None>

# Trust ID for trust scoping. (string value)
#trust_id = <None>

# Cloud name used pick authentication parameters from clouds.* (string value)
#cloud_name = <None>

# Directories where to look for clouds files (list value)
#clouds_file_dirs = .,~/.config/openstack,/etc/openstack

# Clouds file names (list value)
#clouds_file_names = clouds.yaml,clouds.yml,clouds.json


[neutron]
```

(continues on next page)

(continued from previous page)

```
#
# From tobiko
#

# Network for creating ports on an external network (string value)
#external_network = <None>

# Network for creating floating IPs (string value)
#floating_network = <None>

# The CIDR block to allocate IPv4 subnets from (string value)
#ipv4_cidr = 10.100.0.0/16

# The mask bits for IPv4 subnets (integer value)
#ipv4_prefixlen = 24

# List of nameservers IPv4 addresses (list value)
#ipv4_dns_nameservers = <None>

# The CIDR block to allocate IPv6 subnets from (string value)
#ipv6_cidr = 2001:db8::/48

# The mask bits for IPv6 subnets (integer value)
#ipv6_prefixlen = 64

# List of nameservers IPv6 addresses (list value)
#ipv6_dns_nameservers = <None>

# Customized maximum transfer unit size
# Notes:
#  - MTU values as small as 1000 has been seen breaking networking binding due
# to an unknown cause.
#  - Too big MTU values (like greater than 1400) may be refused during network
# creation (integer value)
#custom_mtu_size = 1350

# Host where nameservers files are located (string value)
#nameservers_host = <None>

# File to parse for getting default nameservers list (list value)
#nameservers_filenames = /etc/resolv.conf


[nova]

#
# From tobiko
#

# Default SSH key to login to server instances (string value)
#key_file = ~/.ssh/id_rsa


[octavia]

#
# From tobiko
```

(continues on next page)

(continued from previous page)

```
#

# Interval to check for status changes, in seconds. (integer value)
#check_interval = 5

# Timeout, in seconds, to wait for a status change. (integer value)
#check_timeout = 360


[ping]

#
# From tobiko
#

# Number of ICMP messages to wait before ending ping command execution (integer
# value)
#count = 1

# Max seconds waited from ping command before self terminating himself (integer
# value)
#deadline = 5

# If False it will not allow ICMP messages to be delivered in smaller fragments
# (string value)
#fragmentation = <None>

# Seconds of time interval between consecutive before ICMP messages (string
# value)
#interval = 1

# Size in bytes of ICMP messages (including headers and payload) (integer
# value)
#packet_size = <None>

# Maximum time in seconds a sequence of ICMP messages is sent to a destination
# host before reporting as a failure (integer value)
#timeout = 300.0


[rhel]

#
# From tobiko
#

# Default rhel image name (string value)
#image_name = <None>

# Default rhel image URL (string value)
#image_url = <None>

# Default rhel image filename (string value)
#image_file = <None>

# Default rhel container format (string value)
#container_format = <None>
```

(continues on next page)

(continued from previous page)

```
# Default rhel disk format (string value)
#disk_format = <None>

# Default rhel username (string value)
#username = <None>

# Default rhel password (string value)
#password = <None>

# Default rhel SSH connection timeout (seconds) (floating point value)
#connection_timeout = <None>


[shell]

#
# From tobiko
#

# Default shell command used for executing local commands (string value)
#command = /bin/sh -c

# Default sudo command used for executing commands as superuser or another user
# (string value)
#sudo = sudo


[ssh]

#
# From tobiko
#

# Logout debugging messages of paramiko library (boolean value)
#debug = false

# Default SSH client command (string value)
#command = /usr/bin/ssh

# Default SSH port (string value)
#port = <None>

# Default SSH username (string value)
#username = <None>

# Default user SSH configuration files (list value)
#config_files = ssh_config,.ssh/config

# Default SSH private key file(s) (list value)
#key_file = ~/.ssh/id_rsa,.ssh/id

# Set to False to disable connecting to the SSH agent (boolean value)
#allow_agent = false

# Set to True to turn on compression (boolean value)
#compress = false
```

(continues on next page)

(continued from previous page)

```
# SSH connect timeout in seconds (floating point value)
#timeout = 15.0

# Maximum number of connection attempts to be tried before timeout (integer
# value)
#connection_attempts = 120

# Minimal seconds to wait between every failed SSH connection attempt (floating
# point value)
#connection_interval = 5.0

# Time before stopping retrying establishing an SSH connection (integer value)
#connection_timeout = 200.0

# Default SSH proxy server (string value)
#proxy_jump = <None>

# Default proxy command (string value)
#proxy_command = <None>


[testcase]

#
# From tobiko
#

# Timeout (in seconds) used for interrupting test case execution (floating
# point value)
#timeout = <None>


[topology]

#
# From tobiko
#

# List of hostname nodes (list value)
#nodes = <None>

# Default SSH key to login to cloud nodes (string value)
#key_file = <None>

# Default username for SSH login (string value)
#username = <None>

# Default port for SSH login (string value)
#port = <None>

# Limit connectivity to cloud to IPv4 o IPv6 (string value)
# Possible values:
# '' - <No description provided>
# 4 - <No description provided>
# 6 - <No description provided>
#ip_version = <None>
```

(continues on next page)

(continued from previous page)

```
[tripleo]

#
# From tobiko
#

# hostname or IP address to be used to connect to undercloud host (string
# value)
#undercloud_ssh_hostname = undercloud-0

# TCP port of SSH server on undercloud host (integer value)
#undercloud_ssh_port = <None>

# Username with access to stackrc and overcloudrc files (string value)
#undercloud_ssh_username = stack

# SSH key filename used to login to Undercloud node (string value)
#undercloud_ssh_key_filename = ~/.ssh/id_rsa

# Undercloud RC filename (list value)
#undercloud_rcfile = ~/stackrc

# TCP port of SSH server on overcloud hosts (integer value)
#overcloud_ssh_port = <None>

# Default username used to connect to overcloud nodes (string value)
#overcloud_ssh_username = heat-admin

# SSH key filename used to login to Overcloud nodes (string value)
#overcloud_ssh_key_filename = ~/.ssh/id_overcloud

# Overcloud RC filenames (list value)
#overcloud_rcfile = ~/overcloudrc,~/qe-Cloud-0rc

# Default IP address version to be used to connect to overcloud nodes  (integer
# value)
#overcloud_ip_version = <None>

# Name of network used to connect to overcloud nodes (string value)
#overcloud_network_name = <None>


[ubuntu]

#
# From tobiko
#

# Default ubuntu image name (string value)
#image_name = <None>

# Default ubuntu image URL (string value)
#image_url = <None>

# Default ubuntu image filename (string value)
```

(continues on next page)

(continued from previous page)

```
#image_file = <None>

# Default ubuntu container format (string value)
#container_format = <None>

# Default ubuntu disk format (string value)
#disk_format = <None>

# Default ubuntu username (string value)
#username = <None>

# Default ubuntu password (string value)
#password = <None>

# Default ubuntu SSH connection timeout (seconds) (floating point value)
#connection_timeout = <None>
```