Red Hat OpenStack Platform 17.0

OpenStack Integration Test Suite Guide

Introduction to the OpenStack Integration Test Suite

Last Updated: 2023-09-06

Red Hat OpenStack Platform 17.0 OpenStack Integration Test Suite Guide

Introduction to the OpenStack Integration Test Suite

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

Install, configure, and manage the OpenStack Integration Test Suite (tempest) in a Red Hat OpenStack Platform environment so that you can validate your deployments.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Tell us how we can make it better.

Using the Direct Documentation Feedback (DDF) function

Use the **Add Feedback** DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the *Multi-page HTML* format.
2. Ensure that you see the **Feedback** button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click **Add Feedback**.
5. Complete the **Add Feedback** field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click **Submit**.

CHAPTER 1. OPENSTACK INTEGRATION TEST SUITE (TEMPEST) VALIDATIONS

Because Red Hat OpenStack Platform (RHOSP) consists of many different projects, it is important to test the interoperability of the projects within your RHOSP cluster. The OpenStack Integration Test Suite automates the integration testing of your RHOSP deployment. You can run tests to ensure that your cluster works as expected. Test output to provide early warning of potential problems, especially after an upgrade.

The Integration Test Suite contains tests for OpenStack API validation and scenario testing, as well as unit testing for self-validation. The Integration Test Suite performs black box testing by using the OpenStack public APIs, with tempest as the test runner.

The OpenStack Integration Test Suite (tempest) acts as a gate for commits to the Red Hat OpenStack Platform (RHOSP) core projects, it can stress test to generate load on a cloud deployment, and it can perform CLI tests to check the response formatting of the command line. You can run **scenario tests** and **API tests** against your RHOSP cloud deployment.

Scenario tests

Scenario tests simulate a typical end user action workflow to test the integration points between services. The testing framework conducts the configuration, tests the integration between services, and is then removed automatically. Tag the tests with the services that they relate to clarify, which client libraries the test uses.

The following scenarios are based on a use case:

- Uploading an image to the Image Service
- Deploying an instance from the image
- Attaching a volume to the instance
- Creating a snapshot of the instance
- Detaching the volume from the instance

API tests

API tests validate the OpenStack API. Tests use the OpenStack Integration Test Suite implementation of the OpenStack API. You can use both valid and invalid JSON to ensure that error responses are valid. You can run tests independently and you do not have to rely on the previous test state.

CHAPTER 2. INSTALLING THE INTEGRATION TEST SUITE (TEMPEST)

To manually install the Integration Test Suite, Installing the Integration Test Suite manually .

2.1. PREREQUISITES

- An undercloud installation. For more information, see Installing the undercloud.
- An overcloud deployment. For more information, see Creating a basic overcloud with CLI tools .

2.2. INSTALLING THE INTEGRATION TEST SUITE MANUALLY

If you do not want to install the Integration Test Suite (tempest) automatically with director, you can perform the installation manually later. You must ensure that you have a basic network configuration, install the Integration Test Suite packages, and create a configuration file that contains details about your OpenStack services and other testing behaviour switches.

Procedure

1. Ensure that the following networks are available within your Red Hat OpenStack Platform (RHOSP) environment:
   - An external network that can provide a floating IP.
   - A private network.
     Connect these networks through a router.
     a. To create the private network, specify the following options according to your network deployment:

```
$ openstack network create <network_name> --share
$ openstack subnet create <subnet_name> --subnet-range <address/prefix> \
  --network <network_name>
$ openstack router create <router_name>
$ openstack router add subnet <router_name> <subnet_name>
```

     b. To create the public network, specify the following options according to your network deployment:

```
$ openstack network create <network_name> --external \
  --provider-network-type flat \
  --provider-physical-network datacentre
$ openstack subnet create <subnet_name> --subnet-range <address/prefix> \
  --gateway <default_gateway> --no-dhcp --network <network_name>
$ openstack router set <router_name> --external-gateway <public_network_name>
```

2. Install the packages related to the Integration Test Suite:

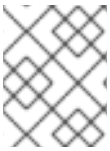
```
$ sudo dnf -y install openstack-tempest
```

   This command does not install any tempest plugins. You must install the plugins manually, depending on your RHOSP installation.

3. Install the appropriate tempest plugin for each component in your environment. For example, enter the following command to install the keystone, neutron, cinder, and telemetry plugins:

```
$ sudo dnf install python3-keystone-tests-tempest python3-neutron-tests-tempest python3-cinder-tests-tempest python3-telemetry-tests-tempest
```

For a full list of packages, see Integration Test Suite packages.



NOTE

You can also install the **openstack-tempest-all** package. This package contains all of the tempest plugins.

2.2.1. Integration Test Suite packages

Use **dnf search** to retrieve a list of tempest test packages:

```
$ sudo dnf search $(openstack service list -c Name -f value) 2>/dev/null | grep test | awk '{print $1}'
```

Component	Package Name
barbican	python3-barbican-tests-tempest
cinder	python3-cinder-tests-tempest
designate	python3-designate-tests-tempest
ec2-api	python3-ec2api-tests-tempest
heat	python3-heat-tests-tempest
ironic	python3-ironic-tests-tempest
keystone	python3-keystone-tests-tempest
kuryr	python3-kuryr-tests-tempest
manila	python3-manila-tests-tempest
mistral	python3-mistral-tests-tempest
networking-bgvpn	python3-networking-bgpvpn-tests-tempest
networking-l2gw	python3-networking-l2gw-tests-tempest
neutron	python3-neutron-tests-tempest
nova-join	python3-novajoin-tests-tempest

Component	Package Name
octavia	python3-octavia-tests-tempest
patrole	python3-patrole-tests-tempest
telemetry	python3-telemetry-tests-tempest
tripleo-common	python3-tripleo-common-tests-tempest
zaqar	python3-zaqar-tests-tempest

NOTE

The **python3-telemetry-tests-tempest** package contains plugins for aodh, panko, gnocchi, and ceilometer tests. The **python3-ironic-tests-tempest** package contains plugins for ironic and ironic-inspector.

CHAPTER 3. CONFIGURING THE INTEGRATION TEST SUITE (TEMPEST)

Before you begin validating your environment with the Integration Test Suite, you must create a workspace and generate the **/etc/tempest.conf** configuration file.

3.1. PREREQUISITES

- An OpenStack environment that contains the Integration Test Suite packages.

3.2. CREATING A WORKSPACE

Create a workspace for your Integration Test Suite (tempest) configuration and output.

Procedure

1. Source the credentials for the target deployment:
   - If the target is in the undercloud, source the credentials for the undercloud:

     ```
     # source stackrc
     ```

   - If the target is in the overcloud, source the credentials for the overcloud:

     ```
     # source overcloudrc
     ```

2. Initialize **tempest**:

   ```
   # tempest init mytempest
   # cd mytempest
   ```

   This command creates a tempest workspace named **mytempest**.

3. Optional: Enter the following command to view a list of existing workspaces:

   ```
   # tempest workspace list
   ```

4. Generate the **etc/tempest.conf** file:

   ```
   # discover-tempest-config --deployer-input ~/tempest-deployer-input.conf \
   --debug --create --network-id <UUID>
   ```

   Replace **UUID** with the UUID of the external network.

   discover-tempest-config was formerly called **config_tempest.py** and uses the same parameters. **python-tempestconf** is as a dependency of **openstack-tempest** and provides the **discover-tempest-config**.

NOTE

To generate the **etc/tempest.conf** file for the undercloud, ensure that the region name in the **tempest-deployer-input.conf** file is the same as the name in the undercloud deployment. If these names do not match, update the region name in the **tempest-deployer-input.conf** file to match the region name of your undercloud.

5. To inspect the region name of your undercloud, enter the following commands:

```
$ source stackrc
$ openstack region list
```

6. To inspect the region name of your overcloud, enter the following commands:

```
$ source overcloudrc
$ openstack region list
```

You might need to modify the default **tempest.conf** file to suit your environment. For more information, see Configuring extension lists and Configuring heat_plugin.

Verification

- Verify your current tempest configuration:

```
# tempest verify-config -o <output>
```

The value of **output** is the output file where Integration Test Suite writes your updated configuration. This is different from your original configuration file.

3.3. CONFIGURING THE INTEGRATION TEST SUITE MANUALLY

The **discover-tempest-config** command generates the **tempest.conf** file automatically. However, you must ensure that the **tempest.conf** file corresponds to the configuration of your environment.

3.3.1. Configuring Integration Test Suite extension lists manually

The default **tempest.conf** file contains lists of extensions for each component. Inspect the **api_extensions** attribute for each component in the **tempest.conf** file and verify that the lists of extensions correspond to your deployment.

If the extensions that are available in your deployment do not correspond to the list of extensions in the **api_extensions** attribute of the **tempest.conf** file, the component fails tempest tests. To prevent this failure, you must identify the extensions that are available in your deployment and include them in the **api_extensions** parameter. To get a list of Network, Compute, Volume, or Identity extensions in your deployment, run the following command:

Procedure

- To retrieve a list of Network, Compute, Volume, or Identity extensions in your deployment, enter the following command:

```
$ openstack extension list [--network] [--compute] [--volume] [--identity]
```

3.3.2. Configuring heat_plugin manually

You can configure **heat_plugin** manually in the **tempest.conf** file.

Procedure

- Use the following example to configure **heat_plugin** according to your deployment:

```
[service_available]
heat = True

[heat_plugin]
username = demo
password = ***
project_name = demo
admin_username = admin
admin_password = ****
admin_project_name = admin
auth_url = http://10.0.0.110:5000//v3
auth_version = 3
user_domain_id = default
project_domain_id = default
user_domain_name = Default
project_domain_name = Default
region = regionOne
fixed_network_name = demo_project_network
network_for_ssh = public
floating_network_name = nova
instance_type = m1.nano
minimal_instance_type = m1.micro
image_ref = 7faed41e-a56c-4971-bf48-24e4e23e69a5
minimal_image_ref = 7faed41e-a56c-4971-bf48-24e4e23e69a5
```

Use the **openstack network list** command to identify networks for the **fixed_network_name**, **network_for_ssh**, and **floating_network_name** parameters.

NOTE

You must set **heat** to **True** in the **[service_available]** section of the **tempest.conf** file, and the user in the **username** attribute of the **[heat_plugin]** section must have the role **member**. For example, enter the following command to add the **member** role to the **demo** user:

```
$ openstack role add --user demo --project demo member
```

3.4. CONFIGURING INTEGRATION TEST SUITE LOGGING

You can change the default location for log files in the **logs** directory within your tempest workspace.

Procedure

1. In **tempest.conf**, under the **[DEFAULT]** section, set **log_dir** to the desired directory:

```
[DEFAULT]
log_dir = <directory>
```

2. If you have your own logging configuration file, in **tempest.conf**, under the **[DEFAULT]** section, set **log_config_append** to your file:

```
[DEFAULT]
log_config_append = <file>
```

If you set the **log_config_append** attribute, the Integration Test Suite ignores all other logging configuration in **tempest.conf**, including the **log_dir** attribute.

3.5. CONFIGURING INTEGRATION TEST SUITE MICROVERSION TESTS

The Integration Test Suite (tempest) provides stable interfaces to test the API microversions. To implement microversion tests by using these interfaces, complete the following steps.

Procedure

1. Configure options in the **tempest.conf** configuration file to specify the target microversions. Configure these options to ensure that the supported microversions correspond to the microversions in the OpenStack cloud.

2. You can specify a range of target microversions to run multiple microversion tests in a single Integration Test Suite operation.
   For example, to limit the range of microversions for the **compute** service, in the **[compute]** section of your configuration file, assign values to the **min_microversion** and **max_microversion** parameters:

```
[compute]
min_microversion = 2.14
max_microversion = latest
```

CHAPTER 4. CLEANING INTEGRATION TEST SUITE (TEMPEST) RESOURCES

Before you validate your deployments by using OpenStack Integration Test Suite (tempest), run the **cleanup** command with the **--init-saved-state** flag. This command scans your environment to discover resources, for example networks, volumes, images, flavors, projects, and users. The discovered resources are saved in a file called **saved_state.json**. When the **tempest cleanup** command is executed all resources not recorded in the **saved_state.json** file are deleted.

Prerequisites

- An OpenStack environment that contains the Integration Test Suite packages.
- An Integration Test Suite configuration that corresponds to your OpenStack environment. For more information, see Creating a workspace.
- One or more completed Integration Test Suite validation tests.

4.1. PERFORMING A DRY RUN

Perform a dry run before you execute the cleanup. A dry run lists the files that Integration Test Suite would delete by a cleanup, without actually deleting any files. The **dry_run.json** file contains the list of files that a cleanup deletes.

Procedure

1. Complete the dry run:

```
# tempest cleanup --dry-run
```

2. Review the **dry_run.json** file to ensure that the cleanup does not delete any files that you require for your environment.

4.2. PERFORMING A TEMPEST CLEAN UP

Before you run any **tempest** tests, you must initialize the saved state. This creates the file **saved_state.json**, which prevents the cleanup from deleting objects that must be kept.

WARNING

If you do not run the **cleanup** command with the **--init-saved-state** flag, RHOSP objects are deleted.

If you create objects after running the **cleanup** command with **--init-saved-state**, those objects can be deleted by subsequent **tempest** commands.

Procedure

1. Initialize the saved state to create the **saved_state.json** file:

```
# tempest cleanup --init-saved-state
```

2. Perform the cleanup:

```
# tempest cleanup
```

The **tempest cleanup** command deletes tempest resources but does not delete projects or the tempest administrator account.

NOTE

You can modify the **saved_state.json** file to include or exclude objects that you want to retain or remove.

CHAPTER 5. VALIDATING YOUR OPENSTACK CLOUD WITH THE INTEGRATION TEST SUITE (TEMPEST)

You can run Integration Test Suite validations in many ways with the **tempest run** command. You can also combine multiple options in a single **tempest run** command.

5.1. PREREQUISITES

- An OpenStack environment that contains the Integration Test Suite packages.
- An Integration Test Suite configuration that corresponds to your OpenStack environment. For more information, see Creating a workspace.

5.2. LISTING AVAILABLE TESTS

Use the **--list-tests** option to list all available tests.

Procedure

- Enter the **tempest run** command with either the **--list-tests** or **-l** options to get a list of available tempest tests:

```
# tempest run -l
```

5.3. RUNNING SMOKE TESTS

Smoke testing is a type of preliminary testing which covers only the most important functionality. Although these tests are not comprehensive, running smoke tests can save time if they do identify a problem.

Procedure

- Enter the **tempest run** command with the **--smoke** option:

```
# tempest run --smoke
```

5.4. PASSING TESTS BY USING ALLOWLIST FILES

An allowlist file is a file that contains regular expressions to select tests that you want to include. If you use one or more regular expressions, specify each expression on a separate line.

Procedure

- Enter the **tempest run** command with either the **--whitelist-file** or **-w** options to use an allowlist file:

```
# tempest run -w <whitelist_file>
```

5.5. SKIPPING TESTS BY USING BLOCKLIST FILES

A blocklist file is a file that contains regular expressions to select tests that you want to exclude. If you use one or more regular expressions, specify each expression on a separate line.

Procedure

- Enter the **tempest run** command with either the **--blacklist-file** or **-b** options to use a blocklist file:

```
# tempest run -b <blacklist_file>
```

5.6. RUNNING TESTS IN PARALLEL OR IN SERIES

You can run tests in parallel, or in series. You can also define the number of workers that you want to use when you run parallel tests. By default, the Integration Test Suite uses one worker for each CPU available.

Choose to run the tests serially or in parallel:

- Run the tests serially:

```
# tempest run --serial
```

- Run the tests in parallel (default):

```
# tempest run --parallel
```

- Use the **--concurrency** or **-c** option to specify the number of workers to use when you run tests in parallel:

```
# tempest run --concurrency <workers>
```

5.7. RUNNING SPECIFIC TESTS

Run specific tests with the **--regex** option. The regular expression must be Python regular expression:

Procedure

- Enter the following command:

```
# tempest run --regex <regex>
```

- For example, use the following example command to run all tests that have names that begin with **tempest.scenario**:

```
# tempest run --regex ^tempest.scenario
```

5.8. DELETING INTEGRATION TEST SUITE OBJECTS

Enter the **tempest cleanup** command to delete all Integration Test Suite (tempest) resources. This command also deletes projects, but the command does not delete the administrator account:

Procedure

- Delete the tempest resources:

```
# tempest cleanup --delete-tempest-conf-objects
```