Red Hat OpenStack Platform 16.2

OpenStack Integration Test Suite Guide

Introduction to the OpenStack Integration Test Suite

Last Updated: 2021-09-28

Red Hat OpenStack Platform 16.2 OpenStack Integration Test Suite Guide

Introduction to the OpenStack Integration Test Suite

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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)

You can install the Integration Test Suite either with director or with a manual installation.

- To install the Integration Test Suite with director, see Installing the Integration Test Suite with director.
- 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 WITH DIRECTOR

Use the Red Hat OpenStack Platform (RHOSP) director to install the test suite automatically.

Prerequisites

- You have installed the **python3-tripleoclient** packages. For more information, see Installing director packages in the Director Installation and Usage guide.

Procedure

1. Log in to the undercloud host as the **stack** user.
2. Edit the **undercloud.conf** file located in the home directory of the **stack** user.
3. Set the **enable_tempest** parameter to **true**.

```
enable_tempest = true
```

4. Run the **openstack undercloud install** command to include the extra configuration in the undercloud:

```
$ openstack undercloud install
```

2.3. 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
$ 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:

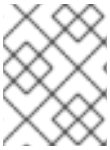
```
$ sudo dnf -y install openstack-tempest
```

   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, horizon, neutron, cinder, and telemetry plugins:

```
$ sudo dnf install python3-keystone-tests-tempest python3-horizon-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.

2.3.1. 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
horizon	python3-horizon-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
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. For more information, see Installing the Integration Test Suite with director .

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** parmameters.

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. 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.

4.1. PREREQUISITES

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

4.2. 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
```

4.3. 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 either the **--whitelist-file** or **-w** options to use a whitelist file:

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

4.4. 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>
```

4.5. 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>
```

4.6. 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>
```

4.7. 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
```

CHAPTER 5. RUNNING THE INTEGRATION TEST SUITE (TEMPEST) FROM A CONTAINER

You can run the Integration Test Suite (tempest) from a container on the undercloud to validate either the undercloud or the overcloud. Containerized and non-containerized Integration Test Suite require the same resources.

5.1. PREPARING THE INTEGRATION TEST SUITE CONTAINER

Download and configure your Integration Test Suite container.

Procedure

1. Change to the **/home/stack** directory.

```
$ cd /home/stack
```

2. Download the tempest container:

```
$ podman pull registry.redhat.io/rhosp-rhel8/openstack-tempest:16.0
```

This container includes all tempest plugins. Running tests globally with this container includes tests for plugins. For example, if you enter the **tempest run --regex '(*.)'** command, Integration Test Suite runs all plug-in tests. These tests fail if your deployment does not contain configuration for all plug-ins. Enter the **tempest list-plugins** command to view all installed plugins. To exclude tests, you must include the tests that you want to exclude in a blocklist file. For more information, see Section 4.5, “Skipping tests by using blocklist files”

3. Create directories to use to exchange data between the host machine and the container:

```
$ mkdir container_tempest tempest_workspace
```

4. Copy the necessary files to the **container_tempest** directory. This directory is the file source for the container:

```
$ cp stackrc overcloudrc tempest-deployer-input.conf container_tempest
```

5. List the available container images:

```
$ podman images
REPOSITORY                                                TAG       IMAGE ID          CREATED
SIZE
registry.redhat.io/rhosp-rhel8/openstack-tempest    latest    881f7ac24d8f        10 days ago
641 MB
```

6. Create an alias to facilitate easier command entry. Ensure that you use absolute paths when you mount the directories:

```
$ alias podman-tempest="podman run -i --privileged=true\
    -v "$(pwd)"/container_tempest:/home/stack/container_tempest:z \
    -v "$(pwd)"/tempest_workspace:/home/stack/tempest_workspace:z \
```

```
    registry.redhat.io/rhosp-rhel8/openstack-tempest:16.0 \
    /bin/bash"
```

7. To retrieve a list of available tempest plugins in the container, enter the following command:

```
$ podman-tempest -c "rpm -qa | grep tempest"
```

5.2. RUNNING THE CONTAINERIZED INTEGRATION TEST SUITE

After you download and configure the Integration Test Suite container, create a script that you can execute within the container to run validation tests.

Procedure

1. Create a tempest script that you can use to execute within the container to generate the **tempest.conf** file and run the tempest tests.

2. Copy the following set of commands and paste them in a Linux console to create your tempest script:

```
$ cat <<'EOF'>> /home/stack/container_tempest/tempest_script.sh
set -e
source /home/stack/container_tempest/overcloudrc
tempest init /home/stack/tempest_workspace
pushd /home/stack/tempest_workspace

export TEMPESTCONF="/usr/bin/discover-tempest-config"

$TEMPESTCONF \
  --out /home/stack/tempest_workspace/etc/tempest.conf \
  --deployer-input /home/stack/container_tempest/tempest-deployer-input.conf \
  --debug \
  --create \
  object-storage.reseller_admin ResellerAdmin

tempest run --smoke
```

   The script performs the following actions:

   - **set -e**
   - Sets the exit status for the command.
   - Sources the **overcloudrc** file if you want to run tempest against the overcloud. Sources the **stackrc** file if you want to run tempest against the undercloud.
   - Runs **tempest init** to create a tempest workspace. Use the shared directory so that the files are also accessible from the host.
   - Changes directory to **tempest_workspace**
   - Exports the TEMPESTCONF environment variable for ease of use at a later stage.

- Executes **discover-tempest-config** to generate the **tempest.conf** file. For more information about the options that you can include in the **discover-tempest-config** command, run **discover-tempest-config --help**.
- Sets **--out** to **home/stack/tempest_workspace/tempest.conf** so that the **tempest.conf** file is accessible from the host machine.
- Sets **--deployer-input** to point to the **tempest-deployer-input.conf** file in the shared directory.
- Runs tempest tests. This example script runs the smoke test **tempest run --smoke**. If you already have a **tempest.conf** file and you want to run only the tempest tests, omit **TEMPESTCONF** from the script and replace it with a command to copy your **tempest.conf** file from the **container_tempest** directory to the **tempest_workspace/etc** directory:

```
$ cp /home/stack/container_tempest/tempest.conf
/home/stack/tempest_workspace/etc/tempest.conf
```

3. Set executable privileges on the **tempest_script.sh** script:

```
$ chmod +x container_tempest/tempest_script.sh
```

4. Run the tempest script from the container by using the alias that you created in a previous step:

```
$ podman-tempest -c 'set -e; /home/stack/container_tempest/tempest_script.sh'
```

5. Inspect the **.stestr** directory for information about the test results.
6. If you want to rerun the tempest tests, you must first remove and recreate the tempest workspace:

```
$ sudo rm -rf /home/stack/tempest_workspace
$ mkdir /home/stack/tempest_workspace
```

5.3. RUNNING THE CONTAINERIZED INTEGRATION TEST SUITE FROM OUTSIDE THE CONTAINER

The container generates or retrieves the **tempest.conf** file and runs tests. You can perform these operations from outside the container:

Procedure

1. If you want to run tempest tests against the overcloud, source the **overcloudrc** file.

```
# source /home/stack/container_tempest/overcloudrc
```

2. If you want to run tempest tests against the undercloud, source the **stackrc** file:

```
# source /home/stack/container_tempest/stackrc
```

3. Create a tempest workspace. Use the shared directory so that the files are also accessible from the host:

```
# tempest init /home/stack/tempest_workspace
```

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

```
# discover-tempest-config \
--out /home/stack/tempest_workspace/tempest.conf \
--deployer-input /home/stack/container_tempest/tempest-deployer-input-conf \
--debug \
--create \
object-storage.reseller_admin ResellerAdmin
```

   For more information about the options that you can include in the **discover-tempest-config** command, enter **discover-tempest-config --help**.

5. Execute the tempest tests. For example, enter the following command to execute the tempest smoke test with the alias you created in a previous step:

```
# podman-tempest -c "tempest run --smoke"
```

6. Inspect the **.stestr** directory for information about the test results.

7. If you want to rerun the tempest tests, you must first remove and recreate the tempest workspace:

```
$ sudo rm -rf /home/stack/tempest_workspace
$ mkdir /home/stack/tempest_workspace
```

CHAPTER 6. CLEANING INTEGRATION TEST SUITE (TEMPEST) RESOURCES

After you run **tempest**, there are files, users and tenants in the testing process that you must delete.

6.1. PREREQUISITES

- An OpenStack environment that contains the Integration Test Suite packages. For more information, see Installing the Integration Test Suite with director .
- 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.

6.2. PERFORMING A CLEAN UP

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

Procedure

1. Initialize the saved state to create the file **saved_state.json**, which prevents the cleanup from deleting necessary objects:

```
# 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.

6.3. 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.

6.4. 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
```