Red Hat OpenStack Platform 13

OpenStack Integration Test Suite 指南

OpenStack Integration Test Suite 简介

Last Updated: 2023-03-29

Red Hat OpenStack Platform 13 OpenStack Integration Test Suite 指南

OpenStack Integration Test Suite 简介

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本指南包含有关在 Red Hat OpenStack Platform 环境中安装、配置和管理 OpenStack Integration Test Suite 的信息。

目录

前言

本指南包含有关在 Red Hat OpenStack Platform 环境中安装、配置和管理 OpenStack Integration Test Suite 的信息。

第 1 章 简介

由于 OpenStack 由许多不同的项目组成，因此测试 OpenStack 集群内项目的互操作性非常重要。OpenStack Integration Test Suite (tempest)自动化您的 Red Hat OpenStack Platform 部署的集成测试。运行测试可确保集群按预期工作，并可能会提前发出警告，特别是在升级后。

Integration Test Suite 包含对 OpenStack API 验证和场景测试的测试，以及对自验证的单元测试。Integration Test Suite 使用 OpenStack 公共 API 执行黑盒测试，并将 tempest 作为测试运行程序。

第 2 章 OPENSTACK INTEGRATION TEST SUITE 测试

OpenStack Integration Test Suite 有许多应用程序。它充当对 OpenStack 核心项目的提交门，它可以压力测试以在云部署上生成负载，并且执行 CLI 测试来检查命令行的响应格式。但是，我们关注的功能是**场景**测试和 **API 测试**。这些测试针对您的 OpenStack 云部署运行。以下小节包含有关实施这些测试的信息。

2.1. 场景测试

场景测试模拟典型的最终用户操作工作流，以测试服务之间的集成点。测试框架执行配置，测试服务间的集成，然后自动删除。使用它们关联的服务标记测试，使其清除测试所使用的客户端库。

场景基于一个用例，例如：

- 将镜像上传到镜像服务
- 从镜像部署实例
- 将卷附加到实例
- 创建实例的快照
- 将卷从实例分离

2.2. API 测试

API 测试验证 OpenStack API。测试使用 OpenStack API 的 OpenStack Integration Test Suite 实施。您可以使用有效和无效的 JSON 来确保错误响应有效。您可以独立运行测试，您不必依赖以前的测试状态。

第 3 章 安装 OPENSTACK INTEGRATION TEST SUITE

本节介绍使用 director 安装 OpenStack Integration Test Suite 或使用手动安装的信息。

3.1. 使用 DIRECTOR

编辑位于 **stack** 用户的主目录中的 **undercloud.conf** 文件。确保将 **enable_tempest** 参数设置为 **true** ：

```
enable_tempest = true
```

如果已安装 undercloud，您可以编辑 **undercloud.conf** 文件，然后运行 **openstack undercloud install** 命令，以在 undercloud 中包含额外的配置：

```
$ openstack undercloud upgrade
```

现在，您可以安装 **tempest** 软件包和插件，如 第 3.3 节 “安装 OpenStack Integration Test Suite 软件包” 所述。

3.2. 准备手动安装

要运行 OpenStack Integration Test Suite，您必须首先安装必要的软件包，并创建一个配置文件，该文件会通知 Integration Test Suite，在其中找到各种 OpenStack 服务和其他测试行为交换机。

要安装 OpenStack Integration Test Suite，必须在 Red Hat OpenStack Platform 环境中提供以下网络：

- 可以提供浮动 IP 的外部网络
- 专用网络

这些网络必须通过路由器进行连接。

创建专用网络。根据您的网络部署指定以下选项：

```
$ 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>
```

创建公共网络。根据您的网络部署指定以下选项：

```
$ 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>
```

现在，您已准备好在 **tempest** 虚拟机中安装和配置 OpenStack Integration Test Suite。更多信息请参阅 第 3.3 节 “安装 OpenStack Integration Test Suite 软件包”。

3.3. 安装 OPENSTACK INTEGRATION TEST SUITE 软件包

1. 安装与 OpenStack Integration Test Suite 相关的软件包：

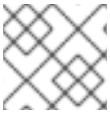
```
$ sudo yum -y install openstack-tempest
```

此命令不会安装任何 tempest 插件。您必须根据 OpenStack 安装手动安装插件。

2. 为环境中的每个组件安装适当的 tempest 插件。例如，运行以下命令来安装 keystone、horizon、neutron、cinder 和 telemetry 插件：

```
$ sudo yum install python-keystone-tests-tempest python-horizon-tests-tempest python-neutron-tests-tempest python-cinder-tests-tempest python-telemetry-tests-tempest
```

如需了解每个 OpenStack 组件的临时插件列表，请参阅 第 3.3.1 节 “Tempest 插件软件包列表”。



注意

您还可以安装 **openstack-tempest-all** 软件包。这个软件包包含所有 tempest 插件。

3.3.1. Tempest 插件软件包列表

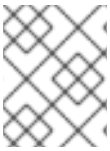
运行以下命令来检索 tempest 测试软件包列表：

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

组件	软件包名称
Barbican	python-barbican-tests-tempest
cinder	python-cinder-tests-tempest
Designate	python-designate-tests-tempest
ec2-api	python-ec2api-tests-tempest
Heat	python-heat-tests-tempest
Horizon	python-horizon-tests-tempest
ironic	python-ironic-tests-tempest
keystone	python-keystone-tests-tempest
kuryr	python-kuryr-tests-tempest
Manila	python-manila-tests-tempest
mistral	python-mistral-tests-tempest
networking-bgvpn	python-networking-bgpvpn-tests-tempest

组件	软件包名称
networking-l2gw	python-networking-l2gw-tests-tempest
neutron	python-neutron-tests-tempest
nova-join	python-novajoin-tests-tempest
octavia	python-octavia-tests-tempest
patrole	python-patrole-tests-tempest
Sahara	python-sahara-tests-tempest
Telemetry	python-telemetry-tests-tempest
tripleo-common	python-tripleo-common-tests-tempest
zaqar	python-zaqar-tests-tempest

Tempest 测试软件包特定于 Python 版本。例如，如果您的系统使用 Python 2，在安装 Tempest 测试软件包时，您必须将 **python-** 替换为 **python2-**。



注意

python-telemetry-tests-tempest 软件包包含 aodh、panko、gnocchi 和 ceilometer 测试的插件。**python-ironic-tests-tempest** 软件包包含 ironic 和 ironic-inspector 的插件。

第 4 章 配置 OPENSTACK INTEGRATION TEST SUITE

4.1. 创建 WORKSPACE

1. 提供目标部署的凭证：

   - 如果目标位于 undercloud 中，请提供 undercloud 的凭据：

     ```
     # source stackrc
     ```

   - 如果目标在 overcloud 中，提供 overcloud 的凭据：

     ```
     # source overcloudrc
     ```

2. 初始化 **tempest** ：

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

   这个命令会创建一个名为 **mytempest** 的临时工作区。

   运行以下命令查看现有工作区列表：

   ```
   # tempest workspace list
   ```

3. 生成 **etc/tempest.conf** 文件：

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

   将 **UUID** 替换为外部网络的 UUID。有关您可以在 **discover-tempest-config** 命令中包含的选项的更多信息，请运行 **discover-tempest-config --help**。例如，使用 **--image** 选项定义您要使用的镜像。如果通过 **--image** 选项分配镜像，您可能需要创建适合分配的镜像的类别。发现 **tempest-config**、m1.nano 和 m1.micro 创建的默认类别可能对该镜像来说太小。创建类别后，您需要在 **tempest.conf** 中将类别 ID 添加到 **flavor_ref** 和 **flavor_ref_alt** 中。

   discover-tempest-config 以前称为 **config_tempest.py**，并采用相同的参数。它由 **python-tempestconf** 提供，它作为 **openstack-tempest** 的依赖项安装。

注意

要为 undercloud 生成 **etc/tempest.conf** 文件，请确保 **tempest-deployer-input.conf** 文件中的区域名称与 undercloud 部署中的名称相同。如果这些名称不匹配，请更新 **tempest-deployer-input.conf** 文件中的区域名称，以匹配 undercloud 的地区名称。

要检查 undercloud 的地区名称，请运行以下命令：

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

要检查 overcloud 的地区名称，请运行以下命令：

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

4.2. 手动配置 TEMPEST

discover-tempest-config 命令自动生成 **tempest.conf** 文件。但是，您必须确保 **tempest.conf** 文件对应于环境的配置。

4.2.1. 手动配置 Tempest 扩展列表

默认 **tempest.conf** 文件包含每个组件的扩展列表。检查 **tempest.conf** 文件中的每个组件的 **api_extensions** 属性，并验证扩展列表是否适合您的部署。

如果您的部署中可用的扩展与 **tempest.conf** 文件的 **api_extensions** 属性中的扩展列表不匹配，则组件会失败 tempest 测试。要防止此失败，您必须识别部署中可用的扩展，并将其包括在 **api_extensions** 参数中。要获取部署中网络、计算、卷或身份扩展列表，请运行以下命令：

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

4.2.2. 手动配置 heat_plugin

根据您的部署配置，手动配置 **heat_plugin** 插件。以下示例包含 **heat_plugin** 的最小 **tempest.conf** 配置：

```
[service_available]
heat_plugin = 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
instance_type = m1.nano
minimal_instance_type = m1.micro
image_ref = 7faed41e-a56c-4971-bf48-24e4e23e69a5
minimal_image_ref = 7faed41e-a56c-4971-bf48-24e4e23e69a5
```

注意

您必须在 **tempest.conf** 文件的 **[service_available]** 部分中将 **heat_plugin** 设置为 **True**，而 **[heat_plugin]** 部分的 **username** 属性中的用户必须具有角色 **_member_**。例如，运行以下命令将 **_member_** 角色添加到 **demo** 用户：

```
$ openstack role add --user demo --project demo '_member_'
```

4.3. 验证 TEMPEST 配置

验证当前的临时配置：

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

输出 是 Tempest 编写您更新的配置的输出文件。这与您原始配置文件不同。

4.4. 更改日志记录配置

日志文件的默认位置是 **tempest** 工作区中的日志目录。

要更改此目录，在 **[DEFAULT]** 部分下的 **tempest.conf** 中，将 **log_dir** 设置为所需的目录：

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

如果您有自己的日志记录配置文件，在 **tempest.conf** 下，在 **[DEFAULT]** 部分下，将 **log_config_append** 设置为您的文件中：

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

如果设置 **log_config_append** 属性，则 Tempest 会忽略 **tempest.conf** 中的所有其他日志记录配置，包括 **log_dir** 属性。

4.5. 配置 MICROVERSION 测试

OpenStack Integration Test Suite 提供稳定的接口来测试 API 微版本。本节包含有关使用这些接口实施微版本测试的信息。

首先，您必须在 **tempest.conf** 配置文件中配置选项，以指定目标微版本。配置这些选项，以确保支持的微版本对应于 OpenStack 云中使用的 microversions。您可以指定一系列目标微版本，在单个集成测试操作中运行多个微版本测试。

例如，若要限制计算服务的微版本范围，在 配置文件的 **[compute]** 部分中，将值分配给 **min_microversion** 和 **max_microversion** 参数：

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

第 5 章 使用 TEMPEST

在本节中运行命令，以执行各种测试操作。您还可以在单个 **tempest run** 命令中组合多个选项。

5.1. 列出可用测试

使用 **--list-tests** 或 **-l** 选项运行 **tempest-run** 命令以获取可用 tempest 测试列表：

```
# tempest run -l
```

5.2. 运行 SMOKE 测试

烟雾测试是一种初步测试，仅介绍最重要的功能。虽然它们不是综合的，但运行烟雾测试可以节省时间（如果他们发现问题）。

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

5.3. 使用白名单文件传递测试

白名单文件是一个包含正则表达式的文件，用于选择您要包括的测试。正则表达式用换行符分隔。

使用 **--whitelist-file** 或 **-w** 选项运行 **tempest run** 命令以使用白名单文件：

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

5.4. 使用黑名单文件跳过测试

blacklist 文件是一个包含正则表达式的文件，用于选择您要排除的测试。正则表达式用换行符分隔。

使用 **--blacklist-file** 或 **-b** 选项运行 **tempest run** 命令，以使用黑名单文件：

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

5.5. 在 PARALLEL CONCURRENTLY 或 SERIALLY 中运行测试

按顺序运行测试：

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

并行运行测试。并行测试是默认测试：

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

使用 **--concurrency** 或 **-c** 选项指定并行运行测试时要使用的 worker 数量：

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

默认情况下，Integration Test Suite 为每个可用 CPU 使用一个 worker。

5.6. 运行特定测试

使用 **--regex** 正则表达式选项运行特定的测试。正则表达式必须是 Python 正则表达式：

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

例如，使用以下命令运行名称以 **tempest.scenario** 开头的所有测试：

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

第 6 章 运行容器化 TEMPEST

本节介绍在 undercloud 上的容器中运行 tempest 的信息。您可以针对 overcloud 或 undercloud 运行温度。容器化 tempest 需要与非容器化 tempest 相同的资源。

该功能在此发行版本中作为*技术预览*提供，因此不享有红帽的全面支持。它只应用于测试，不应部署在生产环境中。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。

6.1. 准备 TEMPEST 容器

完成以下步骤以下载和配置您的临时容器：

1. 进入 **/home/stack** 目录：

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

2. 下载 tempest 容器：

```
$ docker pull registry.redhat.io/rhosp13/openstack-tempest
```

此容器包括所有 tempest 插件。使用这个容器全局运行 tempest 测试包括对插件的测试。例如，如果您运行 **tempest run --regex '(*.)'** 命令，则 tempest 运行所有插件测试。如果您的部署不包含所有插件的配置，则这些临时测试会失败。运行 **tempest list-plugins** 命令以查看所有已安装的插件。要排除测试，您必须在黑名单文件中包括您要排除的测试。更多信息请参阅 第 5 章 *使用 Tempest*。

3. 创建用于在主机和容器间交换数据的目录：

```
$ mkdir container_tempest tempest_workspace
```

4. 将必要的文件复制到 **container_tempest** 目录。这个目录是容器的文件源：

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

5. 列出可用的 Docker 镜像：

```
$ docker images
REPOSITORY                                                TAG       IMAGE ID            CREATED
SIZE
registry.redhat.io/rhosp13-beta/openstack-tempest   latest    881f7ac24d8f        10 days ago
641 MB
```

6. 创建别名以便于命令条目。确定在挂载目录时使用绝对路径：

```
$ alias docker-tempest="docker run -i \
    -v "$(pwd)"/container_tempest:/home/stack/container_tempest \
    -v "$(pwd)"/tempest_workspace:/home/stack/tempest_workspace \
    registry.redhat.io/rhosp13/openstack-tempest \
    /bin/bash"
```

7. 要获取容器中可用临时插件列表，请运行以下命令：

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

6.2. 在容器内运行容器化 TEMPEST

1. 创建一个温度脚本，您可以在容器内执行，以生成 **tempest.conf** 文件并运行 tempest 测试。该脚本执行以下操作：

   - 设置命令 **设置的退出状态 -e**。
   - 如果要针对 overcloud 运行tempest，请 source **overcloudrc** 文件。如果您想针对 undercloud 运行 tempest，则 source **stackrc** 文件。
   - 运行 **tempest init** 以创建温度工作区。使用 共享目录，以便主机也可以访问这些文件。
   - 将目录改为 **tempest_workspace**
   - 导出 TEMPESTCONF 环境变量，以便在以后的阶段使用。
   - 执行 **discover-tempest-config** 以生成 **tempest.conf** 文件。有关您可以在 **discover-tempest-config** 命令中包含的选项的更多信息，请运行 **discover-tempest-config --help**。
   - 将 **--out** 设置为 **home/stack/tempest_workspace/tempest.conf**，以便可以从主机机器访问 **tempest.conf** 文件。
   - 将 **--deployer-input** 设置为指向共享目录中的 **tempest-deployer-input.conf** 文件。
   - 运行 tempest 测试。这个示例脚本运行烟雾测试 **tempest run --smoke**。

     ```
     $ 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

     EOF
     ```

     如果您已有 **tempest.conf** 文件，而您只想运行 tempest 测试，请省略来自脚本的 **TEMPESTCONF**，并使用 命令将您的 **tempest.conf** 文件从 **container_tempest** 目录复制到 **tempest_workspace/etc** 目录：

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

2. 在 **tempest_script.sh** 脚本上设置可执行权限：

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

3. 使用您在上一步中创建的别名从容器运行 tempest 脚本：

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

4. 检查 **.stestr** 目录，以了解有关测试结果的信息。

5. 如果要重新运行 tempest 测试，您必须首先删除并重新创建 tempest 工作区：

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

6.3. 在容器外部运行 CONTAINERIZED TEMPEST

容器生成或检索 **tempest.conf** 文件并运行测试。您可以从容器外部执行这些操作：

1. 如果要针对 overcloud 运行tempest，请 source **overcloudrc** 文件。如果您想针对 undercloud 运行 tempest，则 source **stackrc** 文件：

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

2. 运行 **tempest init** 以创建温度工作区。使用共享目录，以便主机也可以访问这些文件：

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

3. 生成 **tempest.conf** 文件：

```
# 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
```

有关您可以在 **discover-tempest-config** 命令中包含的选项的更多信息，请运行 **discover-tempest-config --help**。

4. 执行温度测试。例如，运行以下命令使用您在上一步中创建的别名执行 tempest 烟雾测试：

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

5. 检查 **.stestr** 目录，以了解有关测试结果的信息。

6. 如果要重新运行 tempest 测试，您必须首先删除并重新创建 tempest 工作区：

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

第 7 章 清理 TEMPEST 资源

运行 **tempest** 后，会存在文件、在测试过程中创建的用户和项目必须被删除。能够自我清理是最温度的设计原则 **之一**。

7.1. 执行清理

首先，您必须初始化保存的状态。这会创建文件 **saved_state.json**，这可防止清理删除需要保留的对象。通常，您将在 tempest run 之前使用 **--init-saved-state** 运行清理。如果情况并非如此，则必须编辑 **saved_state.json** 以删除您要删除的对象。

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

运行清理：

```
# tempest cleanup
```

tempest cleanup 命令删除 tempest 资源，但不会删除项目或温度管理员帐户。

7.2. 执行空运行

建议在执行真实清理前执行空运行。空运行列出了清理将删除但不会删除的任何文件，但不会删除任何文件。文件在 **dry_run.json** 文件中列出。检查 **dry_run.json** 文件，以确保清理不会删除任何环境需要的文件。

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

7.3. 删除 TEMPEST 对象

运行以下命令以删除所有温度资源，包括项目，而不是临时管理员帐户：

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