Red Hat OpenStack Platform 13

OpenStack Integration Test Suite 가이드

OpenStack Integration Test Suite 소개

Last Updated: 2023-04-08

Red Hat OpenStack Platform 13 OpenStack Integration Test Suite 가이드

OpenStack Integration Test Suite 소개

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.

법적 공지

초록

이 가이드에서는 Red Hat OpenStack Platform 환경에서 OpenStack Integration Test Suite를 설치, 구성 및 관리하는 방법에 대해 설명합니다.

차례

머리말 3

1장. 소개 4

2장. OPENSTACK INTEGRATION TEST SUITE 테스트 5
2.1. 시나리오 테스트 5
2.2. API 테스트 5

3장. OPENSTACK INTEGRATION TEST SUITE 설치 6
3.1. DIRECTOR 사용 6
3.2. 수동 설치 준비 6
3.3. OPENSTACK INTEGRATION TEST SUITE 패키지 설치 7
3.3.1. Tempest Plug-in 패키지 목록 7

4장. OPENSTACK INTEGRATION TEST SUITE 구성 10
4.1. 작업 공간 생성 10
4.2. 수동으로 임시 설정 11
4.2.1. 수동으로 임시 확장 목록 구성 11
4.2.2. heat_plugin 수동 구성 12
4.3. TEMPEST 설정 확인 12
4.4. 로깅 구성 변경 12
4.5. MICROVERSION 테스트 구성 13

5장. TEMPEST 사용 14
5.1. 사용 가능한 테스트 나열 14
5.2. SMOKE 테스트 실행 14
5.3. 화이트리스트 파일을 사용하여 테스트 전달 14
5.4. 블랙리스트 파일을 사용하여 테스트 건너뛰기 14
5.5. PARALLEL CONCURRENTLY 또는 SERIALLY에서 테스트 실행 15
5.6. 특정 테스트 실행 15

6장. 컨테이너화된 TEMPEST 실행 16
6.1. TEMPEST 컨테이너 준비 16
6.2. 컨테이너 내부에서 컨테이너화된 TEMPEST 실행 17
6.3. 컨테이너 외부에서 컨테이너화된 TEMPEST 실행 19

7장. 임시 리소스 정리 21
7.1. 정리 작업 수행 21
7.2. DRY RUN 수행 21
7.3. TEMPEST 오브젝트 삭제 21

머리말

이 가이드에서는 Red Hat OpenStack Platform 환경에서 OpenStack Integration Test Suite를 설치, 구성 및 관리하는 방법에 대해 설명합니다.

1장. 소개

OpenStack은 다양한 프로젝트로 구성되어 있으므로 OpenStack 클러스터 내 프로젝트의 상호 운용성을 테스트하는 것이 중요합니다. OpenStack Integration Test Suite(tempest)는 Red Hat OpenStack Platform 배포의 통합 테스트를 자동화합니다. 테스트를 실행하면 클러스터가 예상대로 작동하는지, 특히 업그레이드 후 잠재적인 문제에 대한 조기 경고도 제공할 수 있습니다.

Integration Test Suite에는 OpenStack API 검증 및 시나리오 테스트에 대한 테스트와 자체 유효성 검사를 위한 단위 테스트가 포함되어 있습니다. Integration Test Suite는 테스트 러너로 Tempest를 사용하여 OpenStack 공용 API를 사용하여 블랙 박스 테스트를 수행합니다.

2장. OPENSTACK INTEGRATION TEST SUITE 테스트

OpenStack Integration Test Suite에는 많은 애플리케이션이 있습니다. OpenStack 핵심 프로젝트에 커밋을 위한 게이트 역할을 하며, 클라우드 배포에 대한 부하를 생성하는 테스트를 강조할 수 있으며 CLI 테스트를 수행하여 명령줄의 응답 형식을 확인할 수 있습니다. 그러나 관련 된 기능은 **시나리오 테스트** 및 **API 테스트** 입니다. 이러한 테스트는 OpenStack 클라우드 배포에 대해 실행됩니다. 다음 섹션에는 이러한 각 테스트 구현에 대한 정보가 포함되어 있습니다.

2.1. 시나리오 테스트

시나리오 테스트는 서비스 간 통합 지점을 테스트하기 위해 일반적인 최종 사용자 작업 워크플로우를 시뮬레이션합니다. 테스트 프레임워크는 구성을 수행하고, 서비스 간 통합을 테스트한 다음, 자동으로 제거됩니다. 테스트에서 관련 서비스를 사용하여 테스트에 태그를 지정하여 테스트에서 사용하는 클라이언트 라이브러리를 명확히 합니다.

시나리오는 사용 사례를 기반으로 합니다. 예를 들면 다음과 같습니다.

- 이미지 서비스에 이미지 업로드
- 이미지에서 인스턴스 배포
- 인스턴스에 볼륨 연결
- 인스턴스의 스냅샷 생성
- 인스턴스에서 볼륨 분리

2.2. API 테스트

API 테스트는 OpenStack API를 검증합니다. 테스트에서는 OpenStack API의 OpenStack Integration Test Suite 구현을 사용합니다. 유효한 JSON과 유효하지 않은 JSON을 모두 사용하여 오류 응답이 유효한지 확인할 수 있습니다. 테스트를 독립적으로 실행할 수 있으며 이전 테스트 상태에 의존할 필요가 없습니다.

3장. OPENSTACK INTEGRATION TEST SUITE 설치

이 섹션에는 director 또는 수동 설치와 함께 OpenStack Integration Test Suite 설치에 대한 정보가 포함되어 있습니다.

3.1. DIRECTOR 사용

stack 사용자의 홈 디렉터리에 있는 **undercloud.conf** 파일을 편집합니다. **enable_tempest** 매개변수가 **true** 로 설정되어 있는지 확인합니다.

```
enable_tempest = true
```

언더클라우드가 이미 설치되어 있는 경우 **undercloud.conf** 파일을 편집한 다음 **openstack undercloud install** 명령을 실행하여 언더클라우드에 추가 설정을 포함할 수 있습니다.

```
$ openstack undercloud upgrade
```

이제 3.3절. "OpenStack Integration Test Suite 패키지 설치" 에 설명된 Tempest **패키지** 및 플러그인을 설치할 준비가 되었습니다.

3.2. 수동 설치 준비

OpenStack Integration Test Suite를 실행하려면 먼저 필요한 패키지를 설치하고 다양한 OpenStack 서비스 및 기타 테스트 동작 스위치를 찾을 수 있는 Integration Test Suite를 알리는 구성 파일을 생성해야 합니다.

OpenStack Integration Test Suite를 설치하려면 Red Hat OpenStack Platform 환경에서 다음 네트워크를 사용할 수 있어야 합니다.

- 유동 IP를 제공할 수 있는 외부 네트워크
- 사설 네트워크

이러한 네트워크는 라우터를 통해 연결되어야 합니다.

사설 네트워크를 만듭니다. 네트워크 배포에 따라 다음 옵션을 지정합니다.

```
$ 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>
```

공용 네트워크를 생성합니다. 네트워크 배포에 따라 다음 옵션을 지정합니다.

```
$ 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>
```

이제 가상 머신의 템플릿 내에 OpenStack Integration Test Suite를 설치하고 구성할 준비가 되었습니다. 자세한 내용은 3.3절. “OpenStack Integration Test Suite 패키지 설치”의 내용을 참조하십시오.

3.3. OPENSTACK INTEGRATION TEST SUITE 패키지 설치

1. OpenStack Integration Test Suite와 관련된 패키지를 설치합니다.

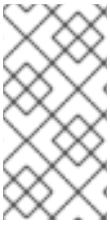
```
$ sudo yum -y install openstack-tempest
```

이 명령은 **tempest** 플러그인을 설치하지 않습니다. OpenStack 설치에 따라 플러그인을 수동으로 설치해야 합니다.

2. 사용자 환경의 각 구성 요소에 적절한 Tempest 플러그인을 설치합니다. 예를 들어 다음 명령을 실행하여 **keystone, horizon, neutron, cinder, telemetry** 플러그인을 설치합니다.

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

각 OpenStack 구성 요소의 템플릿 플러그인 목록은 3.3.1절. “Tempest Plug-in 패키지 목록” 에서 참조하십시오.



참고

openstack-tempest-all 패키지를 설치할 수도 있습니다. 이 패키지에는 모든 Tempest 플러그인이 포함되어 있습니다.

3.3.1. Tempest Plug-in 패키지 목록

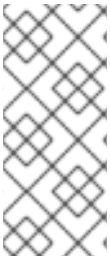
다음 명령을 실행하여 **tempest** 테스트 패키지 목록을 검색합니다.

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

구성 요소	패키지 이름
Barbican	python-barbican-tests-tempest
cinder	python-cinder-tests-tempest

구성 요소	패키지 이름
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
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 테스트 패키지는 Python 버전에 따라 다릅니다. 예를 들어 시스템에서 Python 2를 사용하는 경우 Tempest 테스트 패키지를 설치할 때 python- 을 python2 로 교체해야 합니다.



참고

python-telemetry-tests-tempest 패키지에는 **aodh, panko, gnocchi** 및 **ceilometer** 테스트에 대한 플러그인이 포함되어 있습니다. **python-ironic-tests-tempest** 패키지에는 **ironic** 및 **ironic-inspector**용 플러그인이 포함되어 있습니다.

4장. OPENSTACK INTEGRATION TEST SUITE 구성

4.1. 작업 공간 생성

1. 대상 배포의 인증 정보를 가져옵니다.

   - 대상이 언더클라우드에 있는 경우 언더클라우드의 인증 정보를 가져옵니다.

     ```
     # source stackrc
     ```

   - 대상이 오버클라우드에 있는 경우 오버클라우드의 인증 정보를 가져옵니다.

     ```
     # source overcloudrc
     ```

2. **Tempest(Temp est):**

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

   이 명령은 **mytempest** 라는 **Tempest** 작업 공간을 생성합니다.

   다음 명령을 실행하여 기존 작업 영역 목록을 확인합니다.

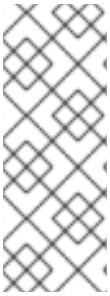
   ```
   # tempest workspace list
   ```

3. **etc/tempest.conf** 파일을 생성하십시오.

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

   UUID 를 외부 네트워크의 **UUID**로 바꿉니다. **discover-tempest-config** 명령에 포함할 수 있는 옵션에 대한 자세한 내용은 **discover-tempest-config --help** 를 실행합니다. 예를 들어 **--image** 옵션을 사용하여 사용하려는 이미지를 정의합니다. **--image** 옵션을 사용하여 이미지를 신속하게 할당하는 경우 할당된 이미지에 적합한 플레이버를 생성해야 할 수 있습니다. **discover-tempest-config, m1.nano** 및 **m1.micro**로 만든 기본 플레이버는 이미지에 비해 너무 작을 수 있습니다. 플레이버를 만든 후 플레이버 **id**를 **flavor_ref** 및 **flavor_ref _alt** 의 **tempest.conf** 에 추가해야 합니다.

discover-tempest-config 는 이전에 **config_tempest.py** 라고 하며 동일한 매개 변수를 사용합니다. **python-tempestconf** 는 **openstack-tempest** 의 종속성으로 설치된 **python-tempestconf**에서 제공합니다.



참고

언더클라우드의 **etc/tempest.conf** 파일을 생성하려면 **tempest-deployer-input.conf** 파일의 리전 이름이 언더클라우드 배포의 이름과 동일한지 확인합니다. 이러한 이름이 일치하지 않으면 **undercloud**의 지역 이름과 일치하도록 **tempest-deployer-input.conf** 파일의 리전 이름을 업데이트합니다.

언더클라우드의 리전 이름을 검사하려면 다음 명령을 실행합니다.

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

오버클라우드의 지역 이름을 검사하려면 다음 명령을 실행합니다.

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

4.2. 수동으로 임시 설정

discover-tempest-config 명령은 **tempest.conf** 파일을 자동으로 생성합니다. 그러나 **tempest.conf** 파일이 환경의 구성에 해당하는지 확인해야 합니다.

4.2.1. 수동으로 임시 확장 목록 구성

기본 **tempest.conf** 파일에는 각 구성 요소에 대한 확장 프로그램 목록이 포함되어 있습니다. **tempest.conf** 파일의 각 구성 요소의 **api_extensions** 속성을 검사하고 확장 목록이 배포에 일치하는지 확인합니다.

배포에서 사용할 수 있는 확장 프로그램이 **tempest.conf** 파일의 **api_extensions** 속성의 **extensions** 목록에 일치하지 않으면 구성 요소가 **tempest** 테스트에서 실패합니다. 이 실패를 방지하려면 배포에서 사용할 수 있는 확장을 식별하고 **api_extensions** 매개변수에 포함해야 합니다. 배포에서 네트워크, 컴퓨팅, 볼륨 또는 **ID** 확장 목록을 가져오려면 다음 명령을 실행합니다.

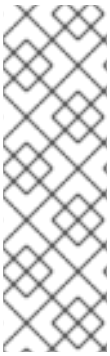
```
$ openstack extension list [--network] [--compute] [--volume] [--identity]
```

4.2.2. heat_plugin 수동 구성

배포 구성에 따라 **heat_plugin** 플러그인을 수동으로 구성합니다. 다음 예제에는 **heat_plugin** 에 대한 최소 **tempest.conf** 설정이 포함되어 있습니다.

```
[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
```



참고

tempest.conf 파일의 **[service_available]** 섹션에서 **heat_plugin** 을 **True** 로 설정해야 하며, **[heat_plugin]** 섹션의 **username** 속성에 있는 사용자는 **_member_** 여야 합니다. 예를 들어 다음 명령을 실행하여 **_member_** 역할을 **demo** 사용자에게 추가합니다.

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

4.3. TEMPEST 설정 확인

현재 **Tempest** 구성을 확인합니다.

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

출력 은 **Tempest**가 업데이트된 구성을 쓰는 출력 파일입니다. 이는 원래 구성 파일과 다릅니다.

4.4. 로깅 구성 변경

로그 파일의 기본 위치는 **tempest** 작업 공간 내의 **logs** 디렉터리입니다.

이 디렉토리를 변경하려면 **[DEFAULT]** 섹션의 **tempest.conf** 에서 **log_dir** 을 원하는 디렉터리로 설정합니다.

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

고유한 로깅 구성 파일이 있는 경우 **tempest.conf** 의 **[DEFAULT]** 섹션에서 **log_config_append** 를 파일로 설정합니다.

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

log_config_append 속성을 설정하면 **Tempest**는 **log_dir** 특성을 포함하여 **tempest.conf** 의 다른 모든 로깅 구성을 무시합니다.

4.5. MICROVERSION 테스트 구성

OpenStack Integration Test Suite는 **API** 마이크로 버전을 테스트하는 안정적인 인터페이스를 제공합니다. 이 섹션에서는 이러한 인터페이스를 사용하여 마이크로버전 테스트를 구현하는 방법을 설명합니다.

먼저 타겟 마이크로버전을 지정하려면 **tempest.conf** 구성 파일의 옵션을 구성해야 합니다. 지원되는 마이크로 버전이 **OpenStack** 클라우드에서 사용된 마이크로버전에 해당하도록 이러한 옵션을 구성합니다. 대상 마이크로 버전 범위를 지정하여 단일 **Integration Test Suite** 작업에서 여러 마이크로 버전 테스트를 실행할 수 있습니다.

예를 들어 구성 파일의 **[compute]** 섹션에서 컴퓨팅 서비스의 마이크로버전 범위를 제한하려면 **min_microversion** 및 **max_microversion** 매개변수에 값을 할당합니다.

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

5장. TEMPEST 사용

이 섹션에서 명령을 실행하여 다양한 테스트 작업을 수행합니다. 단일 **tempest run** 명령에 여러 옵션을 결합할 수도 있습니다.

5.1. 사용 가능한 테스트 나열

--list-tests 또는 **-l** 옵션과 함께 **tempest-run** 명령을 실행하여 사용 가능한 템플릿 테스트 목록을 가져옵니다.

```
# tempest run -l
```

5.2. SMOKE 테스트 실행

연기 테스트는 가장 중요한 기능 만 포함하는 예비 테스트의 유형입니다. 포괄적이지 않지만 스모크 테스트를 실행하면 문제를 식별하는 경우 시간을 절약할 수 있습니다.

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

5.3. 화이트리스트 파일을 사용하여 테스트 전달

허용 목록 파일은 포함하려는 테스트를 선택하는 정규식이 포함된 파일입니다. 정규 표현식은 줄 바꿈으로 구분됩니다.

허용 목록 파일을 사용하려면 **--whitelist-file** 또는 **-w** 옵션과 함께 **tempest run** 명령을 실행합니다.

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

5.4. 블랙리스트 파일을 사용하여 테스트 건너뛰기

블랙리스트 파일은 제외하려는 테스트를 선택하는 정규식이 포함된 파일입니다. 정규 표현식은 줄 바꿈으로 구분됩니다.

블랙리스트 파일을 사용하려면 **--blacklist-file** 또는 **-b** 옵션을 사용하여 **tempest run** 명령을 실행합니다.

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

5.5. PARALLEL CONCURRENTLY 또는 SERIALLY에서 테스트 실행

테스트를 직렬로 실행합니다.

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

테스트를 병렬로 실행합니다. 병렬 테스트는 기본값입니다.

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

테스트를 병렬로 실행할 때 사용할 작업자 수를 지정하려면 **--concurrency** 또는 **-c** 옵션을 사용합니다.

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

기본적으로 **Integration Test Suite**는 사용 가능한 각 **CPU**에 하나의 작업자를 사용합니다.

5.6. 특정 테스트 실행

--regex 정규식 옵션을 사용하여 특정 테스트를 실행합니다. 정규 표현식은 **Python** 정규식이어야 합니다.

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

예를 들어 다음 예제 명령을 사용하여 **tempest.scenario** 로 시작하는 이름을 가진 모든 테스트를 실행합니다.

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

6장. 컨테이너화된 TEMPEST 실행

이 섹션에서는 언더클라우드의 컨테이너에서 **tempest**를 실행하는 방법에 대해 설명합니다. 오버클라우드 또는 언더클라우드에 대해 **tempest**를 실행할 수 있습니다. 컨테이너화된 **Tempest**에는 컨테이너화되지 않은 템포와 동일한 리소스가 필요합니다.

이 기능은 이번 릴리스에서 *기술 프리뷰*로 제공되므로 **Red Hat**에서 완전히 지원되지 않습니다. 테스트 용도로만 사용해야 하며 프로덕션 환경에 배포해서는 안 됩니다. 기술 프리뷰 기능에 대한 자세한 내용은 적용 범위 상세 정보를 참조하십시오.

6.1. TEMPEST 컨테이너 준비

tempest 컨테이너를 다운로드하고 구성하려면 다음 단계를 완료합니다.

1. **/home/stack** 디렉터리로 변경합니다.

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

2. **tempest** 컨테이너를 다운로드합니다.

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

이 컨테이너에는 모든 **Tempest** 플러그인이 포함되어 있습니다. 이 컨테이너에서 **Tempest** 테스트를 전역적으로 실행하면 플러그인에 대한 테스트가 포함됩니다. 예를 들어 **tempest run --regex '(*.)'** 명령을 실행하면 **tempest**가 모든 플러그인 테스트를 실행합니다. 배포에 모든 플러그인에 대한 구성이 포함되어 있지 않으면 이러한 **Tempest** 테스트가 실패합니다. **tempest list-plugins** 명령을 실행하여 설치된 모든 플러그인을 확인합니다. 테스트를 제외하려면 블랙리스트 파일에서 제외하려는 테스트를 포함해야 합니다. 자세한 내용은 5장. *Tempest 사용*의 내용을 참조하십시오.

3. 호스트 머신과 컨테이너 간의 데이터 교환에 사용할 디렉터리를 만듭니다.

```
$ mkdir container_tempest tempest_workspace
```

4. 필요한 파일을 **container_tempest** 디렉터리에 복사합니다. 이 디렉터리는 컨테이너의 파일 소스입니다.

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

5. 사용 가능한 **Docker** 이미지를 나열합니다.

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

6. 더 쉬운 명령 입력을 위해 별칭을 만듭니다. 디렉터리를 마운트할 때 절대 경로를 사용해야 합니다.

```
$ 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. 컨테이너에서 사용 가능한 **Tempest** 플러그인 목록을 가져오려면 다음 명령을 실행합니다.

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

6.2. 컨테이너 내부에서 컨테이너화된 TEMPEST 실행

1. 컨테이너 내에서 실행할 수 있는 **tempest** 스크립트를 생성하여 **tempest.conf** 파일을 생성하고 **tempest** 테스트를 실행합니다. 이 스크립트는 다음 작업을 수행합니다.

   - **set -e** 명령의 종료 상태를 설정합니다.
   - 오버클라우드에 대해 **tempest**를 실행하려면 **overcloudrc** 파일을 소싱합니다. 언더클라우드에 대해 **tempest**를 실행하려면 **stackrc** 파일을 소싱합니다.
   - **tempest init** 을 실행하여 **tempest Workspace**를 생성합니다. 호스트에서 파일도 액세스할 수 있도록 공유 디렉터리를 사용합니다.
   - 디렉토리를 **tempest_workspace**로 변경

- 이후 단계에서 쉽게 사용할 수 있도록 **TEMPESTCONF** 환경 변수를 내보냅니다.

- **discover-tempest-config** 를 실행하여 **tempest.conf** 파일을 생성합니다. **discover-tempest-config** 명령에 포함할 수 있는 옵션에 대한 자세한 내용은 **discover-tempest-config --help** 를 실행합니다.

- 호스트 시스템에서 **tempest.conf** 파일에 액세스할 수 있도록 **--out** 을 **home/stack/tempest_workspace/tempest.conf** 로 설정합니다.

- 공유 디렉터리에서 **tempest-deployer-input.conf** 파일을 가리키도록 **--deployer-input.conf**를 설정합니다.

- **Tempest** 테스트를 실행합니다. 이 예제 스크립트는 스모크 테스트 **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
```

이미 **tempest.conf** 파일이 있고 **tempest** 테스트만 실행하려면 스크립트에서 **TEMPESTCONF** 를 생략하고 이를 **container_tempest** 디렉터리의 **tempest.conf** 파일을 **tempest_workspace/etc** 디렉터리로 복사하십시오.

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

2. **tempest_script.sh** 스크립트에 실행 가능한 권한을 설정합니다.

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

3. 이전 단계에서 생성한 별칭을 사용하여 컨테이너에서 **tempest** 스크립트를 실행합니다.

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

4. 테스트 결과에 대한 정보는 **.stestr** 디렉토리를 검사합니다.

5. **tempest** 테스트를 다시 실행하려면 먼저 **tempest** 작업 영역을 제거하고 다시 생성해야 합니다.

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

6.3. 컨테이너 외부에서 컨테이너화된 TEMPEST 실행

컨테이너는 **tempest.conf** 파일을 생성하거나 검색하고 테스트를 실행합니다. 컨테이너 외부에서 다음 작업을 수행할 수 있습니다.

1. 오버클라우드에 대해 **tempest**를 실행하려면 **overcloudrc** 파일을 소싱합니다. 언더클라우드에 대해 **tempest**를 실행하려면 **stackrc** 파일을 소싱합니다.

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

2. **tempest init** 을 실행하여 **tempest Workspace**를 생성합니다. 호스트에서 파일도 액세스할 수 있도록 공유 디렉터리를 사용합니다.

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

3. **tempest.conf** 파일을 생성합니다.

```
# 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
```

discover-tempest-config 명령에 포함할 수 있는 옵션에 대한 자세한 내용은 **discover-tempest-config --help** 를 실행합니다.

4. **tempest** 테스트를 실행합니다. 예를 들어 이전 단계에서 생성한 별칭을 사용하여 **tempest** 스모크 테스트를 실행하려면 다음 명령을 실행합니다.

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

5. 테스트 결과에 대한 정보는 **.stestr** 디렉토리를 검사합니다.

6. **tempest** 테스트를 다시 실행하려면 먼저 **tempest** 작업 영역을 제거하고 다시 생성해야 합니다.

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

7장. 임시 리소스 정리

Tempest 를 실행한 후 파일, 테스트 프로세스에 생성된 사용자 및 프로젝트를 삭제해야 합니다. 자체 클리닝할 수 있다는 것은 템포의 디자인 원칙 중 하나입니다.

7.1. 정리 작업 수행

먼저 저장된 상태를 초기화해야 합니다. 이렇게 하면 **saved_state.json** 파일이 생성되어 정리에서 유지해야 하는 오브젝트를 삭제할 수 없습니다. 일반적으로 **tempest** 실행 전에 **--init-saved-state** 를 사용하여 정리를 실행합니다. 그렇지 않은 경우 **save_state.json** 을 편집하여 정리를 삭제하려는 개체를 제거해야 합니다.

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

정리를 실행합니다.

```
# tempest cleanup
```

tempest cleanup 명령은 **tempest** 리소스를 삭제하지만 프로젝트 또는 임시 관리자 계정은 삭제하지 않습니다.

7.2. DRY RUN 수행

실제 정리를 실행하기 전에 예행 연습을 수행하는 것이 좋습니다. 예행 연습에서 정리를 통해 삭제할 파일이 나열되지만 파일은 삭제하지 않습니다. 파일은 **dry_run.json** 파일에 나열됩니다. **dry_run.json** 파일을 확인하여 정리가 환경에 필요한 파일을 삭제하지 않는지 확인합니다.

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

7.3. TEMPEST 오브젝트 삭제

다음 명령을 실행하여 **tempest** 관리자 계정이 아닌 프로젝트를 포함하여 모든 **tempest** 리소스를 삭제합니다.

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