Red Hat OpenStack Platform 11

OpenStack Integration Test Suite ガイド

OpenStack Integration Test Suite の概要

Last Updated: 2018-03-01

Red Hat OpenStack Platform 11 OpenStack Integration Test Suite ガイド

OpenStack Integration Test Suite の概要

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本ガイドは、Red Hat OpenStack Platform 環境での OpenStack Integration Test Suite のインストール、設定、管理について説明します。

目次

前書き

本ガイドは、Red Hat OpenStack Platform 環境での OpenStack Integration Test Suite のインストール、設定、管理について説明します。

第1章 はじめに

OpenStack は多数の異なるプロジェクトで構成されているので、それらの相互運用性を OpenStack クラスター内でテストすることが重要となります。OpenStack Integration Test Suite (tempest) は、Red Hat OpenStack Platform デプロイメントの統合テストを自動化します。テストを実行することにより、クラスターが想定通りに機能することを確認できます。また、特にアップグレード後に、潜在的な問題の早期警告を提供することができます。

Integration Test Suite には、OpenStack API の検証およびシナリオテストと、自己検証向けの単体テスト用のテストが含まれています。Integration Test Suite は、ostestr をテストランナーとし、OpenStack パブリック API を使用してブラックボックステストを実行します。

第2章 OPENSTACK INTEGRATION TEST SUITE のテスト

OpenStack Integration Test Suite には多数のアプリケーションが含まれています。これは、OpenStack のコアプロジェクトに対するコミットのためのゲートとして機能し、ストレステストを実行してクラウドデプロイメント上に負荷を生成したり、CLI テストを実行してコマンドラインの応答フォーマットをチェックしたりすることができます。ただし、本書で取り上げるのは **`scenario tests`** と **`API tests`** の機能です。これらのテストはお使いの OpenStack クラウドデプロイメントに対して実行されます。以下の項には、各テストについての簡単な説明と実装方法を記載します。

2.1. シナリオテスト

シナリオテストは、標準的なエンドユーザーのアクションワークフローをシミュレーションして、サービス間の統合ポイントをテストします。テストフレームワークは、設定、テスト、サービス間の統合を実行した後に解体されます。テストは、関連するサービスでタグ付けして、テストが使用するクライアントライブラリーを明確化すべきです。

シナリオはユースケースに基づいています。以下に例を示します。

- Image サービスにイメージをアップロードします。
- イメージからインスタンスをデプロイします。
- インスタンスにボリュームを接続します。
- インスタンスのスナップショットを作成します。
- インスタンスからボリュームを切断します。

2.2. API テスト

API テストは、OpenStack API を検証します。このテストは、OpenStack API の OpenStack Integration Test Suite 実装を使用します。有効な JSON と無効な JSON の両方を使用してエラーの応答が有効であることを確認することができます。テストは個別に実行することが可能で、前回のテストで残された状態には依存しません。

第3章 OPENSTACK INTEGRATION TEST SUITE のインストール

本項では、director または手動のインストール手順を使用した OpenStack Integration Test Suite のインストール方法について説明します。

3.1. DIRECTOR を使用した方法

stack ユーザーのホームディレクトリーにある **undercloud.conf** ファイルを編集します。デフォルトでは、**enable_tempest** は **false** に設定されています。この値を **true** に変更します。

```
enable_tempest = true
```

これで、「OpenStack Integration Test Suite パッケージのインストール」に記載の **tempest** パッケージとプラグインをインストールする準備が整いました。

3.2. 手動でのインストールの準備

OpenStack Integration Test Suite を実行するには、まず必要なパッケージをインストールして、Integration Test Suite に対してさまざまな OpenStack サービスやその他の動作スイッチの場所を示す設定ファイルを作成します。

コントローラーノードで、**root** ユーザーとして **tempest** という名前の仮想マシンを作成します。このマシンは、Red Hat Enterprise Linux 7.3 以上のバージョンを実行している必要があります。また、クラウドにアクセス可能でなければなりませんが、必ずしもクラウドの一部である必要はありません。詳しい情報は、「Virt-Manager を使用したゲストの作成」を参照してください。

また、OpenStack Integration Test Suite をインストールする前には、Red Hat OpenStack Platform 環境内に次のネットワークが設定されている必要があります。

- Floating IP を提供することができる外部ネットワーク
- プライベートネットワーク

これらのネットワークは、1 つのルーターを介して接続されている必要があります。

プライベートネットワークを作成します。

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

パブリックネットワークを作成します。

```
$ 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>_
```

これで、OpenStack Integration Test Suite を **tempest** 仮想マシン内にインストールおよび設定する準

備が整いました。詳しい情報は、「OpenStack Integration Test Suite パッケージのインストール」を参照してください。

3.3. OPENSTACK INTEGRATION TEST SUITE パッケージのインストール

1. OpenStack Integration Test Suite に関連するパッケージをインストールします。

```
# yum -y install openstack-tempest
```

ただし、このコマンドでは、tempest プラグインは一切インストールされません。tempest プラグインは、お使いの OpenStack インストール環境に応じて手動でインストールする必要があります。

2. マシンにインストール済みの全 OpenStack コンポーネントを確認します。

```
# openstack-status
```

3. 各コンポーネントに適切な tempest プラグインをインストールします。以下に例を示します。

```
# yum install python-glance-tests python-keystone-tests python-
horizon-tests-tempest python-neutron-tests python-cinder-tests
python-nova-tests python-swift-tests python-ceilometer-tests python-
gnocchi-tests python-aodh-tests
```

各 OpenStack コンポーネント向けの tempest プラグインの一覧は、「tempest プラグインパッケージの一覧」を参照してください。

3.3.1. tempest プラグインパッケージの一覧

コンポーネント	パッケージ名
aodh	python-aodh-tests
ceilometer	python-ceilometer-tests
cinder	python-cinder-tests
designate	python-designate-tests-tempest
glance	python-glance-tests
gnocchi	python-gnocchi-tests
heat	python-heat-tests
horizon	python-horizon-tests-tempest
ironic	python-ironic-tests

コンポーネント	パッケージ名
ironic-inspector	python-ironic-inspector-tests
keystone	python-keystone-tests
magnum	python-magnum-tests
manila	python-manila-tests
mistral	python-mistral-tests
murano	python-murano-tests
neutron	python-neutron-tests
neutron-fwaas	python-neutron-fwaas-tests
neutron-lbaas	python-neutron-lbaas-tests
neutron-vpnaas	python-neutron-vpnaas-tests
nova	python-nova-tests
sahara	python-sahara-tests-tempest
swift	python-swift-tests
trove	python-trove-tests
watcher	python-watcher-tests-tempest
zaqar	python-zaqar-tests

第4章 OPENSTACK INTEGRATION TEST SUITE の設定

4.1. ワークスペースの作成

1. 管理者の認証情報を読み込みます。
   アンダークラウドでは、以下のコマンドを実行します。

   ```
   # source stackrc
   ```

   また、オーバークラウドでは、以下のコマンドを実行します。

   ```
   # source overcloudrc
   ```

2. **tempest** を初期化します。

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

   これで、**mytempest** という名前の tempest ワークスペースが作成されます。

   既存のワークスペースの一覧を表示することができます。

   ```
   # tempest workspace list
   ```

3. **etc/tempest.conf** ファイルを生成します。

   ```
   # discover-tempest-config --deployer-input ~/tempest-deployer-
   input.conf \
   --debug --create identity.uri $OS_AUTH_URL identity.admin_password \
   $OS_PASSWORD --network-id _<uuid>_
   ```

   uuid は、外部ネットワークの UUID です。

   discover-tempest-config は、以前は **config_tempest.py** という名前で、同じパラメーターを取ります。これは、**openstack-tempest** の依存関係としてインストールされる **python-tempestconf** によって提供されます。

4.2. TEMPEST の設定の確認

現在の tempest の設定を確認します。

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

output は、更新した設定を書き込むための出力ファイルです。これは、元の設定ファイルとは異なります。

4.3. ロギング設定の変更

ログファイルのデフォルトの場所は、tempest ワークプレース内の **logs** ディレクトリーです。

このディレクトリーを変更するには、**tempest.conf** の **[DEFAULT]** セクション下で、**log_dir** を必要なディレクトリーに設定します。

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

独自のログ設定ファイルがある場合には、**tempest.conf** の **[DEFAULT]** セクション下で **log_config_append** をそのファイルに指定します。

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

このパラメーターが設定されている場合には、**log_dir** を含む、**tempest.conf** 内のその他のログ設定はすべて無視されます。

4.4. マイクロバージョンテストの設定

OpenStack Integration Test Suite は、API マイクロバージョンをテストする安定性のあるインターフェースを提供します。以下のセクションは、これらのインターフェースを使用してマイクロバージョンテストの実装方法を説明します。

まず最初に、**tempest.conf** 設定ファイルのオプションを設定して、ターゲットのマイクロバージョンを指定する必要があります。これは、サポートされているマイクロバージョンが OpenStack 使用されているマイクロバージョンと必ず一致するようにするためです。ターゲットバージョンの範囲を指定すると、Integration Test Suite の 1 回の操作で複数のマイクロバージョンテストを実行することが可能です。

たとえば、 **compute** サービスのマイクロバージョンの範囲を限定するには、設定ファイルの **[compute]** セクションで **min_microversion** および **max_microversion** パラメーターの値を割り当てます。

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

第5章 OSTESTR を使用した TEMPEST の実行

ostestr は、**testr** テストランナーの OpenStack ラッパーです。

5.1. スモークテストの実行

スモークテストとは、最も重要な機能のみを対象とする事前テストの一種です。スモークテストは包括的ではありませんが、実行して問題が検出された場合には、時間を節約できます。

スモークテストを実行するには、以下のコマンドを実行します。

```
# ostestr '.*smoke'
```

5.2. ホワイトリストファイルを使用したテストの指定

ホワイトリストファイルは、含めるテストを選択するための正規表現が記載されたファイルです。正規表現は、改行で区切ります。

```
# ostestr --whitelist-file _<whitelist_file>_
```

または

```
# ostestr -w _<whitelist_file>_
```

5.3. ブラックリストファイルを使用したテストの省略

ブラックリストファイルは、除外するテストを選択するための正規表現が記載されたファイルです。正規表現は、改行で区切ります。

ブラックリストファイルを使用するには、以下のコマンドを実行します。

```
# ostestr --blacklist-file _<blacklist_file>_
```

または

```
# ostestr -b _<blacklist_file>_
```

5.4. テストの並行同時または順次の実行

テストを順次に実行するには、以下のコマンドを実行します。

```
# ostestr --serial
```

複数のテストを同時に実行するには (これがデフォルトです)、以下のコマンドを実行します。

```
# ostestr --parallel
```

テストを並行して実行する場合に使用するワーカー数を指定します。

```
# ostestr --concurrency _<workers>_
```

-

または

```
# ostestr -c _<workers>_
```

デフォルトでは、この値は CPU 数に設定されます。

第6章 TEMPEST リソースのクリーニング

tempest の実行後には、テストプロセス中にユーザーとテナントによって作成されたファイルが残るので、削除する必要があります。セルフクリーニングが可能であることは、**tempest** の設計原則の 1 つです。

6.1. クリーンアップの実行

最初に、保存されている状態を初期化する必要があります。これにより、 **saved_state.json** が作成され、保持する必要のあるオブジェクトが cleanup によって削除されるのを防ぎます。tempest の実行前には通常、cleanup に **--init-saved-state** オプションを指定して実行します。そうでない場合には、**saved_state.json** を編集して、cleanup で消去したいオブジェクトを削除します。

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

cleanup を実行します。

```
# tempest cleanup
```

6.2. ドライランの実行

ドライランは、クリーンアップによって削除されるファイルを一覧表示しますが、ファイルは一切削除しません。ファイルは、**dry_run.json** ファイル内に一覧表示されます。

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

6.3. TEMPEST オブジェクトの削除

tempest により作成されたユーザーとテナントを削除します。

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