Red Hat OpenStack Platform 17.1

Red Hat OpenStack Platform Integration Test Suite を使用したクラウドの検証

Red Hat OpenStack Platform デプロイメントの検証

Last Updated: 2024-08-12

Red Hat OpenStack Platform 17.1 Red Hat OpenStack Platform Integration Test Suite を使用したクラウドの検証

Red Hat OpenStack Platform デプロイメントの検証

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

デプロイメントを検証できるように、Red Hat OpenStack Platform 環境に OpenStack Integration Test Suite (tempest) をインストールし、設定して管理します。

目次

多様性を受け入れるオープンソースの強化 3

RED HAT ドキュメントへのフィードバック (英語のみ) 4

第1章 OPENSTACK INTEGRATION TEST SUITE (TEMPEST) の検証 5

第2章 INTEGRATION TEST SUITE (TEMPEST) のインストール 6
2.1. 前提条件 6
2.2. INTEGRATION TEST SUITE の手動インストール 6

第3章 INTEGRATION TEST SUITE (TEMPEST) の設定 9
3.1. 前提条件 9
3.2. ワークスペースの作成 9
3.3. INTEGRATION TEST SUITE の手動設定 10
3.4. INTEGRATION TEST SUITE ロギングの設定 11
3.5. INTEGRATION TEST SUITE マイクロバージョンテストの設定 12

第4章 INTEGRATION TEST SUITE (TEMPEST) リソースのクリーンアップ 13
4.1. ドライランの実行 13
4.2. テンペストクリーンアップの実行 13

第5章 INTEGRATION TEST SUITE (TEMPEST) を使用した OPENSTACK クラウドの検証 15
5.1. 前提条件 15
5.2. 利用可能なテストのリスト表示 15
5.3. SMOKE テストの実行 15
5.4. 許可リストファイルを使用したテストのパス 15
5.5. ブロックリストファイルを使用したテストのスキップ 15
5.6. 並行または連続してテストの実行 16
5.7. 特定のテストの実行 16
5.8. INTEGRATION TEST SUITE オブジェクトの削除 16

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

Jira でドキュメントのフィードバックを提供する

ドキュメントに関するフィードバックを提供するには、Create Issue フォームを使用します。Red Hat OpenStack Platform Jira プロジェクトで Jira Issue が作成され、フィードバックの進行状況を追跡できます。

1. Jira にログインしていることを確認してください。Jira アカウントをお持ちでない場合は、アカウントを作成してフィードバックを送信してください。
2. Create Issue をクリックして、**Create Issue** ページを開きます。
3. **Summary** フィールドと **Description** フィールドに入力します。**Description** フィールドに、ドキュメントの URL、章またはセクション番号、および問題の詳しい説明を入力します。フォーム内の他のフィールドは変更しないでください。
4. **Create** をクリックします。

第1章 OPENSTACK INTEGRATION TEST SUITE (TEMPEST) の検証

Red Hat OpenStack Platform (RHOSP) は多くの異なるプロジェクトで構成されるため、RHOSP クラスター内のプロジェクトの相互運用性をテストすることが重要です。OpenStack Integration Test Suite は、RHOSP デプロイメントの統合テストを自動化します。テストを実行して、クラスターが想定どおりに機能することを確認できます。テスト出力で、特にアップグレード後の潜在的な問題を早期に警告します。

Integration Test Suite には、OpenStack API 検証とシナリオテストのテスト、および自己検証のユニットテストが含まれています。Integration Test Suite は、OpenStack パブリック API を使用し、テストランナーとして tempest を使用してブラックボックステストを実行します。

OpenStack Integration Test Suite (tempest) は、Red Hat OpenStack Platform (RHOSP) コアプロジェクトへのコミットのゲートとして動作し、クラウドデプロイメントの負荷を生成するためにストレステストを行い、CLI テストを実行してコマンドラインの応答形式を確認できます。RHOSP クラウドデプロイメントに対して、**scenario tests** および **API tests** を実行できます。

シナリオテスト

シナリオテストは、サービス間の統合ポイントをテストする一般的なエンドユーザーアクションワークフローをシミュレートします。テストフレームワークは、設定を実施し、サービス間の統合をテストしてから、自動的に削除されます。テストに関連するサービスでテストにタグを付け、テストが使用するクライアントライブラリーを明確にします。

次のシナリオは、ユースケースに基づいています。

- Image サービスへのイメージのアップロード
- イメージからのインスタンスのデプロイ
- インスタンスへのボリュームの接続
- インスタンスのスナップショットの作成
- インスタンスからのボリュームの切断

API テスト

API テストは、OpenStack API を検証します。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 の両方を使用すると、エラーの応答が有効であることを確認できます。テストを個別に実行し、以前のテスト状態に依存する必要はありません。

第2章 INTEGRATION TEST SUITE (TEMPEST) のインストール

Integration Test Suite を手動でインストールする場合は、Integration Test Suite の手動インストール を参照してください。

2.1. 前提条件

- アンダークラウドのインストール。詳細は、アンダークラウドのインストール を参照してください。
- オーバークラウドのデプロイメント。詳細は、CLI ツールを使用した基本的なオーバークラウドの作成 を参照してください。

2.2. INTEGRATION TEST SUITE の手動インストール

director を使用して Integration Test Suite (tempest) を自動的にインストールしない場合は、後で手動でインストールを行うことができます。基本的なネットワーク設定を定義し、Integration Test Suite パッケージをインストールし、OpenStack サービスおよびその他のテスト動作スイッチの詳細が含まれる設定ファイルを作成していることを確認する必要があります。

手順

1. 以下のネットワークが Red Hat OpenStack Platform (RHOSP) 環境内で利用可能であることを確認します。
   - Floating IP を提供できる外部ネットワーク
   - プライベートネットワーク
     ルーターを使用してこれらのネットワークに接続します。
     a. プライベートネットワークを作成するには、ネットワークデプロイメントに応じて以下のオプションを指定します。

```
$ 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. パブリックネットワークを作成するには、ネットワークデプロイメントに従って以下のオプションを指定します。

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

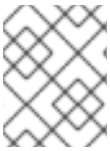
```
$ sudo dnf -y install openstack-tempest
```

このコマンドでは、tempest プラグインはインストールされません。RHOSP のインストールに応じて、プラグインを手動でインストールする必要があります。

3. 環境内の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。たとえば、keystone、neutron、cinder、および telemetry プラグインをインストールするには、以下のコマンドを入力します。

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

パッケージの全リストは、Integration Test Suite のパッケージ を参照してください。



注記

openstack-tempest-all パッケージをインストールすることもできます。このパッケージには、tempest プラグインがすべて含まれます。

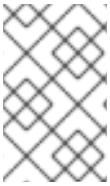
2.2.1. Integration Test Suite のパッケージ

dnf search を使用して、tempest テストパッケージのリストを取得します。

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

コンポーネント	パッケージ名
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

コンポーネント	パッケージ名
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



注記

python3-telemetry-tests-tempest パッケージには、aodh、panko、gnocchi、および ceilometer テスト用のプラグインが含まれます。**python3-ironic-tests-tempest** パッケージには、ironic および ironic-inspector のプラグインが含まれます。

第3章 INTEGRATION TEST SUITE (TEMPEST) の設定

Integration Test Suite で環境の検証を開始する前に、ワークスペースを作成して **/etc/tempest.conf** 設定ファイルを生成する必要があります。

3.1. 前提条件

- Integration Test Suite のパッケージが含まれる OpenStack 環境。

3.2. ワークスペースの作成

Integration Test Suite (tempest) 設定および出力用にワークスペースを作成します。

手順

1. ターゲットデプロイメントの認証情報を読み込みます。
   - ターゲットがアンダークラウドにある場合は、source コマンドでアンダークラウドの認証情報を読み込みます。

     ```
     # source stackrc
     ```

   - ターゲットがオーバークラウドにある場合、source コマンドでオーバークラウドの認証情報を読み込みます。

     ```
     # source overcloudrc
     ```

2. **tempest** を初期化します。

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

   このコマンドは、**mytempest** という名前の tempest ワークスペースを作成します。

3. オプション: 以下のコマンドを入力して、既存のワークスペースのリストを表示します。

   ```
   # tempest workspace list
   ```

4. **etc/tempest.conf** ファイルを生成します。

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

   UUID を外部ネットワークの UUID に置き換えます。

   discover-tempest-config は、以前は **config_tempest.py** と呼ばれ、同じパラメーターを使用します。**python-tempestconf** は **openstack-tempest** の依存関係として、**discover-tempest-config** を提供しています。

注記

アンダークラウドの **etc/tempest.conf** ファイルを生成するには、**tempest-deployer-input.conf** ファイルのリージョン名がアンダークラウドデプロイメントの名前と同じであることを確認します。これらの名前が一致しない場合は、**tempest-deployer-input.conf** ファイルのリージョン名を更新して、アンダークラウドのリージョン名と一致するように更新します。

5. アンダークラウドのリージョン名を検証するには、以下のコマンドを入力します。

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

6. オーバークラウドのリージョン名を検証するには、以下のコマンドを入力します。

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

お使いの環境に応じて、デフォルトの **tempest.conf** ファイルを変更する必要がある場合があります。詳しくは、拡張リストの設定 および heat_plugin の設定 を参照してください。

検証

- 現在の tempest 設定を検証します。

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

output の値は、Integration Test Suite が更新された設定を書き込む出力ファイルです。これは、元の設定ファイルとは異なります。

3.3. INTEGRATION TEST SUITE の手動設定

discover-tempest-config コマンドは、**tempest.conf** ファイルを自動的に生成します。ただし、**tempest.conf** ファイルが環境の設定に対応していることを確認する必要があります。

3.3.1. Integration Test Suite 拡張リストの手動設定

デフォルトの **tempest.conf** ファイルには、各コンポーネントの拡張リストが含まれます。**tempest.conf** ファイルの各コンポーネントの **api_extensions** 属性を検査し、拡張機能のリストがデプロイメントに対応することを確認します。

デプロイメントで利用可能な拡張機能が **tempest.conf** ファイルの **api_extensions** 属性の拡張機能のリストと一致しない場合、コンポーネントは tempest テストに失敗します。この失敗を回避するには、デプロイメントで利用可能な拡張機能を特定し、**api_extensions** パラメーターに含める必要があります。デプロイメント内の Network、Compute、Volume、または Identity 拡張機能のリストを取得するには、以下のコマンドを実行します。

手順

- デプロイメント内の Network、Compute、Volume、または Identity 拡張機能のリストを取得するには、以下のコマンドを入力します。

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

3.3.2. heat_plugin の手動設定

tempest.conf ファイルで、**heat_plugin** を手動で設定できます。

手順

- 以下の例を使用して、デプロイメントに応じて **heat_plugin** を設定します。

```
[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
```

openstack network list コマンドを使用して、**fixed_network_name**、**network_for_ssh**、および **floating_network_name** のネットワークを特定します。

注記

tempest.conf ファイルの **[service_available]** セクションで **heat** を **True** に設定する必要があります。また、**[heat_plugin]** セクションの **username** 属性にあるユーザーは、**member** である必要があります。たとえば、以下のコマンドを入力して **member** ロールを **demo** ユーザーに追加します。

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

3.4. INTEGRATION TEST SUITE ロギングの設定

tempest ワークスペース内の **logs** ディレクトリーのログファイルのデフォルトの場所を変更することができます。

手順

1. **tempest.conf** の **[DEFAULT]** セクションで、**log_dir** を目的のディレクトリーに設定します。

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

2. **tempest.conf** に独自のロギング設定ファイルを使用している場合は、使用しているファイルの **[DEFAULT]** セクションの下に **log_config_append** を設定します。

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

log_config_append 属性を設定すると、Integration Test Suite は **log_dir** 属性を含む **tempest.conf** の他のすべてのロギング設定を無視します。

3.5. INTEGRATION TEST SUITE マイクロバージョンテストの設定

Integration Test Suite (tempest) は、API マイクロバージョンをテストする安定したインターフェイスを提供します。これらのインターフェイスを使用してマイクロバージョンテストを実装するには、次の手順を実行します。

手順

1. **tempest.conf** 設定ファイルでオプションを設定し、ターゲットマイクロバージョンを指定します。これらのオプションを設定して、サポートされているマイクロバージョンが OpenStack クラウド内のマイクロバージョンに対応するようにします。

2. 単一の Integration Test Suite 操作で複数のマイクロバージョンテストを実行するターゲットマイクロバージョンの範囲を指定できます。
   たとえば、設定ファイルの **[compute]** セクションで、**compute** サービスのマイクロバージョンの範囲を制限するには、**min_microversion** および **max_microversion** パラメーターに値を割り当てます。

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

第4章 INTEGRATION TEST SUITE (TEMPEST) リソースのクリーンアップ

OpenStack Integration Test Suite (tempest) を使用してデプロイメントを検証する前に、**--init-saved-state** フラグを指定して **cleanup** コマンドを実行します。このコマンドは、環境をスキャンしてリソース (ネットワーク、ボリューム、イメージ、フレーバー、プロジェクト、ユーザーなど) を検出します。検出されたリソースは、**saved_state.json** というファイルに保存されます。**tempest cleanup** コマンドが実行すると、**saved_state.json** ファイルに記録されていないすべてのリソースが削除されます。

前提条件

- Integration Test Suite のパッケージが含まれる OpenStack 環境。
- OpenStack 環境に対応する Integration Test Suite 設定。詳細は、ワークスペースの作成 参照してください。
- 1 つ以上の完了した Integration Test Suite 検証テスト。

4.1. ドライランの実行

クリーンアップを実行する前にドライランを実行します。ドライランは、Integration Test Suite が実際にファイルを削除せずに、クリーンアップによって削除されるファイルをリスト表示します。**dry_run.json** ファイルには、クリーンアップによって削除されるファイルのリストが含まれます。

手順

1. ドライランを完了します。

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

2. **dry_run.json** ファイルをチェックして、クリーンアップにより環境に必要なファイルが削除されないようにします。

4.2. テンペストクリーンアップの実行

tempest テストを実行する前に、保存された状態を初期化する必要があります。これにより **saved_state.json** ファイルが作成され、保持する必要があるオブジェクトがクリーンアップにより削除されるのを防ぎます。

警告

--init-saved-state フラグを指定して **cleanup** コマンドを実行しない場合は、RHOSP オブジェクトが削除されます。

--init-saved-state を指定して **cleanup** コマンドを実行した後にオブジェクトを作成する場合、このオブジェクトは後続の **tempest** コマンドで削除できます。

手順

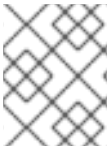
1. 保存された状態を初期化し、**saved_state.json** ファイルを作成します。

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

2. クリーンアップを実行します。

```
# tempest cleanup
```

tempest cleanup コマンドは tempest リソースを削除しますが、プロジェクトや tempest の管理者アカウントは削除しません。



注記

saved_state.json ファイルを修正して、保持または削除するオブジェクトの指定や除外を行うことができます。

第5章 INTEGRATION TEST SUITE (TEMPEST) を使用した OPENSTACK クラウドの検証

Integration Test Suite の検証は、**tempest run** コマンドで数多くの方法で実行することができます。1 つの **tempest run** コマンドで、複数のオプションを組み合わせることもできます。

5.1. 前提条件

- Integration Test Suite のパッケージが含まれる OpenStack 環境。
- OpenStack 環境に対応する Integration Test Suite 設定。詳細は、ワークスペースの作成 参照してください。

5.2. 利用可能なテストのリスト表示

--list-tests オプションを使用して、利用可能なすべてのテストをリスト表示します。

手順

- **--list-tests** または **-l** オプションのいずれかを指定して **tempest-run** コマンドを入力し、利用可能な tempest テストのリストを取得します。

```
# tempest run -l
```

5.3. SMOKE テストの実行

smoke テストは、最も重要な機能のみを対象とした予備的なテストの種類です。このテストは包括的ではありませんが、smoke テストの実行で問題が特定できれば時間を節約できます。

手順

- **--smoke** オプションを指定して tempest **run** コマンドを入力します。

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

5.4. 許可リストファイルを使用したテストのパス

許可リストファイルは、追加するテストを選択する正規表現が含まれるファイルです。1 つ以上の正規表現を使用する場合は、各行に各式を指定します。

手順

- **--whitelist-file** または **-w** オプションのいずれかを指定して **tempest run** コマンドを入力し、許可リストファイルを使用します。

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

5.5. ブロックリストファイルを使用したテストのスキップ

ブロックリストファイルは、除外するテストを選択する正規表現が含まれるファイルです。1 つ以上の正規表現を使用する場合は、各行に各式を指定します。

手順

- **--blacklist-file** または **-b** オプションのいずれかを指定して **tempest run** コマンドを入力し、ブラックリストファイルを使用します。

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

5.6. 並行または連続してテストの実行

テストは並行して実行することも、連続して実行することができます。また、並列テストを実行する際に使用するワーカー数を定義することもできます。デフォルトでは、Integration Test Suite は利用可能な CPU ごとに 1 つのワーカーを使用します。

テストを順次実行するか、並行して実行することを選択します。

- テストを順次実行します。

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

- テストを並行して実行します (デフォルト)。

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

- **--concurrency** または **-c** オプションを使用して、テストを並行して実行する時に使用するワーカーの数を指定します。

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

5.7. 特定のテストの実行

--regex オプションを使用して特定のテストを実行します。正規表現は Python 正規表現を使用する必要があります。

手順

- 以下のコマンドを入力します。

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

- たとえば、以下の例に示すコマンドを使用して、**tempest.scenario** で始まる名前のテストをすべて実行します。

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

5.8. INTEGRATION TEST SUITE オブジェクトの削除

tempest cleanup コマンドを入力し、すべての Integration Test Suite (tempest) リソースを削除します。このコマンドではプロジェクトも削除されますが、管理者アカウントは削除されません。

手順

- tempest リソースを削除します。

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