python-tempestconf Documentation

Release 3.2.2.dev8

Red Hat, Inc.

Nov 18, 2021

CONTENTS

python-tempestconf is a tool for automatic generation of tempest configuration based on users cloud.

CHAPTER

ONE

CONTENT:

Note: In Ocata release new features were presented. `discover-tempest-config` is the new name of the **old** `config_tempest.py` script and it **accepts the same parameters.** More about the new features can be found here

1.1 Overview

python-tempestconf will automatically generate the tempest configuration based on your cloud.

- Free software: Apache license
- Documentation: https://docs.openstack.org/python-tempestconf/latest/
- Source: https://opendev.org/openinfra/python-tempestconf
- Bugs: https://storyboard.openstack.org/#!/project/912
- Release notes: https://docs.openstack.org/releasenotes/python-tempestconf

1.2 Install Guide

1.2.1 Installation

Git

1. Clone and change to the directory:

```
$ git clone https://opendev.org/openinfra/python-tempestconf
$ cd python-tempestconf
```

2. Create a virtual environment using **`virtualenv`**:

```
$ virtualenv .venv
$ source .venv/bin/activate
```

3. Install requirements in the newly created virtual environment:

```
(.venv) $ pip install .
```

4. *(optional)* Instead of manual installation described in steps 2 and 3 above, tox can be used for installing the requirements as well. To create python 3.6 environment run following:

```
$ tox -epy36
$ source .tox/py36/bin/activate
```

RPM Installation (RDO)

- `python-tempestconf` package can be installed as follows:

```
$ sudo yum install python-tempestconf
```

- `python-tempestconf` is installed together with `openstack-tempest`, as a new dependency (starting **from** the **Ocata** release):

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

Pip installation

Install `python-tempestconf` via pip as follows:

```
$ pip install python-tempestconf
```

1.3 CLI Documentation

1.3.1 CLI Options

discover-tempest-config

```
usage: discover-tempest-config [-h] [--os-cloud <name>]
                               [--os-auth-type <name>]
                               [--os-auth-url OS_AUTH_URL]
                               [--os-system-scope OS_SYSTEM_SCOPE]
                               [--os-domain-id OS_DOMAIN_ID]
                               [--os-domain-name OS_DOMAIN_NAME]
                               [--os-project-id OS_PROJECT_ID]
                               [--os-project-name OS_PROJECT_NAME]
                               [--os-project-domain-id OS_PROJECT_DOMAIN_ID]
                               [--os-project-domain-name OS_PROJECT_DOMAIN_
↪NAME]
                               [--os-trust-id OS_TRUST_ID]
                               [--os-default-domain-id OS_DEFAULT_DOMAIN_ID]
                               [--os-default-domain-name OS_DEFAULT_DOMAIN_
↪NAME]
```

(continues on next page)

(continued from previous page)

```
                              [--os-user-id OS_USER_ID]
                              [--os-username OS_USERNAME]
                              [--os-user-domain-id OS_USER_DOMAIN_ID]
                              [--os-user-domain-name OS_USER_DOMAIN_NAME]
                              [--os-password OS_PASSWORD] [--insecure]
                              [--os-cacert <ca-certificate>]
                              [--os-cert <certificate>] [--os-key <key>]
                              [--timeout <seconds>] [--collect-timing]
                              [--os-service-type <name>]
                              [--os-service-name <name>]
                              [--os-interface <name>]
                              [--os-region-name <name>]
                              [--os-endpoint-override <name>]
                              [--os-api-version <name>] [--create]
                              [--out OUT] [--deployer-input DEPLOYER_INPUT]
                              [--no-default-deployer] [--debug] [--verbose]
                              [--no-rng] [--non-admin] [--test-accounts PATH]
                              [--create-accounts-file PATH] [--profile PATH]
                              [--generate-profile PATH]
                              [--image-disk-format IMAGE_DISK_FORMAT]
                              [--image IMAGE] [--retry-image]
                              [--flavor-min-mem FLAVOR_MIN_MEM]
                              [--flavor-min-disk FLAVOR_MIN_DISK]
                              [--convert-to-raw] [--network-id NETWORK_ID]
                              [--append SECTION.KEY=VALUE[,VALUE]]
                              [--remove SECTION.KEY=VALUE[,VALUE]]
                              [overrides [overrides ...]]
```

Positional Arguments

overrides

Override options Key value pairs used to hardcode values in *tempest.conf*. The key is a section.key where section is a section header in the conf file. For example:

$ discover-tempest-config identity.username myname identity.password mypass

Default: []

Named Arguments

--os-cloud Named cloud to connect to

--os-auth-type, --os-auth-plugin Authentication type to use

Default: password

--create

Create Tempest resources Make *python-tempestconf* to create Tempest resources such as flavors needed for running Tempest tests.

Default: False

--out
: **Output file** A name of the file where the discovered Tempest configuration will be written to.

  Default: etc/tempest.conf

--deployer-input
: **Path to deployer file** A file in the format of tempest.conf that will override the default values. It is usually created by an installer and contains environment specific options.

  The deployer-input file is an alternative to providing key/value pairs. If there are also key/value pairs they will be applied after the deployer-input file.

  If the option is **not defined** and **no-default-deployer** is **not used**, python-tempestconf **will try** to look for the file in *$HOME/tempest-deployer-input.conf* location.

--no-default-deployer
: **Do not check for the default deployer input in** *$HOME/tempest-deployer-input.conf*

  Default: False

--debug
: Print debugging information.

  Default: False

--verbose, -v
: Print more information about the execution.

  Default: False

--no-rng
: **Create new flavors and upload images without** random number generator device.

  Default: False

--non-admin
: **Simulate non-admin credentials.** When True, the credentials are used as non-admin ones. No resources are created.

  Default: False

--test-accounts
: **Tempest accounts.yaml file** Defines a path to a Tempest accounts.yaml file. For example:

  test-accounts $HOME/tempest/accounts.yaml

--create-accounts-file
: **Generate Tempest accounts file** Minimal accounts file will be created in the specified path. For example:

  create-accounts-file $HOME/accounts.yaml

--profile
: **python-tempestconfs profile.yaml file** A file which contains definition of python-tempestconfs arguments. NOTE: If this argument is used, other arguments cannot be defined!

--generate-profile
: **Generate a sample profile.yaml file.** A sample profile.yaml will be generated in the specified path. After that python-tempestconf ends. For example:

  generate-profile $HOME/profile.yaml

--image-disk-format
: **A format of an image to be uploaded to glance.** Default is qcow2

Default: qcow2

--image **An image name/path/url to be uploaded to** glance if its not already there. The name of the image is the leaf name of the path. Default is https://download.cirros-cloud.net/0.5.2/cirros-0.5.2-x86_64-disk.img

Default: https://download.cirros-cloud.net/0.5.2/cirros-0.5.2-x86_64-disk.img

--retry-image **Allow tempestconf to retry download an image,** in case of failure, from these urls: [https://download.cirros-cloud.net/0.5.2/cirros-0.5.2-x86_64-disk.img, http://images.rdoproject.org/cirros/cirros-0.5.2-x86_64-disk.img]

Default: False

--flavor-min-mem **Specify minimum memory for new** flavours, default is 128.

Default: 128

--flavor-min-disk **Specify minimum disk size for new** flavours, default is 1.

Default: 1

--convert-to-raw **Convert images to raw format before uploading** to glance.

Default: False

--network-id **Specify which network with external connectivity** should be used by the tests.

--append **Append values to tempest.conf** Key value pair to be appended to the configuration file. NOTE: Multiple values are supposed to be divided by a COLON only, WITHOUT spaces. For example:

$ discover-tempest-config append features.ext=tag[,tag-ext] append section.ext=ext[,another-ext]

Default: []

--remove **Remove values from tempest.conf** Key value pair to be removed from the configuration file. NOTE: Multiple values are supposed to be divided by a COLON only, WITHOUT spaces. For example:

$ discover-tempest-config remove identity.username=myname remove feature-enabled.api_ext=http[,https]

Default: []

Authentication Options

Options specific to the password plugin.

--os-auth-url Authentication URL

--os-system-scope Scope for system operations

--os-domain-id Domain ID to scope to

--os-domain-name Domain name to scope to

--os-project-id, --os-tenant-id Project ID to scope to

--os-project-name, --os-tenant-name Project name to scope to

--os-project-domain-id Domain ID containing project

--os-project-domain-name Domain name containing project

--os-trust-id Trust ID

--os-default-domain-id Optional domain ID to use with v3 and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

--os-default-domain-name Optional domain name to use with v3 API and v2 parameters. It will be used for both the user and project domain in v3 and ignored in v2 authentication.

--os-user-id User id

--os-username, --os-user-name Username

--os-user-domain-id Users domain id

--os-user-domain-name Users domain name

--os-password Users password

API Connection Options

Options controlling the HTTP API Connections

--insecure Explicitly allow client to perform insecure TLS (https) requests. The servers certificate will not be verified against any certificate authorities. This option should be used with caution.

Default: False

--os-cacert Specify a CA bundle file to use in verifying a TLS (https) server certificate. Defaults to env[OS_CACERT].

--os-cert Defaults to env[OS_CERT].

--os-key Defaults to env[OS_KEY].

--timeout Set request timeout (in seconds).

Default: 600

--collect-timing	Collect per-API call timing information. Default: False

Service Options

Options controlling the specialization of the API Connection from information found in the catalog

--os-service-type	Service type to request from the catalog
--os-service-name	Service name to request from the catalog
--os-interface	API Interface to use [public, internal, admin] Default: public
--os-region-name	Region of the cloud to use
--os-endpoint-override	Endpoint to use instead of the endpoint in the catalog
--os-api-version	Which version of the service API to use

1.4 User Guide

1.4.1 Usage

To install `python-tempestconf` follow our Installation Guide

For a successful execution of `python-tempestconf` a user needs to do one of the following:

- source OpenStack RC file before running **`discover-tempest-config`** command, see *Examples of usage with sourced credentials*
- use `clouds.yaml` file and take advantage of `openstacksdk` support and use a named cloud, see *Examples of usage with a named cloud*

If a user doesnt use `--create`, no resources, which require admin credentials, are created. See *Resources* section.

Examples of usage with sourced credentials

All of the examples in this section mentioned below **use** the following step **as a prerequisite**:

- Source your OpenStack RC file containing the cloud credentials. Lets say you have an overcloud_rc file with the following content:

```
$ cat overcloud_rc
unset OS_SERVICE_TOKEN
export OS_USERNAME=demo
export OS_PASSWORD='password'
export OS_AUTH_URL=http://172.16.52.15/identity/v3
export PS1='[\u@\h \W(keystone_demo)]\$ '
export OS_PROJECT_NAME=demo
export OS_USER_DOMAIN_NAME=default
```

(continues on next page)

(continued from previous page)

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_IDENTITY_API_VERSION=3
```

Then it can be sourced by:

```
$ source overcloud_rc
```

Note: Thanks to openstacksdk support, `python-tempestconf` is able to read cloud credentials from the shell environment, which means, they **dont need** to be explicitly passed via CLI.

Override values

Override values can be useful when a user wants to set a key-value pair in generated `tempest.conf` from one of the two following reasons:

- `python-tempestconf` is **not** able to discover it and therefore set the desired key-value pair in `tempest.conf` by itself
- `python-tempestconf` is able to discover it, but a user wants to set it differently

Values specified as overrides will be set to tempest.conf no matter if they were discovered or not. If a section or a key dont exist, they will be created.

In the following example we make the tool to print debugging information, we set that `tempest.conf` will be written to `etc/` directory and we pass some override values.

```
$ discover-tempest-config \
    --debug \
    --out etc/tempest.conf \
    auth.tempest_roles Member \
    identity.username MyOverrideUsername \
    section.key MyValue
```

Note: Please, notice that override values are written together (only then theyre parsed correctly) and after all other arguments (thats due to better readability).

The generated `tempest.conf` will look like:

```
$ cat etc/tempest.conf
<omitted some content>
[auth]
tempest_roles = Member
<omitted some content>

[identity]
username = MyOverrideUsername
<omitted some content>

```

(continues on next page)

(continued from previous page)

```
[section]
key = value
<omitted some content>
```

Prevent some key-value pairs to be set in tempest.conf

A user can define key-value pairs which are not wanted to be written to the generated `tempest.conf`. This can be useful in case when `python-tempestconf` discovers something which is not wanted by a user to have in `tempest.conf`. If the option is used, `python-tempestconf` will make sure that the defined values are not written to tempest.conf no matter if they were discovered or not.

```
$ discover-tempest-config \
    --remove section1.key1 \
    --remove section2.key2=value \
    --remove section3.key3=value1,value2
```

In the following case **all** api_extensions will be removed and `tempest.conf` will **not contain** the api_extensions key under volume-feature-enabled section.

```
$ discover-tempest-config \
    --remove volume-feature-enabled.api_extensions
```

In the following case **only** NMN api extension will be removed from the api_extensions list.

```
$ discover-tempest-config \
    --remove volume-feature-enabled.api_extensions=NMN
```

In the following case only NMN **and** OS-EXT-IPS api extensions will be removed.

```
$ discover-tempest-config \
    --remove volume-feature-enabled.api_extensions=NMN,OS-EXT-IPS
```

Note: `--remove` option will remove even values set as overrides

Note: This arguments functionality is opposite to `--append` one, see *Append values to tempest.conf*

Append values to tempest.conf

In a case when `python-tempestconf` is not able to discover some wanted api_extensions, you can make `python-tempestconf` append any extensions by using `--append` argument.

The following will make `python-tempestconf` append my_ext extension to volume-feature-enabled.api_extensions and tag and tag-ext extensions to network-feature-enabled.api_extensions.

```
$ discover-tempest-config \
    --append volume-feature-enabled.api_extensions=my_ext \
    --append network-feature-enabled.api_extensions=tag,tag-ext
```

Note: This arguments functionality is opposite to `--remove` one, see *Prevent some key-value pairs to be set in tempest.conf*

Usage with tempest accounts file

To read more about `accounts.yaml` file and how to generate it follow these links:

- what is accounts.yaml?
- how to generate it?

When `--test-accounts` argument is used, `python-tempestconf` will not write any credentials to generated `tempest.conf` file, it will add a **test_accounts_file** key to **auth** section with value equal to the path provided by the `--test-accounts` argument. Also **use_dynamic_credentials** under **auth** section will be set to False as tempest documentation suggests.

This argument can be useful when a user doesnt want to store credentials in `tempest.conf`, f.e: the user wants to share the `tempest.conf`.

If you already have the file created, you can run **`discover-tempest-config`** command with `--test-accounts` argument:

```
$ discover-tempest-config \
    --out etc/tempest.conf \
    --test-accounts /path/to/my/accounts.yaml
```

The generated `tempest.conf` will look like:

```
$ cat etc/tempest.conf
<omitted some content>
[auth]
test_accounts_file = /path/to/my/accounts.yaml
use_dynamic_credentials = False
<omitted some content>
```

non-admin argument

If your credentials are **non-admin ones**, which means that you are **not allowed** to create any resources in your cloud, then please specify `--non-admin` argument. When this argument is used, `python-tempestconf` will **not create** any resources.

```
$ discover-tempest-config \
    -v \
    --debug \
    --non-admin
```

Examples of usage with a named cloud

`python-tempestconf` supports openstacksdk so instead of sourcing an OpenStack RC file a user can use clouds.yml file. Location where this file should be stored and syntax which is used to define it can be found here

Lets say there is a `clouds.yaml` file located in `/etc/openstack/` with the following content:

```
$ cat /etc/openstack/clouds.yaml
clouds:
  devstack:
    auth:
      auth_url: http://172.16.52.15/identity/v3
      password: password
      project_domain_id: default
      project_name: demo
      user_domain_id: default
      username: demo
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '2'
```

Then if you use `--os-cloud` argument you can run **`discover-tempest-config` without** setting any OS_* environment variable (for example by sourcing any OpenStack RC file).

`--os-cloud` specifies one of the cloud names located in the `clouds.yaml` file.

```
$ discover-tempest-config \
    --debug \
    --os-cloud devstack
```

So the call from *non-admin argument* section would for example look like:

```
$ discover-tempest-config \
    -v \
    --debug \
    --non-admin \
    --os-cloud devstack
```

The call from *Usage with tempest accounts file* section would for example look like:

```
$ discover-tempest-config \
    --os-cloud devstack \
    --out etc/tempest.conf \
    --test-accounts /path/to/my/accounts.yaml
```

Resources

Without specifying `--create` argument, no resources which require admin credentials are crated during the `python-tempestconf` execution. For the documentation on how to use `--create` argument see Admin User Guide

This affects these types of resources:

- users
- images
- flavors

Users

For a successful execution of Tempest at least two users need to be created (the default concurrency is 2). Therefore `python-tempestconf` looks for the following two users:

- the user who started `python-tempestconf`
- the alt user defined by:
  - identity.alt_username
  - identity.alt_password
  - identity.alt_project_name

Note: These values are set by default, have a look at default values which `python-tempestconf` sets to a `tempest.conf`

If the users are not found, they cant be created, so **`discover-tempest-config`** ends with an exception.

Images

Any user can create an image, therefore `--create` argument doesnt have to be used in order to have created images, necessary for tempest execution, by `python-tempestconf`.

However, when non-admin credentials are used, the created images will have **community** visibility. Its because users without admin credentials cant create a public image and private images are not visible for other users - tempest tests **would fail** finding the image, because they are usually run under a **different user.**

When admin credentials are used, the images are created as public ones.

`--image` argument is used to specify an image which will be uploaded to glance and used later by tempest tests for booting VMs.

The following example will upload `/my/path/to/myImage.img` image to glance twice. First **compute.image_ref** will be equal to the ID of the uploaded image. Then the image is uploaded to glance again but **compute.image_alt_ref** is set to the new corresponding ID:

```
$ discover-tempest-config \
    --os-cloud myCloud \
    --image /my/path/to/myImage.img
```

In the following example, an override value is used to set **compute.image_ref**, which means that the image specified by `--image` is uploaded and only **compute.image_alt_ref** is set to the ID of newly created image.

```
$ discover-tempest-config \
    --os-cloud myCloud \
    compute.image_ref 2eb9f6c9-bd32-427d-850d-c3bb3cfaaa87
```

Note: `python-tempestconf` checks by image name, if it is already present in glance and only in case its not present there, will upload the image.

Note: If the image ID specified as an override is not found, the image where `--image` points to is used.

If `--image` is not defined, the default image (see CLI options) is chosen to be uploaded.

Converting images to .raw format

By using `--convert-to-raw` argument you can make `python-tempestconf` convert the image given by `--image` argument to **.raw** format before uploading it to glance. If Ceph is used as a backend, the boot time of the image will be faster when the image is already in **.raw** format.

In the following example the `/my/path/to/myImage.img` image will be downloaded, then converted to **.raw** format and then uploaded to glance.

```
$ discover-tempest-config \
    --os-cloud myCloud \
    --image /my/path/to/myImage.img \
    --convert-to-raw
```

Flavors

`python-tempestconf` looks by default for these two flavors:

- *m1.nano* with 64 MB of RAM, which will be set as **compute.flavor_ref**
- *m1.micro* with 128 MB of RAM, which will be set as **compute.flavor_alt_ref**

If a user used `--flavor-min-mem` argument, `python-tempestconf` will look for these two flavors:

- *custom*
- *custom_alt*

Note: `python-tempestconf` looks for flavors by name, so if a user has had a flavor with name *custom/custom_alt* already created, those flavors IDs will be set as **compute.flavor_ref/compute.flavor_ref_alt** without checking if theirs RAM size is equal to the one specified by `--flavor-min-mem`.

If they are not found and `--create` argument is not used, the tool will try to auto discover two smallest flavors available in the system. If at least two flavors are not found, the tool ends with an exception.

If two flavors are found, their IDs will be set to `tempest.conf`, see the following example:

```
$ discover-tempest-config \
    --out etc/tempest.conf
```

The generated tempest.conf will look like:

```
$ cat etc/tempest.conf
<omitted some content>
[compute]
# typically an ID of the smaller flavor found
flavor_ref = <ID_1>
# typically an ID of the bigger flavor found
flavor_alt_ref = <ID_2>
<omitted some content>
```

In the following example, an override option specifies **compute.flavor_ref** ID, which if its found, the tool continues with looking for a **m1.micro** flavor to be set as **compute.flavor_alt_ref** as was explained above.

```
$ discover-tempest-config \
    --out etc/tempest.conf \
    compute.flavor_ref 123
```

Note: If the **compute.flavor_ref** ID is not found, the tool ends with an exception.

1.4.2 Use python-tempestconf as Python module

`python-tempestconf` can be imported and used from a different Python project.

Warning: The import of config_tempest is possible **only when the version of the tool is at least 2.0.0**.

Installation

See our Install Guide on how to install `python-tempestconf`.

Import

Import `python-tempestconf` in your project as follows:

```
from config_tempest import main as tempestconf
```

`python-tempestconf` needs cloud credentials in order to create a tempest configuration file. There is a helper method for obtaining cloud credentials which uses openstacksdk for parsing the cloud for credentials.

The following example shows how to get cloud credentials and how to pass it to the configuration tool:

```
# The following call will return a dict containing cloud credentials,
# for example:
# >>> tempestconf.get_cloud_creds(args_namespace)
# {
#     'username': 'demo',
#     'project_name': 'demo',
#     'user_domain_name': 'Default',
#     'auth_url': 'http://172.16.52.8:5000/v3',
#     'password': 'f0921edc3c2b4fc8',
#     'project_domain_name': 'Default'
# }
cloud_creds = tempestconf.get_cloud_creds(args_namespace)

# Then the configuration step can be run using:
tempestconf.config_tempest(cloud_creds=cloud_creds)
```

Note: If *args_namespace* contains `--os-cloud` argument, the *get_cloud_creds* method returns cloud credentials related to that cloud, otherwise, it returns credentials of the current cloud (according to the sourced credentials).

List of arguments which may be passed to *config_tempest*

- cloud_creds
- create
- create_accounts_file
- debug
- deployer_input
- image_disk_format
- image_path

- network_id
- non_admin
- os_cloud
- out
- overrides
- remove
- test_accounts
- verbose

OR

- profile, see why **or** in CLI documentation

Note: For detailed description of the options see our CLI documentation

Example implementation

1. Save following code snippet as `example.py`:

```
import argparse
from config_tempest import main as tempestconf

parser = argparse.ArgumentParser(description='Example␣
↪implementation.')
args = parser.parse_args()

# get the credentials of the current cloud according to
# the sourced credentials
cloud_creds = tempestconf.get_cloud_creds(args)

tempestconf.config_tempest(non_admin=True,
                           out='./etc/tempest.conf',
                           cloud_creds=cloud_creds)
```

2. Source your OpenStack RC file containing the cloud credentials. Lets say you have a overcloud_rc file with the following content:

```
$ cat overcloud_rc
unset OS_SERVICE_TOKEN
export OS_USERNAME=demo
export OS_PASSWORD='password'
export OS_AUTH_URL=http://172.16.52.15/identity/v3
export PS1='[\u@\h \W(keystone_demo)]\$ '
export OS_PROJECT_NAME=demo
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_DOMAIN_NAME=default
export OS_IDENTITY_API_VERSION=3
```

Then it can be source by:

```
$ source overcloud_rc
```

3. Run `example.py`:

```
$ python example.py
```

Example implementation with a named cloud

1. Lets say there is a `clouds.yaml` file located in `/etc/openstack/` with the following content:

```
$ cat /etc/openstack/clouds.yaml
clouds:
  devstack:
    auth:
      auth_url: http://172.16.52.15/identity/v3
      password: password
      project_domain_id: default
      project_name: demo
      user_domain_id: default
      username: demo
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '2'
```

2. Save following code snippet as `example.py`:

```
import argparse
from config_tempest import main as tempestconf

parser = argparse.ArgumentParser(description='Example␣
↪implementation.')
# Let's add an os_cloud option which will be passed
# to config_tempest later.
parser.add_argument('--os-cloud', help='Name of a named cloud.')
args = parser.parse_args()

# get the credentials to the devstack cloud
cloud_creds = tempestconf.get_cloud_creds(args)

tempestconf.config_tempest(non_admin=True,
                           out='./etc/tempest.conf',
                           cloud_creds=cloud_creds)
```

3. Run `example.py`:

```
$ python example.py --os-cloud devstack
```

Note: In this example you **dont need** to source cloud credentials. The credentials are obtained from the /etc/openstack/clouds.yaml file thanks to --os-cloud argument.

1.4.3 Use python-tempestconf with a profile.yaml file

A profile.yaml is helpful mainly in jobs running on different versions of OpenStack, because arguments for python-tempestconf may differ in each OpenStack version. Using the --profile argument those jobs can use a profile.yaml file for each OpenStack version which makes the code of those jobs much clearer and more readable, as for example they contain less if else statements,

Note: Apart from --deployer-input config file which specifies content of a tempest.conf, profile.yaml file defines arguments which are passed to python-tempestconf, see CLI documentation.

Warning: If this argument is used, other arguments cannot be defined, it means, user uses either CLI arguments or profile.yaml file.

Generating a sample profile.yaml file

```
$ discover-tempest-config --generate-profile ./etc/profile.yaml
```

```
$ cat ./etc/profile.yaml
collect_timing: false
create: false
create_accounts_file: null
debug: false
deployer_input: null
endpoint_type: null
generate_profile: ./etc/profile.yaml
http_timeout: null
image: http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img
image_disk_format: qcow2
insecure: false
network_id: null
no_default_deployer: false
non_admin: false
os_api_version: null
os_auth_type: password
os_auth_url: null
os_cacert: null
os_cert: null
os_cloud: null
os_default_domain_id: null
os_default_domain_name: null
os_domain_id: null
os_domain_name: null
```

(continues on next page)

(continued from previous page)

```
os_endpoint_override: null
os_endpoint_type: null
os_interface: public
os_key: null
os_password: null
os_project_domain_id: null
os_project_domain_name: null
os_project_id: null
os_project_name: null
os_region_name: null
os_service_name: null
os_service_type: null
os_system_scope: null
os_trust_id: null
os_user_domain_id: null
os_user_domain_name: null
os_user_id: null
os_username: null
out: etc/tempest.conf
test_accounts: null
timeout: 600
verbose: false
append: {}
  #identity.username: username
  #network-feature-enabled.api_extensions:
  #  - dvr
  #  - extension
overrides: {}
  #identity.username: username
  #identity.password:
  #  - my_password
  #network-feature-enabled.api_extensions:
  #  - dvr
  #  - extension
remove: {}
  #identity.username: username
  #network-feature-enabled.api_extensions:
  #  - dvr
  #  - extension
```

Note: The generated sample of a `profile.yaml` file contains all `python-tempestconf` arguments set to their default values. That means, that you can **remove arguments you didnt modify** to keep the file simple and more readable.

`python-tempestconf` accepts both of the following inputs, so you can use what suits you better, either strings or lists:

```
create: True
```

(continues on next page)

(continued from previous page)

```
out: ./etc/tempest.conf
deployer-input: ./deploy.txt
no-default-deployer: False
overrides:
  identity.username: my_override
  identity.password: my_password
  network-feature-enabled.api_extensions: all
  volume-feature-enabled.api_extensions: dvr,mine
remove:
  auth.identity: username
  network-feature-enabled.api_extensions: ''
  volume-feature-enabled.api_extensions: dvr,mine
```

```
create: True
out: ./etc/tempest.conf
deployer-input: ./deploy.txt
no-default-deployer: False
overrides:
  identity.username: my_override
  identity.password:
    - my_password
  network-feature-enabled.api_extensions:
    - all
  volume-feature-enabled.api_extensions:
    - dvr
    - mine
remove:
  auth.identity: username
  network-feature-enabled.api_extensions:
    - ''
  volume-feature-enabled.api_extensions:
    - dvr
    - mine
```

Using profile.yaml file

After youve created your customized `profile.yaml` file, lets say in `./etc/profile.yaml`, use it as follows:

```
$ discover-tempest-config --profile ./etc/profile.yaml
```

1.4.4 Default values

`python-tempestconf` provides sensitive default values for many options in order to simplify its usage, reducing the amount of options that needs to be specified.

Here is the list of tempest options, which are set by default:

```
[DEFAULT]
debug = true
use_stderr = false
log_file = tempest.log

[identity]
username = demo_tempestconf
password = secrete
project_name = demo
alt_username = alt_demo_tempestconf
alt_password = secrete
alt_project_name = alt_demo

[auth]
; if _member_ role is not present in the system, python-tempestconf
; looks for member role and if the member is also not present
; tempest_roles option is not set
tempest_roles = _member_
admin_username = admin
admin_project_name = admin
admin_domain_name = Default

[object-storage]
reseller_admin_role = ResellerAdmin

[oslo-concurrency]
lock_path = /tmp

[compute-feature-enabled]
# Default deployment does not use shared storage
live_migration = false
live_migrate_paused_instances = true
preserve_ports = true

[network-feature-enabled]
ipv6_subnet_attributes = true
```

1.5 Admin User Guide

1.5.1 Usage

Before reading this page, **its recommended** to go through User Guide first as the content on this site is more advanced and uses knowledge gained from the User Guide.

This page shows examples of usage of `python-tempestconf` where **admin credentials** are **required**. That means, only users with admin credentials will run **`discover-tempest-config`** with arguments described on this page successfully.

Why admin credentials? Its because `python-tempestconf` can create resources **necessary** for tempest execution in order to make users life easier.

The following resources are created **only when** `--create` argument is used:

- flavors, to see what flavors are created, see User Guide, Flavors section
- users, to see what users are created, see User Guide, Users section

Examples

In the following example, `python-tempestconf` will create all necessary resources (Flavors and Users) if they dont exist already:

```
$ discover-tempest-config \
    --os-cloud devstack-admin \
    --create
```

If a user wants to use a custom image (instead of the default cirros one), a minimum memory and disk size for new flavors can be defined by `--flavor-min-mem` and `--flavor-min-disk` arguments.

```
$ discover-tempest-config \
    --image <path/url to custom image> \
    --flavor-min-mem 1024 \
    --flavor-min-disk 10
```

In the example above `python-tempestconf` will create *custom* flavor with 1024 MB of RAM and 10 GB of disk size and *custom_alt** flavor with 1024 + 1 MB of RAM and 10 GB of disk size.

`python-tempestconf` can also create a minimal accounts file when `--create-accounts-file` is used. It can be useful when a user doesnt have any `accounts.yaml` and wants to create it. It can be done with one call:

```
$ discover-tempest-config \
    --os-cloud devstack-admin \
    --create \
    --create-accounts-file ~/accounts.yaml
```

The call above will behave the same as if `--test-accounts` argument was used, see here. The generated accounts file will look similarly to this one:

```
$ cat ~/accounts.yaml
# A minimal accounts.yaml file
# Will likely not work with swift, since additional
# roles are required. For more documentation see:
# https://opendev.org/openstack/tempest/src/branch/master/etc/accounts.yaml.
↪sample

- password: password
  project_name: admin
  username: admin
```

Note: More about accounts file can be found in our documentation about Usage with tempest accounts file

1.6 Contributor Guide

1.6.1 How to Contribute

`python-tempestconf` source code is publicly available. You can contribute code to individual projects, documentation, report bugs and vulnerabilities and request features.

Reporting Bugs

We have a storyboard project created to track any change required for `python-tempestconf`. If you have found any bug, please, report it there.

Important information **to mention**:

- **System** on which the problem occurred (e.g. CentOS, Ubuntu,)
- The source of `python-tempestconf` you have used. The **package version number** in case of RPM or the **branch used** in case of installation from git.
- The **exact command** with all arguments you have used.
- Its always better to include the **console output** as well.

Requesting Features

Create a story with a task for our project containing all the relevant information, mainly:

- **description** of the feature
- **inputs** (new CLI option,) and **outputs** (desired configuration in tempest.conf) of the feature
- the **reason why** it should be implemented

Fixing bugs

1. If you have found a bug and you know how to fix it, please, check our storyboard project for any stories which may relate to the issue. If you havent found any related stories, please, create one. Check *Reporting Bugs*.
2. Follow *Contributing Code* and submit a code review in https://review.opendev.org/.

Contributing Code

Like any other project part of OpenStack, the development of `python-tempestconf` follows the OpenStack guidelines for contribution.

Learn how to contribute into OpenStack.

If you have made any changes in the source code, **run tests locally before posting a review**. You can do so by running tox.

If youve made any changes in the documentation (under `doc/`) run:

```
$ tox -edocs
```

If youve made any changes in the source code run unit tests as follows:

```
$ tox -epy36
```

and **pep8** check like following:

```
$ tox -epep8
```

If youve written also a releasenote, make sure the syntax is correct by running:

```
$ tox -ereleasenotes
```

If youve made any changes which are related to a task in a story in our storyboard project, please, **include a story and task number in the commit message**.

- search