Tuskar Documentation

Release 2013.2.dev15

OpenStack Foundation

January 19, 2015

Contents

1 High-Level Overview **3**
1.1 Related Projects 3

2 Developer Information **5**
2.1 Install and Contribute 5
2.2 API version 2 10

3 Contact Us **13**

Tuskar is a management service for planning TripleO deployments.

Interested in seeing the full Tuskar and Tuskar UI setup? Watch the demo.

CHAPTER 1

High-Level Overview

TODO Add project overview

- *TODO* feature examples
- *TODO* link to high-level portion of FAQ
- *Recommended reading*

1.1 Related Projects

- tuskar-ui - tuskar-ui provides dashboard access to Tuskar functionality as a Horizon plugin. See the Tuskar UI documentation
- python-tuskarclient - A Python client for the Tuskar API, python-tuskarclient is utilized by tuskar-ui.

CHAPTER 2

Developer Information

2.1 Install and Contribute

2.1.1 Developer Installation Guide

The Tuskar source code should be pulled directly from git.

```
git clone https://git.openstack.org/openstack/tuskar
```

Dependencies

Setting up a local environment for development can be done with tox.

```
# install prerequisites
* Fedora/RHEL:
$ sudo yum install python-devel python-pip libxml2-devel \
    libxslt-devel postgresql-devel mariadb-devel

* Ubuntu/Debian:
$ sudo apt-get install python-dev python-pip libxml2-dev \
    libxslt-dev libpq-dev libmysqlclient-dev
```

Note: If you wish you run Tuskar against MySQL or PostgreSQL you will need also install and configure these at this point. Otherwise you can run Tuskar with an sqlite database.

To run the Tuskar test suite you will also need to install Tox.

```
$ sudo pip install tox
```

Note: An issue with tox requires that you use a version <1.70 or >= 1.7.2.

Now create your virtualenv.

```
$ cd <your_src_dir>/tuskar
$ tox -e venv
```

Note: If `pip install` fails due to an outdated setuptools, you can try to update it first.

```
$ sudo pip install --upgrade setuptools
```

To run the test suite use the following command. This will run against Python 2.6, Python 2.7 and run the flake8 code linting.

```
$ tox
```

Note: If you only have access to Python 2.6 or 2.7 locally pass in *-e py26* or *-e py27* respectively.

Configuration

Copy the sample configuration file:

```
$ cp etc/tuskar/tuskar.conf.sample etc/tuskar/tuskar.conf
```

We need to tell tuskar where to connect to database. Edit the config file in `database` section and change

```
#connection=<None>
```

to

```
connection=sqlite:///tuskar/tuskar.sqlite
```

Note: If you are using a different database backend, you will need to enter a SQLAlchemy compatible conection string for this setting.

We need to initialise the database schema.

```
# activate the virtualenv
$ source .tox/venv/bin/activate

# if you delete tuskar.sqlite this will force creation of tables again - e.g.
# if you added a new resource table definitions etc in an existing migration
# file
$ tuskar-dbsync --config-file etc/tuskar/tuskar.conf
```

You can verify this was successful (in addition to seeing no error output) with.

```
$ sqlite3 tuskar/tuskar.sqlite .schema
```

Then, launch the app.

```
$ tuskar-api --config-file etc/tuskar/tuskar.conf
```

You can then verify that everything worked by running.

```
$ curl -v -X GET -H 'Accept: application/json' http://0.0.0.0:8585/v2/plans/ | python -mjson.tool
```

This command should return JSON with an empty result set.

Running Tuskar API

Whenever you want to run the API again, just switch to the virtualenv and run *tuskar-api* command.

```
$ source .tox/venv/bin/activate
$ tuskar-api --config-file etc/tuskar/tuskar.conf
```

Loading Initial Roles

Tuskar needs to be provided with a set of roles that can be added to a deployment plan. The following steps will add the roles from the TripleO Heat Templates repository.

```
$ git clone http://git.openstack.org/cgit/openstack/tripleo-heat-templates/
$ cd tripleo-heat-templates
$ tuskar-load-roles --config-file etc/tuskar/tuskar.conf \
    -r compute.yaml \
    -r controller.yaml
```

After this, if the Tuskar API isn't running, start it with the above command and the following curl command should show you the loaded roles.

```
$ curl -v -X GET -H 'Accept: application/json' http://0.0.0.0:8585/v2/roles/ | python -mjson.tool
```

Keystone Configuration

By default, Tuskar is configured to skip authentication for REST API calls. Keystone authentication can be enabled by making the appropriate changes to the `tuskar.conf` file as described in the keystone documentation

Contributing

For additional developer information, take a look at *the contributing guide*.

2.1.2 Contributing to Tuskar

Tuskar follows the OpenStack development processes for code and communication. The repository is hosted on git.openstack.org, bugs and blueprints are on Launchpad and we use the openstack-dev mailing list (subject *[tuskar]*) and the *#tripleo* IRC channel for communication.

As Tuskar is under the TripleO umbrella of projects you will also want to look at the TripleO contributing guidelines.

Coding Standards

We comply with the OpenStack coding standards.

Be sure to familiarise yourself with OpenStack's Gerrit Workflow.

Before submitting your code, please make sure you have completed the following checklist:

1. Update the API docs (if needed)
2. Update the tests (if needed)

Finding your way around

There are various pieces of the codebase that may not be immediately obvious to a newcomer to the project, so we attempt to explain some of that in this section.

Where do the tuskar commands come from? (tuskar-api, tuskar-dbsync, etc) The project-specific commands live in tuskar/cmd, and are implementations that use the oslo.config project as a base. They are generated and put into your venv when you run 'python setup.py develop'. Adding a new one consists of:

1. Creating a new file in tuskar/cmd
2. Adding the appropriate name and package reference to the entry_points section of setup.cfg

How do I add a new controller? Controllers are contained in tuskar/api/controllers/v2.py. To add a new controller, you need to add an 'HTTP Representation' of whatever model you wish to expose with this controller. This is a simple python object that extends Base, and describes the key and value types that the object will return. For example, say there is a Foo model object you wish to return.

```
class Foo(Base):
    id = int
    name = wtypes.text
    fred = Fred  # Fred is another object defined in this file
```

Then add a controller for it (anywhere above the Controller class, which is the last in the file. For example:

```
class FoosController(rest.RestController):
    @wsme_pecan.wsexpose([Foo])
    def get_all(self)
        result = []
        """Do some things to get your list of Foos"""
        return result
```

Lastly, add a reference to the controller in the Controller class at the bottom of the file as so.

```
class Controller(object):
    foos = FoosController()
```

The name you give the controller above will be how it is accessed by the client, so in the above case, you could get the list of foos with.

```
curl http://0.0.0.0:8585/v1/foos
```

For doing something simple, like a poc controller that doesn't return any objects, you can return plain text as so

```
class FarkleController(rest.RestController):
    @wsme_pecan.wsexpose(None, wtypes.text)
    def get_all(self):
        return "Hi, I am farkle!"
```

Where are my changes to the app? You may make a change to, say, a controller, and wonder why your change does not seem to happen when you call your curl command on that resource. This is because, at least at the current time, you must ctrl+c to kill the tuskar-api server, and then restart it again to pick up your changes.

How do I create a new model? Models live in tuskar/db/sqlalchemy/. There are two files here of relevance for describing the model (we will get to defining the table in the next section), api.py and models.py. The models.py file contains the definition of the columns to expose to the client for the model objects, as well as a mapping of the object in this file to the tablename define in the migration (below). In api.py, we have utility methods, as well as validation rules and other custom methods for interacting with the models.

How do I define the table for my new model? This is described in a migration file, located in tuskar/db/sqlalchemy/migrate_repo/versions/. Each new table or change to an existing table should get a new file here with a descriptive name, starting with a 3 digit number. Each new file should increment the number to avoid collisions. The primary part of this file is the definition of your table, which s done via a Table object, and you describe the columns, using, surprisingly enough, a Column object. There are upgrade nd downgrade methods in these migrations to describe what to do for creating a given set of tables, as well as dropping them, or rolling back to what was done before the upgrade.

Writing and Running tests

We use testtools for our unit tests, and mox for mock objects.

You can run tests using Tox:

```
$ tox
```

This will run tests under Python 2.6, 2.7 and verify PEP 8 compliance. The identical test suite is run by OpenStack's Jenkins whenever you send a patch.

2.1.3 Recommended Reading

Tuskar Design Discussions

- Juno Planning
- Template storage planning
- TripleO Specifications

Relevant OpenStack Projects

- TripleO
- Heat
- oslo.db
- oslo.config
- hacking This enforces openstack community coding style guidelines

General Python/Frameworks

- dive into python
- pecan
- sqlalchemy
- style guide This guide is the baseline for 'hacking' above.

2.2 API version 2

2.2.1 cURL Commands for API ver. 2

Resources

- Plan
- Role

Plan

Example of JSON Representation of Plan

```
{
  "created_at": "2014-09-26T20:23:14.222815",
  "description": "Development testing cloud",
  "name": "dev-cloud",
  "parameters":
  [
    {
      "default": "guest",
      "description": "The password for RabbitMQ",
      "hidden": true,
      "label": null,
      "name": "compute-1::RabbitPassword",
      "value": "secret-password"
    }
  ],
  "roles":
  [
    {
      "description": "OpenStack hypervisor node. Can be wrapped in a ResourceGroup for scaling.\n",
      "name": "compute",
      "uuid": "b7b1583c-5c80-481f-a25b-708ed4a39734",
      "version": 1
    }
  ],
  "updated_at": null,
  "uuid": "53268a27-afc8-4b21-839f-90227dd7a001"
}
```

List All Plans

```
curl -v -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' http://0.0.0.0:8585/
```

Retrieve a Single Plan

```
curl -v -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' http://0.0.0.0:8585/
```

Create a New Plan

```
curl -v -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -d  '
 {
   "name": "dev-cloud",
   "description": "Development testing cloud",
 }
 ' http://0.0.0.0:8585/v2/plans
```

This command will create new Plan without any Roles associated with it. To assign a Role to Plan see How to Add a Role to a Plan.

Delete an Existing Plan

```
curl -v -X DELETE http://localhost:8585/v2/plans/53268a27-afc8-4b21-839f-90227dd7a001
```

Changing a Plan's Configuration Values

```
curl -v -X PATCH -H 'Content-Type: application/json' -H 'Accept: application/json' -d '
[
  {
    "name" : "database_host",
    "value" : "10.11.12.13"
  },
  {
    "name" : "database_password",
    "value" : "secret"
  }
]
' http://0.0.0.0:8585/v2/plans/53268a27-afc8-4b21-839f-90227dd7a001
```

You can change only existing parameters in Plan.

Retrieve a Plan's Template Files

```
curl -v -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' http://0.0.0.0:8585/
```

Example of JSON representation:

```
{
  "environment.yaml" : "... content of template file ...",
  "plan.yaml" : "... content of template file ...",
  "provider-compute-1.yaml" : "... content of template file ..."
}
```

back to top

Role

Example of JSON Representation of Role

```
{
  "description": "OpenStack hypervisor node. Can be wrapped in a ResourceGroup for scaling.\n",
  "name": "compute",
  "uuid": "b7b1583c-5c80-481f-a25b-708ed4a39734",
  "version": 1
}
```

Retrieving Possible Roles

```
curl -v -X GET -H 'Content-Type: application/json' -H 'Accept: application/json' http://0.0.0.0:8585/
```

Adding a Role to a Plan

```
curl -v -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -d  '
 {
   "uuid": "b7b1583c-5c80-481f-a25b-708ed4a39734"
 }
 ' http://0.0.0.0:8585/v2/plans/53268a27-afc8-4b21-839f-90227dd7a001
```

Removing a Role from a Plan

```
curl -v -X DELETE http://localhost:8585/v2/plans/53268a27-afc8-4b21-839f-90227dd7a001/roles/b7b1583c-
```

back to top

CHAPTER 3

Contact Us

Join us on IRC (Internet Relay Chat):

```
Network: Freenode (irc.freenode.net/tuskar)
Channel: #tripleo and #tuskar
```