gabbi-tempest Documentation

Chris Dent

Oct 29, 2018

Contents

CHAPTER 1

Using With Zuul

gabbi-tempest is designed to work well with OpenStack infrastructure. A base zuul job, named 'gabbi-tempest', is provided. Children of this job need to only define the path of where their gabbi YAML files are located and then request that the job run. For example a job for nova where the YAML files live in a directory named `gate/gabbits` from the root of the repository would look like this:

```
- job:
  name: nova-gabbi-tempest
  parent: gabbi-tempest
  timeout: 10800
  description: |
      Test live nova using gabbi.
  vars:
    gabbi_tempest_path: "{{ devstack_base_dir|default('/opt/stack') }}/nova/gate/
↪gabbits"
```

Once that job is in place, further tests are added by adding more YAML files to the `gate/gabbits` directory.

1.1 More Detail

The `gabbi-tempest` zuul job defines a small number of required settings, but has its parent, `devstack-tempest`, do most of the work. The settings:

- Add `openstack/gabbi-tempest` to `required-projects`, making sure the latest version of the plugin is available.
- Chooses the `all` tox environment when running tempest.
- Passes `gabbi` as the test regular expression.
- Sets concurrency to 1 to make sure the gabbi tests run in order (otherwise duplicate tests can run).
- Sets `TEMPEST_PLUGINS`.
- Turns on Python 3..

Gabbi-tempest is a Tempest plugin that enables testing the APIs of running OpenStack services, integrated with tempest but without needing to write Python. Instead the YAML format provided by gabbi is used to write and evaluate HTTP requests and responses.

Tests are placed in YAML files in one or more directories. Those directories are added to a `GABBI_TEMPEST_PATH` environment variable. When that variable is passed into a tempest test runner that is aware of the gabbi plugin, the files on that path will be used to create tempests tests.

The test harness sets a series of enviornment variables that can be used in the YAML to reach the available services. The available variables may be extended in two ways:

- Adding them to the environment that calls tempest if the values are known.
- Setting them in a subclass of the plugin if the values need to be calculated from what tempest knows.

For each service in the service catalog there are `<SERVICE_TYPE>_SERVICE` and `<SERVICE_TYPE>_BASE` variables (e.g., `PLACEMENT_SERVICE` and `PLACEMENT_BASE`). A useful `SERVICE_TOKEN`, `IMAGE_REF`, `FLAVOR_REF` and `FLAVOR_REF_ALT` are also available.

For the time being the `SERVICE_TOKEN` is `admin`.

gabbi-tempest can be used with *Using With Zuul*.

CHAPTER 2

Trying It

To experiment with this you need a working tempest installation and configuration. One way to do that is to use devstack with the following added to the local.conf:

```
enable_service tempest
INSTALL_TEMPEST=True
```

in local.conf.

Once tempest is confirmed to be working, gabbi-tempest must be installed. Either install it from PyPI:

```
pip install gabbi-tempest
```

Or make a clone of this repo, cd into it, and do the equivalent of:

```
pip install -e .
```

If you are using virtualenvs or need sudo, your form will be different.

Create some gabbi tests that exercise the OpenStack services. There are sample files in the `samples` directory in the repo.

Go to the tempest directory (often `/opt/stack/tempest`) and run tempest as follows. Adding the `regex` will limit the test run to just gabbi related tests:

```
GABBI_TEMPEST_PATH=/path/one:/path/two tempest run --regex gabbi
```

This will run the tests described by the YAML files in `/path/one` and `/path/two`.

CHAPTER 3

History

This code is based on the work of Mehdi Abaakouk who made a tempest plugin for gnocchi that worked with gabbi. He figured out the details of the plugin structure.