Ripcord Documentation

Release

Paul Belanger

March 19, 2014

Contents

Contents:

CHAPTER 1

API Complete Reference

1.1 Ripcord API Reference

1.1.1 API Operations

CHAPTER 2

Developers Docs

2.1 Developer Guide

2.1.1 Programming HowTos and Tutorials

Setting Up a Development Environment

Note: This document's look and feel originates from the OpenStack Nova project. Since we also use a similar toolset for testing, we can also use some of their documentaion. We have obviously made changes that only affect our project but we credit the OpenStack project for the original [1].

This page describes how to setup a working Python development environment that can be used in developing ripcord on Ubuntu. These instructions assume you're already familiar with git.

Following these instructions will allow you to run the ripcord unit tests.

Virtual environments

Ripcord development uses a set of shell scripts from OpenStack's DevStack. Virtual enviroments with venv are also available with the source code.

Linux Systems

Note: This section is tested for Ripcord on Ubuntu (12.04-64) distribution. Feel free to add notes and change according to your experiences or operating system.

Install the prerequisite packages.

On Ubuntu Precise (12.04):

[1] See http://docs.openstack.org/developer/nova/devref/development.environment.html

```
$ sudo apt-get install python-dev python-pip git-core libffi-dev libxml2-dev libxslt1-dev libmysqlcli
```

Getting the code

Grab the code from GitHub:

```
$ git clone https://github.com/kickstandproject/ripcord.git
$ cd ripcord
```

Installing the proper version of tox

This project uses `tox` to automatically download and install any necessary project dependencies. If you haven't previously done so, you will need to install version 1.6.1 of `tox`.

```
$ pip install tox==1.6.1 --user
```

This command installs `tox` version 1.6.1 in the `.local/bin` subdirectory of your home directory.

Using tox to satisfy dependencies and run tests

Next, we will use `tox` to download the project dependencies and run a series of tests on the code, to ensure it's working properly. Please make sure that you're still in the `ripcord` directory, and run:

```
$ ~/.local/bin/tox -v
```

We've added the `-v` parameter to make `tox` be a little more verbose about its progress. After a few minutes, if everything goes successfully, you should see a message similar to this:

```
_____________________________________ summary ______________________________________
  docs: commands succeeded
  py27: commands succeeded
  pep8: commands succeeded
  congratulations :)
```

Using tox for virtual environments

Ripcord development uses tox as a wrapper around virtualenv to track and manage Python dependencies while in development and testing.
To activate the Ripcord virtualenv for the extent of your current shell session you can run:

```
$ source .tox/py27/bin/activate
```

To deactivate the virtualenv sandbox and return to your regular shell, type:

```
$ deactivate
```

Or, if you prefer, you can run commands in the virtualenv on a case by case basis by running the following (substituting your actual command instead of `echo hello`):

```
$ ~/.local/bin/tox -evenv -- echo hello
```

Also, please note that `make test` will automatically use the virtualenv sandbox.

Setting up Ripcord

With Ripcord installed and passing tests, it's time to start configuring it. To begin, we'll activate the Ripcord virtual environment:

```
$ source .tox/py27/bin/activate
```

Once inside the virtual environment, you should notice that your shell prompt has changed to indicate that you're inside the virtual environment.

Next, let's sync the Ripcord database:

```
$ ripcord-manage db-sync
```

After that, we can start configuring Ripcord. Initially, we'll copy the sample configuration file to `ripcord.conf`:

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

Now we can edit the `ripcord.conf` file. When you first get started with developing for Ripcord, it's useful to disable the authentication. Of course, you'll want to re-enable authentication before using Ripcord in a production environment. To disable authentication, edit the `etc/ripcord/ripcord.conf` configuration file, find the line that says `#auth_strategy=keystone`. and change it to:

```
auth_strategy=noauth
```

Note: Please note that the `auth_strategy` line was commented out in the configuration file, but that when you change it to `noauth`, you'll need to make sure it doesn't begin with a `#` symbol. Also, please remember to turn authentication back on before running Ripcord in a production environment.

Now that we've configured Ripcord, we can start it. When starting the program, it blocks the current terminal, so we recommend running it inside of a `screen` session. (If you're not familiar with `screen`, it's a screen multiplexor that allows you to run more than one program from a single terminal window.) Let's start Ripcord in a screen window:

```
$ screen ripcord-api --config-file=etc/ripcord/ripcord.conf -v
```

If Ripcord has started correctly, you should see something like this at the bottom of your terminal window:

```
2014-03-10 18:14:49.465 7373 INFO ripcord.cmd.api [-] ********************************************
```

Since that terminal window is busy showing us the logs from Ripcord, let's use `screen` to create a second virtual terminal window, and do a quick double-check to make sure Ripcord is listening for network connections.

Simply press `CTRL-A` and then `C` to create a new virtual terminal window in screen. This should give you another shell prompt. You can then press `CTRL-A` and then `N` to cycle between the two virtual terminals.

In the new virtual terminal window, run the following to ensure that Ripcord is listening for network connections:

```
$ netstat --ltnp
```

You should see a line in the output like the following, showing that our python program (Ripcord in this instance) is listening for network connections on port `9869`:

```
tcp        0      0 0.0.0.0:9869            0.0.0.0:*               LISTEN      7373/python2.7
```

Once you've confirmed that Ripcord is listening for network connections, you can connect to Ripcord by opening a web browser and entering `http://127.0.0.1:9869` in the address bar and pressing enter.

You should get back an response that shows something similar to

```
404 Not Found
The resource could not be found.
```

This is the expected response at this time. At this point, you're ready to install the Ripcord command-line client.

To leave Ripcord running in the background and disconnect from the screen session, type `CTRL-A` and then `D`. You can later reconnect to your screen session by typing:

```
$ screen -x
```

Running unit tests

The unit tests will run by default inside the `tox` environment in the `.tox` directory. Run the unit tests by running the following command in the `ripcord` directory:

```
$ ~/.local/bin/tox
```

See *Unit Tests* for more details.

Contributing Your Work

Once your work is complete you may wish to contribute it to the project. Refer to HowToContribute for information. The Kickstand Project uses the Gerrit code review system. For information on how to submit your branch to Gerrit, see GerritWorkflow.

Unit Tests

Note: This document originates from the OpenStack Nova project. Since we also use the same toolset for testing, we can also use some of their documentaion. We have obviously made changes that only affect our project but we credit the OpenStack project for the original [2].

Ripcord contains a suite of unit tests, in the ripcord/tests directory.

Any proposed code change will be automatically rejected by the Kickstand Project Jenkins server if the change causes unit test failures.

Running the tests

Run the unit tests by doing:

```
tox
```

This script is a wrapper around the nose testrunner and the pep8 checker.

[2] See http://docs.openstack.org/developer/nova/devref/unit_tests.html

CHAPTER 3

Man pages

3.1 ripcord-api

3.1.1 Server for the Ripcord APIs

Author paul.belanger@polybeacon.com

Date March 19, 2014

Copyright PolyBeacon, Inc

Version

Manual section 1

SYNOPSIS

ripcord-api [options]

DESCRIPTION

ripcord-api is a server daemon that serves the ripcord APIs

OPTIONS

General options

FILES

- /etc/ripcord/ripcord.conf

SEE ALSO

- ripcord

BUGS

- Ripcord is sourced in Github so you can view current bugs at ripcord

3.2 ripcord-manage

3.2.1 Control and manage ripcord

Author paul.belanger@polybeacon.com

Date March 19, 2014

Copyright PolyBeacon, Inc

Version

Manual section 1

SYNOPSIS

ripcord-manage <category> <action> [<args>]

DESCRIPTION

ripcord-manage is a CLI tool to control ripcord

OPTIONS

General options

ripcord DB

`ripcord-manage db sync`

Sync the database up to the most recent version.

`ripcord-manage db version`

Print the current database version.

FILES

- /etc/ripcord/ripcord.conf

SEE ALSO

- ripcord

BUGS

- Ripcord is sourced in Github so you can view current bugs at ripcord

CHAPTER 4

Modules

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*