Delorean Documentation

Release 1.0

RDO Community

February 25, 2016

Contents

1 Introduction **3**

2 Installation **5**
2.1 Configuration 5
2.2 Configuring your httpd 6

3 Repositories **7**
3.1 Building new packages and repositories 7
3.2 Delorean repository: delorean-deps 7
3.3 Delorean repository: current 8
3.4 Delorean repository: consistent 8
3.5 Delorean repository: current-passed-ci 8

4 Usage **11**
4.1 Parameters 11
4.2 Initial build 11
4.3 Output and log files 12

5 Contributing **13**
5.1 Setting up a development environment in an OpenStack VM using cloud-init 13
5.2 Setting up a development environment manually 14
5.3 Submitting pull requests 15
5.4 Generating the documentation 15

6 Indices and tables **17**

Contents:

CHAPTER 1

Introduction

Delorean builds and maintains yum repositories following Openstack's upstream repositories.

CHAPTER 2

Installation

Installing prerequisites:

```
$ sudo yum install git createrepo python-virtualenv git-hg mock
```

Add the user you intend to run as to the mock group:

```
$ sudo usermod -a -G mock $USER
$ newgrp mock
$ newgrp $USER
```

If you want to serv the built packages and the status reports:

```
$ systemctl start httpd
```

Install Delorean:

```
$ pip install delorean
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv delorean
$ pip install delorean
```

The httpd module is not strictly required, Delorean does not use it. However, it will output it's results in a way that is suitable for a web-server to serve. This means you can easily set up a web-server to serve the finished `.rpm` and `.log` files.

2.1 Configuration

Configuration is done in an INI-file. An example file called `projects.ini` is included. The configuration file looks like this:

```
[DEFAULT]
datadir=./data
baseurl=https://trunk.rdoproject.org/
distro=rpm-master
smtpserver=
maxretries=3
target=centos
```

- `datadir` is the directory where the packages and repositories will be created.

- `baseurl` is the URL to the data-directory, as hosted by your web-server. Unless you are installing Delorean for local use only, this must be a publicly accessible URL.
- `distro` is the branch to use for building the packages.
- `target` is the distribution to use for building the packages (`centos` or `fedora`).
- `smtpserver` is the address of the mail server for sending out notification emails. If this is empty no emails will be sent out. If you are running Delorean locally, then do not set an smtpserver.
- `maxretries` is the maximum number of retries on known errors before marking the build as failed. If a build fails, Delorean will check the log files for known, transient errors such as network issues. If the build fails for that reason more than maxretries times, it will be marked as failed.

2.2 Configuring your httpd

The output generated by Delorean is a file structure suitable for serving with a web-server. You can either add a section in the server configuration where you map a URL to the data directories, or just make a symbolic link:

```
$ cd /var/www/html
$ sudo ln -s <datadir>/repos .
```

CHAPTER 3

Repositories

Delorean doesn't stop at building packages, it also generates yum repositories you can install the packages from.

Delorean repositories are all hosted on http://trunk.rdoproject.org.

This documentation goes through the various repositories and what they are used for.

3.1 Building new packages and repositories

Delorean watches upstream git repositories for new commits. When there is one, Delorean builds a new version of the project's package with the new commit.

On a successful build, Delorean will generate a new repository with the latest version of every package that successfully built.

A package build can fail due to different reasons, for example when a new dependency was introduced that needs to be added to the RPM spec file. If there is a build failure, no repository is generated and the project's package is not updated.

The package will not be updated for as long as it fails to build. This means that newer repositories generated from other projects' commits would not contain all the latest commits of the project that failed to build.

Delorean does not delete any generated repositories. This means we can use any previously built repositories if necessary.

Generated repositories are unique and each have their own hash. For example, you might be using the Delorean `/centos7/current/delorean.repo` repository but in fact this corresponds to `/centos7/42/0c/420c638d6325d1ccf50eb5fe430c5d255dcbfb94_52cbbfe7`.

Delorean manages these references as simple symbolic links for the `current` and `consistent` repositories. The `current-passed-ci` repository is a symbolic link managed automatically by RDO's continuous integration pipeline and is not managed or known by Delorean itself.

3.2 Delorean repository: delorean-deps

OpenStack projects are typically built into the Delorean repositories. These projects require dependencies that Delorean does not build, for example python-requests, python-prettytable and so on.

The RDO project provides a mirror which contains all of these dependencies and the repository configuration is available at `/delorean-deps.repo` for each release.

For example:

- Trunk: http://trunk.rdoproject.org/centos7/delorean-deps.repo
- Liberty: http://trunk.rdoproject.org/centos7-liberty/delorean-deps.repo

3.3 Delorean repository: current

On a successful build, Delorean will generate a new repository with the latest version of every package that successfully built.

This new repository will be tagged as `current`.

A Delorean current repository might not contain all the latest upstream commits if there are any ongoing build failures that are unresolved.

This repository is available at `/current/delorean.repo` for each release.

For example:

- Trunk: http://trunk.rdoproject.org/centos7/current/delorean.repo
- Liberty: http://trunk.rdoproject.org/centos7-liberty/current/delorean.repo

3.4 Delorean repository: consistent

Delorean `consistent` repositories are generated for any given set of packages that have no current build failures.

These repositories have the latest and greatest of every package and all upstream commits have been successfully built up until that point.

The continuous integration done to test RDO packages target the Delorean consistent repositories.

This repository is available at `/consistent/delorean.repo` for each release.

For example:

- Trunk: http://trunk.rdoproject.org/centos7/consistent/delorean.repo
- Liberty: http://trunk.rdoproject.org/centos7-liberty/consistent/delorean.repo

3.5 Delorean repository: current-passed-ci

The RDO project has a continuous integration pipeline that consists of multiple jobs that deploy and test OpenStack as accomplished by different installers.

This vast test coverage attempts to ensure that there are no known issues either in packaging, in code or in the installers themselves.

Once a Delorean consistent repository has undergone these tests successfully, it will be promoted to `current-passed-ci`.

current-passed-ci represents the latest and greatest version of RDO trunk packages that were tested together successfully.

We encourage installer projects and users of RDO to use this repository to keep up with trunk while maintaining a certain level of stability provided by RDO's CI.

This repository is available at `/current-passed-ci/delorean.repo` for each release.

For example:

- Trunk: http://trunk.rdoproject.org/centos7/current-passed-ci/delorean.repo
- Liberty: http://trunk.rdoproject.org/centos7-liberty/current-passed-ci/delorean.repo

CHAPTER 4

Usage

4.1 Parameters

```
usage: delorean [-h] --config-file CONFIG_FILE [--info-repo INFO_REPO]
                [--build-env BUILD_ENV] [--local] [--head-only]
                [--package-name PACKAGE_NAME] [--dev] [--log-commands]
                [--use-public] [--order]
arguments:
  -h, --help            show this help message and exit
  --config-file CONFIG_FILE
                        Config file (required)
  --info-repo INFO_REPO
                        use a local rdoinfo repo instead of fetching the
                        default one using rdopkg.
  --build-env BUILD_ENV
                        Variables for the build environment.
  --local               Use local git repos if possible
  --head-only           Build from the most recent Git commit only.
  --package-name PACKAGE_NAME
                        Build a specific package name only.
  --dev                 Don't reset packaging git repo, force build and add
                        public master repo for dependencies (dev mode).
  --log-commands        Log the commands run by delorean.
  --use-public          Use the public master repo for dependencies when doing
                        install verification.
  --order               Compute the build order according to the spec files
                        instead of the dates of the commits.
  --version             show program's version number and exit
```

4.2 Initial build

Some of the projects require others to build. As a result, use the special option `--order` to build in the order computed from the BuildRequires and Requires fields of the spec files. If this option is not specified, Delorean builds the packages in the order of the timestamps of the commits.

```
$ delorean --config-file projects.ini --order
```

4.3 Output and log files

The output of Delorean is generated in the `<datadir>/repos` directory. It consists of the finished `.rpm` files for download, located in `/repos/current`, and reports of the failures in `/repos/status_report.html`, and a report of all builds in `/repos/report.html`.

CHAPTER 5

Contributing

5.1 Setting up a development environment in an OpenStack VM using cloud-init

The following cloud-config script can be passed as a –user-data argument to *nova boot*. This will result in a fully operational delorean environment to hack on.

```
#cloud-config
disable_root: 0
groups:
  - docker: [root]
package_upgrade: true
packages:
  - vim
  - docker-io
  - git
  - python-pip
  - git-remote-hg
  - git-hg
  - python-virtualenv
  - httpd
  - gcc
  - createrepo
  - screen
  - python3
  - python-tox
  - git-review
write_files:
  - content: |
        #!/bin/bash
        setenforce 0
        systemctl enable docker.service
        systemctl start docker.service
        systemctl enable httpd.service
        systemctl start httpd.service
        cd ~
        git clone https://github.com/openstack-packages/delorean
        cd ~/delorean
        virtualenv .venv
        source .venv/bin/activate
        pip install -r requirements.txt
        pip install -r test-requirements.txt
```

```
        python setup.py develop
        chcon -t docker_exec_t scripts/*
        scripts/create_build_image.sh
        FLOAT=`curl http://169.254.169.254/latest/meta-data/public-ipv4`
        sed -i s:trunk.rdoproject.org:$FLOAT: ~/delorean/projects.ini
        sed -i s:./data:/root/delorean/data: ~/delorean/projects.ini
        chmod +x /root
        ln -s /root/delorean/data/repos /var/www/html/
        delorean --config-file ~/delorean/projects.ini
        sqlite3 ~/delorean/commits.sqlite < ~/fix-fails.sql
        echo "*/5 * * * * root /root/run.sh" >> /etc/crontab
    path: /root/setup.sh
    permissions: 0744

  - content: |
        #!/bin/bash
        LOCK='/root/delorean.lock'
        set -e

        exec 200>$LOCK
        flock -n 200 || exit 1

        source ~/delorean/.venv/bin/activate
        LOGFILE=/var/log/delorean/run.$(date +%s).log
        cd ~/delorean

        echo `date` "Starting delorean run." >> $LOGFILE
        delorean --config-file ~/delorean/projects.ini 2>> $LOGFILE
        echo `date` "Delorean run complete." >> $LOGFILE
    path:  /root/run.sh
    permissions: 0744

  - content: |
        delete from commits where status == "FAILED";
    path:  /root/fix-fails.sql
    permissions: 0644

  - content: |
        [user]
                email = test@example.com
                name = Tester Testerson
    path: /root/.gitconfig
    permisssions: 0664

runcmd:
  - mkdir /var/log/delorean
  - script -c "/root/setup.sh" /var/log/delorean/setup.log

final_message: "Delorean installed, after $UPTIME seconds."
```

5.2 Setting up a development environment manually

Installing prerequisites:

```
$ sudo yum install docker-io git createrepo python-virtualenv git-hg
$ sudo systemctl start httpd
```

```
$ sudo systemctl start docker
```

Add the user you intend to run as to the docker group:

```
$ sudo usermod -a -G docker $USER
$ newgrp docker
$ newgrp $USER
```

Checkout the Source code and install a virtualenv:

```
$ git clone https://github.com/openstack-packages/delorean.git
$ cd delorean
$ virtualenv .venv
$ source .venv/bin/activate
$ pip install -r requirements.txt
$ pip install -r test-requirements.txt
$ python setup.py develop
```

5.3 Submitting pull requests

Pull requests submitted through GitHub will be ignored. They should be sent to GerritHub instead, using git-review. Once submitted, they will show up here:

https://review.gerrithub.io/#/q/status:open+and+project:openstack-packages/delorean

5.4 Generating the documentation

Please note that the Master Packaging Guide also contains instructions for Delorean. If you modify the documentation, please make sure the Master Packaging Guide is also up to date. The source code is located at https://github.com/redhat-openstack/openstack-packaging-doc/blob/master/doc/rdo-packaging.txt .

The documentation is generated with Sphinx. To generate the documentation, go to the documentation directory and run the make file:

```
$ cd delorean/doc/source
$ make html
```

The output will be in delorean/doc/build/html

CHAPTER 6

Indices and tables

- genindex
- modindex
- search