Starforge Documentation

Release 0.4.0.dev0

Galaxy Project and Community

Apr 25, 2018

Contents

Contents:

Starforge: Build Galaxy things in virtualization

Things you can do with Starforge:

- Build Galaxy Tool Shed dependencies
- Build Python Wheels (e.g. for the Galaxy Wheels Server)
- Rebuild Debian or Ubuntu source packages (for modifications)

These things will be built in Docker. Additionally, wheels can be built in QEMU/KVM virtualized systems.

Documentation can be found at starforge.readthedocs.org

Starforge is maintained by the Galaxy Project and community. A list of contributors to the project can be found on GitHub.

CHAPTER 1

Quick Start

For all types of builds, begin by installing Docker.

1.1 Tool Shed Dependencies

There are two scripts that can be used, depending on the package recipes available:

```
$ ./build.sh <package>
$ python build.py <package> --version 1.0
```

`build.sh` is the older format, and simply uses a single `<package>build.sh` file, like Atlas. *build.py* is the newer format, and uses yaml metadata in `<package>/<version>/build.yml`.

The base image for Galaxy packages is Debian Squeeze. This will hopefully produce binaries usable on Galaxy's targeted platforms (at time of writing: CentOS 6+, Debian 6.0+, Ubuntu 12.04+).

```
$ ./build galaxy <package>
$ python build.py <package>
```

To build packages against a different OS, you can use the *–image* flag, e.g.:

```
$ ./build <dist>[:tag] <package>
$ python build.py <package> --image <dist>[:tag]
```

e.g.

```
$ ./build ubuntu:trusty nginx
$ python build.py nginx --image debian:squeeze
```

Building all the things:

There's a separate `build-all.sh` which allows you to build all of the packages using their preferred build mechanism

1.1.1 Notes on the two build scripts

build.py

The `<version>` option is optional, and defaults to the string `'default'`, which is useful for recipes that don't have version specific changes (E.g. bcftools 1.0 builds identically to 1.2)

1.2 Python Wheels

Starforge can build both pure Python and C-extension Python modules into wheels for Linux under Docker and for Mac OS X under QEMU/KVM. To do this, you'll want to install Starforge (preferably in a Python virtualenv) using `pip install starforge` (to install from PyPI) or `python setup.py install` to install from the source.

Docker (and QEMU) images to use are specified in starforge/config/default.yml. To modify this file, copy it to `$XDG_CONFIG_HOME/galaxy-starforge/config.yml` (`$XDG_CONFIG_HOME` is, by default `~/.config`). The sample file wheels/build/wheels.yml is used to determine what wheels can be built and their rules for building. To use this file, use the `--wheels-config` argument to `starforge wheel` or copy `wheels.yml` to `$XDG_CONFIG_HOME/galaxy-starforge/wheels.yml`.

Wheels can be built using `starforge wheel <package>`, e.g.:

```
$ starforge wheel pycrypto
$ starforge wheel --no-qemu pysam   # only build on docker
```

See the output of `starforge --help` for help using the Starforge command-line interface.

1.2.1 Pull Request wheel builder

Pull requests to the Starforge repository on Github that modify wheels/build/wheels.yml can automatically be built for all specified platforms on a dedicated Starforge build server by the Galaxy Jenkins service. To utilize, modify wheels.yml as appropriate and create a pull request. Any member of the Galaxy Committers group can then authorize Jenkins to initiate the build. If it fails, you can modify the pull request and further builds can be triggered.

1.2.2 Notes on images

Linux

Images used to build wheels are uploaded to the Starforge Docker Hub repo and will be pulled as necessary. Typically you will only use the manylinux1-wheel and manylinux1-32-wheel images, which are manylinux CentOS 5-based images that will usually produce wheels usable on all Galaxy-supported platforms.

You can also produce "platform-specific" wheels by using the `full-wheel` imageset. This is useful if you want to link to distribution-specific system versions of non-standard libraries without bundling them in to the wheel.

Mac OS X

Mac OS X images are not provided due to legal reasons. Consult the *Building on Mac OS X under QEMU/KVM* documentation for details.

CHAPTER 2

Building on Mac OS X under QEMU/KVM

Due to legal reasons, a Mac OS X image can not be provided. However, an Ansible playbook is available at wheels/image/osx-playbook.yml. that can be used to perform most of the image bootstrapping.

The Starforge developers make no claims as to the legality of virtualizing Mac OS X under QEMU/KVM. Common readings of the Mac OS X license suggest that virtualization of Mac OS X on Linux is legal as long as the underlying hardware is an Apple computer. However, as with the rest of the Starforge project, the authors are not liable for any claim, damages or other liability, as laid out in the Starforge License.

Once the image is available (and configured in `config.yml`), Starforge will use Btrfs to create snapshots of the image. Thus, you will need to store the Mac OS X image in a btrfs subvolume on the host OS.

2.1 Configuring the host OS

TODO

2.2 Creating the image

TODO

2.3 Configuring the image

TODO

2.4 Configuring Starforge

TODO

CHAPTER 3

(Re)building Debian Packages

Starforge can be used to build or rebuild Debian or Ubuntu source packages with very little effort. These are suitable for use on a private APT repository or Ubuntu PPA.

The only example of this Starforge build type currently in the repository is the nginx package, which is modified to include the nginx upload module.

How to build packages:

1. Create a build recipe. You can use the nginx package as an example.
2. Run the appropriate build method (`build` or `build.py`).
3. Packages and related artifacts will be output to the package directory. From here, on the (Debian-based) host system, you can sign the packages using:

```
% debsign -S <pkg>-<version>_source.changes
```

4. Upload to a PPA with:

```
% dput ppa:<owner>/<repo> <pkg>-<version>_source.changes
```

CHAPTER 4

History

4.1 0.4.0.dev0

4.2 0.3.5 (2017-10-01)

- Support xz/lzma tarballs for wheel builds Pull Request 166

4.3 0.3.4 (2017-09-12)

- Native support for auditwheel and delocate. (#160)

4.4 0.3.3 (2017-09-08)

- Do not build sdists with the *wheel* subcommand by default. (#155)

4.5 0.3.2 (2017-09-08)

- Fix a bug where the wrong working directory was set when building wheels with multiple sources. (#154)

4.6 0.3.1 (2017-09-08)

- Fix a bug with `sudo` and `brew install` on macOS. (#151).
- Short circuit platform caching on OS X (#150).

4.7 0.3 (2017-01-10)

- Drop the dependency on the "Galaxy" wheel fork, which makes installation much easier. "Platform-specific" wheels can still be built.
- When building Docker images, install Starforge from the local source instead of installing from PyPI or Github.

4.8 0.2.1 (2016-05-27)

- Do a case-insensitive comparison for cached tarball names (uWSGI's project name is `uWSGI` but its source tarballs are named `uwsgi-*`). 7672547

4.9 0.2 (2016-05-19)

- Added support for building manylinux1 wheels. 0dbecb7

4.10 0.1.1 (2016-01-20)

- Only running prebuild during wheel builds (and not sdists) was too naive, since this prevents changing the version number of sdists in the prebuild action (a common use of the prebuild action). Instead, allow for separate `wheel`, `sdist`, and `all` prebuild actions. Reverts the behavior of 9008c57. Issue 64
- Install Galaxy pip from Github instead of wheels.galaxyproject.org so that Starforge images can be built with new versions of Galaxy pip before they are released. 97b4ba4

4.11 0.1 (2016-01-12)

- Reimplemented the wheel building scripts as a library and `starforge` command line
- Wrote some documentation

4.12 Older than 0.1

Originally Galaxy docker-build and later renamed Starforge, but as a collection of disjointed shell scripts, Python, and YAML used to build Galaxy Tool Shed dependencies, as well as rebuilding Debian and Ubuntu source packages with modifications (which itself came from a project created to do the same via Vagrant and Ansible called vadebuildsible).

CHAPTER 5

Indices and tables

- genindex
- modindex
- search