Dry Dock Documentation

Release 0.6.11

Taylor "Nekroze" Lawson

December 19, 2014

Contents

A Docker cluster construction utility.

DryDock takes a simple (YAML) specification file then can construct and configure a cluster of Docker containers. DryDock will automatically setup a reverse proxy, exposure of ports, and even persistent storage to allow for easy future upgrading by simply rebuilding the DryDock specification!

CHAPTER 1

Features

- Simple YAML configuration.
- Automatic Docker cluster provisioning/configuration
- Nginx reverse proxy configuration with HTTPS/SSL support
- Easy setup for persistent volumes.
- Share your DryDock specifications with the world.
- Container supervisor utilizing the Docker API.

CHAPTER 2

TODO

- Provide a better update path, specific update commands, custom pre-post.
- Container monitoring, ensure a container is working else reconstruct it.
- Better support for stateless-ness and volumes
- Better unittest coverage.
- Ability to provide a custom docker images repository for master containers.
- An external facing DNS server or config generation.
- Control over a master container cluster, Drydock as a Service?.
- Web interface for control, logging and monitoring. Far future.

2.1 Contents:

2.1.1 Installation

At the command line either via easy_install or pip:

```
$ easy_install drydock
$ pip install drydock
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv drydock
$ pip install drydock
```

2.1.2 Usage

DryDock has multiple functions, the end goal of which is to setup and configure a cluster of Docker containers.

Master

The `master` command will prepare and provide the command for a new Docker container under the given name based upon `nekroze/drydock` . This container is setup and ready to use **DryDock** to run a cluster of Docker containers in a Docker container (dind). This is designed to easily contain a **DryDock** cluster but is not required.

By default the master container will take over the host ports; 80, 443, for; HTTP, and HTTPS. This can be customized by providing the root specification a `http_port` and `https_port` respectively. If the specification describes any exposed ports for sub-containers the external ports for those will also be exposed through the master container.

Once a master container has been prepared **DryDock** will provide the user with the command required to run that specific cluster. It may be worth adding the docker `--rm=true` switch to the provided command. using this in an upstart script for example would mean that each boot or restart of the master container will be fresh from the original image that was created. If you wish to run it in the background immediately then the `-d` do that for you.

Note: Setting up a **DryDock** master container is entirely optional.

Warning: The resulting master container runs in `--privileged` mode and retains all security concerns of such a Docker container.

Prepare

This command will setup a few Docker containers, generate an ssl certificate, and must be run before running `construct` on a specification.

The following containers will be setup and run:

1. `skydns`: Small dns server.
2. `skydock`: Docker skydns registry.
3. `nginx`: An nginx powered reverse proxy container.

The `nginx` container will have a volume mapped to the hosts `/etc/nginx/sites-enabled` directory for the matching `nginx` directory.

Construct

The main function for **DryDock**, `construct`, takes a YAML specification file and will create the required configuration files (supervisor, and nginx) before running and naming containers as defined in the specification.

Here is an example of a **DryDock** specification file that will construct `nekroze.com` with *wordpress* and *gitlab* available at `blog.nekroze.com` and `lab.nekroze.com`. Finally the config describes a special `root` container that serves the root of the domain, in this case `nekroze.com` gets passed to the `root` sub-container running *drupal*.

```
name: nekroze.com-1
domain: nekroze.com

subcontainers:

  - name: blog
    base: skxskx/wordpress
    exposed_ports:
        2222: 22
    http_port: 80
    volumes: [/var/lib/mysql]

  - name: lab
    base: crashsystems/gitlab-docker
    exposed_ports:
        22: 22
    http_port: 80
```

```
- name: root
  base: moul/drupal
  exposed_ports:
      2221: 22
  http_port: 80
```

The YAML specification file consists of two main parts; cluster information, and container specification. Together these define a *DryDock Specification* which gets constructed into running Docker containers and accompanying configuration files!

All peristant data should be stored in `/var/lib/{domain}/` where domain is the specifications domain

Note: This command assumes that both Docker and supervisor are currently installed on the system.

Pull

This command is used almost identically to the `construct` and `deconstruct` command and will download all of the base images in the given specification. This can be useful for caching and or testing.

Supervise

This is a supervisor for a **DryDock** container cluster and can be used to ensure sub-container uptime. The `supervise` command can be run without running prepare or even construct as it will maintain and construct containers as needed.

Note: When using a master container, its default entrypoint command will be to supervise the given specification.

Deconstruct

The `deconstruct` command is used the same way as the `construct` command, however it will remove any thing created by the corresponding `construct` command.

2.1.3 Specification Reference

Information fields are displayed as follows:

Optional field with default value

`info`: `default`: description

Required field

`info`: description

DryDock Specification

DryDock can specify any of the of following fields (some of which may overwrite any sub-containers corresponding field).

`name`: name of resulting Docker container.

`domain`: domain that all sub containers will server unless specified otherwise.

`subcontainers`: a list of sub-container specifications.

While technically any sub-container fields can be specified in the top level **DryDock** specification their behaviour is either unspecified, undefined, or untested.

Sub-Container Specification

Any container can define the following information.

`name`: name of resulting Docker container.

`base`: `ubuntu`: base image for container.

`domain`: `none`: domain that subcontainers will serve.

`specification: ''none`: a link to a container specification that will be used as the base for this container. Any fields defined in this specification will override the given link's specification. With this docker container developers can provide its own specification and leave the implementation fields (ie, domain) to the end user.

`command`: `none`: if provided this command will be passed to the new container for its first time setup.

`envs`: `none`: map of variable names and values to pass to the container.

`external`: `Yes`: if `No` container will allow only lan connections.

`exposed_ports`: `none`: a set of external to internal port maps.

`http_port`: `80`: port that serves http for the reverse proxy to point to.

`https_port`: `443`: port that serves https for the reverse proxy to point to.

`volumes`: `none`: list of paths to be externally available under `/var/lib/{domain}/{containername}/{volume}`.

`data: No`: if `Yes` this container will have an additional volume map from the hosts `/var/lib/{domain}/drydock/data` to the container path `/mnt/data`.

2.1.4 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/Nekroze/drydock/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

Dry Dock could always use more documentation, whether as part of the official Dry Dock docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/Nekroze/drydock/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *drydock* for local development.

1. Fork the *drydock* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/drydock.git
   ```

3. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass style and unit tests, including testing other Python versions with tox:

   ```
   $ tox
   ```

To get tox, just pip install it.

5. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/Nekroze/drydock under pull requests for active pull requests or run the `tox` command and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test test/test_drydock.py
```

2.1.5 Credits

Development Lead

- Taylor "Nekroze" Lawson <nekroze@eturnilnetwork.com>

Contributors

None yet. Why not be the first?

2.1.6 History

0.6.11 (10-08-2014)

- Fixed: Improved environment variable handling

0.6.12 (19-06-2014)

- Changed: increased nginx upload limit for flexability

0.6.10 (19-06-2014)

- Changed: pass /etc/localtime to containers rather then /etc/timezone

0.6.9 (18-06-2014)

- Added: `supervise` now prints out each system command before calling it.

0.6.8 (14-06-2014)

- Fixed: ssl key generation commands now work and produce strong keys.

0.6.7 (14-06-2014)

- Removed: `prepare` no longer constructs base containers.

0.6.6 (14-04-2014)

- Fixed: Missing semicolon line ending for nginx https.

0.6.5 (14-04-2014)

- Added: Full options to start and supervise to allow for custom dns settings.

0.6.4 (31-03-2014)

- Changed: Altered the HTTPS/SSL reverse proxy config to hopefully work.

0.6.3 (23-03-2014)

- Fix: stale NFS file handles should no longer crash the supervisor.

0.6.2 (18-03-2014)

- Fix: remove conflicting args for containers `-d` and `--rm`.

0.6.1 (17-03-2014)

- Fix: detect networking before running `supervise`.

0.6.0 (17-03-2014)

- Changed: `supervise` now prepares and constructs all containers for a

DryDock cluster and will re-create them on failure. * Changed: After running `master` two command will be provided, one to pull all required images and first time setup, another to run the **DryDock** cluster.

0.5.15 (17-03-2014)

- Removed: got rid of master containers constructing special images.
- Changed: the master command just prepares the files and gives you a run

command for the master. * Added: a volume map from the internal cluster containers based on FQDN and the master containers `/var/lib/{name}` persistance directory.

0.5.14 (17-03-2014)

- Changed: `supervise` will no longer reconstruct nginx at start.
- Changed: moved full master and cluster container construction to

construction of master image not first time run run. * Changed: master images will now have `-master` appended to the end of their name * Changed: master images will now be named based on the specification name not domain, as will their location in `/var/lib/...` * Fix: Attempted to provide a better final run command for the master image.

0.5.13 (17-03-2014)

- Changed: only construct master cluster when supervising.

0.5.12 (17-03-2014)

- Added: new master container scripts to manage drydock cluster.

0.5.11 (16-03-2014)

- Fix: disabled nginx event config.

0.5.10 (16-03-2014)

- Fix: properly wrap final master command in qoutes for bash.

0.5.9 (16-03-2014)

- Fix: each image only gets pulled once via the pull command.
- Fix: more specific nginx config for use with skydock.

0.5.8 (16-03-2014)

- Changed: The shared data directory to map

`/var/lib/{domain}/drydock/data` to the containers `/mnt/data`. * Added: Shared data container now works for master containers.

0.5.7 (16-03-2014)

- Changed: Moved master container prepare to first time run.

0.5.6 (16-03-2014)

- Fix: only create nginx configs where needed.

0.5.5 (16-03-2014)

- Changed: master containers only pull on creation and will construct at

first run.

0.5.4 (16-03-2014)

- Fix: volume mapped master containers drydock volume for persistance.

0.5.3 (16-03-2014)

- Added: display of final master container run command to end user.

0.5.2 (16-03-2014)

- Fix: master containers nginx sites are now read write.

0.5.1 (15-03-2014)

- Added: Customize your network interfaces, ips, and dns from cli.
- Fix: cleaned up template storage to allow manipulation.

0.5.0 (15-03-2014)

This release has a major emphasis on the `master` command and containers and is released early to facilitate testing.

- Changed: `master` command now takes a specification to automatically

build it. * Removed: any usage of `supervisord` in favor of the new `drydock supervisor`.

0.4.3 (12-03-2014)

- Fix: use `--name` for naming containers for future proofing.
- Fix: use `--dns` for future proofing.

0.4.2 (11-03-2014)

- Fix: allow self connections to the host when nginx blocks external.

0.4.1 (09-03-2014)

- Fix: check for config files before removing them.
- Fix: `supervise` command will now recreate the nginx container each run.

0.4.0 (08-03-2014)

- Added: `supervise` command line command. DryDock has its own supervisor!
- Added: `start` and `stop` command line commands.
- Added: `data` in subcontainer specification maps volumes at

`/mnt/drydock`. * Fix: All containers are passed their FQDN as their hostname * Fix: `pull` command also grabs the containers required for the `prepare` command. * Fix: Pass host timezone to subcontainers. * Fix: volumes now go map to `/var/lib/{domain}/{name}/`.

0.3.0 (28-02-2014)

- Added: reports at the end of running all the major commands.
- Added: `envs` to specification for environment variable definitions.
- Added: `command` to specification for run command definition.
- Added: `pull` command to download all images required for the

specification. * Added `specification` to specification for external specification links.

0.2.0 (25-02-2014)

- Added: supervisor config writing is now an option.
- Added: deconstruct command to remove a specification.

0.1.0 (25-02-2014)

- First release on PyPI.

2.2 Feedback

If you have any suggestions or questions about **DryDock** feel free to email me at nekroze@eturnilnetwork.com.

If you encounter any errors or problems with **DryDock**, please let me know! Open an Issue at the GitHub http://github.com/Nekroze/drydock main repository.