Deck Chores Documentation

Release 0.2

Frank Sachsenheim

Jan 22, 2018

Contents

Contents:

CHAPTER 1

Usage

1.1 Invocation

Usually you would run *deck-chores* in a container:

```
$ docker run --rm -v /var/run/docker.sock:/var/run/docker.sock funkyfuture/deck-chores
```

Note: There's a manifest on the Docker Hub that maps images to builds targeting `amd64` and `arm` architectures. Thus you don't need to specify any platform indicator, the Docker client will figure out which one is the proper image to pull.

Likewise, docker-compose can be used with such configuration:

```
version: '2'

services:
  officer:
    image: funkyfuture/deck-chores
    restart: unless-stopped
    environment:
      TIMEZONE: Asia/Tel Aviv
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
```

You could also install *deck-chores* from the Python Package Index with `pip` or `pipsi` (recommended):

```
$ pipsi install deck-chores
```

and then run it:

```
$ deck-chores
```

Now one instance of *deck-chores* is running and will handle all job definitions that it discovers on containers that run on the Docker host.

1.2 Caveats & Tips

Caution: There's yet no way to distinguish container events that happen during an **image build** from others (#6 and #15211). Thus when an image is built, *deck-chores* will register and remove jobs on all intermediate containers following labels that define jobs. It would possibly trigger these jobs, which might lead to a corrupted build. You can avoid this risk by building images on a host that is not observed by *deck-chores* or by pausing it during image builds.

1.2.1 Containers without an enduring main process

If the container is supposed to only run the scheduled commands and not a main process, use a non-stopping no-op command as main process like in this snippet of a `docker-compose.yml` file:

```
services:
  neverending:
    # ...
    command: >
      tail -f /dev/null
    labels:
      deck-chores.short.command: daily_command ...
      deck-chores.short.interval: daily
```

1.3 Job definitions

Job definitions are parsed from a container's metadata aka labels. A label's key must be in the namespace defined by *LABEL_NAMESPACE* (default: `deck-chores`) to be considered. A job has an own namespace that holds all its attributes. Thus an attribute's key has this schema:

```
$LABEL_NAMESPACE.<job name>.<job attribute>
```

The *job name* `options` cannot be used as it is reserved for setting *Container options*.

The following attributes are available:

Attribute	Description
command	the command to run
cron	a *cron* definition
date	a *date* definition
interval	a *interval* definition
max	the maximum of simultaneously running command instances, defaults to *DEFAULT_MAX*
timezone	the timezone that *cron* and *date* relate to, defaults to *TIMEZONE*
user	the user to run the command, defaults to *DEFAULT_USER*

The attribute `command` and one of `cron`, `date` or `interval` are *required* for each job.

Example snippet from a `docker-compose.yml` file:

```
services:
  web:
    # ...
    labels:
      deck-chores.clear-caches.command: drush cc all
      deck-chores.clear-caches.interval: daily
      deck-chores.clear-caches.user: www-data
```

Or baked into an image:

```
LABEL deck-chores.clear-caches.command="drush cc all" \
      deck-chores.clear-caches.interval="daily" \
      deck-chores.clear-caches.user="www-data"
```

1.4 Job triggers

1.4.1 cron

cron triggers allow definitions for repeated run times like for the well-known *cron* daemon. In contrast to the classic, the sequence of fields is flipped, starting with the greatest unit on the left. The fields are separated by spaces, missing fields are filled up with `*` on the left.

The fields from left to right define:

- `year`
- `month`
- `day` (of month)
- `week` (of year)
- `day_of_week`
- `hour`
- `minute`
- `second`

See APScheduler's documentation for details on its versatile expressions.

Examples

```
* * * * * */3 0 0  # run on all hours dividable by 3
*/3 0 0            # as shortened expression
* * * * 6 1 0 0    # run every Sunday at 1:00
6 1 0 0            # as shortened expression
sun 1 0 0          # as 'speaking' variant
* * * * * 1-4 0 0  # run daily at 1:00, 2:00, 3:00 and 4:00
1-4 0 0            # as shortened expression
```

1.4.2 date

A one-time trigger that is formatted as `YYYY-MM-DD [HH:MM:SS]`.

An omitted time is interpreted as `0:00:00`. Note that times must include a seconds field.

1.4.3 interval

This trigger defines a repetition by a fixed interval. The interval is added up by the fields *weeks*, *days*, *hours*, *minutes* and *seconds*. Possible field separators are `.`, `:`, `/` and spaces. Missing fields are filled up with `0` on the left.

Examples

```
42:00:00      # run every fourty-two hours
100/00:00:00  # run every one hundred days
```

There are also the convenience shortcuts `weekly`, `daily`, `hourly`, `every minute` and `every second`.

1.5 Container options

Option flags control *deck-chores*'s behaviour with regard to the labeled container and override the setting of *DEFAULT_OPTIONS*. The schema for an option label name is:

```
$LABEL_NAMESPACE.options
```

Options are set as comma-separated list of flags. An option set by *DEFAULT_OPTIONS* can be unset by prefixing with `no`.

These options are available:

`image`
: Job definitions in the container's basing image labels are also parsed while container label keys override these.

`service`
: Restricts jobs to one container of those that are identified with the same service.

  See *SERVICE_ID_LABELS* regarding service identity.

1.6 Environment variables

deck-chore's behaviour is defined by these environment variables:

`CLIENT_TIMEOUT`
: The timeout for responses from the Docker daemon in seconds without unit indicator. The default is imported from *docker-py*.

`DOCKER_HOST`
: default: `unix://var/run/docker.sock`

  The URL of the Docker daemon to connect to.

`DEBUG`
: default: `no`

  Log debugging messages, enabled by `on`, `true` or `yes`.

DEFAULT_MAX
default: `1`

The default for a job's `max` attribute.

DEFAULT_OPTIONS
default: `image,service`

The default for a job's *options* attribute.

DEFAULT_USER
default: `root`

The default for a job's `user` attribute.

LABEL_NAMESPACE
default: `deck-chores`

The label namespace to look for job definitions.

LOG_FORMAT
default: `{asctime}|{levelname:8}|{message}`

Pattern that formats log record attributes.

SERVICE_ID_LABELS
default: `com.docker.compose.project,com.docker.compose.service`

A comma-separated list of container labels that identify a unique service with possibly multiple container instances. This has an impact on how the *`service`* option behaves.

TIMEZONE

default: `UTC`

The job scheduler's timezone and the default for a job's `timezone` attribute.

1.6.1 TLS options

ASSERT_HOSTNAME
default: `no`

Enabled by `on`, `true` or `yes`.

SSL_VERSION
default: `TLS` (selects the highest version supported by the client and the daemon)

For other options see the names provided by Python's ssl library prefixed with `PROTOCOL_`.

Authentication related files are expected to be available at `/config/ca.pem`, `/config/cert.pem` respectively `/config/key.pem`.

CHAPTER 2

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:

2.1 Types of Contributions

2.1.1 Report Bugs

If you run into problems, make sure you are running the latest image and run it with *DEBUG* set to `true`.

Report bugs at https://github.com/funkyfuture/deck-chores/issues.

If you are reporting a bug, please include:

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

2.1.2 Fix Bugs

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

2.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

2.1.4 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/funkyfuture/deck-chores/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 :)

2.2 Get Started!

Ready to contribute? Here's how to set up *deck-chores* for local development.

1. Fork the *deck-chores* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/deck-chores.git
```

3. Install your local copy into a virtualenv. Assuming you have pew installed, this is how you set up your fork for local development:

```
$ cd deck-chores
$ pew new -p $(which python) -a $(pwd) deck-chores
$ python setup.py develop
```

4. Create a branch for local development:

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

   Now you can make your changes locally.

5a. When you're done making changes, check that your changes pass flake8 and the tests:

```
 $ tox


To get flake8 and pytest, just pip install them into your virtualenv.
```

5b. If you want to run a container for testing purposes:

```
$ make run-dev
```

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

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push
```

7. Submit a pull request through the GitHub website.

2.3 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.

CHAPTER 3

History

3.1 0.2 (2018-02-23)

- *new*: documentation how to run scheduled jobs only (#25 by @binnisb)
- *fix*: documentation on cron triggers (#27 by @alpine-digger)

3.2 0.2-rc3 (2017-12-23)

- *changed*: arm builds base on python:3.6-alpine that are executed on an ARMv7l architecture
- *changed*: Updated dependencies *APScheduler* and *docker-py*

3.3 0.2-rc2 (2017-08-05)

- *changed*: arm builds base on arm32v6/python
- *changed*: therefore `arm32v6` replaces the `arm`-suffix in image tags
- *changed*: there are no more images that get tagged with `latest-$architecture`

3.4 0.2-rc1 (2017-07-01)

- *refactoring*: uses the Python Docker SDK 2 (#14)
- *removed*: `ASSERT_FINGERPRINT` environment variable
- *renamed*: `DOCKER_DAEMON` to `DOCKER_HOST` to comply with the SDK
- *fix*: check on fixed labels (#18 by @aeri4list)

- documentation updates

3.5 0.1 (2017-03-02)

- *fix*: docker-py returns `None` for labels of images that were created with older Docker versions (#7)

3.6 0.1.beta3 (2017-01-22)

- *new*: there's now a build for arm architectures
- *new*: an architecture agnostic manifest is pushed to the image registry for release images

3.7 0.1.beta2 (2016-12-08)

- *new:* set log format per :envvar:LOG_FORMAT
- *new:* an options label to set behavioural flags
- *new:* containers can be identified as a service by configurable labels
- *new:* job definitions for further containers of a service are ignored (default, opt-out can be configured)
- *new:* image labels can also be parsed for job definitions (default, opt-out can be configured)

3.8 0.1.beta1 (2016-12-04)

- First release with full documentation

CHAPTER 4

deck-chores

A job scheduler for Docker containers, configured via container labels.

- Documentation: https://deck-chores.readthedocs.io
- Image repository: https://hub.docker.com/r/funkyfuture/deck-chores
- Code repository: https://github.com/funkyfuture/deck-chores
- Issue tracker: https://github.com/funkyfuture/deck-chores/issues
- Free software: ISC license

4.1 Features

- define regular jobs to run within a container context with container and optionally with image labels
- use date, interval and cron-like triggers
- set a maximum of simultaneously running instances per job
- restrict job scheduling to one container per service
- multi-architecture image supports `amd64` and `armv7l` platforms, no emulator involved

4.2 Example

Let's say you want to dump the database of a Wordpress once a day. Here's a `docker-compose.yml` that defines a job that will be handled by *deck-chores*:

```
version: '2'

services:
  wordpress:
    image: wordpress
```

```
  mysql:
    image: mariadb
    volumes:
      - ./database_dumps:/dumps
    labels:
      deck-chores.dump.command: sh -c "mysqldump --all-databases > /dumps/dump-$
↪$(date -Idate)"
      deck-chores.dump.interval: daily
```

It is however recommended to use scripts with a proper shebang for such actions. Their outputs to `stdout` and `stderr` as well as their exit code will be logged by *deck-chores*.

4.3 Limitations

At the moment *deck-chores* is designed to run on a single Docker node, not within a cluster of these. Code and documentation contribution covering this are highly encouraged.

4.4 Acknowledgements

It wouldn't be as charming to write this piece of software without these projects:

- APScheduler for managing jobs
- cerberus for processing metadata
- docker-py for Docker interaction
- flake8, mypy, pytest and tox for testing
- Python

4.5 Roadmap

4.5.1 0.3

- parse time units for interval triggers
- handle a global limit on concurrent jobs
- print jobs when receiving SIGUSR1
- support for configuring APScheduler's jitter option on Cron- & IntervalTrigger

4.5.2 0.4

- keep output of job executions
- a rudimentary web ui

4.6 Authors

- Frank Sachsenheim (maintaining)
- aeri4list
- alpine-digger
- Brynjar Smári Bjarnason

Index