pyentrypoint Documentation

Release 0.5.2

Christophe Mehay

Jan 09, 2018

Contents

pyentrypoint is a tool written in `Python` to manage Docker containers `ENTRYPOINT`.

This tool avoids writing shell scripts to:

- Handle commands and sub commands
- Identify linked containers
- Auto configure container using *jinja2* templates
- Run commands before starting service
- Clean environment before running service
- Reload service when configuration has changed
- Increase security by setuid/setgid service

Contents:

CHAPTER 1

Installation

All you need to do is to setup a `yaml` file called `entrypoint-config.yml` and to install **pyentrypoint** in your `Dockerfile` using pip.

```
FROM        debian
# Installing git for example
RUN         apt-get update && apt-get install git python-pip -y
# Install pyentrypoint
RUN         pip install pyentrypoint
# Copy config file in the current WORKDIR
COPY        entrypoint-config.yml .
# Set ENTRYPOINT
ENTRYPOINT  ['pyentrypoint']
# git will be the default command
CMD         ['git']
```

```
FROM        alpine
# Installing git for example
RUN         apk add --update py-pip git
# Install pyentrypoint
RUN         pip install pyentrypoint
# Copy config file in the current WORKDIR
COPY        entrypoint-config.yml .
# Set ENTRYPOINT
ENTRYPOINT  ['pyentrypoint']
# git will be the default command
CMD         ['git']
```

CHAPTER 2

pyentrypoint-config.yml

This is an example of `entrypoint-config.yml` file.

```
# Entrypoint configuration example

# This entry should reflect CMD in Dockerfile
command: git

# This is a list with some subcommands to handle
# when CMD is not `git` here.
# By default, all args started with hyphen are handled.
subcommands:
    - "-*"
    - clone
    - init
    - ls-files
    # etc...

# User and group to run the cmd.
# Can be name or uid/gid.
# Affect only command handled.
# Dockerfile USER value by default.
user: 1000
group: 1000

# These files should exist (ADD or COPY)
# and should be jinja templated.
# Note: if config files end with ".tpl", the extension will be removed.
config_files:
    - /etc/gitconfig
    - .ssh/config.tpl # Will apply to ".ssh/config"
    - /tmp/id_rsa: .ssh/id_rsa # Will apply "/tmp/id_rsa" template to ".ssh/id_rsa"

# These environment variables will be wiped before
# exec command to keep them secret
# CAUTION: if the container is linked to another one,
```

```
# theses variables will passed to it anyway
secret_env:
    - SSHKEY
    - '*' # Support globbing, all environment will be wiped

# Links are handled here
# Port, name, protocol or env variable can be used to identify the links
# Raise an error if the link could not be identified
# This is not supported when using docker network or docker-compose v2.
links:
    'ssh':
        port: 22
        name: 'ssh*'
        protocol: tcp
        # env can be list, dict or string
        env:
            FOO: bar
        # Single doesn't allow multiple links for this ID
        # false by default
        single: true
        # Set to false to get optional link
        # true by default
        required: true

# Commands to run before applying configuration
pre_conf_commands:
    - echo something > to_this_file

# commands to run after applying configuration
post_conf_commands:
    - echo "something else" > to_this_another_file

post_run_commands:
    - echo run commands after started service

# Reload service when configuration change by sending a signal to process
reload:
    signal: SIGHUP # Optional, signal to send, default is SIGHUP
    watch_config_files: true # Optional, watch defined config files, default True
    files: # Optional, list of files to watch
        - /etc/conf/to/watch
# can also be enabled with a boolean:
reload: true

# Cleanup environment from variables created by linked containers
# before running command (True by default)
clean_env: true

# Enable debug to debug
debug: true

# Do not output anything except error
quiet: false
```

2.1 yaml references

2.1.1 command

`command` should reflect CMD in Dockerfile.

If the container is not started with this commande, the configuration will not be applied.

2.1.2 subcommands

`subcommands` is a list with some subcommands to handle.

Running container with a matching subcommand run it with setuped `command`.

```
subcommands:
    - "-*"
    - clone
    - init
    - ls-files
```

Note: Globbing pattern is enabled here.

By default, all args started with hyphen are handled.

2.1.3 user, group

User and group to run the `command`. Can be name or uid/gid. Affect only command handled.

```
user: 1000
group: root
```

Note: Dockerfile USER value by default.

Can be expended from environment in `ENTRYPOINT_USER` and `ENTRYPOINT_GROUP`.

2.1.4 config_files

These files should exist (ADD or COPY) and should be jinja templated.

```
config_files:
    - /etc/gitconfig
    - .ssh/config.tpl # Will apply to ".ssh/config"
    - /tmp/id_rsa: .ssh/id_rsa # Will apply "/tmp/id_rsa" template to ".ssh/id_rsa"
```

Note: if config files end with ".tpl", the extension will be removed.

2.1.5 secret_env

These environment variables will be wiped before running command to keep them secret.

```
secret_env:
    - SSHKEY
    - APIKEY
```

CAUTION: if the container is linked to another one, theses variables will be sent to it anyway.

2.1.6 links

Not supported when using docker network or docker-compose v2.

Links are handled here.

Port, name, protocol or environment variables can be used to identify the links.

```
links:
    'ssh': # This is the name to handle link in templates
        port: 22
        name: 'ssh*'
        protocol: tcp
        # env can be list, dictionary or string
        env:
            FOO: bar
        # Single doesn't allow multiple links for this ID
        # false by default
        single: true
        # Set to false to get optional link
        # true by default
        required: true
```

Note: All parameters are optionals.

Raise an error if the link could not be identified.

2.1.7 pre_conf_commands

List of shell commands to run before applying configuration

```
pre_conf_commands:
    - echo something > to_this_file
```

2.1.8 post_conf_commands

List of shell commands to run after applying configuration

```
post_conf_commands:
    - echo "something else" > to_this_another_file
```

2.1.9 post_run_commands

List of shell commands to run after service is started

```
post_run_commands:
    - sleep 5
    - echo "something else" > to_this_another_file
```

2.1.10 reload

Send SIGHUP to PID 1 to reload service when configuration change

Accept boolean or dictionary

```
reload:
    signal: SIGHUP # Optional, signal to send, default is SIGHUP
    watch_config_files: true # Optional, watch defined config files, default True
    files: # Optional, list of files to watch
        - /etc/conf/to/watch
        - /file/support/*.matching
# can also be enabled with a boolean:
reload: true
```

2.1.11 clean_env

Cleanup environment from variables created by linked containers before running command (True by default)

2.1.12 debug

Print some debug.

2.1.13 quiet

Do not output anything except error

CHAPTER 3

Templates

You can generate configuration for your service with jinga2 template.

Here is an example for an hypothetical ssh config file:

```
host server:
    hostname {{links.ssh.ip}}
    port {{links.ssh.port}}
```

Templates will be replaced with ip address and port of the identified link. All links can be accessed from `links.all`, this is a tuple of links you can iterate on it.

```
{% for link in links.all %}
host {{link.names[0]}}
    hostname {{link.ip}}
    port {{links.port}}
{% endfor %}
```

If you change the option `single` to `false` in the `entrypoint-config.yml`, the identified link `ssh` will become a tuple of links. You must iterate on it in the `jinja` template.

```
{% for link in links.ssh %}
host {{link.names[0]}}
    hostname {{link.ip}}
    port {{links.port}}
{% endfor %}
```

Accessing environment in template.

```
{% if 'SSHKEY in env' %}
{{env['SSHKEY']}}
{% endfor %}
```

3.1 Accessible objects

You have 4 available objects in your templates.

- `config`
- `links`
- `containers`
- `environ`

3.1.1 config

`Config` reflect the config file. You can retrieve any setup in this object.

(see `config.py`)

3.1.2 links

Not supported when using docker network or docker-compose v2.

`Links` handles `Link` objects. You can identify links using wildcard patterns in the configuration file.

`link` is related to one physical link (one ip and one port).

`link` handles the following attributes: - `ip` - link ip - `port` - link port (integer) - `environ` - related container environment - `protocol` - link protocol (`tcp` or `udp`) - `uri` - link URI (example: `tcp://10.0.0.3:80`) - `names` - tuple of related container names

3.1.3 containers

Not supported when using docker network or docker-compose v2.

`containers` handles a tuple of `container` object.

`container` handles the following attributes: - `ip` - container ip - `environ` - container environment - `names` - List of containers names - Names are sorted by length, but container ID will be the last element. - `id` - Hexadecimal container ID (if available, empty string else) - `links` - Tuple of `link` objects related to this container

3.1.4 environ

`environ` is the environment of the container (os.environ).

`env` is an alias to `environ`.

3.1.5 `yaml` and `json`

`yaml` and `json` objects are respectively an import of *PyYAML <http://pyyaml.org/>* and ‘json <https://docs.python.org/2/library/json.html> modules.

They are useful to load and dump serialized data from environment.

CHAPTER 4

Options setup

Some setups can be overridden using environment variables in the container.

- `ENTRYPOINT_CONFIG` overrides path of `entrypoint-config.yml` file.
- `ENTRYPOINT_FORCE` applies configuration and runs pre and post conf commands even if the `command` provided is not handled.
- `ENTRYPOINT_PRECONF_COMMAND` run an extra pre conf shell command after all pre conf commands.
- `ENTRYPOINT_POSTCONF_COMMAND` run an extra post conf shell command after all post conf commands.
- `ENTRYPOINT_DEBUG` enables debug logs.
- `ENTRYPOINT_RAW` does not use logging to display pre and post conf commands. This can be useful if output is serialized.
- `ENTRYPOINT_DISABLE_RELOAD` disable reload system even if it is enabled in `entrypoint-config.yml`.
- `ENTRYPOINT_USER` overrides `user` in config.
- `ENTRYPOINT_GROUP` overrides `group` in config.
- `ENTRYPOINT_DISABLE_SERVICE` exits container with 0 before doing anything. Useful to disable container using environement.

CHAPTER 5

Working examples

- Tor hidden service

CHAPTER 6

Indices and tables

- genindex
- modindex
- search