set-square

Building Docker images your own way

osoco software company

Jose San Leandro @rydnr

OSOCO

Contents

Dockerfiles (I)

In theory

- Automates building Docker images.
- Focuses on readability.
- Uses a caching mechanism to speed up the process.
- Advocates for deterministic/idempotent recipes: the same Dockerfile produces equivalent images.

Dockerfiles (II)

kstaken's Apache dockerfile

```
FROM ubuntu:12.04

MAINTAINER Kimbro Staken version: 0.1

RUN apt-get update && apt-get install -y apache2 && \
    apt-get clean && rm -rf /var/lib/apt/lists/*

ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

Dockerfiles (III)

In practice

- It's **not a language**, only a variable-less DSL.
- Using the operating system's package manager makes **idempotence mostly unachievable**.
- Not using the operating system's package manager **complicates** Dockerfiles.
- Images tend to **grow so much** the Dockerfiles need to join steps together at the expense of simplicity.

Features

set-square

1. Adds support for simple **variables** in Dockerfiles.
2. It's **not** a template engine (it's based on `envsubst`).
3. Maintains Dockerfile's *original readability*.
4. **Simplifies** the process of building images.

Example

gradle/Dockerfile.template

```
FROM ${NAMESPACE}/sdkman:${TAG}
MAINTAINER ${MAINTAINER}

USER developer

RUN cd /home/developer && \
    source /home/developer/.sdkman/bin/sdkman-init.sh && \
    sdk install gradle ${GRADLE_VERSION}
```

Example: default values

By default, we'd be interested in the latest stable version.

gradle/build-settings.sh

```
defineEnvVar GRADLE_VERSION "The Gradle version" "2.8";
```

Example: default values

By default, we'd be interested in the latest stable version.

gradle/build-settings.sh

```
defineEnvVar GRADLE_VERSION "The Gradle version" "2.8";
```

Of course, we can customize the default values.

gradle/.build-settings.sh

```
overrideEnvVar GRADLE_VERSION "2.7";
```

Example: default values

By default, we'd be interested in the latest stable version.

gradle/build-settings.sh

```
defineEnvVar GRADLE_VERSION "The Gradle version" "2.8";
```

Of course, we can customize the default values.

gradle/.build-settings.sh

```
overrideEnvVar GRADLE_VERSION "2.7";
```

Even further!

```
GRADLE_VERSION="2.6" ./build.sh [..] gradle
```

Common variables

There're some variables we'll want to override, globally:

.build.inc.sh

```
overrideEnvVar AUTHOR "rydnr";
overrideEnvVar AUTHOR_EMAIL "jose.sanleandro@osoco.es";
overrideEnvVar NAMESPACE "osoco";
```

Build process (I)

```
$ mkdir liquibase
```

Build process (I)

```
$ mkdir liquibase
```

liquibase/Dockerfile.template

```
FROM ${NAMESPACE}/fixme:${TAG}
MAINTAINER ${MAINTAINER}

# USER root
COPY Dockerfile /Dockerfile

# RUN apt-get update ...
# CMD [ "args" ]
# EXPOSE [port]
# VOLUME [my-folder]
```

Build process (II)

```
$ ./build.sh liquibase
```

Build process (II)

```
$ ./build.sh liquibase
```

```
[2015/11/08 20:50:05 build:sanity-check] Checking mandatory constants     [done]
[2015/11/08 20:50:05 build:sanity-check] Checking mandatory functions     [done]
[2015/11/08 20:50:05 build:sanity-check] Checking declared requirements   [done]
[2015/11/08 20:50:05 build:sanity-check] Checking input                   [valid]
[2015/11/08 20:50:06 build:main] Building osoco/liquibase:201511
Sending build context to Docker daemon 3.072 kB
Sending build context to Docker daemon
Step 0 : FROM osoco/base:201511
 ---> 4e111b99814b
Step 1 : MAINTAINER Jose San Leandro <jose.sanleandro@osoco.es>
 ---> Using cache
 ---> 2112e7bdc2c4
Step 2 : COPY Dockerfile /Dockerfile
 ---> b94be96794f2
Removing intermediate container 841272bd344b
Successfully built b94be96794f2
[2015/11/08 20:50:14 build:main] osoco/liquibase:201511                   [built]
[2015/11/08 20:50:14 build:main] Cleaning up 1 stale container(s)          [done]
```

Build process (III)

Build the image, run it, test it, fix it.

liquibase/Dockerfile.template

```
FROM ${NAMESPACE}/java:${TAG}
MAINTAINER ${MAINTAINER}

COPY liquibase.sh /usr/local/bin/liquibase
ENTRYPOINT [ "/usr/local/bin/liquibase" ]

RUN mkdir /opt/liquibase-${LIQUIBASE_VERSION} && \
    cd /opt/liquibase-${LIQUIBASE_VERSION} && \
    wget ${LIQUIBASE_URL} && \
    tar xvf ${LIQUIBASE_ARTIFACT} -C /opt/liquibase-${LIQUIBASE_VERSION} && \
    rm -f /opt/${LIQUIBASE_ARTIFACT} && \
    chmod +x /usr/local/bin/liquibase

# Run with
# docker run --link [db-container]:db £{NAMESPACE}/£{IMAGE}:£{TAG}
```

Build process (IV)

You can use templates for other files too.

liquibase/liquibase.sh.template

```
#!/bin/bash

cd /opt/liquibase-${LIQUIBASE_VERSION}
echo "Running $*"
./liquibase --changeLogFile=/changelogs/changelog.yml $*
```

Conclusion (I)

Simple but useful

- Simple and straight-forward.
- No hidden magic.
- Avoids changing values manually.
- Useful for CI jobs.

Conclusion (I)

Simple but useful

- Simple and straight-forward.
- No hidden magic.
- Avoids changing values manually.
- Useful for CI jobs.

Try it

https://github.com/rydnr/set-square

Conclusion (II)

set-square-phusion-images

- Catalog of Docker images.
- Based on phusion/baseimage-docker.
- Out-of-the-box (but optional):
  - monitoring,
  - backup,
  - logging.