

# Pemrograman Framework Java

Pertemuan 14

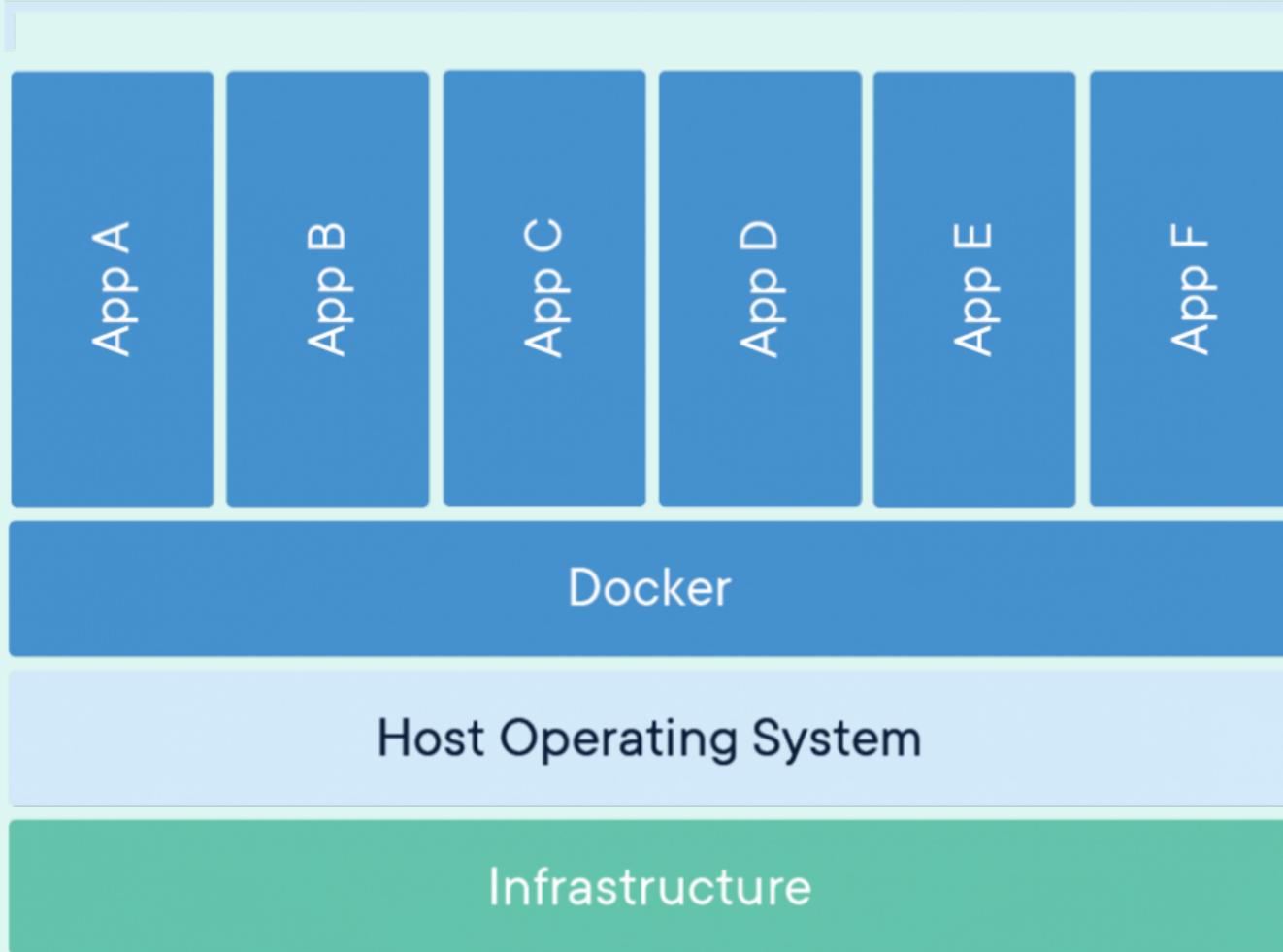
# Spring Docker

- Docker adalah sebuah teknologi “container” aplikasi, sehingga satu aplikasi akan terisolasi dengan yang lainnya.
- Sebuah kontainer mengandung semua kode aplikasi termasuk dependensinya, sehingga bisa digunakan untuk semua lingkungan

# Cont'd

- Beberapa kontainer dapat berjalan di mesin yang sama, dan berbagi kernel OS dengan kontainer lainnya.
- Dan setiap aplikasi yang berjalan di user space terisolasi satu sama lain
- Sehingga aplikasi dengan menggunakan Spring dapat dimasukkan menjadi Image Container

# Containerized Applications



# Container Engine

- Container memerlukan Engine agar bisa menjalankan aplikasi yang diinginkan
- Engine yang digunakan biasanya *crossplatform* sehingga dapat digunakan untuk semua sistem operasi khususnya Linux dan Windows

# Cont'd

- Docker : Engine ini menargetkan sistem Enterprise dibandingkan dengan Desktop biasa
- Snapcraft : Engine packaging aplikasi khususnya Linux, terpasang otomatis di Ubuntu
- Flatpak : Engine packaging aplikasi namun dengan fitur yang berbeda dengan Snap

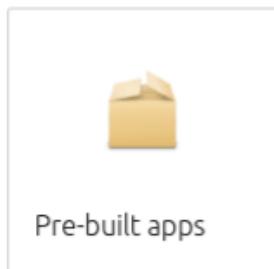
# Dukungan Snap

Learn how to snap an app in 30 minutes

What language or framework does your app use?



Python



Pre-built apps



C/C++



Go



Java



Node.js



Electron



Ruby



Rust



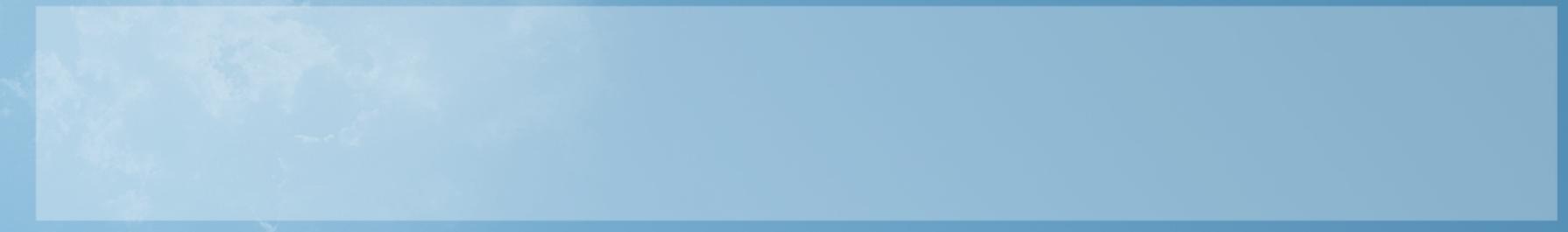
MOOS



ROS



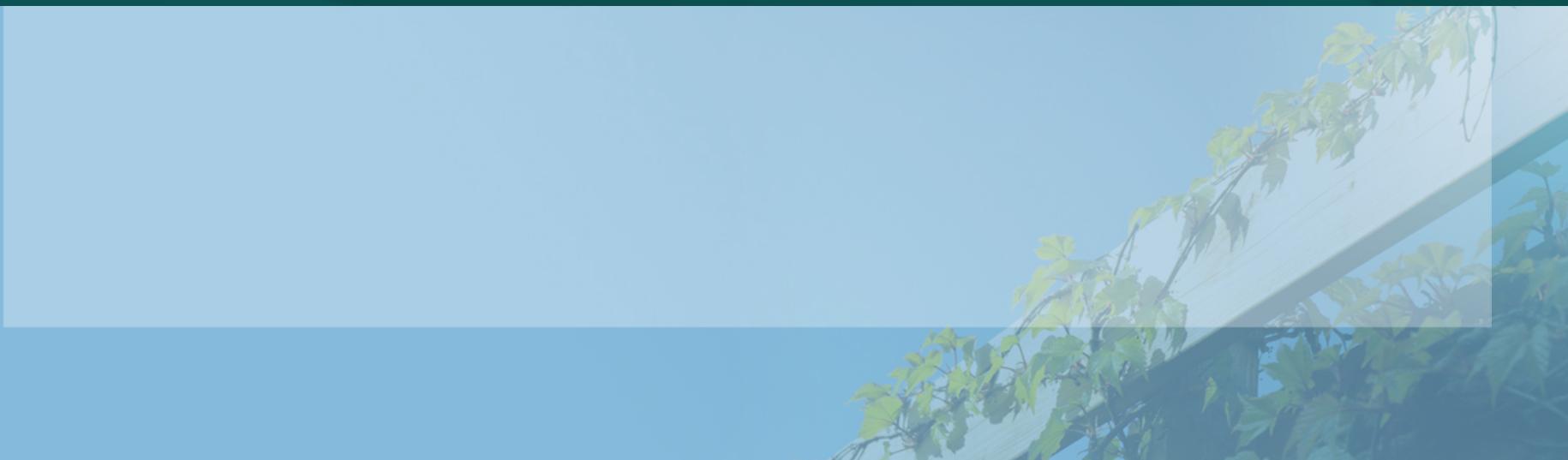
ROS 2



Search thousands of snaps used by millions of people across 41 Linux distributions

Search thousands of snaps

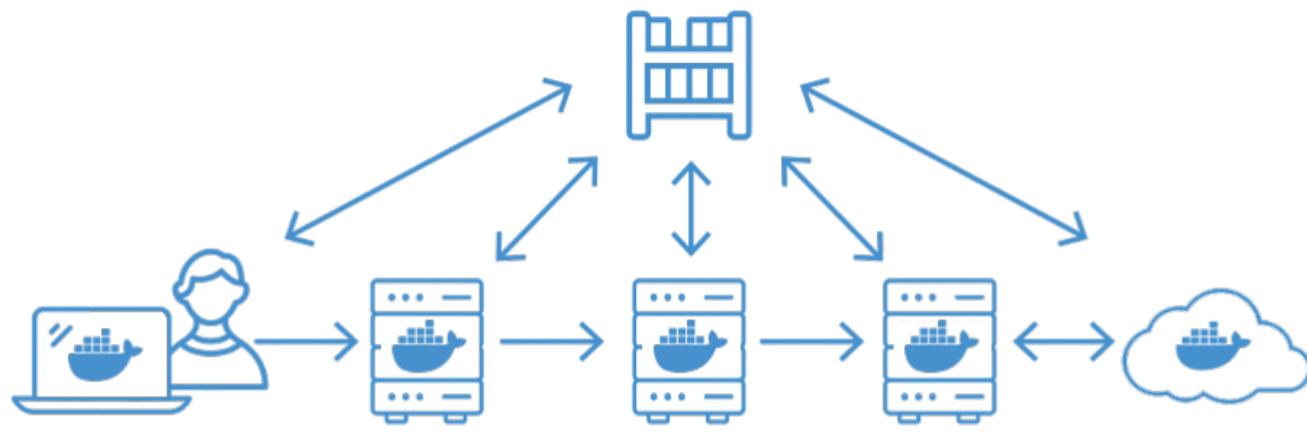
Search



# Docker Image Registry

- Aplikasi yang telah dibuat akan disimpan di Docker Image Registry demi kemudahan untuk *Build* bahkan berbagi ke Developer lainnya

## Docker Trusted Registry



Development

Docker Desktop

Test

Staging

Production

Signature verification

Native encryption



Automated Policies



Scanning



Signing

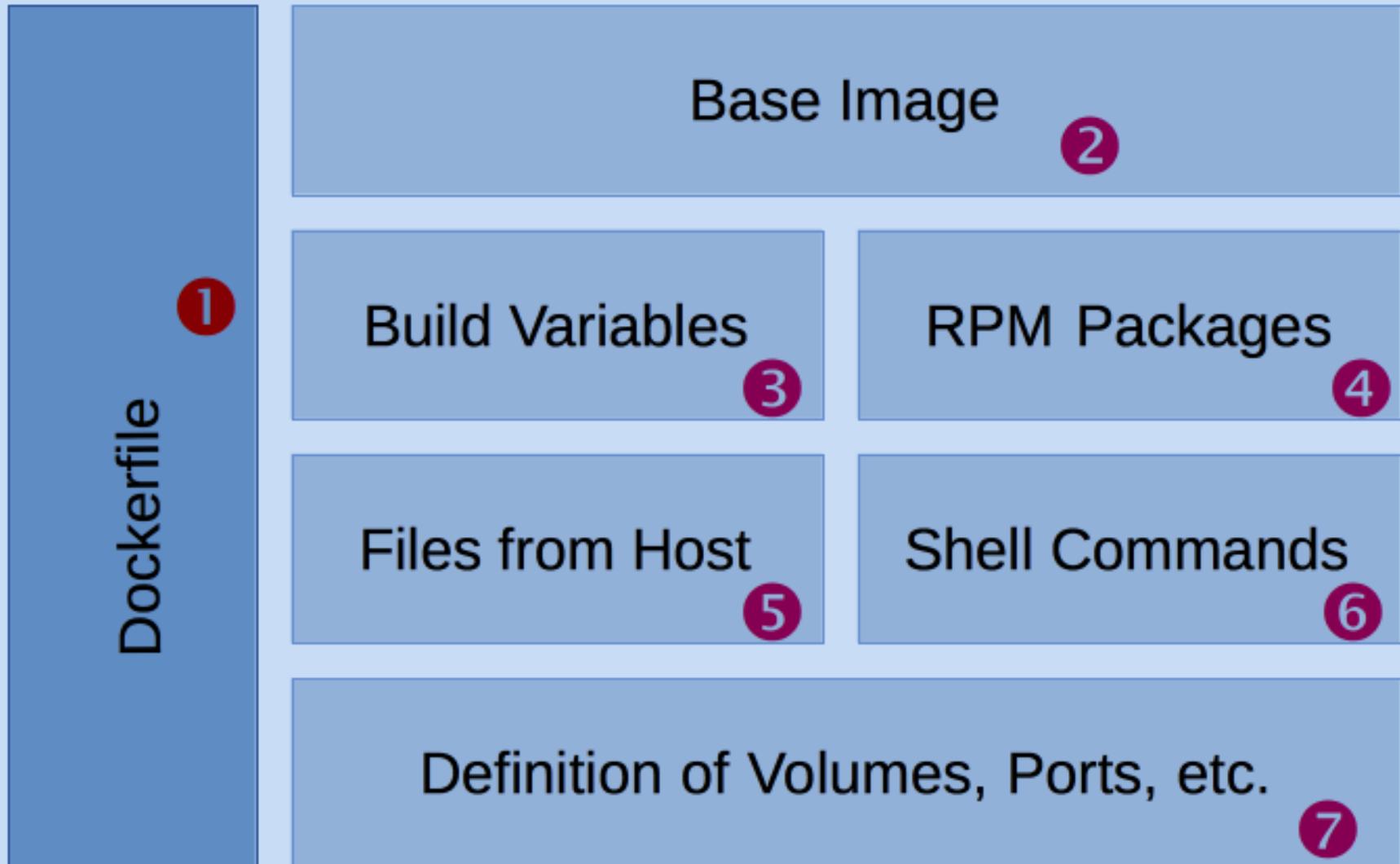
# Spring + Docker

- Code dan Share!
- Aplikasi dengan Abstraksi untuk memudahkan eksekusi di berbagai jenis lingkungan
- Masalah dengan dependensi akan semakin berkurang

# Cont'd

- Dalam proses pembuatannya, koding berlangsung layaknya aplikasi biasa. Termasuk pengujian apakah jalan atau tidak.
- Kemudian beralih ke proses Kontenarisasi membuat **Dockerfile**

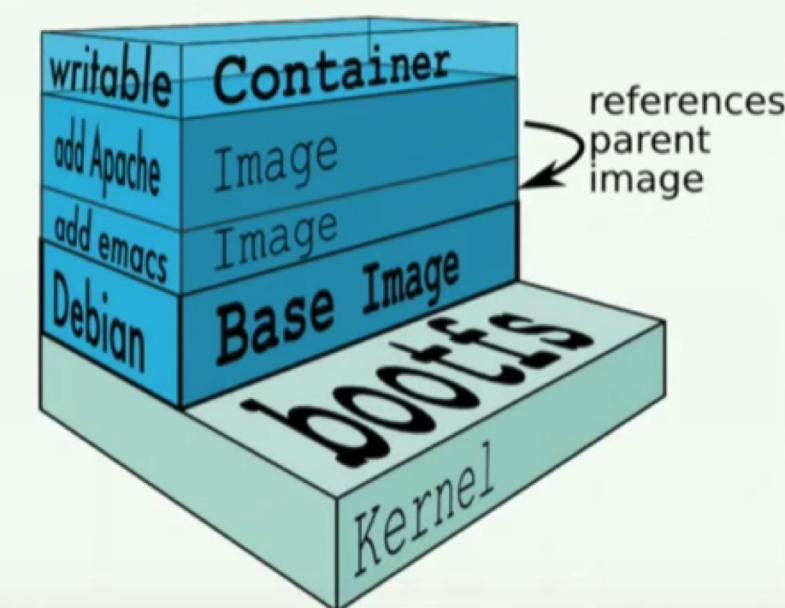
# Container Image – Build Process



# Cont'd

- Docker memiliki satu file **Dockerfile** yang digunakan untuk menspesifikasikan lapisan-lapisan di dalam image nantinya.

Image layers



# Cont'd

**Runtime file system (read-write)**

**WAR file added (read-only)**

**Tomcat installed (read-only)**

**Java 8 installed (read-only)**

**Apache HTTP installed (read-only)**

**Debian Wheezy (read-only base image)**

**Linux Kernel 3.13.3 (read-only bootfs)**

# Dockerfile

- Sebuah dokumen yang mengandung semua perintah yang dapat digunakan untuk melakukan pemasangan image.

```
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE=target/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT
["java","-Djava.security.egd=file:/dev/.urandom","-jar","/app.jar"]
```

# Cont'd

- **FROM:** Memberitahu Docker untuk menggunakan image yang diberikan sebagai builder dasar.
- **VOLUME:** Instruksi ini membuat mount point dengan nama yang telah ditentukan.
- **COPY:** Instruksi mengopi file baru atau direktori dari sumber ke file system kontainer

# Cont'd

- Setelah Dockerfile dibuat, hal selanjutnya adalah melakukan building.

```
docker build -t springio/gs-spring-boot-docker .
```

# Cont'd

- Sending build context to Docker daemon 15.36 kB
- Step 1/4 : FROM alpine:3.2
  - ---> 31f630c65071
- Step 2/4 : MAINTAINER SvenDowideit@home.org.au
  - ---> Using cache
  - ---> 2a1c91448f5f
- Step 3/4 : RUN apk update && apk add socat && rm -r /var/cache/
  - ---> Using cache
  - ---> 21ed6e7fbb73
- Step 4/4 : CMD env | grep \_TCP= | (sed 's/.\*\_PORT\_\([0-9]\*\)\_TCP=tcp://\1:\1/s/\1/\1/g' && socat -t 100000000 TCP4-LISTEN:\1,fork,reuseaddr TCP4:\2:\3 \&/ && echo wait) | sh
  - ---> Using cache
  - ---> 7ea8aef582cc
- Successfully built 7ea8aef582cc

# Tweaks

- Menggunakan ***spring-context-indexer*** namun tidak terlalu berguna untuk aplikasi kecil
- Sebaiknya tidak menggunakan **Actuator**
- Mematikan fitur **JMX**

# Eksekusi Image

- \$ docker run -p 8080:8080 -t springio/gs-spring-boot-docker
- ....
- 2015-03-31 13:25:48.035 INFO 1 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
- 2015-03-31 13:25:48.037 INFO 1 --- [main] hello.Application : Started Application in 5.613 seconds (JVM running for 7.293)

# Cont'd

- **docker run -p 8080:8080 -t springio/gs-spring-boot-docker**
- Memiliki arti untuk menjalankan aplikasi dengan container **springio/gs-spring-boot-docker** dengan menggunakan port **8080:8080**