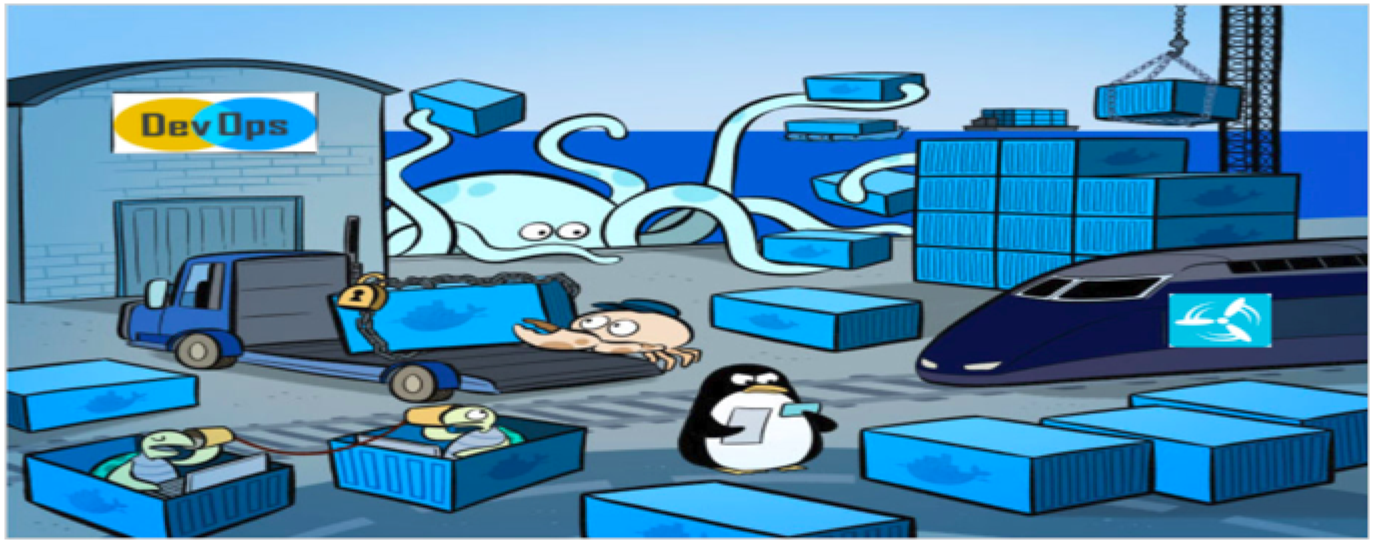
Programm Overview

This one-day program will give strong foundation knowledge on docker deployment in single server and operations on containers, images and networking

- Docker
- Virtualization
- Docker container
- Docker image
- Docker Networking
- Docker Volumes
- Docker Compose

Docker



Rama Shanker

Understanding Docker

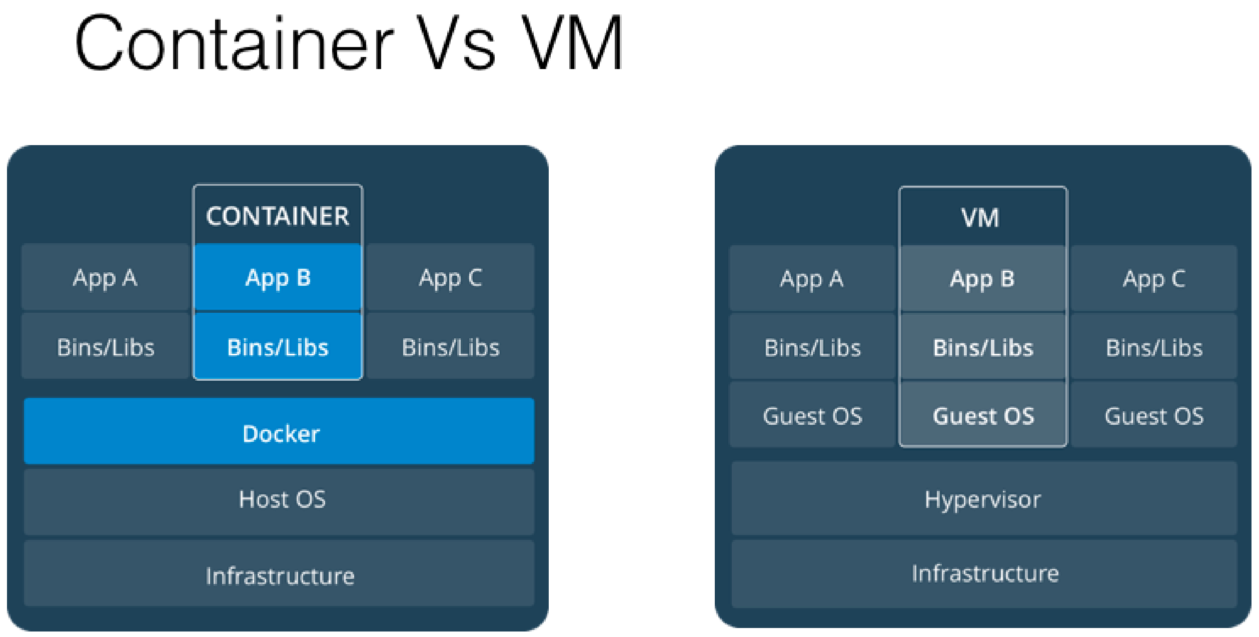
- Docker is a platform for developers and sys admins to **develop, deploy, and run** applications with containers. The use of Linux containers to deploy applications is called *containerization*. Containers are not new, but their use for easily deploying applications is.
- Containerization is increasingly popular because containers are:
- Flexible: Even the most complex applications can be containerized.
- Lightweight: Containers leverage and share the host kernel.
- Interchangeable: You can deploy updates and upgrades on-the-fly.
- Portable: You can build locally, deploy to the cloud, and run anywhere.
- Scalable: You can increase and automatically distribute container replicas.
- Stackable: You can stack services vertically and on-the-fly.

https://docs.docker.com/get-started/

Containers and virtual machines

- A **container** runs *natively* on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight.
- By contrast, a **virtual machine** (VM) runs a full-blown “guest” operating system with *virtual* access to host resources through a hypervisor. In general, VMs provide an environment with more resources than most applications need.

Container And VM



Installing Docker

- Installing docker on Windows:
- https://docs.docker.com/docker-for-windows/install/

Installing docker on ubuntu

- https://tecadmin.net/install-docker-on-ubuntu/
- https://docs.docker.com/install/linux/docker-ce/ubuntu/

Installing docker on mac

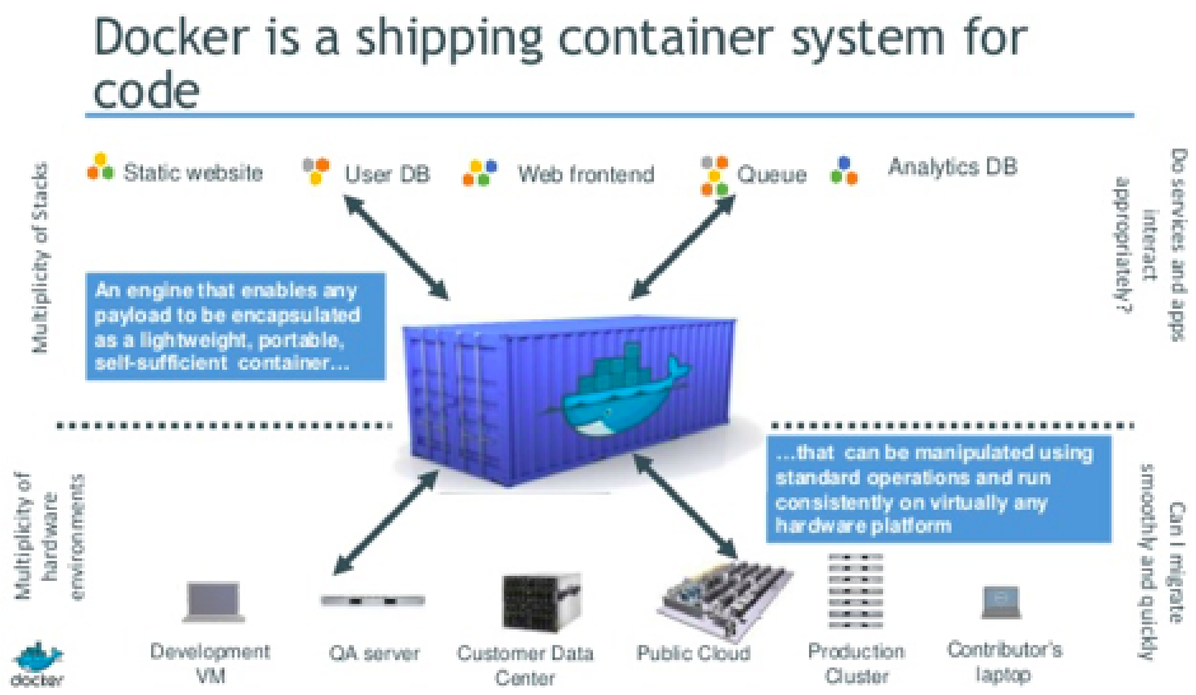
- https://docs.docker.com/docker-for-mac/install/

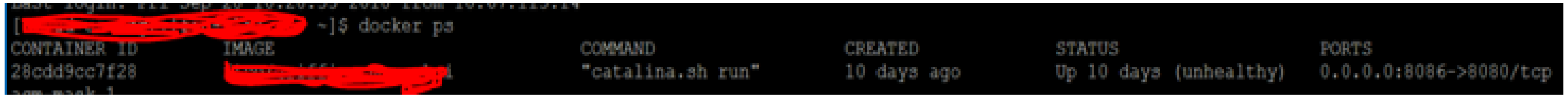
Container

- https://www.docker.com/resources/what-container

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

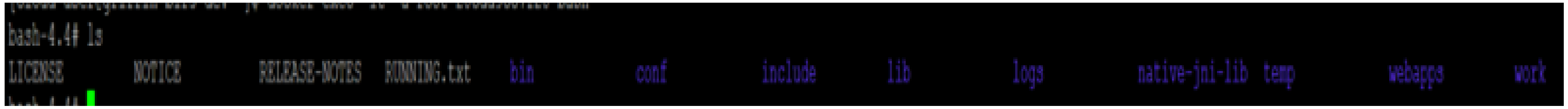
Container


Docker is a shipping container system for code
Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…
…that can be manipulated using standard operations and run consistently on virtually any hardware platform
Multiplicity of hardware environments
Can I migrate smoothly and quickly
docker
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop



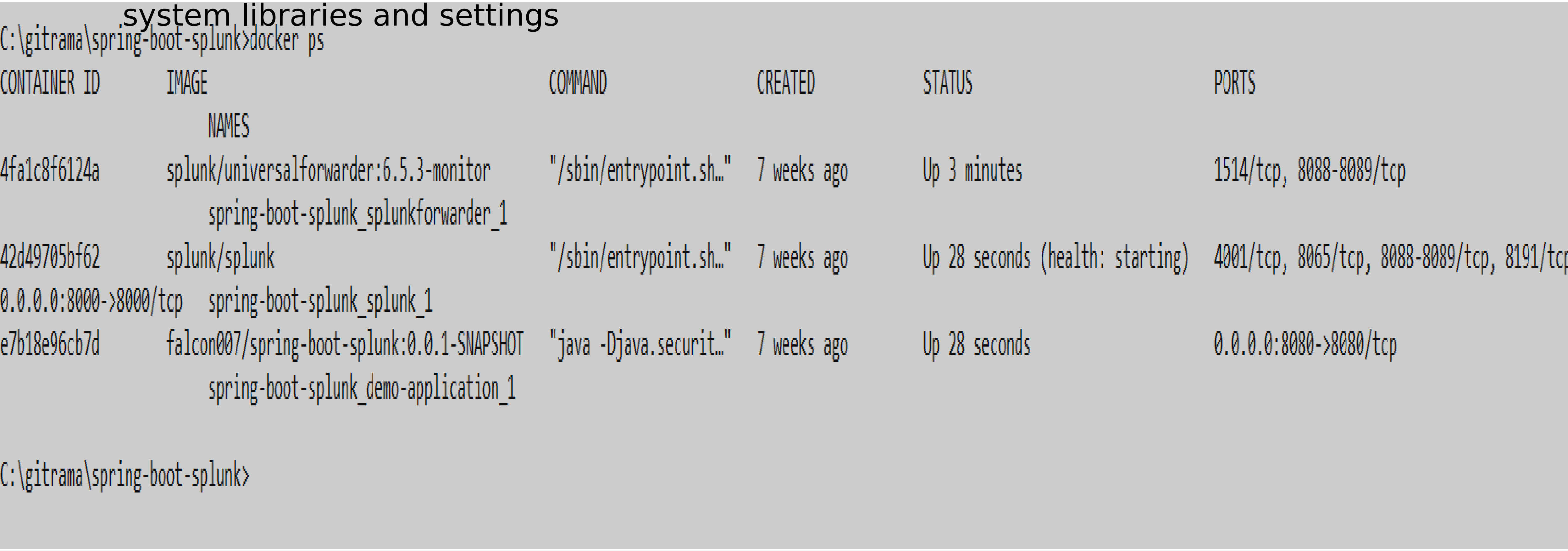
```
[[redacted] ~]$ docker ps
CONTAINER ID     IMAGE                           COMMAND             CREATED        STATUS                  PORTS
28cdd9cc7f28     [redacted]                      "catalina.sh run"   10 days ago    Up 10 days (unhealthy)  0.0.0.0:8086->8080/tcp
```

docker exec -it -u root 28cdd9cc7f28 bash



```
bash-4.4# ls
LICENSE    NOTICE    RELEASE-NOTES  RUNNING.txt  bin    conf    include    lib    logs    native-jni-lib  temp    webapps    work
```

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings

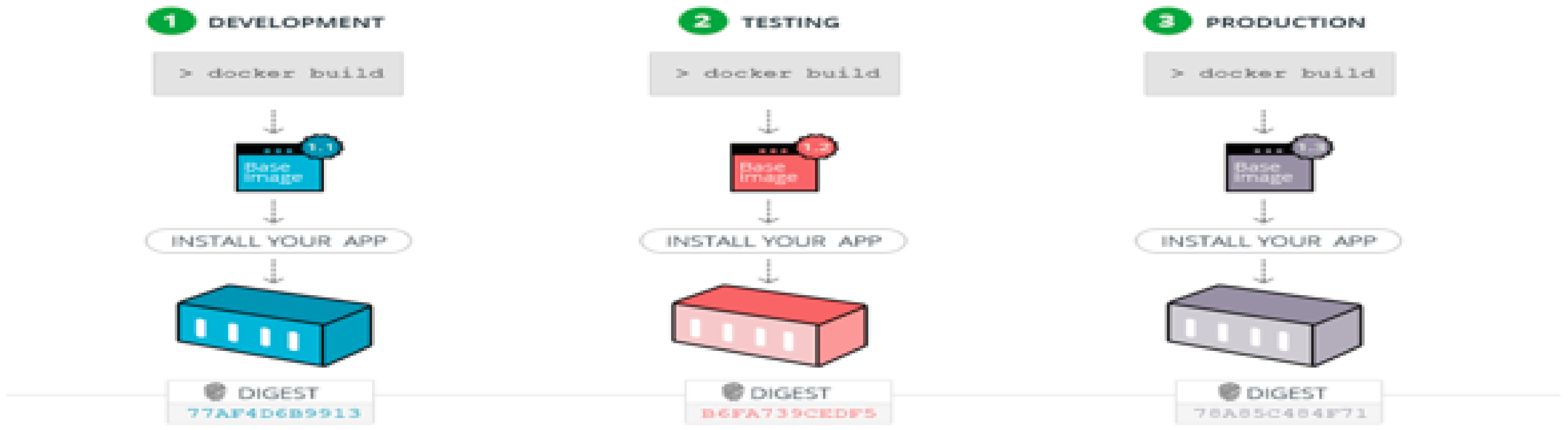


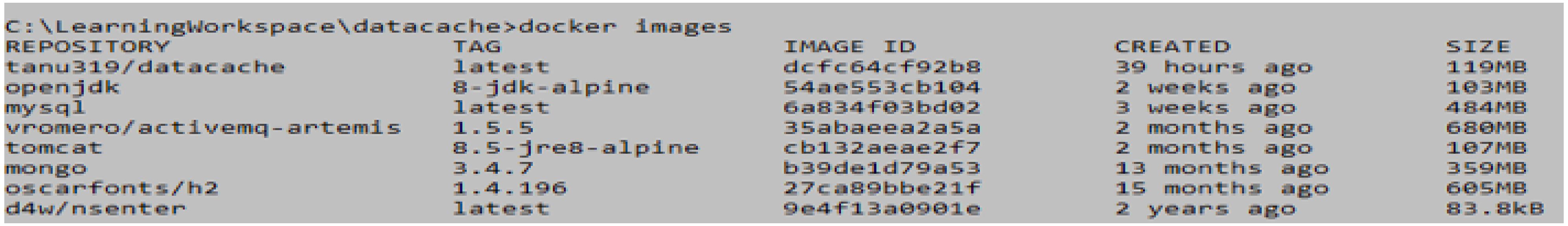
```
C:\gitrama\spring-boot-splunk>docker ps
CONTAINER ID        IMAGE                                   COMMAND                  CREATED             STATUS                              PORTS
                    NAMES
4fa1c8f6124a        splunk/universalforwarder:6.5.3-monitor "/sbin/entrypoint.sh…"   7 weeks ago         Up 3 minutes                        1514/tcp, 8088-8089/tcp
                    spring-boot-splunk_splunkforwarder_1
42d49705bf62        splunk/splunk                           "/sbin/entrypoint.sh…"   7 weeks ago         Up 28 seconds (health: starting)    4001/tcp, 8065/tcp, 8088-8089/tcp, 8191/tc
0.0.0.0:8000->8000/tcp   spring-boot-splunk_splunk_1
e7b18e96cb7d        falcon007/spring-boot-splunk:0.0.1-SNAPSHOT  "java -Djava.securit…"   7 weeks ago    Up 28 seconds                       0.0.0.0:8080->8080/tcp
                    spring-boot-splunk_demo-application_1

C:\gitrama\spring-boot-splunk>
```

Docker Image

- A **Docker image** is a file, comprised of multiple layers, used to execute code in a **Docker container**.
  An **image**is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel.

Image

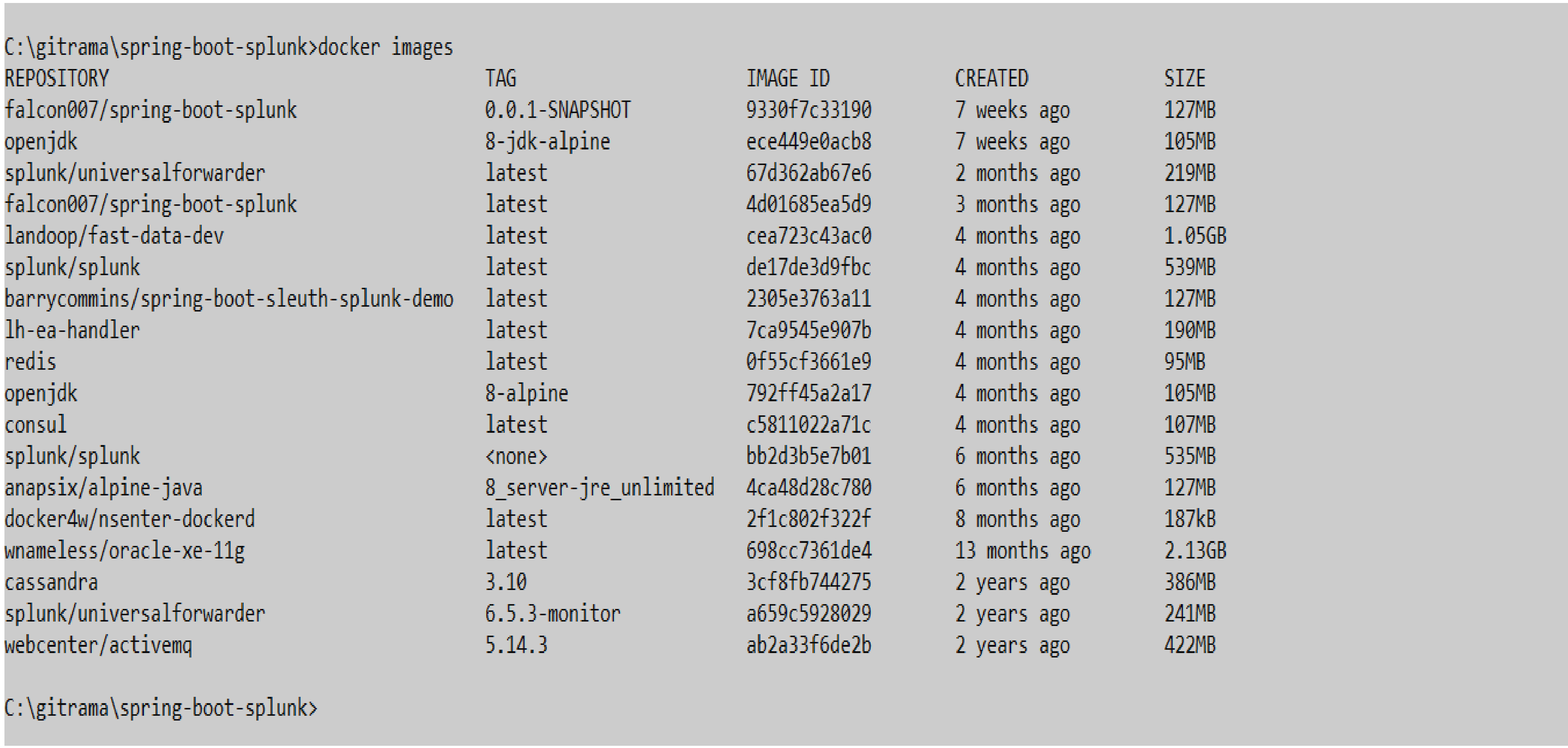




```
C:\LearningWorkspace\datacache>docker images
REPOSITORY                   TAG                   IMAGE ID            CREATED             SIZE
tanu319/datacache            latest                dcfc64cf92b8        39 hours ago        119MB
openjdk                      8-jdk-alpine          54ae553cb104        2 weeks ago         103MB
mysql                        latest                6a834f03bd02        3 weeks ago         484MB
vromero/activemq-artemis     1.5.5                 35abaeea2a5a        2 months ago        680MB
tomcat                       8.5-jre8-alpine       cb132aeae2f7        2 months ago        107MB
mongo                        3.4.7                 b39de1d79a53        13 months ago       359MB
oscarfonts/h2                1.4.196               27ca89bbe21f        15 months ago       605MB
d4w/nsenter                  latest                9e4f13a0901e        2 years ago         83.8kB
```

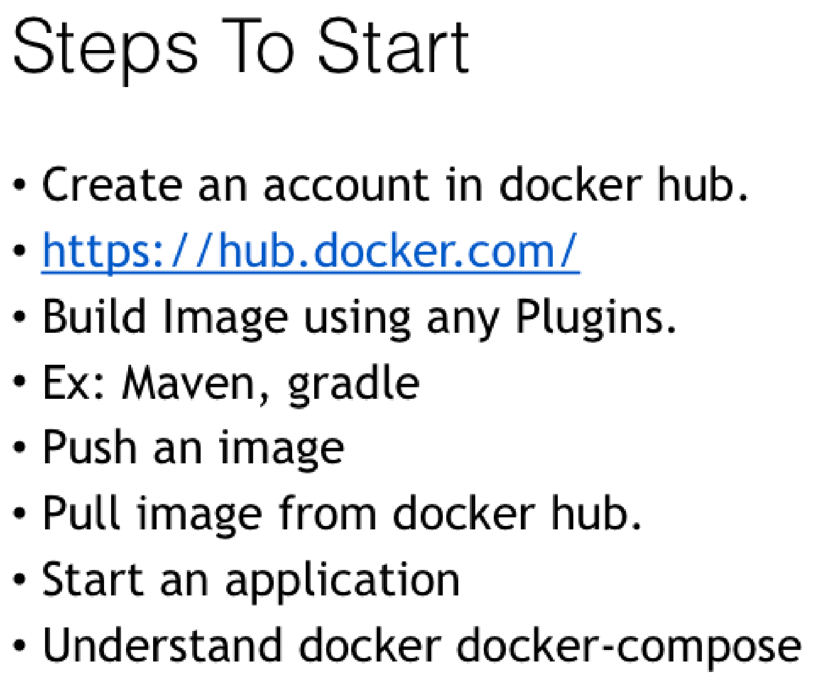
- An **image** is an executable package that includes everything needed to run an applicatio the code, a runtime, libraries, environment variables, and configuration files.

Docker images



```
C:\gitrama\spring-boot-splunk>docker images
REPOSITORY                              TAG                       IMAGE ID            CREATED             SIZE
falcon007/spring-boot-splunk            0.0.1-SNAPSHOT            9330f7c33190        7 weeks ago         127MB
openjdk                                 8-jdk-alpine              ece449e0acb8        7 weeks ago         105MB
splunk/universalforwarder               latest                    67d362ab67e6        2 months ago        219MB
falcon007/spring-boot-splunk            latest                    4d01685ea5d9        3 months ago        127MB
landoop/fast-data-dev                   latest                    cea723c43ac0        4 months ago        1.05GB
splunk/splunk                           latest                    de17de3d9fbc        4 months ago        539MB
barrycommins/spring-boot-sleuth-splunk-demo latest                2305e3763a11        4 months ago        127MB
lh-ea-handler                           latest                    7ca9545e907b        4 months ago        190MB
redis                                   latest                    0f55cf3661e9        4 months ago        95MB
openjdk                                 8-alpine                  792ff45a2a17        4 months ago        105MB
consul                                  latest                    c5811022a71c        4 months ago        107MB
splunk/splunk                           <none>                    bb2d3b5e7b01        6 months ago        535MB
anapsix/alpine-java                     8_server-jre_unlimited    4ca48d28c780        6 months ago        127MB
docker4w/nsenter-dockerd                latest                    2f1c802f322f        8 months ago        187kB
wnameless/oracle-xe-11g                 latest                    698cc7361de4        13 months ago       2.13GB
cassandra                               3.10                      3cf8fb744275        2 years ago         386MB
splunk/universalforwarder               6.5.3-monitor             a659c5928029        2 years ago         241MB
webcenter/activemq                      5.14.3                    ab2a33f6de2b        2 years ago         422MB

C:\gitrama\spring-boot-splunk>
```

Lab

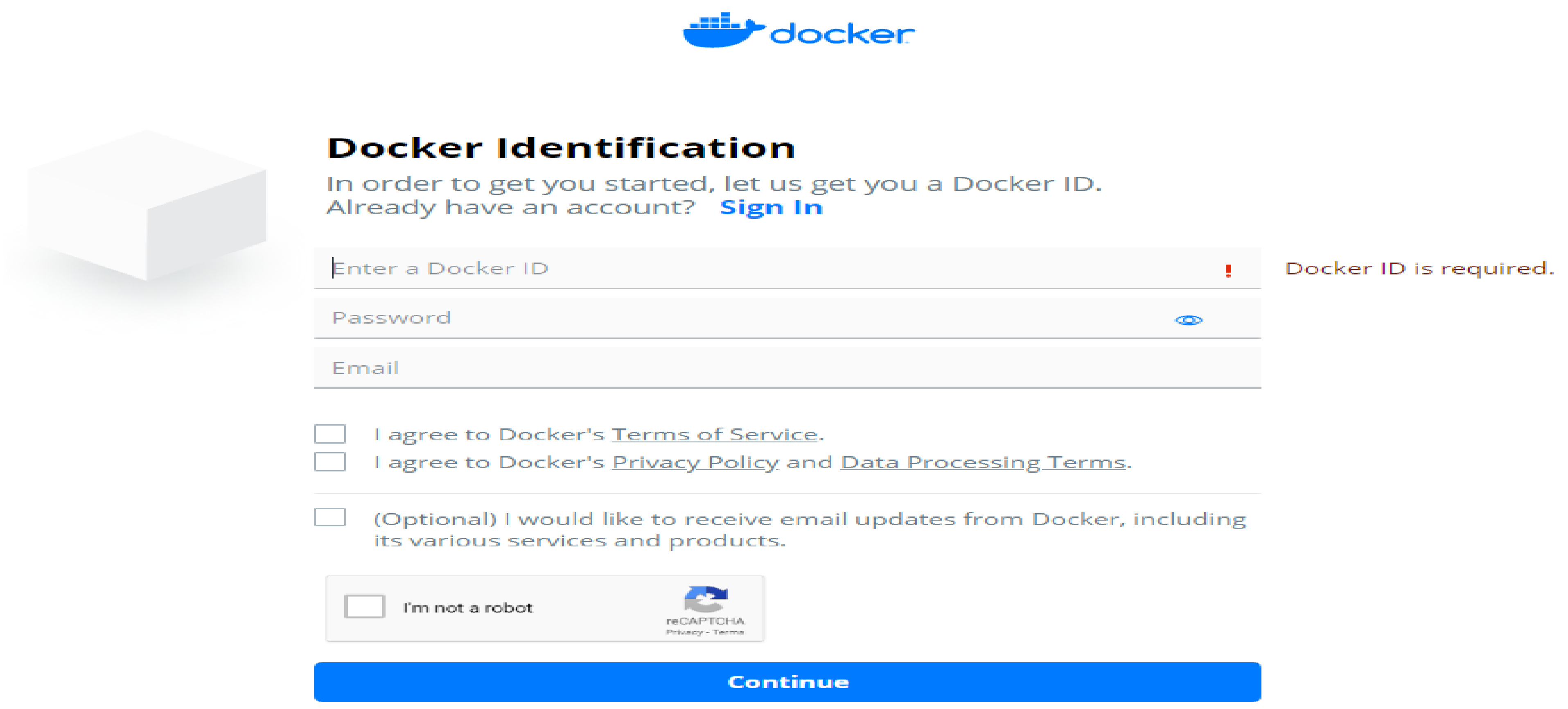


Steps To Start

- Create an account in docker hub.
- https://hub.docker.com/
- Build Image using any Plugins.
- Ex: Maven, gradle
- Push an image
- Pull image from docker hub.
- Start an application
- Understand docker docker-compose

Docker Hub Signup

- https://hub.docker.com/



docker

Docker Identification

In order to get you started, let us get you a Docker ID.
Already have an account? **Sign In**

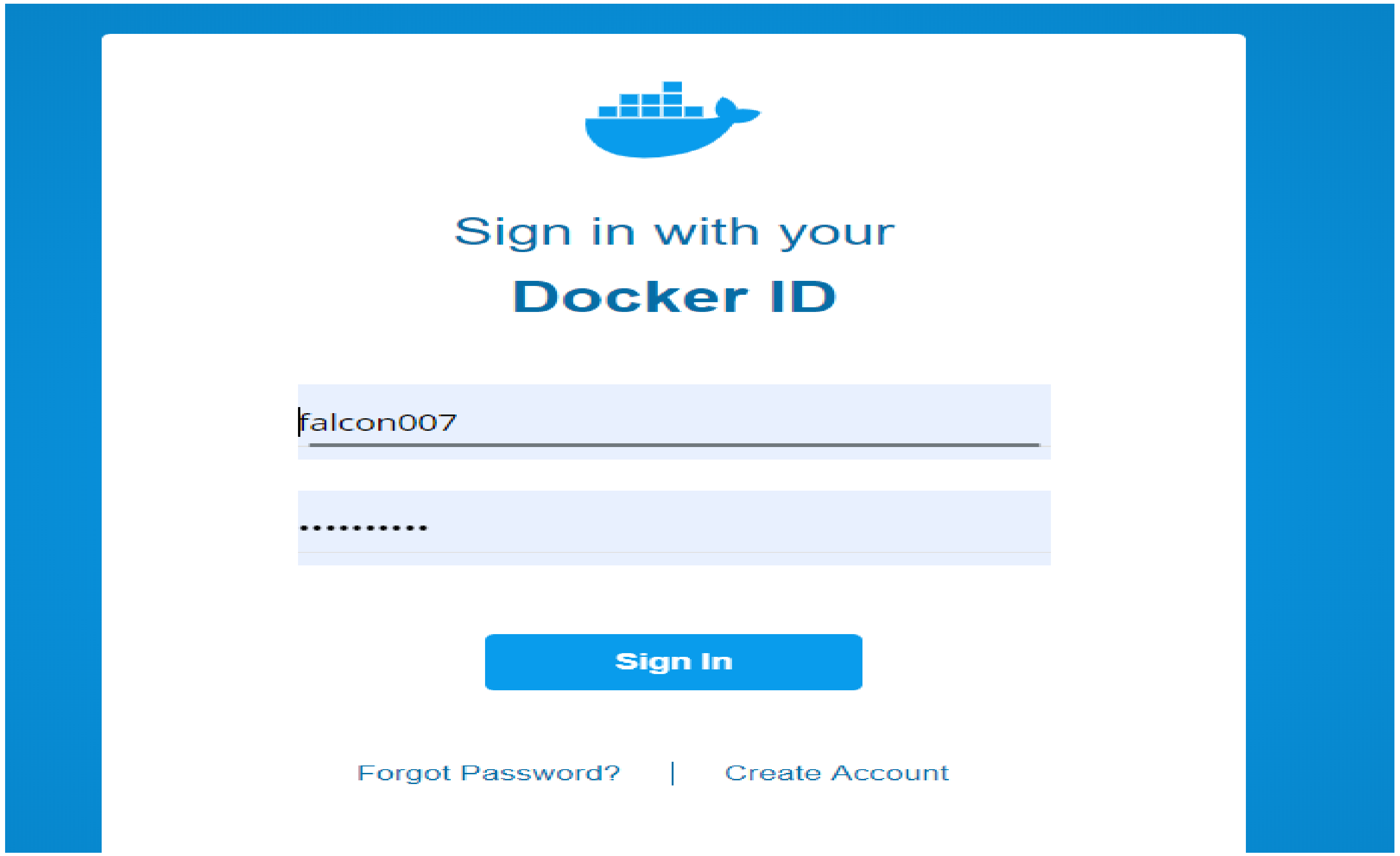
Enter a Docker ID — Docker ID is required.

Password

Email

- [ ] I agree to Docker's Terms of Service.
- [ ] I agree to Docker's Privacy Policy and Data Processing Terms.
- [ ] (Optional) I would like to receive email updates from Docker, including its various services and products.

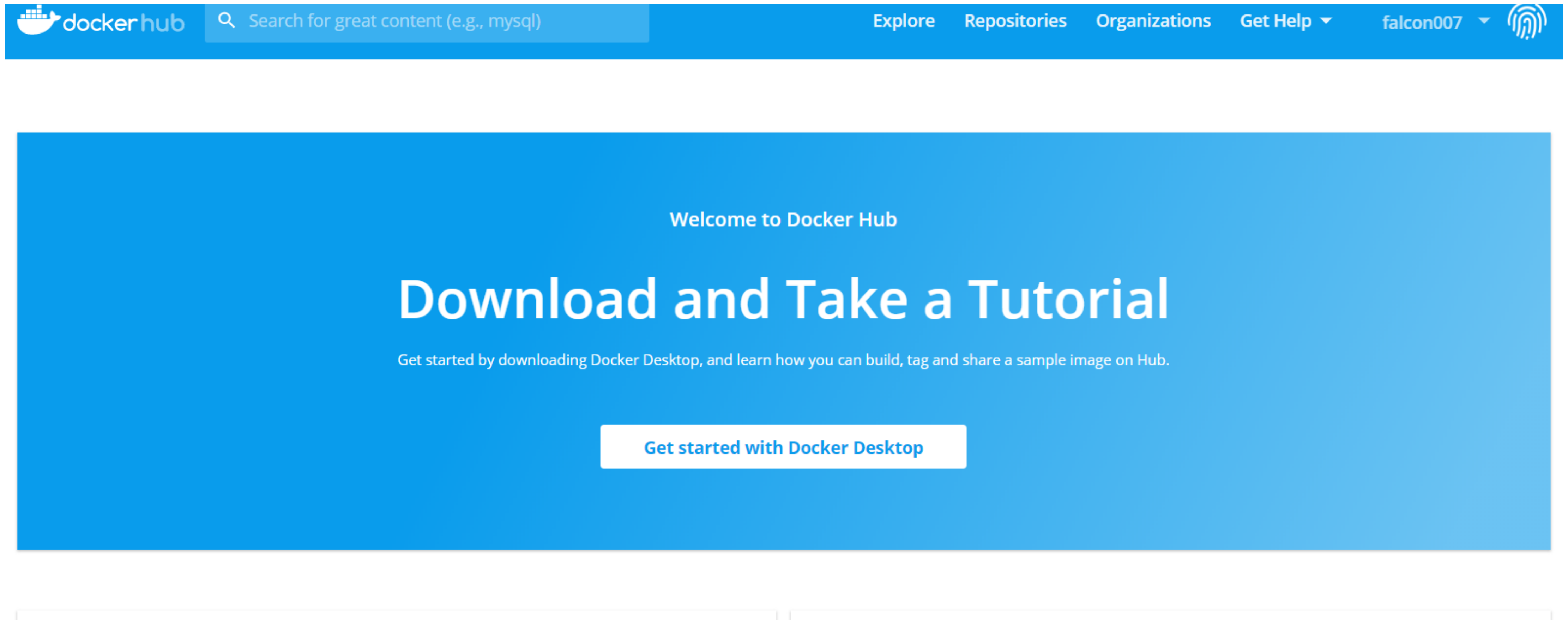
Docker Hub Sign In


Sign in with your
Docker ID

falcon007

••••••••••

Login Screen


docker hub
Search for great content (e.g., mysql)
Explore
Repositories
Organizations
Get Help
falcon007

Welcome to Docker Hub

Download and Take a Tutorial

Get started by downloading Docker Desktop, and learn how you can build, tag and share a sample image on Hub.

Get started with Docker Desktop

Application:

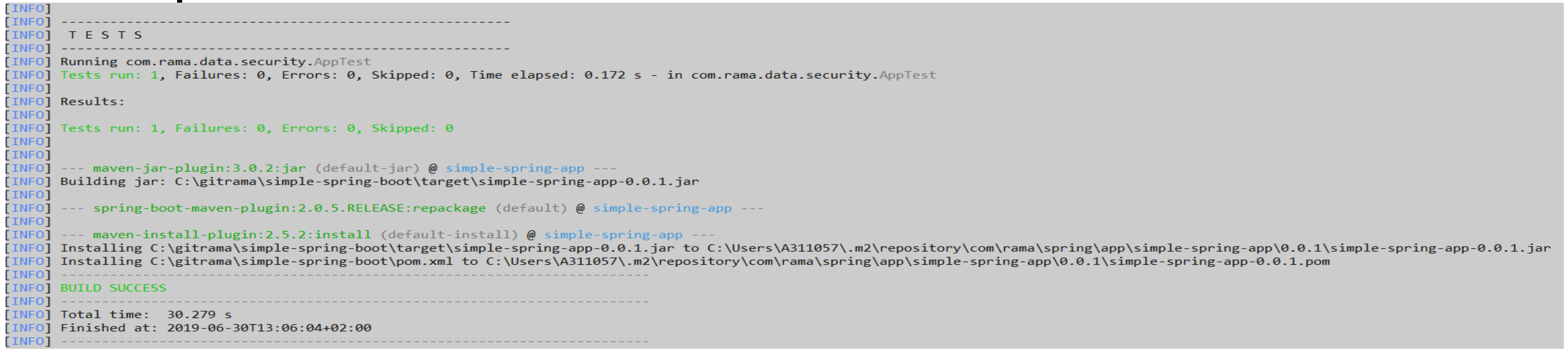
- Create simple spring-boot Application
- https://github.com/ramashanker/training/tree/master/simple-spring-boot

Tools Need to install

- Maven
- Eclipse
- JDK

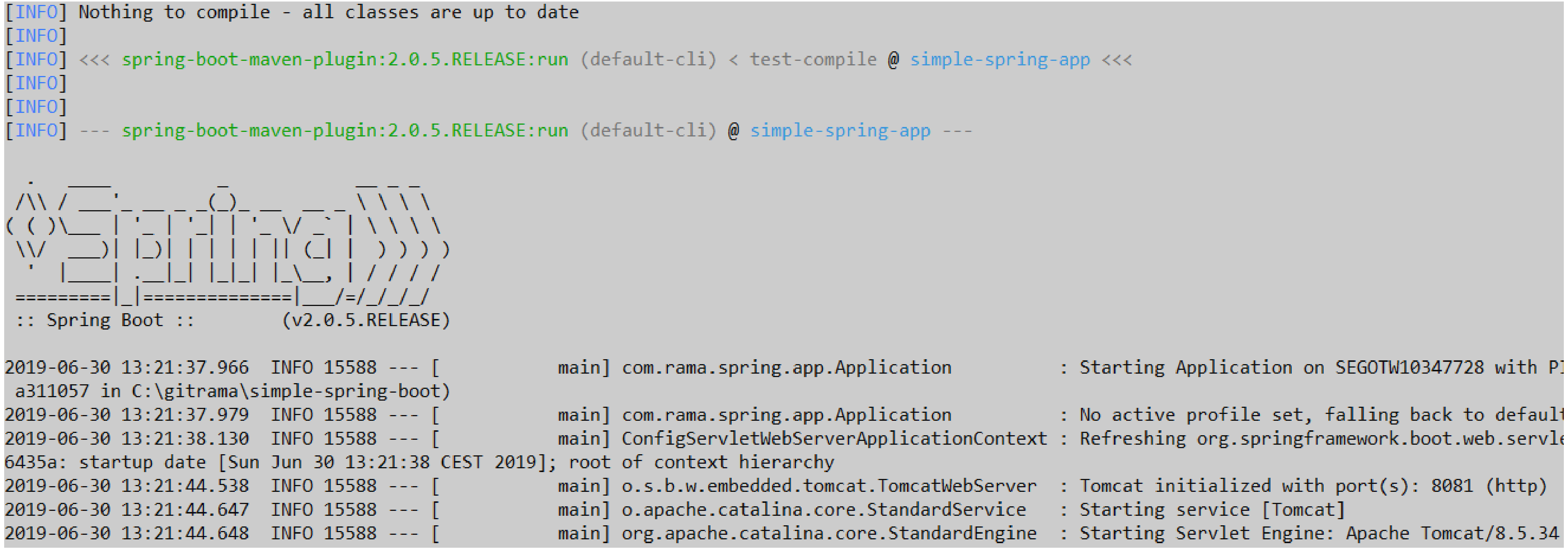
  C:\Program Files\Java\jdk1.8.0_191

- Compile the code: mvn clean install



```
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.rama.data.security.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.172 s - in com.rama.data.security.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ simple-spring-app ---
[INFO] Building jar: C:\gitrama\simple-spring-boot\target\simple-spring-app-0.0.1.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.0.5.RELEASE:repackage (default) @ simple-spring-app ---
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ simple-spring-app ---
[INFO] Installing C:\gitrama\simple-spring-boot\target\simple-spring-app-0.0.1.jar to C:\Users\A311057\.m2\repository\com\rama\spring\app\simple-spring-app\0.0.1\simple-spring-app-0.0.1.jar
[INFO] Installing C:\gitrama\simple-spring-boot\pom.xml to C:\Users\A311057\.m2\repository\com\rama\spring\app\simple-spring-app\0.0.1\simple-spring-app-0.0.1.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  30.279 s
[INFO] Finished at: 2019-06-30T13:06:04+02:00
[INFO] ------------------------------------------------------------------------
```

- Run the application: mvn spring-boot:run



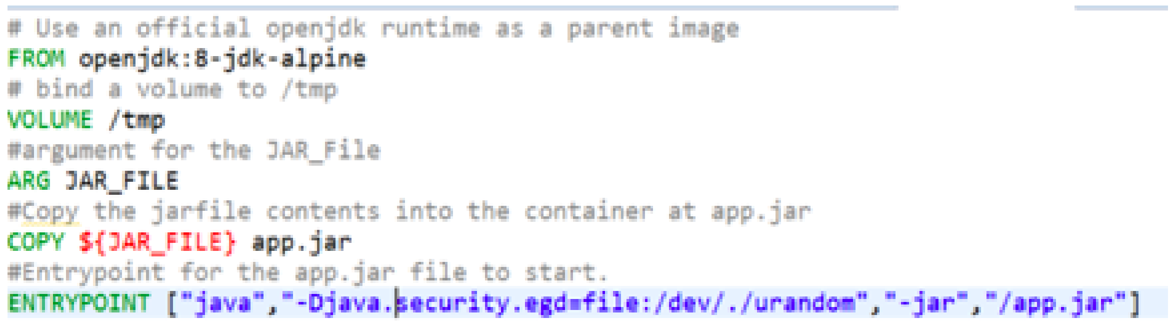
```
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] <<< spring-boot-maven-plugin:2.0.5.RELEASE:run (default-cli) < test-compile @ simple-spring-app <<<
[INFO]
[INFO]
[INFO] --- spring-boot-maven-plugin:2.0.5.RELEASE:run (default-cli) @ simple-spring-app ---

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.5.RELEASE)

2019-06-30 13:21:37.966  INFO 15588 --- [           main] com.rama.spring.app.Application          : Starting Application on SEGOTW10347728 with PI
 a311057 in C:\gitrama\simple-spring-boot)
2019-06-30 13:21:37.979  INFO 15588 --- [           main] com.rama.spring.app.Application          : No active profile set, falling back to default
2019-06-30 13:21:38.130  INFO 15588 --- [           main] ConfigServletWebServerApplicationContext : Refreshing org.springframework.boot.web.servle
6435a: startup date [Sun Jun 30 13:21:38 CEST 2019]; root of context hierarchy
2019-06-30 13:21:44.538  INFO 15588 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8081 (http)
2019-06-30 13:21:44.647  INFO 15588 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-06-30 13:21:44.648  INFO 15588 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/8.5.34
```

Creating Docker Image

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

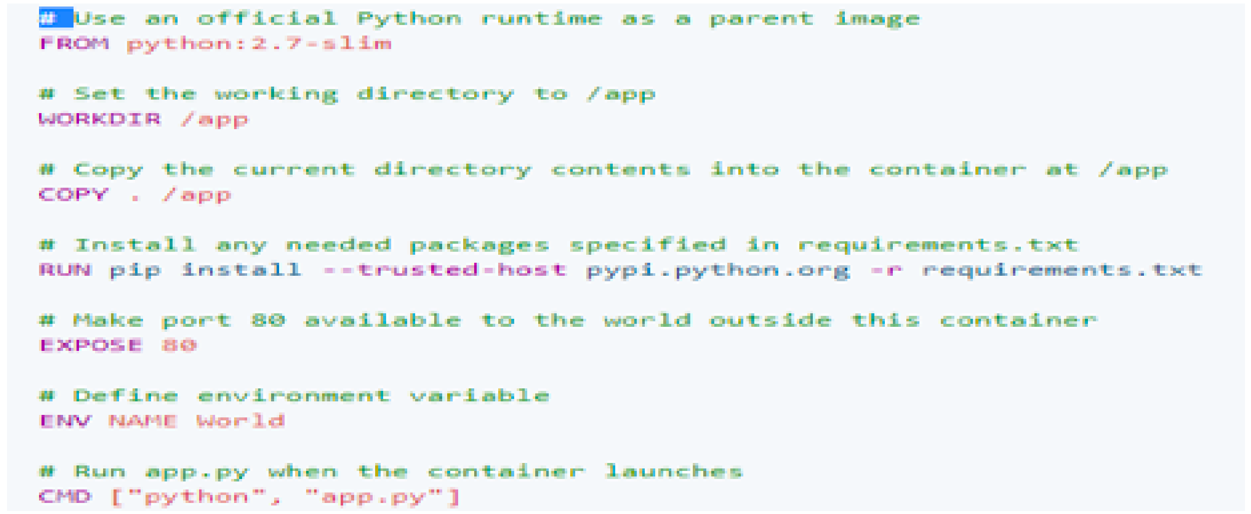
Dockerfile

- For Java:



```
# Use an official openjdk runtime as a parent image
FROM openjdk:8-jdk-alpine
# bind a volume to /tmp
VOLUME /tmp
#argument for the JAR_File
ARG JAR_FILE
#Copy the jarfile contents into the container at app.jar
COPY ${JAR_FILE} app.jar
#Entrypoint for the app.jar file to start.
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

- For Python:



```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

DockerFile

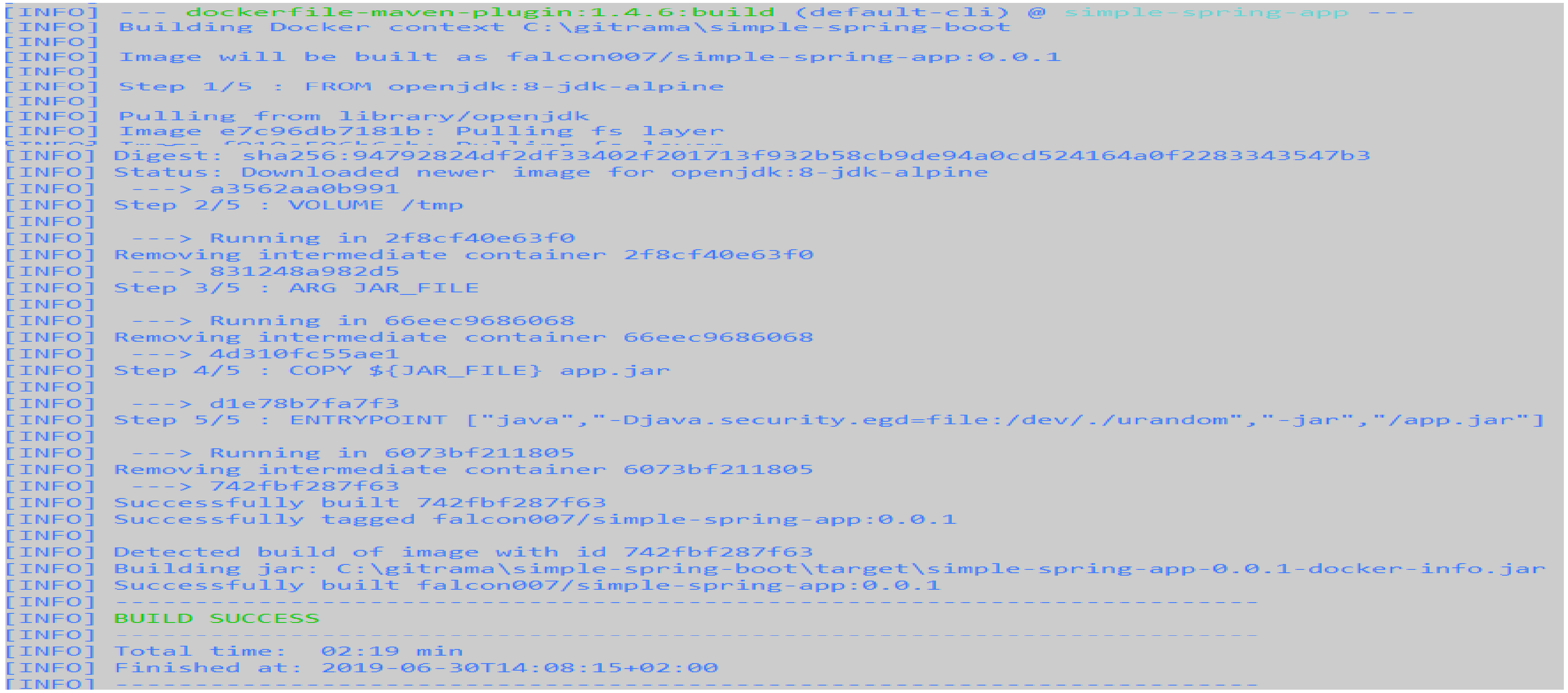
- Spotify Docker plugin
- fabric8 Docker plugin

- **FROM**: As base for our image we will take the *Java*-enabled *Alpine Linux*, created in the previous section
- **COPY**: We let *Docker* copy our jar file into the image
- **ENV**: This command lets us define some environment variables, which will be respected by the application running in the container. Here we define a customized *Spring Boot Application***configuration**, to hand-over to the jar-executable later
- **ENTRYPOINT/CMD**: This will be the executable to start when the container is booting. We must define them as *JSON-Array*, because we will use an *ENTRYPOINT* in combination with a *CMD* for some application arguments
- **VOLUME**: Because our container will be running in an isolated environment, with no direct network access, we have to define a mountpoint-placeholder for our configuration repository

Creating Docker Image

Creating With Maven

- mvn install dockerfile:build

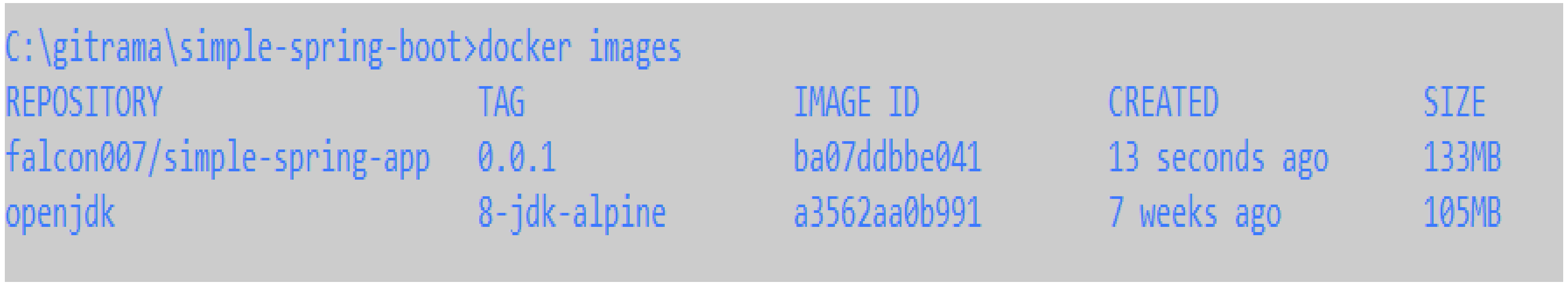


```
[INFO] --- dockerfile-maven-plugin:1.4.6:build (default-cli) @ simple-spring-app ---
[INFO] Building Docker context C:\gitrama\simple-spring-boot
[INFO]
[INFO] Image will be built as falcon007/simple-spring-app:0.0.1
[INFO]
[INFO] Step 1/5 : FROM openjdk:8-jdk-alpine
[INFO]
[INFO] Pulling from library/openjdk
[INFO] Image e7c96db7181b: Pulling fs layer
[illegible]
[INFO] Digest: sha256:94792824df2df33402f201713f932b58cb9de94a0cd524164a0f2283343547b3
[INFO] Status: Downloaded newer image for openjdk:8-jdk-alpine
[INFO]  ---> a3562aa0b991
[INFO] Step 2/5 : VOLUME /tmp
[INFO]
[INFO]  ---> Running in 2f8cf40e63f0
[INFO] Removing intermediate container 2f8cf40e63f0
[INFO]  ---> 831248a982d5
[INFO] Step 3/5 : ARG JAR_FILE
[INFO]
[INFO]  ---> Running in 66eec9686068
[INFO] Removing intermediate container 66eec9686068
[INFO]  ---> 4d310fc55ae1
[INFO] Step 4/5 : COPY ${JAR_FILE} app.jar
[INFO]
[INFO]  ---> d1e78b7fa7f3
[INFO] Step 5/5 : ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
[INFO]
[INFO]  ---> Running in 6073bf211805
[INFO] Removing intermediate container 6073bf211805
[INFO]  ---> 742fbf287f63
[INFO] Successfully built 742fbf287f63
[INFO] Successfully tagged falcon007/simple-spring-app:0.0.1
[INFO]
[INFO] Detected build of image with id 742fbf287f63
[INFO] Building jar: C:\gitrama\simple-spring-boot\target\simple-spring-app-0.0.1-docker-info.jar
[INFO] Successfully built falcon007/simple-spring-app:0.0.1
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:19 min
[INFO] Finished at: 2019-06-30T14:08:15+02:00
[INFO] ------------------------------------------------------------------------
```

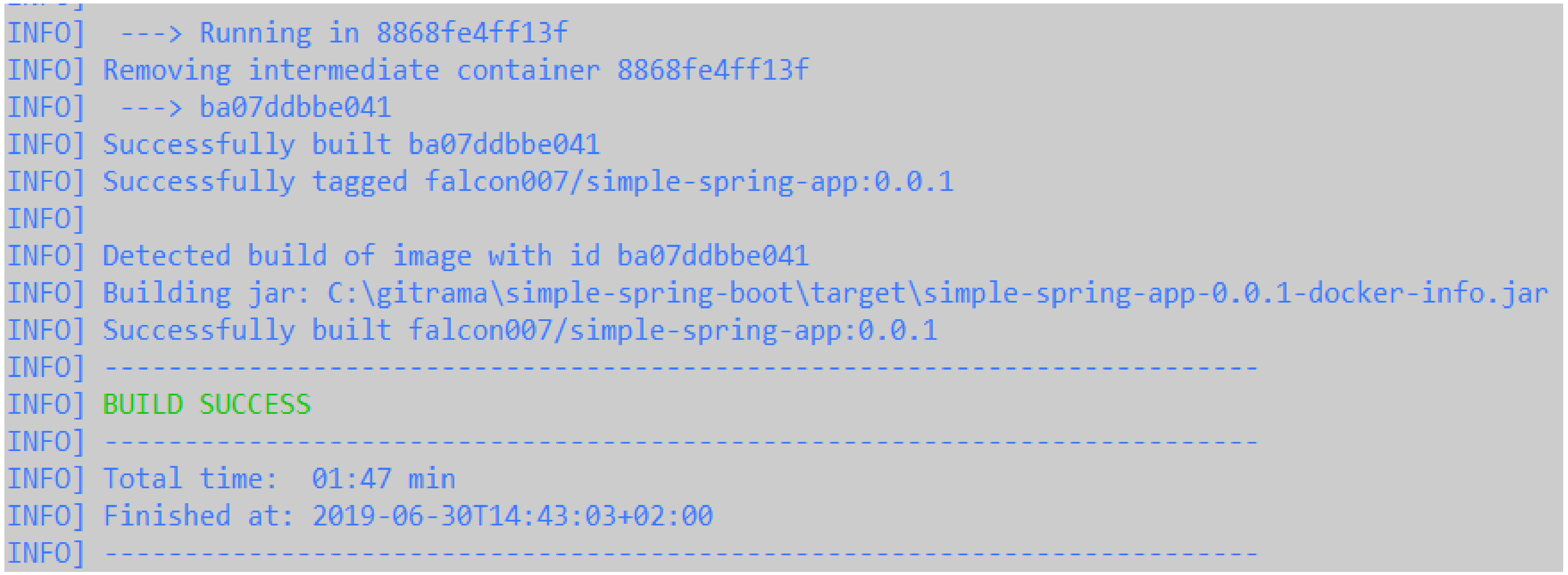
Creating With gradle

- gradle build docker

Docker image

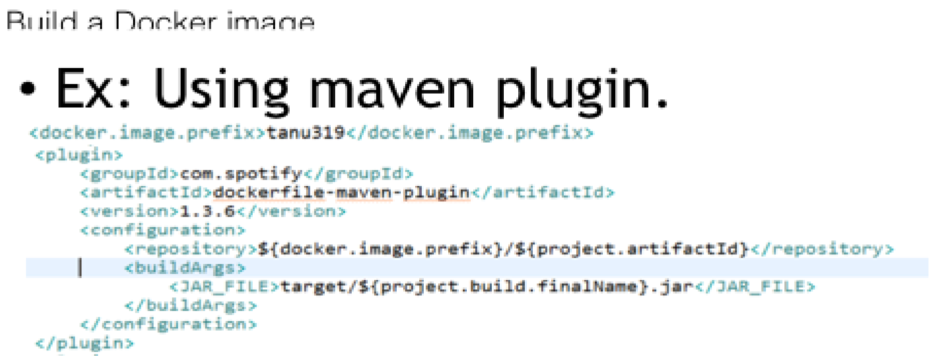


```
C:\gitrama\simple-spring-boot>docker images
REPOSITORY                    TAG             IMAGE ID        CREATED          SIZE
falcon007/simple-spring-app   0.0.1           ba07ddbbe041    13 seconds ago   133MB
openjdk                       8-jdk-alpine    a3562aa0b991    7 weeks ago      105MB
```



```
INFO]  ---> Running in 8868fe4ff13f
INFO] Removing intermediate container 8868fe4ff13f
INFO]  ---> ba07ddbbe041
INFO] Successfully built ba07ddbbe041
INFO] Successfully tagged falcon007/simple-spring-app:0.0.1
INFO]
INFO] Detected build of image with id ba07ddbbe041
INFO] Building jar: C:\gitrama\simple-spring-boot\target\simple-spring-app-0.0.1-docker-info.jar
INFO] Successfully built falcon007/simple-spring-app:0.0.1
INFO] ------------------------------------------------------------------------
INFO] BUILD SUCCESS
INFO] ------------------------------------------------------------------------
INFO] Total time:  01:47 min
INFO] Finished at: 2019-06-30T14:43:03+02:00
INFO] ------------------------------------------------------------------------
```

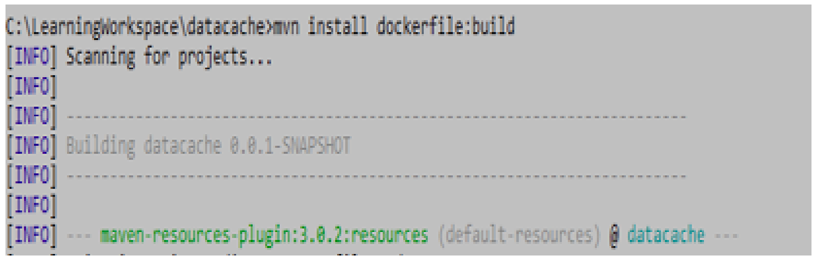
Simple Flow:



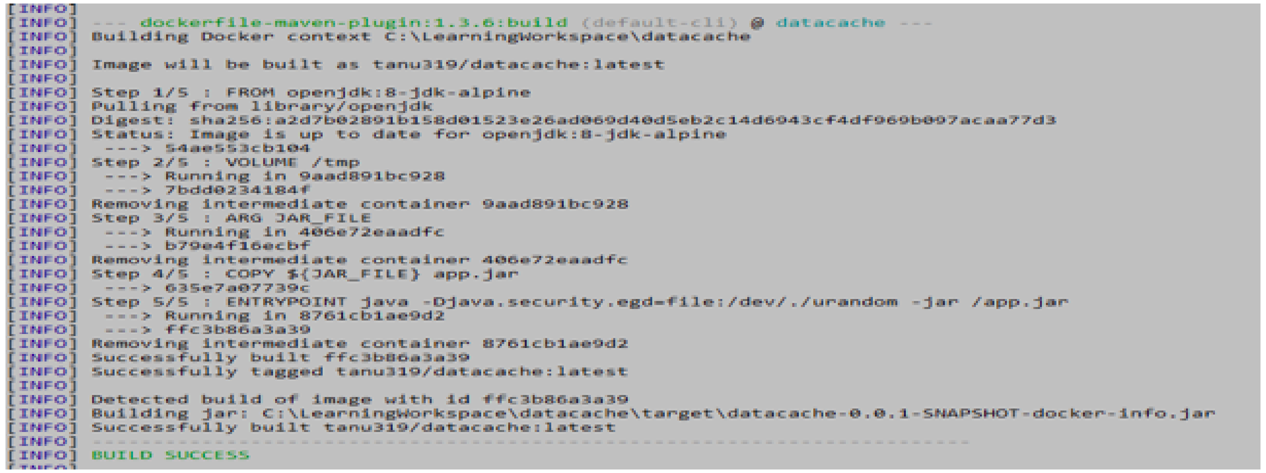
Build a Docker image

- Ex: Using maven plugin.

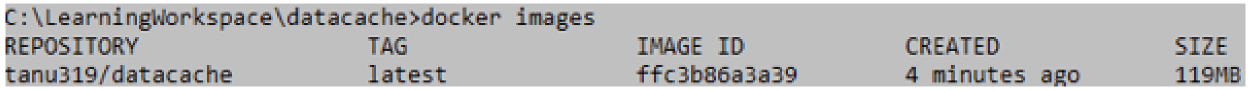
```
<docker.image.prefix>tanu319</docker.image.prefix>
 <plugin>
     <groupId>com.spotify</groupId>
     <artifactId>dockerfile-maven-plugin</artifactId>
     <version>1.3.6</version>
     <configuration>
         <repository>${docker.image.prefix}/${project.artifactId}</repository>
         <buildArgs>
             <JAR_FILE>target/${project.build.finalName}.jar</JAR_FILE>
         </buildArgs>
     </configuration>
 </plugin>
```



```
C:\LearningWorkspace\datacache>mvn install dockerfile:build
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building datacache 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ datacache ---
```



```
[INFO]
[INFO] --- dockerfile-maven-plugin:1.3.6:build (default-cli) @ datacache ---
[INFO] Building Docker context C:\LearningWorkspace\datacache
[INFO]
[INFO] Image will be built as tanu319/datacache:latest
[INFO]
[INFO] Step 1/5 : FROM openjdk:8-jdk-alpine
[INFO] Pulling from library/openjdk
[INFO] Digest: sha256:a2d7b02891b158d01523e26ad069d40d5eb2c14d6943cf4df969b097acaa77d3
[INFO] Status: Image is up to date for openjdk:8-jdk-alpine
[INFO]  ---> 54ae553cb104
[INFO] Step 2/5 : VOLUME /tmp
[INFO]  ---> Running in 9aad891bc928
[INFO]  ---> 7bdd0234184f
[INFO] Removing intermediate container 9aad891bc928
[INFO] Step 3/5 : ARG JAR_FILE
[INFO]  ---> Running in 406e72eaadfc
[INFO]  ---> b79e4f16ecbf
[INFO] Removing intermediate container 406e72eaadfc
[INFO] Step 4/5 : COPY ${JAR_FILE} app.jar
[INFO]  ---> 635e7a07739c
[INFO] Step 5/5 : ENTRYPOINT java -Djava.security.egd=file:/dev/./urandom -jar /app.jar
[INFO]  ---> Running in 8761cb1ae9d2
[INFO]  ---> ffc3b86a3a39
[INFO] Removing intermediate container 8761cb1ae9d2
[INFO] Successfully built ffc3b86a3a39
[INFO] Successfully tagged tanu319/datacache:latest
[INFO]
[INFO] Detected build of image with id ffc3b86a3a39
[INFO] Building jar: C:\LearningWorkspace\datacache\target\datacache-0.0.1-SNAPSHOT-docker-info.jar
[INFO] Successfully built tanu319/datacache:latest
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```



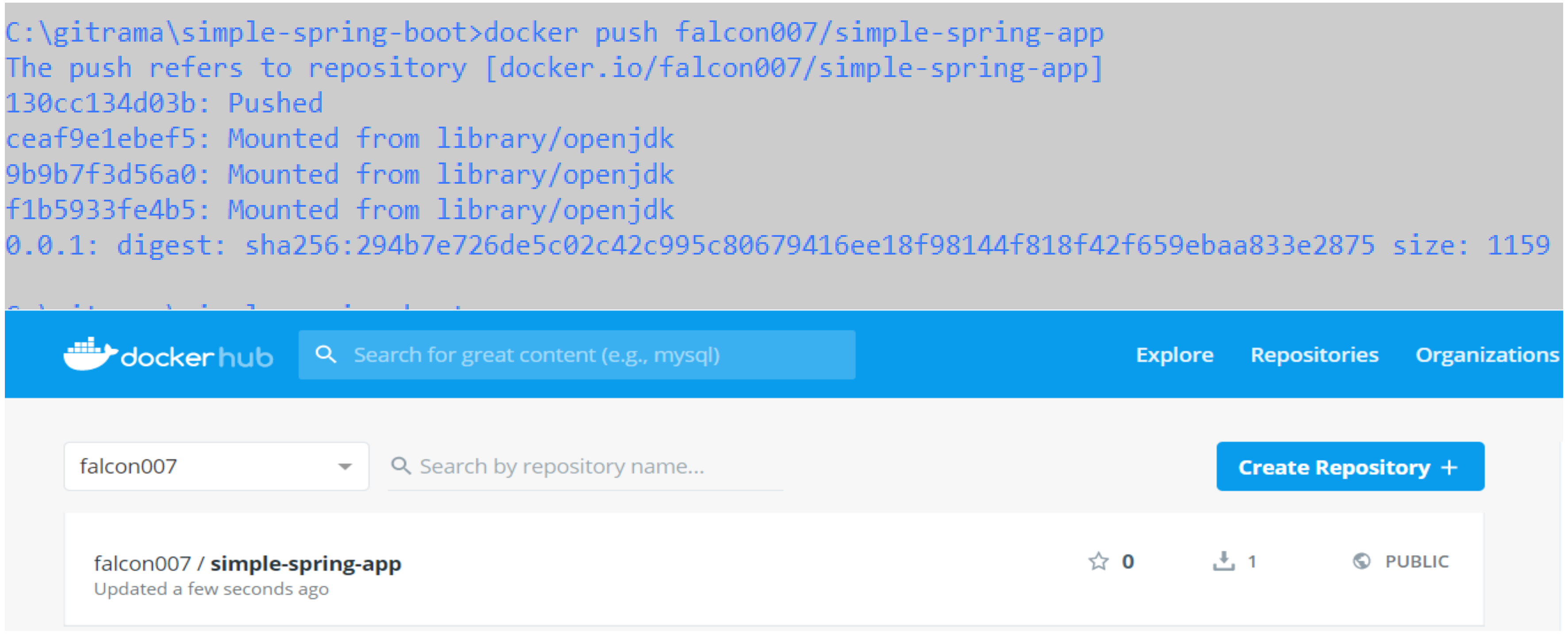
```
C:\LearningWorkspace\datacache>docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
tanu319/datacache           latest              ffc3b86a3a39        4 minutes ago       119MB
```

Understand More Detail:

- https://www.youtube.com/watch?v=q7ERqUOE1Bw&t=1045s

Push docker image

- docker login
- Docker push
- docker push falcon007/simple-spring-app



```
C:\gitrama\simple-spring-boot>docker push falcon007/simple-spring-app
The push refers to repository [docker.io/falcon007/simple-spring-app]
130cc134d03b: Pushed
ceaf9e1ebef5: Mounted from library/openjdk
9b9b7f3d56a0: Mounted from library/openjdk
f1b5933fe4b5: Mounted from library/openjdk
0.0.1: digest: sha256:294b7e726de5c02c42c995c80679416ee18f98144f818f42f659ebaa833e2875 size: 1159
```

falcon007 / **simple-spring-app**
Updated a few seconds ago

0 1 PUBLIC

Push Docker image

Push Image

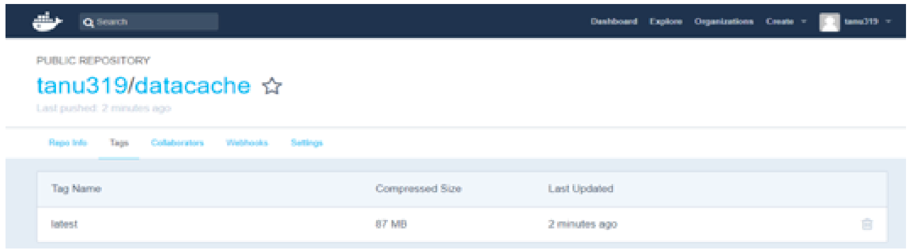
- Login to docker hub using command

```
C:\LearningWorkspace\datacache>docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username (tanu319): tanu319
Password:
Login Succeeded
```

- Push image to docker hub

```
C:\LearningWorkspace\datacache>docker push tanu319/datacache
The push refers to a repository [docker.io/tanu319/datacache]
cd339a0120c4: Pushed
f2ec1bba02a6: Layer already exists
0c3170905795: Layer already exists
df64d3292fd6: Layer already exists
latest: digest: sha256:57fd9d87cf907a8ed939880e28eb69c8baffc2a04fb83ba67714653ba9c14e40 size: 1159
```

- View Image in docker hub.



PUBLIC REPOSITORY

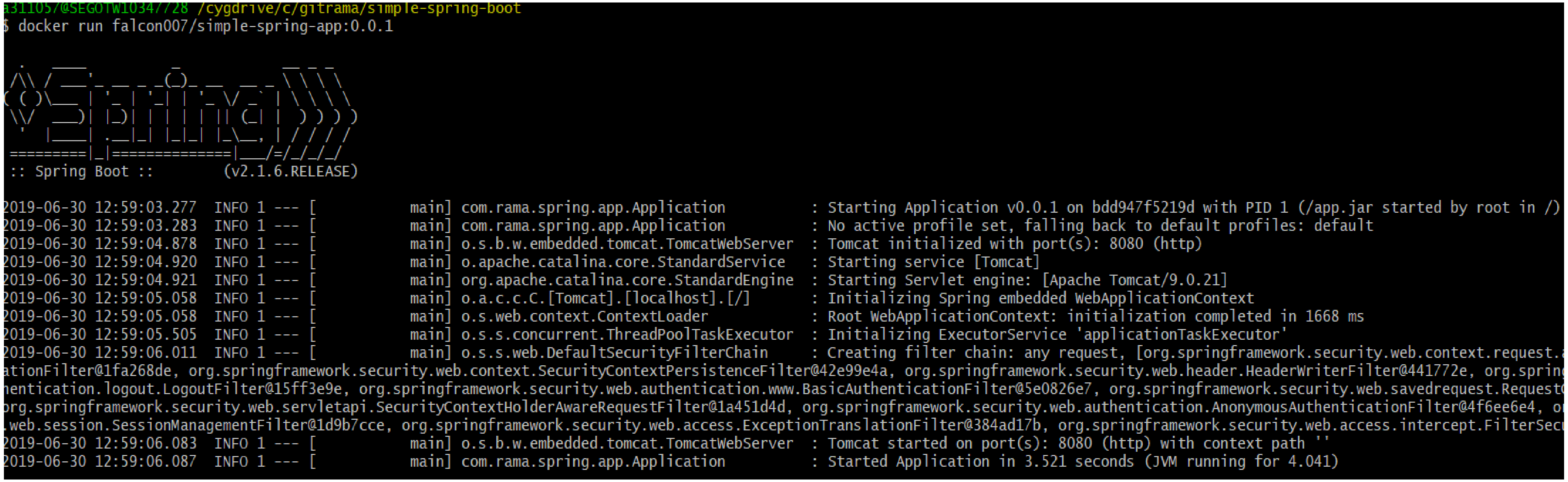
tanu319/datacache

Last pushed: 2 minutes ago

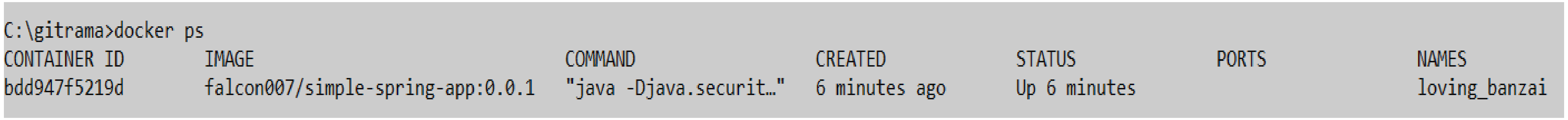
Tag Name	Compressed Size	Last Updated
latest	87 MB	2 minutes ago

Docker Run:failed

- docker run falcon007/simple-spring-app:0.0.1



```
a311057@SEGOTW10347728 /cygdrive/c/gitrama/simple-spring-boot
$ docker run falcon007/simple-spring-app:0.0.1

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.6.RELEASE)

2019-06-30 12:59:03.277  INFO 1 --- [           main] com.rama.spring.app.Application          : Starting Application v0.0.1 on bdd947f5219d with PID 1 (/app.jar started by root in /)
2019-06-30 12:59:03.283  INFO 1 --- [           main] com.rama.spring.app.Application          : No active profile set, falling back to default profiles: default
2019-06-30 12:59:04.878  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2019-06-30 12:59:04.920  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-06-30 12:59:04.921  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.21]
2019-06-30 12:59:05.058  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-06-30 12:59:05.058  INFO 1 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1668 ms
2019-06-30 12:59:05.505  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2019-06-30 12:59:06.011  INFO 1 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Creating filter chain: any request, [org.springframework.security.web.context.request.a
ationFilter@1fa268de, org.springframework.security.web.context.SecurityContextPersistenceFilter@42e99e4a, org.springframework.security.web.header.HeaderWriterFilter@441772e, org.spring
hentication.logout.LogoutFilter@15ff3e9e, org.springframework.security.web.authentication.www.BasicAuthenticationFilter@5e0826e7, org.springframework.security.web.savedrequest.RequestC
org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@1a451d4d, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@4f6ee6e4, or
.web.session.SessionManagementFilter@1d9b7cce, org.springframework.security.web.access.ExceptionTranslationFilter@384ad17b, org.springframework.security.web.access.intercept.FilterSecu
2019-06-30 12:59:06.083  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2019-06-30 12:59:06.087  INFO 1 --- [           main] com.rama.spring.app.Application          : Started Application in 3.521 seconds (JVM running for 4.041)
```

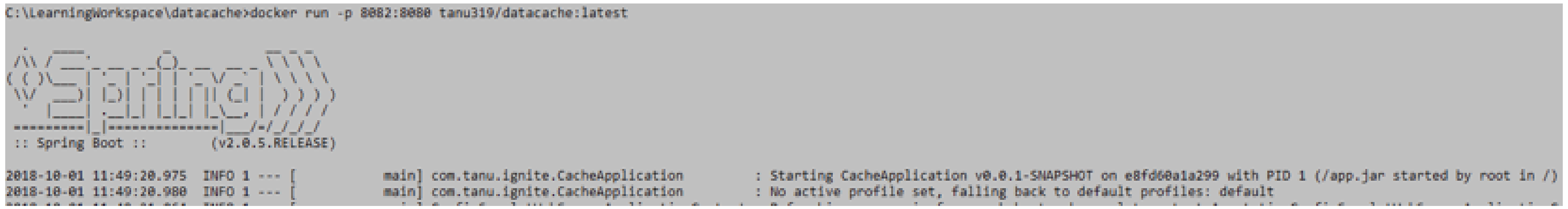


```
C:\gitrama>docker ps
CONTAINER ID        IMAGE                               COMMAND                  CREATED             STATUS              PORTS               NAMES
bdd947f5219d        falcon007/simple-spring-app:0.0.1   "java -Djava.securit…"   6 minutes ago       Up 6 minutes                            loving_banzai
```

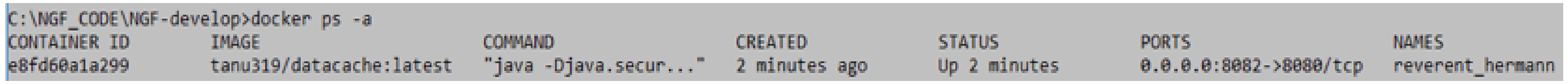
Docker Run With Port

Start application:

```
docker run -p exposed_port:running_port username/repository:tag
```

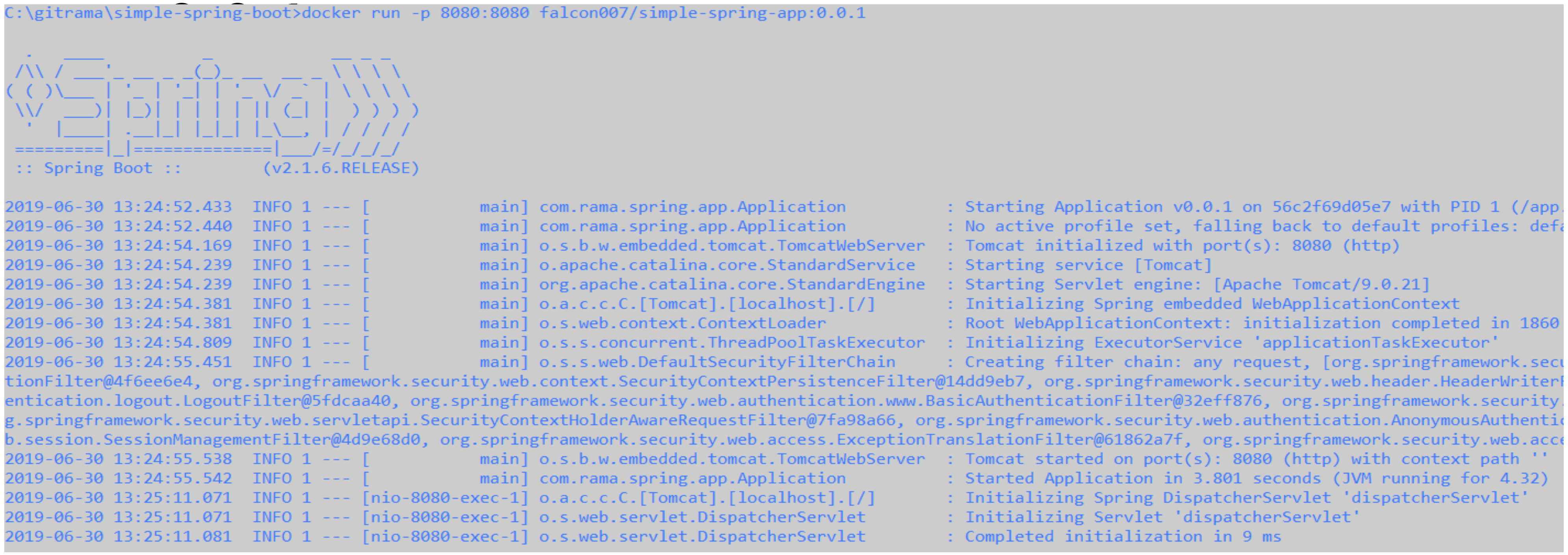


```
C:\LearningWorkspace\datacache>docker run -p 8082:8080 tanu319/datacache:latest

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.5.RELEASE)

2018-10-01 11:49:20.975  INFO 1 --- [           main] com.tanu.ignite.CacheApplication         : Starting CacheApplication v0.0.1-SNAPSHOT on e8fd60a1a299 with PID 1 (/app.jar started by root in /)
2018-10-01 11:49:20.980  INFO 1 --- [           main] com.tanu.ignite.CacheApplication         : No active profile set, falling back to default profiles: default
```

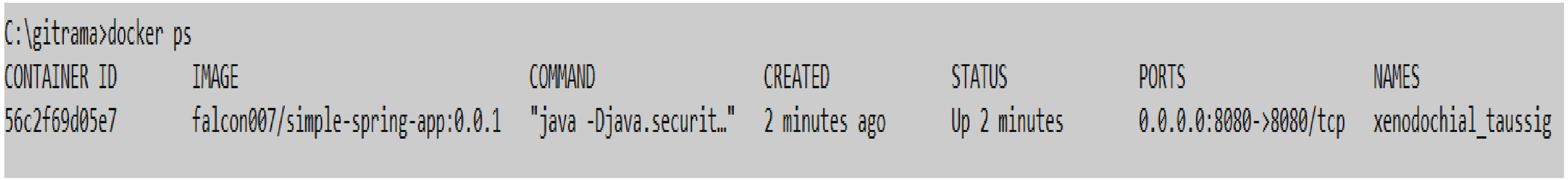


```
C:\NGF_CODE\NGF-develop>docker ps -a
CONTAINER ID        IMAGE                     COMMAND                  CREATED             STATUS              PORTS                    NAMES
e8fd60a1a299        tanu319/datacache:latest  "java -Djava.secur..."   2 minutes ago       Up 2 minutes        0.0.0.0:8082->8080/tcp   reverent_hermann
```

- docker run –p exposed_port:running_port repository:tag
- docker run -p 8080:8080 falcon007/simple-spring-



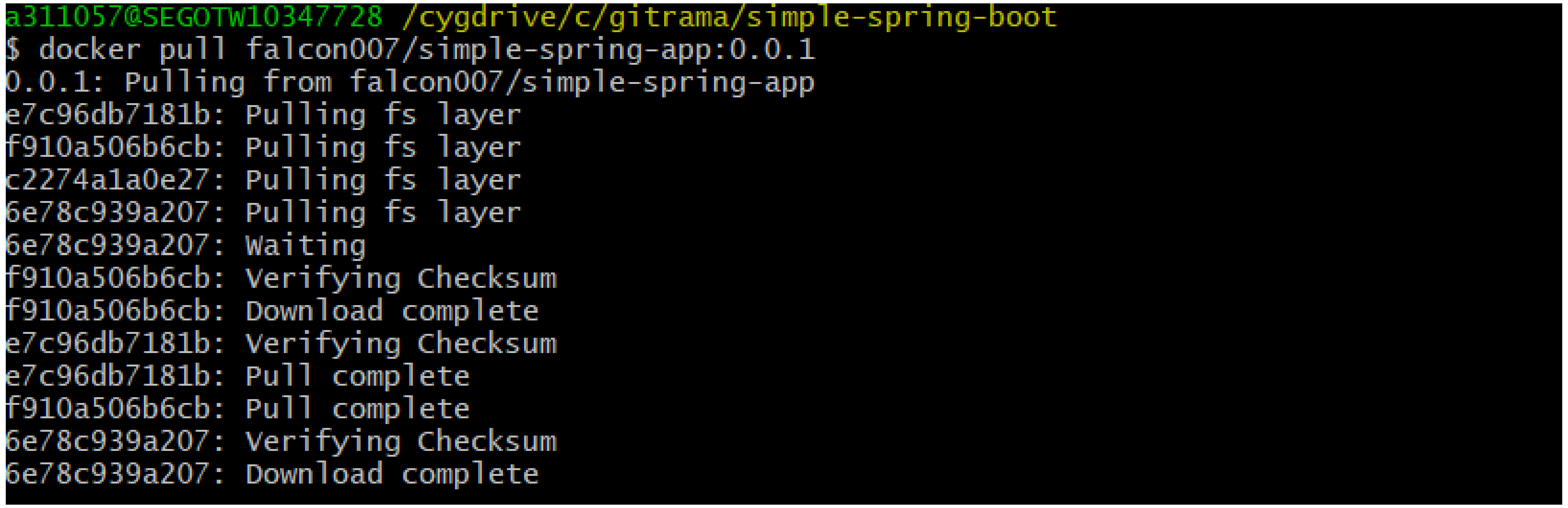
```
C:\gitrama\simple-spring-boot>docker run -p 8080:8080 falcon007/simple-spring-app:0.0.1

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.1.6.RELEASE)

2019-06-30 13:24:52.433  INFO 1 --- [           main] com.rama.spring.app.Application          : Starting Application v0.0.1 on 56c2f69d05e7 with PID 1 (/app.
2019-06-30 13:24:52.440  INFO 1 --- [           main] com.rama.spring.app.Application          : No active profile set, falling back to default profiles: defa
2019-06-30 13:24:54.169  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2019-06-30 13:24:54.239  INFO 1 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-06-30 13:24:54.239  INFO 1 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.21]
2019-06-30 13:24:54.381  INFO 1 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-06-30 13:24:54.381  INFO 1 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 1860
2019-06-30 13:24:54.809  INFO 1 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2019-06-30 13:24:55.451  INFO 1 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Creating filter chain: any request, [org.springframework.secu
tionFilter@4f6ee6e4, org.springframework.security.web.context.SecurityContextPersistenceFilter@14dd9eb7, org.springframework.security.web.header.HeaderWriterF
entication.logout.LogoutFilter@5fdcaa40, org.springframework.security.web.authentication.www.BasicAuthenticationFilter@32eff876, org.springframework.security.
g.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@7fa98a66, org.springframework.security.web.authentication.AnonymousAuthentic
b.session.SessionManagementFilter@4d9e68d0, org.springframework.security.web.access.ExceptionTranslationFilter@61862a7f, org.springframework.security.web.acce
2019-06-30 13:24:55.538  INFO 1 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2019-06-30 13:24:55.542  INFO 1 --- [           main] com.rama.spring.app.Application          : Started Application in 3.801 seconds (JVM running for 4.32)
2019-06-30 13:25:11.071  INFO 1 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2019-06-30 13:25:11.071  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2019-06-30 13:25:11.081  INFO 1 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 9 ms
```



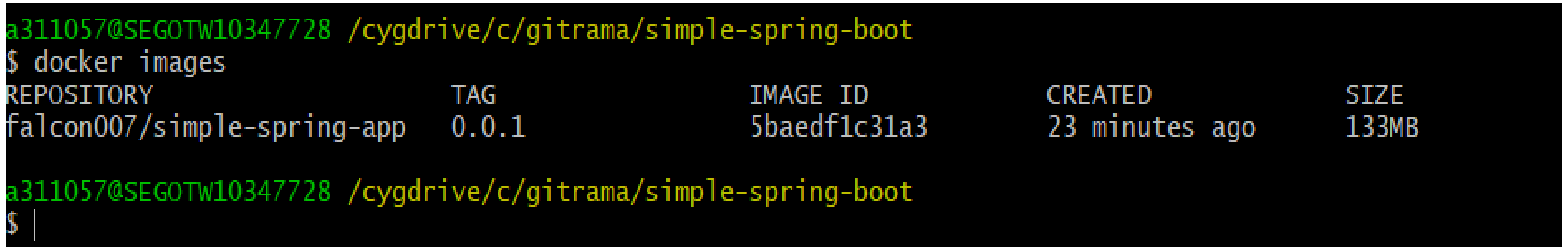
```
C:\gitrama>docker ps
CONTAINER ID   IMAGE                             COMMAND                  CREATED         STATUS         PORTS                    NAMES
56c2f69d05e7   falcon007/simple-spring-app:0.0.1 "java -Djava.securit…"   2 minutes ago   Up 2 minutes   0.0.0.0:8080->8080/tcp   xenodochial_taussig
```

Docker pull

- docker pull falcon007/simple-spring-app:0.0.1



```
a311057@SEGOTW10347728 /cygdrive/c/gitrama/simple-spring-boot
$ docker pull falcon007/simple-spring-app:0.0.1
0.0.1: Pulling from falcon007/simple-spring-app
e7c96db7181b: Pulling fs layer
f910a506b6cb: Pulling fs layer
c2274a1a0e27: Pulling fs layer
6e78c939a207: Pulling fs layer
6e78c939a207: Waiting
f910a506b6cb: Verifying Checksum
f910a506b6cb: Download complete
e7c96db7181b: Verifying Checksum
e7c96db7181b: Pull complete
f910a506b6cb: Pull complete
6e78c939a207: Verifying Checksum
6e78c939a207: Download complete
```



```
a311057@SEGOTW10347728 /cygdrive/c/gitrama/simple-spring-boot
$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
falcon007/simple-spring-app   0.0.1               5baedf1c31a3        23 minutes ago      133MB

a311057@SEGOTW10347728 /cygdrive/c/gitrama/simple-spring-boot
$
```

Docker Network

- https://docs.docker.com/network/

One of the reasons Docker containers and services are so powerful is that you can connect them together, or connect them to non-Docker workloads. Docker containers and services do not even need to be aware that they are deployed on Docker, or whether their peers are also Docker workloads or not. Whether your Docker hosts run Linux, Windows, or a mix of the two, you can use Docker to manage them in a platform-agnostic way.

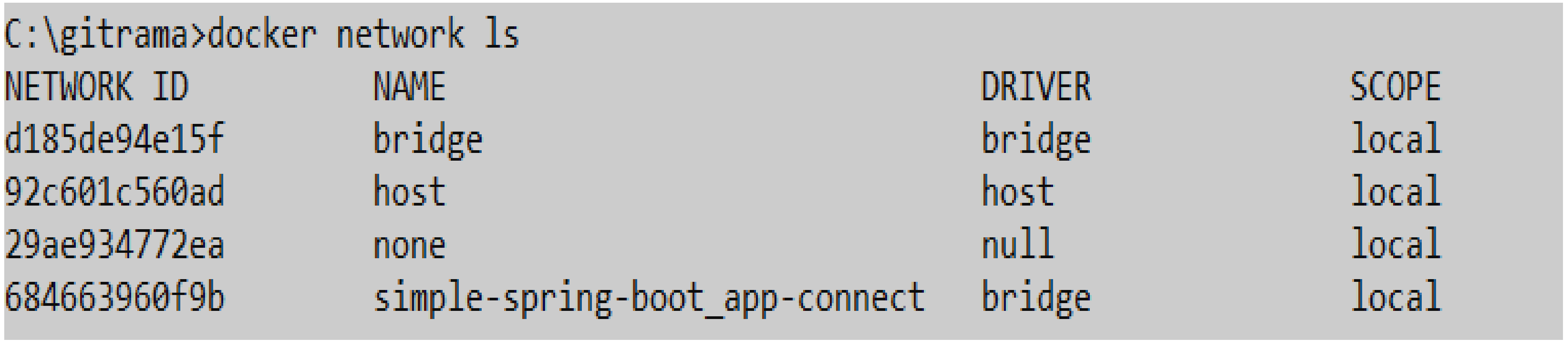
Network drivers:

- Docker’s networking subsystem is pluggable, using drivers. Several drivers exist by default, and provide core networking functionality:
- **bridge**: The default network driver. If you don’t specify a driver, this is the type of network you are creating. Bridge networks are usually used when your applications run in standalone containers that need to communicate.
- **host**: For standalone containers, remove network isolation between the container and the Docker host, and use the host’s networking directly. host is only available for swarm services on Docker 17.06 and higher.
- **overlay**: Overlay networks connect multiple Docker daemons together and enable swarm services to communicate with each other. You can also use overlay networks to facilitate communication between a swarm service and a standalone container, or between two standalone containers on different Docker daemons. This strategy removes the need to do OS-level routing between these containers.
- **macvlan**: Macvlan networks allow you to assign a MAC address to a container, making it appear as a physical device on your network. The Docker daemon routes traffic to containers by their MAC addresses. Using the macvlan driver is sometimes the best choice when dealing with legacy applications that expect to be directly connected to the physical network, rather than routed through the Docker host’s network stack.
- **none**: For this container, disable all networking. Usually used in conjunction with a custom network driver. none is not available for swarm services. See disable container networking.
- **Network plugins**: You can install and use third-party network plugins with Docker. These plugins are available from Docker Hub or from third-party vendors. See the vendor’s documentation for installing and using a given network plugin.

Commands

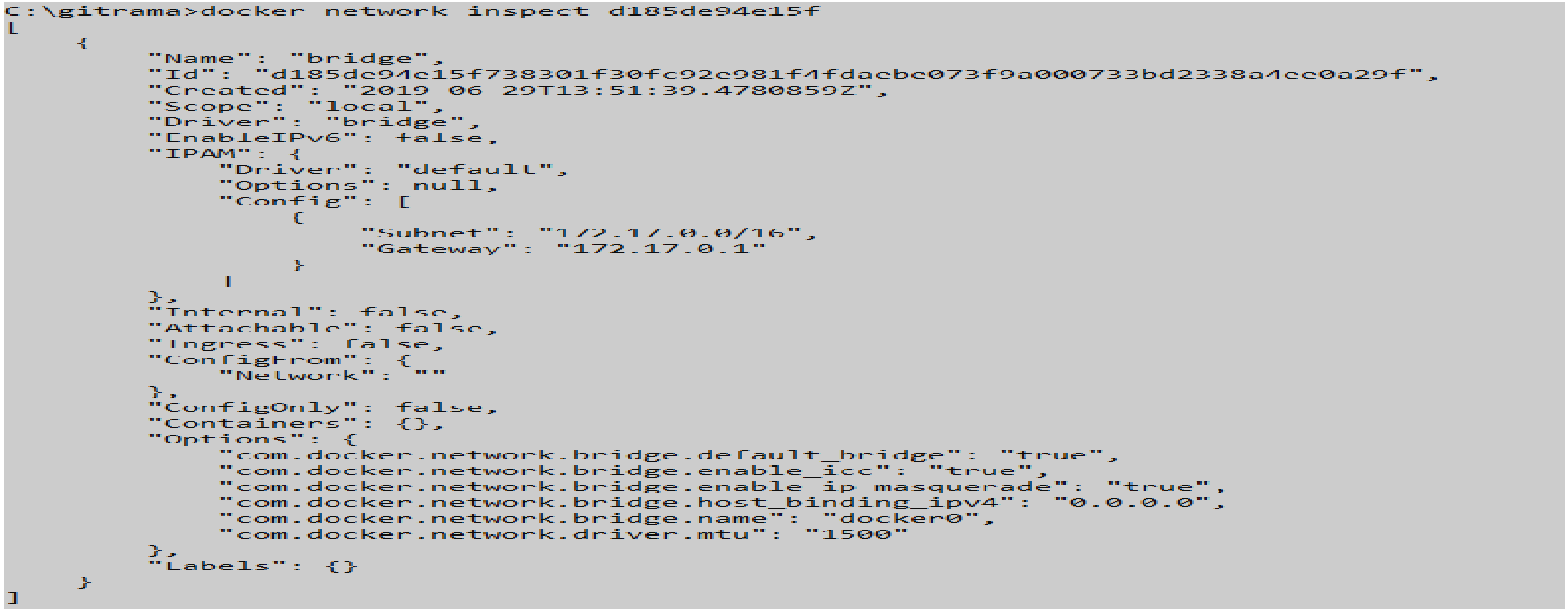
Command	Description
docker network connect	Connect a container to a network
docker network create	Create a network
docker network disconnect	Disconnect a container from a network
docker network inspect	Display detailed information on one or more networks
docker network ls	List networks
docker network prune	Remove all unused networks
docker network rm	Remove one or more networks

Docker network List



```
C:\gitrama>docker network ls
NETWORK ID          NAME                                DRIVER              SCOPE
d185de94e15f        bridge                              bridge              local
92c601c560ad        host                                host                local
29ae934772ea        none                                null                local
684663960f9b        simple-spring-boot_app-connect      bridge              local
```

Network Inspect:bridge



```
C:\gitrama>docker network inspect d185de94e15f
[
    {
        "Name": "bridge",
        "Id": "d185de94e15f738301f30fc92e981f4fdaebe073f9a000733bd2338a4ee0a29f",
        "Created": "2019-06-29T13:51:39.4780859Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
```

https://docs.docker.com/network/bridge/

- Bridge networks apply to containers running on the same Docker daemon host. For communication among containers running on different Docker daemon hosts, you can either manage routing at the OS level, or you can use an overlay network.

- When you start Docker, a default bridge network (also called bridge) is created automatically, and newly-started containers connect to it unless otherwise specified. You can also create user-defined custom bridge networks. User-defined bridge networks are superior to the default bridge network.

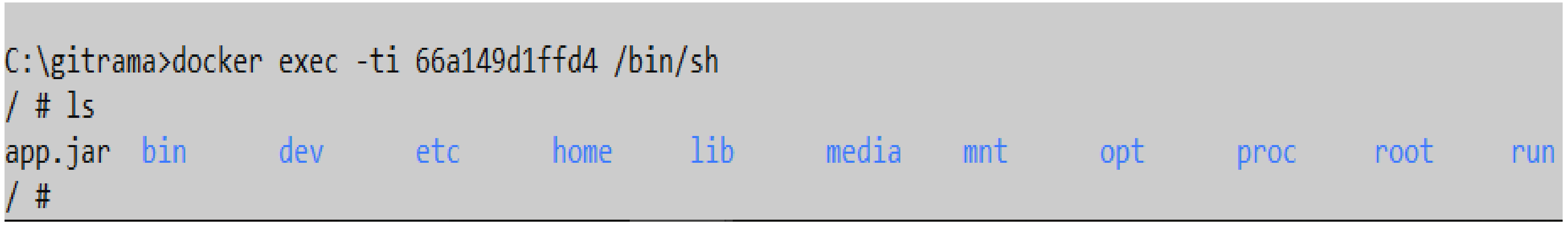
Port Mapping

By default, when you create a container, it does not publish any of its ports to the outside world. To make a port available to services outside of Docker, or to Docker containers which are not connected to the container’s network, use the --publish or -p flag. This creates a firewall rule which maps a container port to a port on the Docker host. Here are some examples.

Flag value	**Description**
-p 8080:80	Map TCP port 80 in the container to port 8080 on the Docker host.
-p 192.168.1.100:8080:80	Map TCP port 80 in the container to port 8080 on the Docker host for connections to host IP 192.168.1.100.
-p 8080:80/udp	Map UDP port 80 in the container to port 8080 on the Docker host.
-p 8080:80/tcp -p 8080:80/udp	Map TCP port 80 in the container to TCP port 8080 on the Docker host, and map UDP port 80 in the container to UDP port 8080 on the Docker host.

Container Inspection:

- docker exec -ti 66a149d1ffd4 /bin/sh



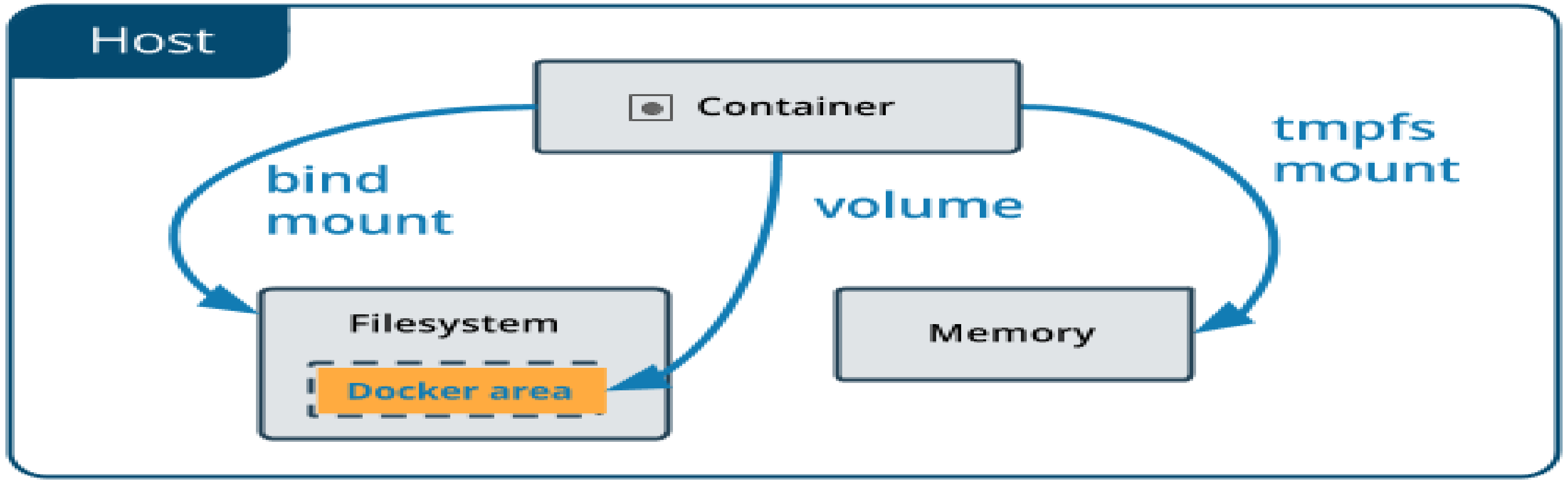
```
C:\gitrama>docker exec -ti 66a149d1ffd4 /bin/sh
/ # ls
app.jar  bin      dev      etc      home     lib      media    mnt      opt      proc     root     run
/ #
```

Docker Volume:

Docker volumes are very useful when we need to persist data in Docker containers or share data between containers.

Docker volumes are important because when a Docker container is destroyed, it’s entire file system is destroyed too. So if we want to keep this data, it is necessary that we use Docker volumes.

Docker volumes are attached to containers during a docker run command by using the -v flag



Bind mounts

- A bind mount is just mapping a directory on the host machine to a directory in the container. However, when the container is removed, it does not affect the directory.
- If the -v or --volume flag’s value is a path, then it is assumed to be a bind mount. If the directory does not exist, then it will be created. When we do this, the directory /path/to/app/directory will contain the same files as /var/www in the container.
- Ex:docker run -d --name rama_app -v /path/to/app/directory:/var/www falcon007/simple-spring-app:0.0.1

Docker Volume:

- Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure of the host machine, volumes are completely managed by Docker. Volumes have several advantages over bind mounts:

- Volumes are easier to back up or migrate than bind mounts.
- You can manage volumes using Docker CLI commands or the Docker API.
- Volumes work on both Linux and Windows containers.
- Volumes can be more safely shared among multiple containers.
- Volume drivers allow you to store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
- A new volume’s contents can be pre-populated by a container.
- In addition, volumes are often a better choice than persisting data in a container’s writable layer, because using a volume does not increase the size of containers using it, and the volume’s contents exist outside the lifecycle of a given container.

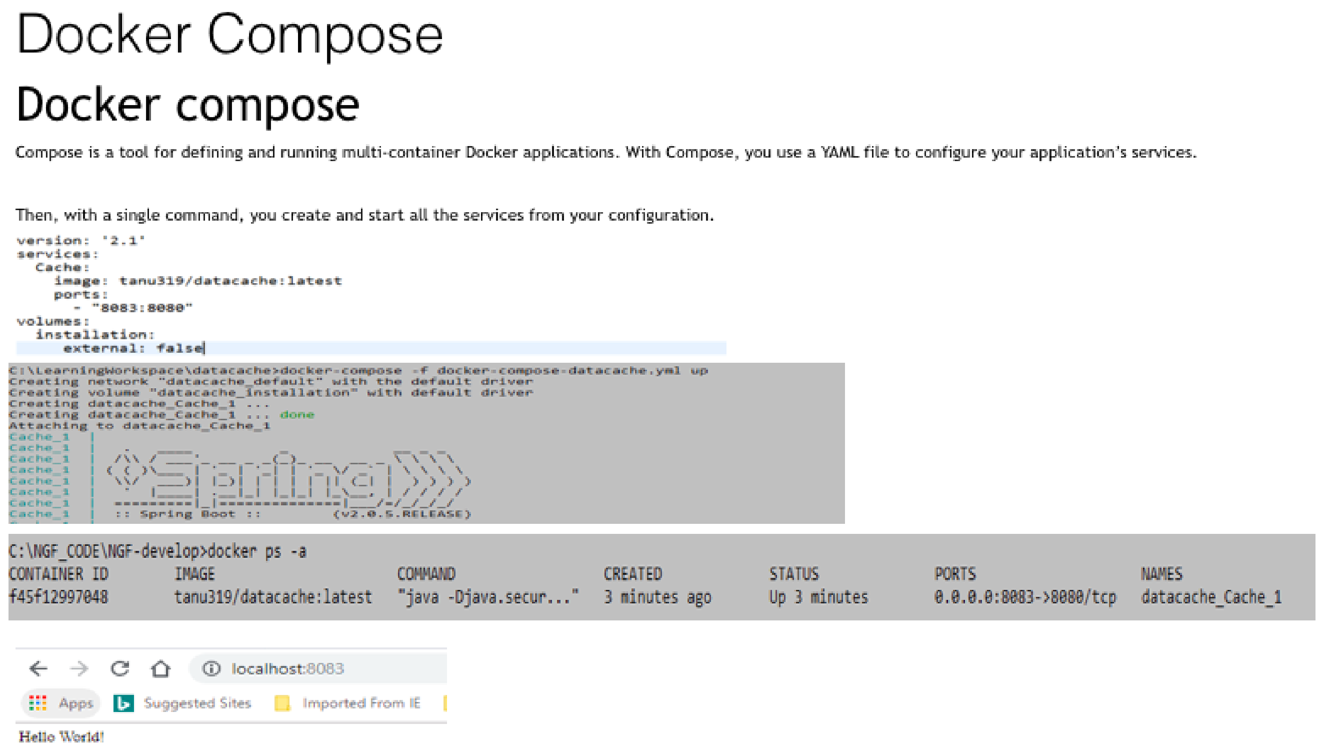
Attaching Volume:

- https://stephenafamo.com/blog/docker-volumes-introduction/

Docker cheet sheet

- docker rmi -f $(docker images -a -q)
- docker start $(docker ps -a -q --filter "status=exited")
- docker restart $(docker ps -a -q)

Docker composer



Docker Compose

Docker compose

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services.

Then, with a single command, you create and start all the services from your configuration.

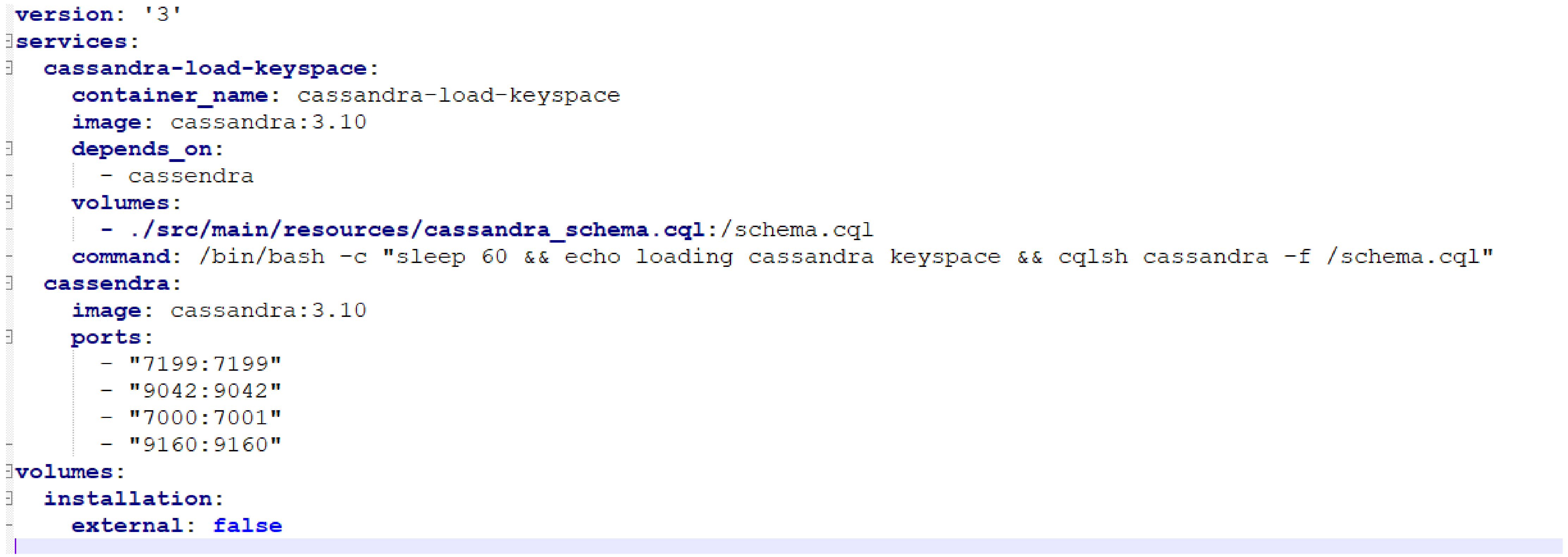
```
version: '2.1'
services:
  Cache:
    image: tanu319/datacache:latest
    ports:
      - "8083:8080"
volumes:
  installation:
     external: false
```

```
C:\LearningWorkspace\datacache>docker-compose -f docker-compose-datacache.yml up
Creating network "datacache_default" with the default driver
Creating volume "datacache_installation" with default driver
Creating datacache_Cache_1 ...
Creating datacache_Cache_1 ... done
Attaching to datacache_Cache_1
Cache_1  |
Cache_1  |   .   ____          _            __ _ _
Cache_1  |  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
Cache_1  | ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
Cache_1  |  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
Cache_1  |   '  |____| .__|_| |_|_| |_\__, | / / / /
Cache_1  |  =========|_|==============|___/=/_/_/_/
Cache_1  |  :: Spring Boot ::        (v2.0.5.RELEASE)
```

```
C:\NGF_CODE\NGF-develop>docker ps -a
CONTAINER ID        IMAGE                       COMMAND                  CREATED             STATUS              PORTS                    NAMES
f45f12997048        tanu319/datacache:latest    "java -Djava.secur..."   3 minutes ago       Up 3 minutes        0.0.0.0:8083->8080/tcp   datacache_Cache_1
```

Hello World!

Composer



```yaml
version: '3'
services:
  cassandra-load-keyspace:
    container_name: cassandra-load-keyspace
    image: cassandra:3.10
    depends_on:
      - cassandra
    volumes:
      - ./src/main/resources/cassandra_schema.cql:/schema.cql
    command: /bin/bash -c "sleep 60 && echo loading cassandra keyspace && cqlsh cassandra -f /schema.cql"
  cassendra:
    image: cassandra:3.10
    ports:
      - "7199:7199"
      - "9042:9042"
      - "7000:7001"
      - "9160:9160"
volumes:
  installation:
    external: false
```