Docker for Java Developers

Fabiane Nardon, @fabianenardon
Arun Gupta, @arungupta

Java Champion

Duke's Choice Award Winner (2 years)

SouJava Founder

Data Scientist / Big Data expert

Docker Captain

Java Champion

JavaOne Rock Star (4 years)

NetBeans Dream Team

Silicon Valley JUG Leader

Author

Runner

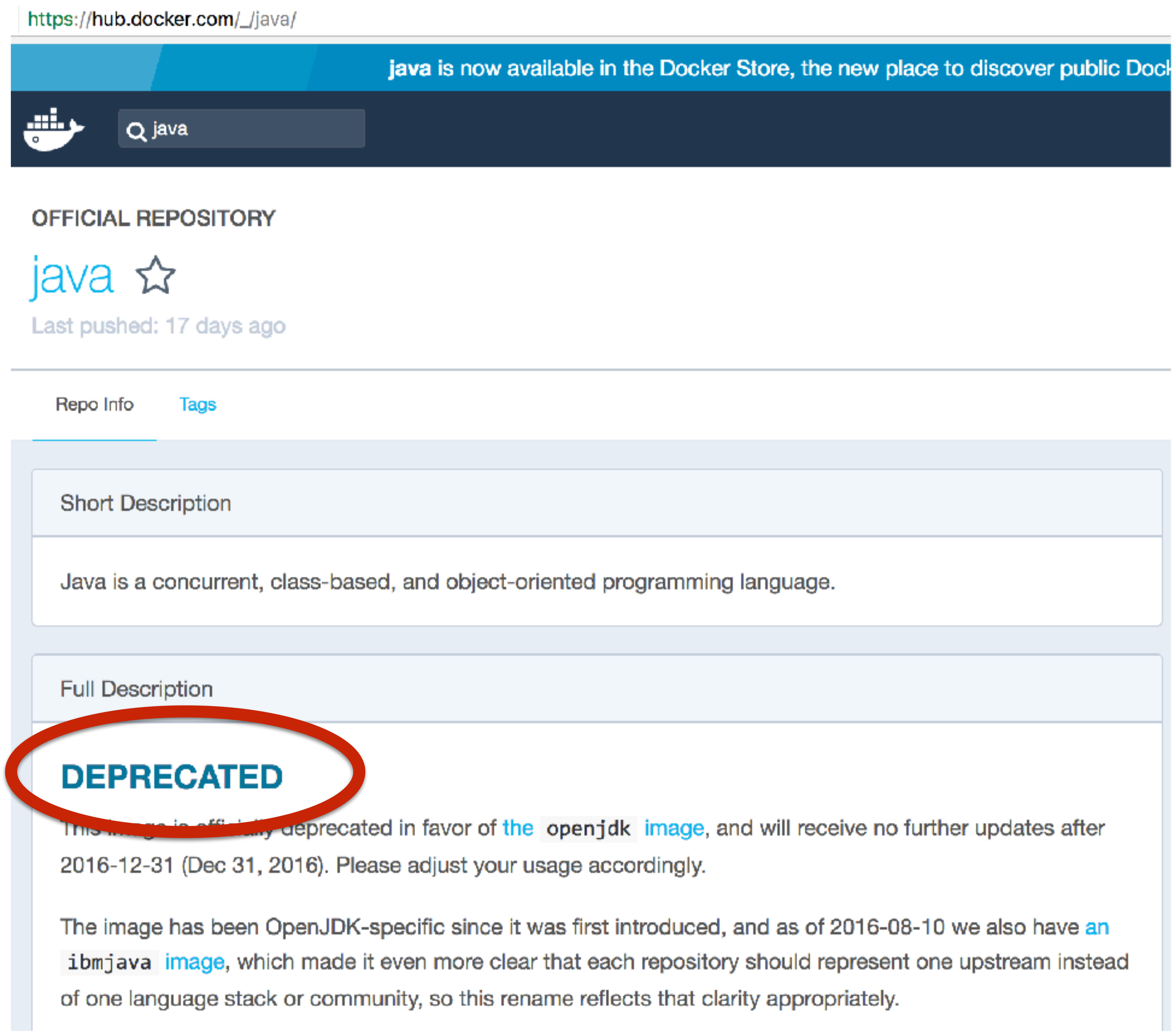
Lifelong learner

What we plan to cover?

- Java Base Image
- Package Java application using Maven and Gradle
- Multi-container application on single/multiple host(s)
- Scaling apps on AWS or Azure
- Memory management for Java Applications
- Debugging Java Applications
- Monitor Java Applications
- Integration Testing

Java Base Image

Java base image #1


https://hub.docker.com/_/java/

java is now available in the Docker Store, the new place to discover public Dock

OFFICIAL REPOSITORY

java

Last pushed: 17 days ago

Short Description

Java is a concurrent, class-based, and object-oriented programming language.

Full Description

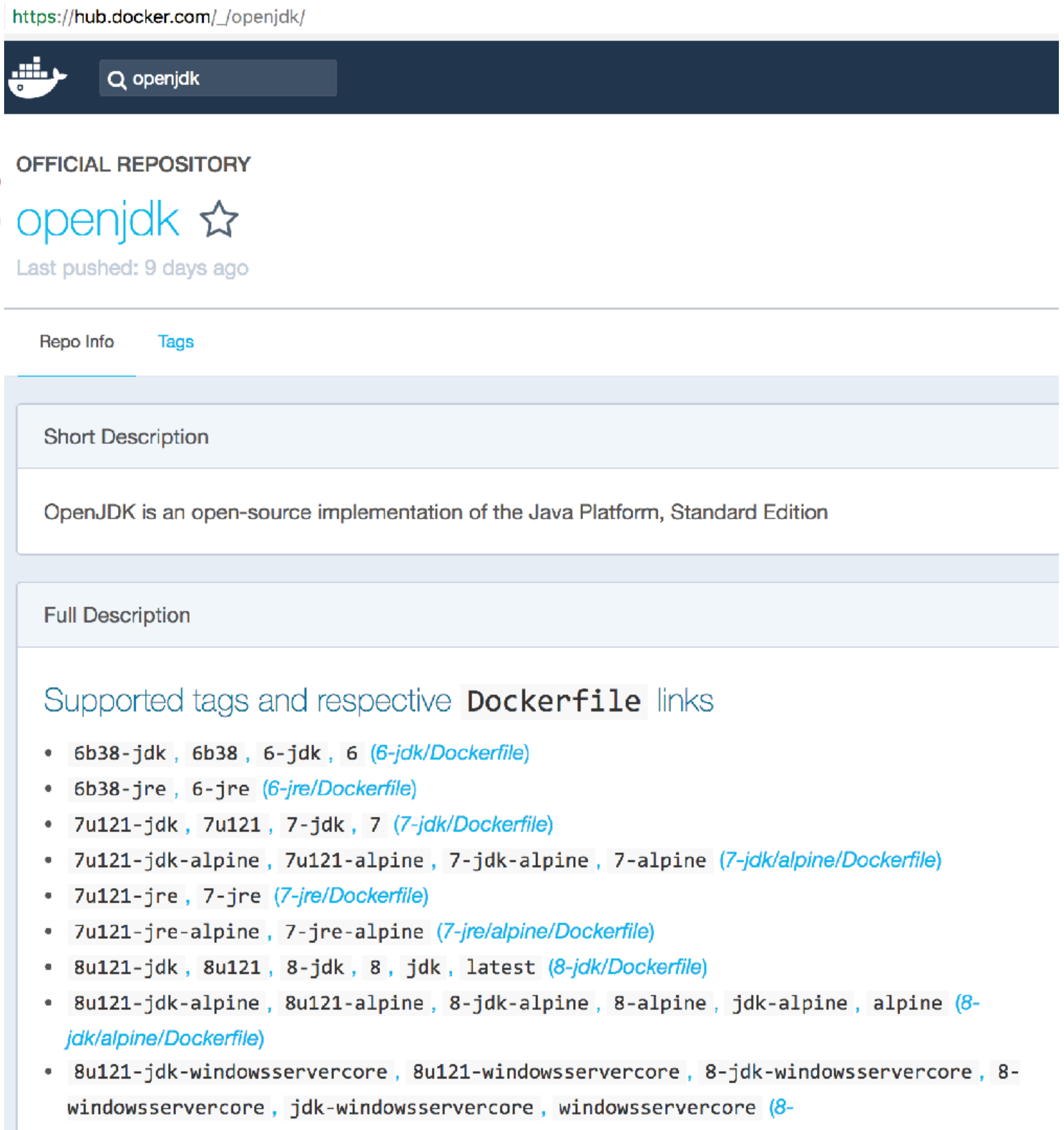
DEPRECATED

This image is officially deprecated in favor of the `openjdk` image, and will receive no further updates after 2016-12-31 (Dec 31, 2016). Please adjust your usage accordingly.

The image has been OpenJDK-specific since it was first introduced, and as of 2016-08-10 we also have an `ibmjava` image, which made it even more clear that each repository should represent one upstream instead of one language stack or community, so this rename reflects that clarity appropriately.

Java base image #2

	Debian	Alpine
jdk	244MB	71MB
jre	124MB	56MB



https://hub.docker.com/_/openjdk/

openjdk

OFFICIAL REPOSITORY

openjdk

Last pushed: 9 days ago

Repo Info | Tags

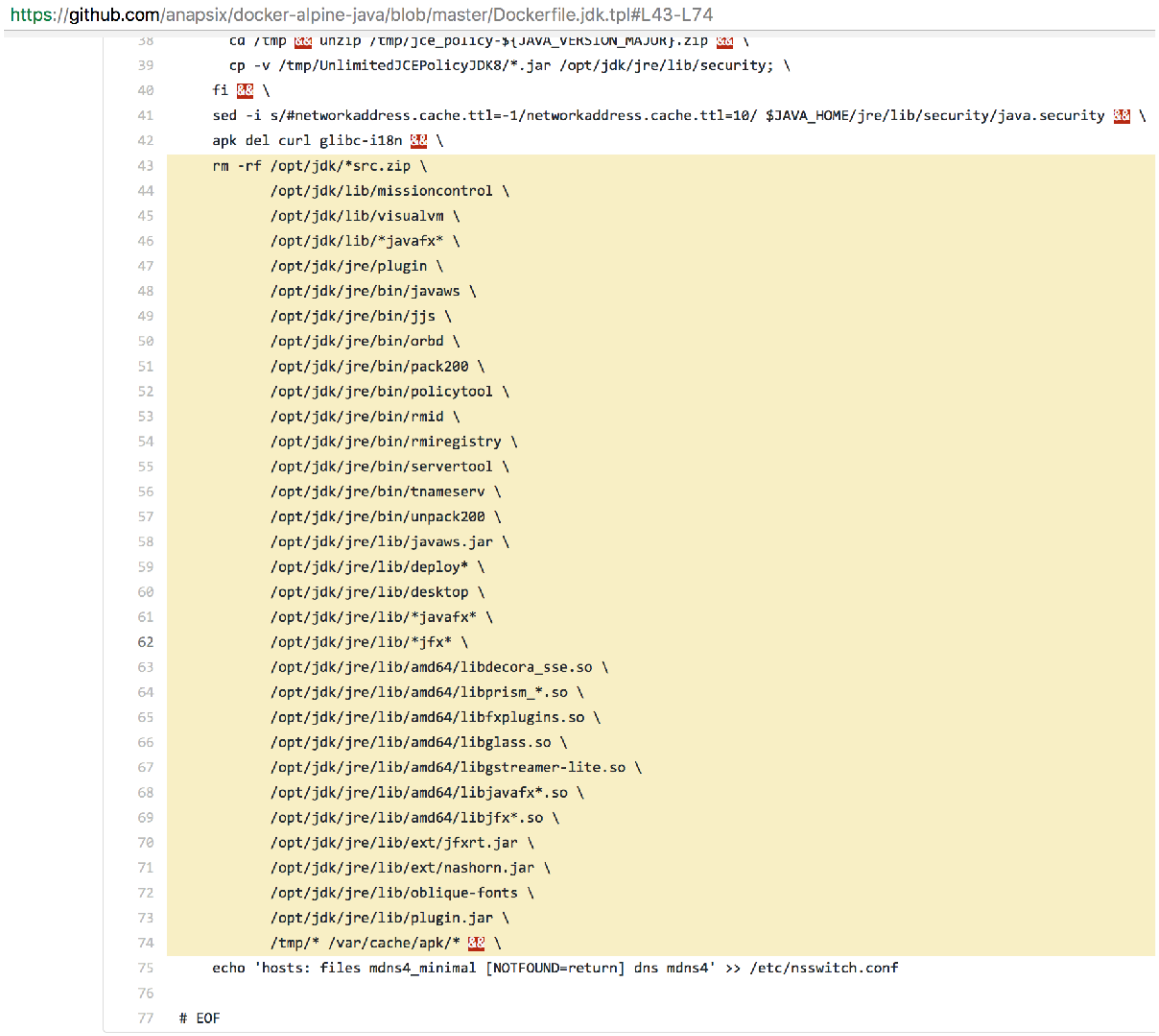
Short Description

OpenJDK is an open-source implementation of the Java Platform, Standard Edition

Full Description

Supported tags and respective `Dockerfile` links

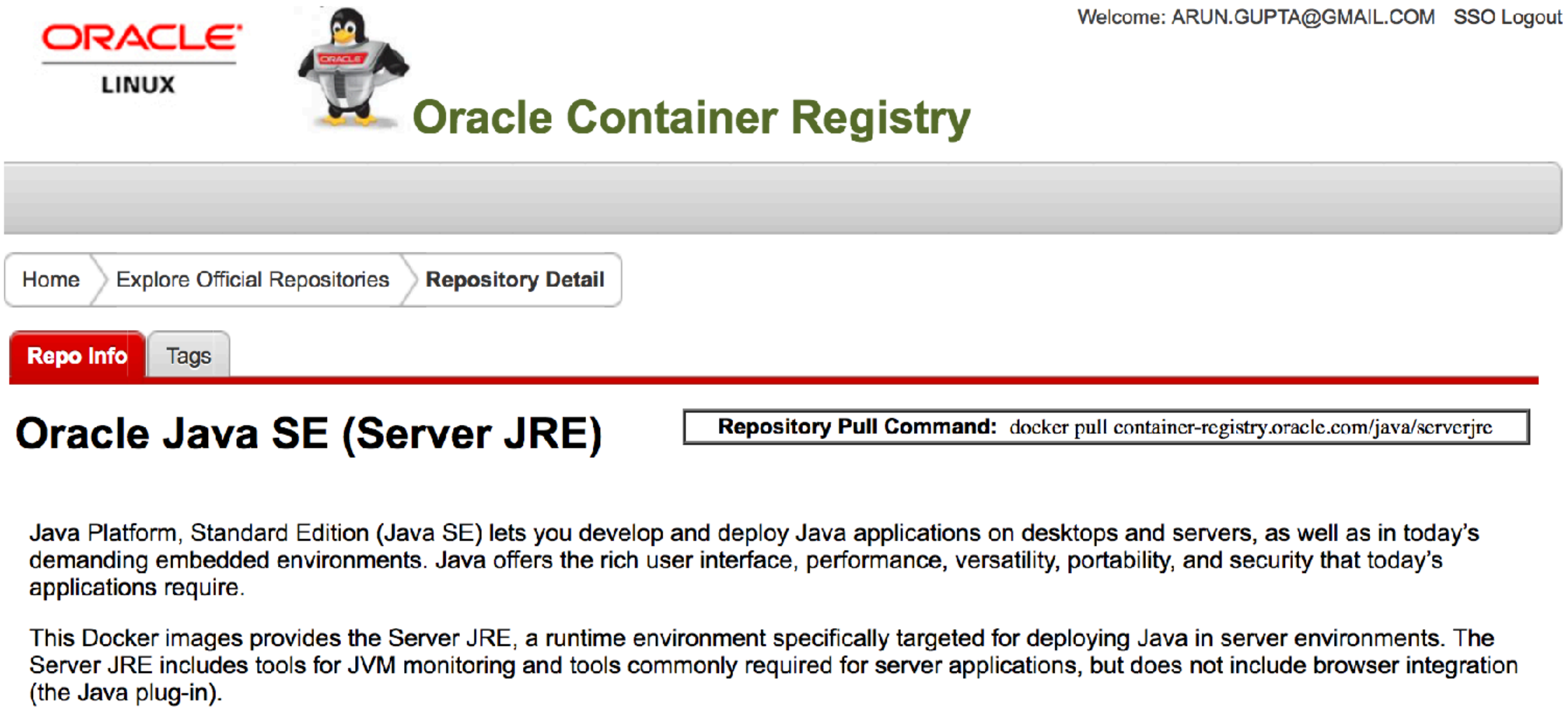
- `6b38-jdk` , `6b38` , `6-jdk` , `6` *(6-jdk/Dockerfile)*
- `6b38-jre` , `6-jre` *(6-jre/Dockerfile)*
- `7u121-jdk` , `7u121` , `7-jdk` , `7` *(7-jdk/Dockerfile)*
- `7u121-jdk-alpine` , `7u121-alpine` , `7-jdk-alpine` , `7-alpine` *(7-jdk/alpine/Dockerfile)*
- `7u121-jre` , `7-jre` *(7-jre/Dockerfile)*
- `7u121-jre-alpine` , `7-jre-alpine` *(7-jre/alpine/Dockerfile)*
- `8u121-jdk` , `8u121` , `8-jdk` , `8` , `jdk` , `latest` *(8-jdk/Dockerfile)*
- `8u121-jdk-alpine` , `8u121-alpine` , `8-jdk-alpine` , `8-alpine` , `jdk-alpine` , `alpine` *(8-jdk/alpine/Dockerfile)*
- `8u121-jdk-windowsservercore` , `8u121-windowsservercore` , `8-jdk-windowsservercore` , `8-windowsservercore` , `jdk-windowsservercore` , `windowsservercore` *(8-*



https://github.com/anapsix/docker-alpine-java/blob/master/Dockerfile.jdk.tpl#L43-L74

```
      cd /tmp && unzip /tmp/jce_policy-${JAVA_VERSION_MAJOR}.zip && \
      cp -v /tmp/UnlimitedJCEPolicyJDK8/*.jar /opt/jdk/jre/lib/security; \
    fi && \
    sed -i s/#networkaddress.cache.ttl=-1/networkaddress.cache.ttl=10/ $JAVA_HOME/jre/lib/security/java.security && \
    apk del curl glibc-i18n && \
    rm -rf /opt/jdk/*src.zip \
           /opt/jdk/lib/missioncontrol \
           /opt/jdk/lib/visualvm \
           /opt/jdk/lib/*javafx* \
           /opt/jdk/jre/plugin \
           /opt/jdk/jre/bin/javaws \
           /opt/jdk/jre/bin/jjs \
           /opt/jdk/jre/bin/orbd \
           /opt/jdk/jre/bin/pack200 \
           /opt/jdk/jre/bin/policytool \
           /opt/jdk/jre/bin/rmid \
           /opt/jdk/jre/bin/rmiregistry \
           /opt/jdk/jre/bin/servertool \
           /opt/jdk/jre/bin/tnameserv \
           /opt/jdk/jre/bin/unpack200 \
           /opt/jdk/jre/lib/javaws.jar \
           /opt/jdk/jre/lib/deploy* \
           /opt/jdk/jre/lib/desktop \
           /opt/jdk/jre/lib/*javafx* \
           /opt/jdk/jre/lib/*jfx* \
           /opt/jdk/jre/lib/amd64/libdecora_sse.so \
           /opt/jdk/jre/lib/amd64/libprism_*.so \
           /opt/jdk/jre/lib/amd64/libfxplugins.so \
           /opt/jdk/jre/lib/amd64/libglass.so \
           /opt/jdk/jre/lib/amd64/libgstreamer-lite.so \
           /opt/jdk/jre/lib/amd64/libjavafx*.so \
           /opt/jdk/jre/lib/amd64/libjfx*.so \
           /opt/jdk/jre/lib/ext/jfxrt.jar \
           /opt/jdk/jre/lib/ext/nashorn.jar \
           /opt/jdk/jre/lib/oblique-fonts \
           /opt/jdk/jre/lib/plugin.jar \
           /tmp/* /var/cache/apk/* && \
    echo 'hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4' >> /etc/nsswitch.conf

# EOF
```

Java base image #3


Welcome: ARUN.GUPTA@GMAIL.COM SSO Logout

Oracle Container Registry

Home > Explore Official Repositories > Repository Detail

Repo Info | Tags

Oracle Java SE (Server JRE)

Repository Pull Command: docker pull container-registry.oracle.com/java/serverjre

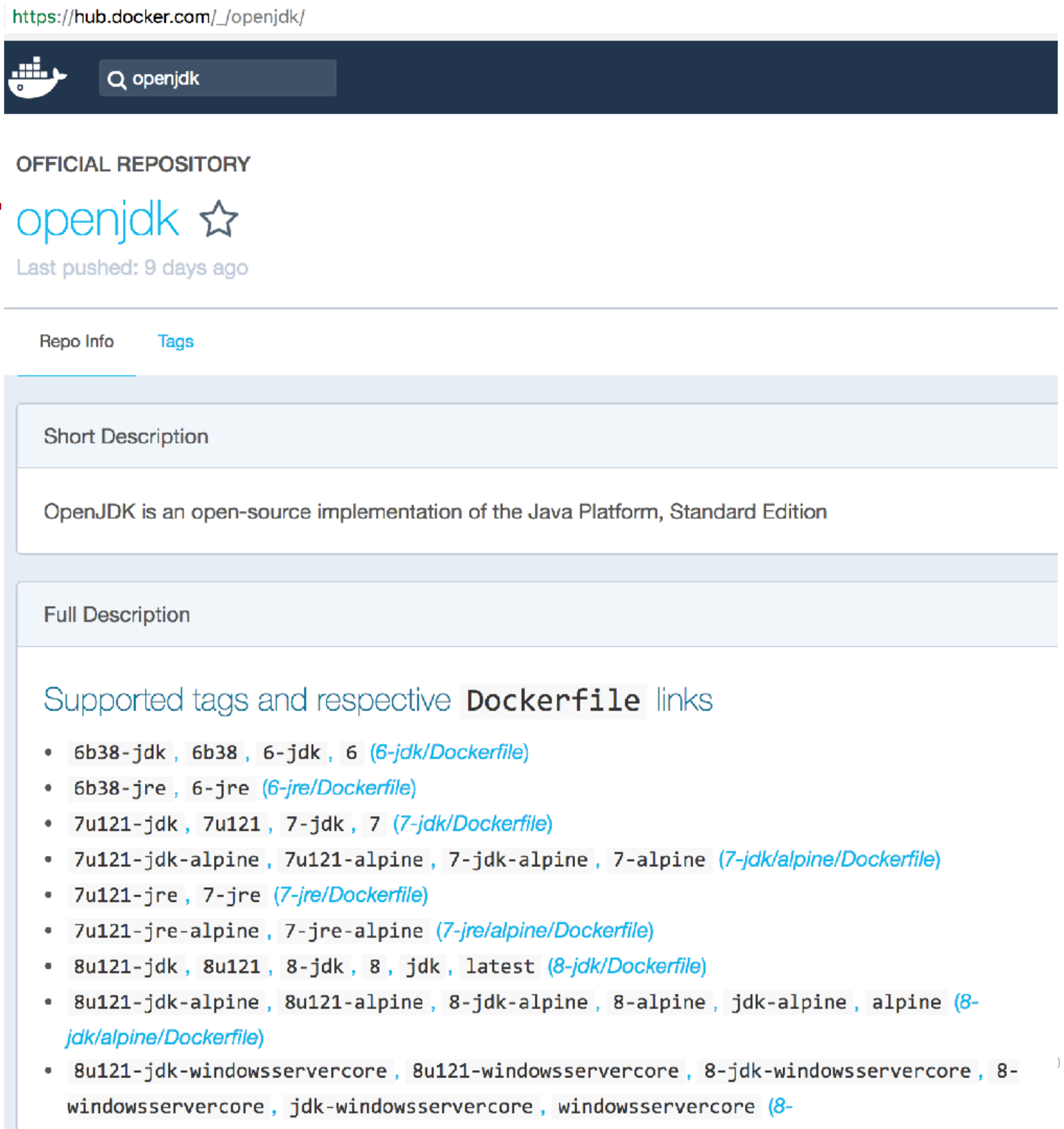
Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on desktops and servers, as well as in today's demanding embedded environments. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

This Docker images provides the Server JRE, a runtime environment specifically targeted for deploying Java in server environments. The Server JRE includes tools for JVM monitoring and tools commonly required for server applications, but does not include browser integration (the Java plug-in).

Java base image #4

Snapshot from EC2 Container Registry

Also available in Artifactory


https://hub.docker.com/_/openjdk/

openjdk

OFFICIAL REPOSITORY

openjdk

Last pushed: 9 days ago

Repo Info Tags

Short Description

OpenJDK is an open-source implementation of the Java Platform, Standard Edition

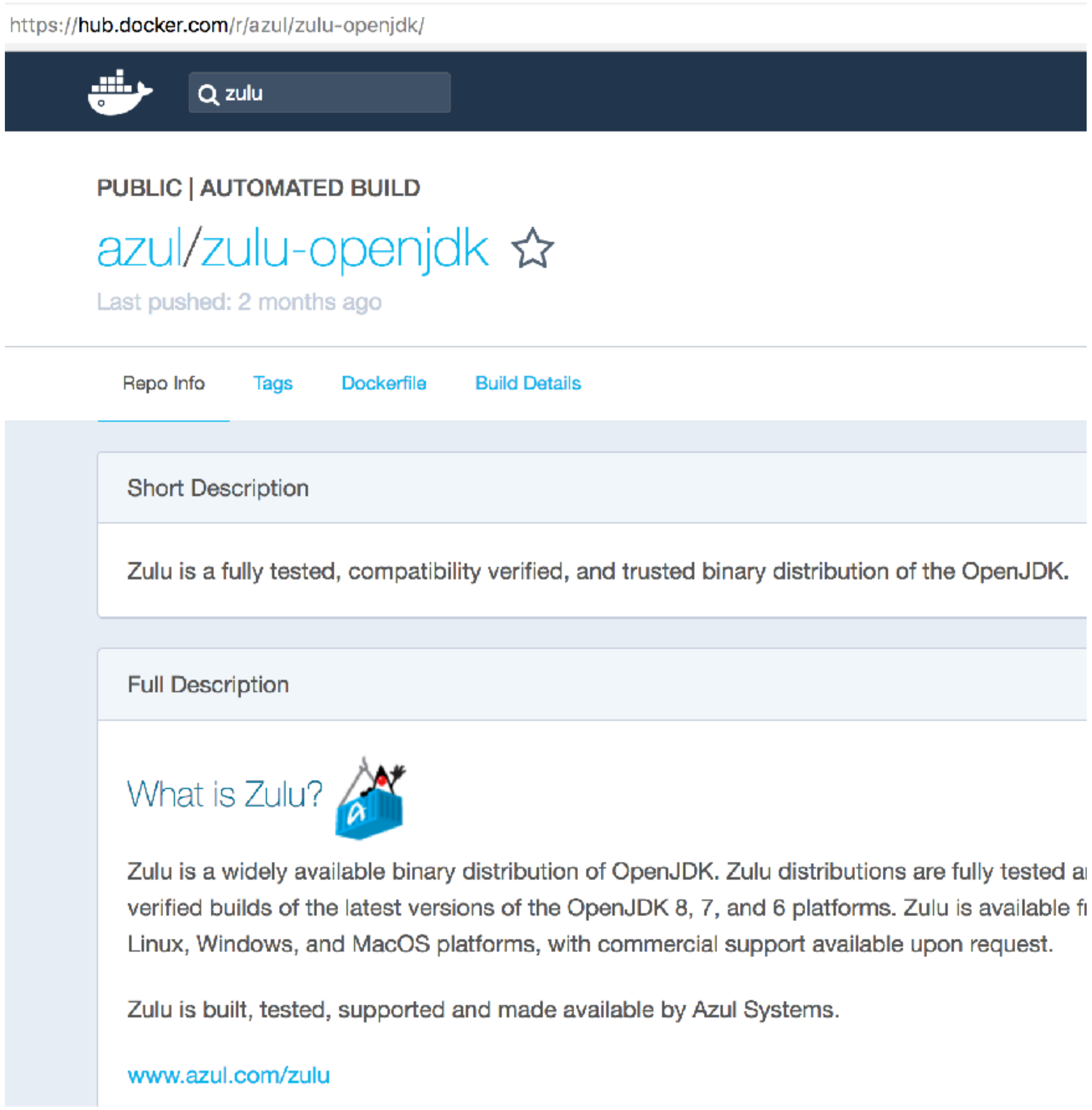
Full Description

Supported tags and respective `Dockerfile` links

- `6b38-jdk`, `6b38`, `6-jdk`, `6` (6-jdk/Dockerfile)
- `6b38-jre`, `6-jre` (6-jre/Dockerfile)
- `7u121-jdk`, `7u121`, `7-jdk`, `7` (7-jdk/Dockerfile)
- `7u121-jdk-alpine`, `7u121-alpine`, `7-jdk-alpine`, `7-alpine` (7-jdk/alpine/Dockerfile)
- `7u121-jre`, `7-jre` (7-jre/Dockerfile)
- `7u121-jre-alpine`, `7-jre-alpine` (7-jre/alpine/Dockerfile)
- `8u121-jdk`, `8u121`, `8-jdk`, `8`, `jdk`, `latest` (8-jdk/Dockerfile)
- `8u121-jdk-alpine`, `8u121-alpine`, `8-jdk-alpine`, `8-alpine`, `jdk-alpine`, `alpine` (8-jdk/alpine/Dockerfile)
- `8u121-jdk-windowsservercore`, `8u121-windowsservercore`, `8-jdk-windowsservercore`, `8-windowsservercore`, `jdk-windowsservercore`, `windowsservercore` (8-

Java base image #5

openjdk	244MB	Debian
zulu-openjdk	161MB	Ubuntu



https://hub.docker.com/r/azul/zulu-openjdk/

PUBLIC | AUTOMATED BUILD

azul/zulu-openjdk

Last pushed: 2 months ago

Short Description

Zulu is a fully tested, compatibility verified, and trusted binary distribution of the OpenJDK.

Full Description

What is Zulu?

Zulu is a widely available binary distribution of OpenJDK. Zulu distributions are fully tested a
verified builds of the latest versions of the OpenJDK 8, 7, and 6 platforms. Zulu is available f
Linux, Windows, and MacOS platforms, with commercial support available upon request.

Zulu is built, tested, supported and made available by Azul Systems.

www.azul.com/zulu

Package Java Application using Maven or Gradle

Maven Plugin

- Plugin

```
<groupId>io.fabric8</groupId>
<artifactId>docker-maven-plugin</artifactId>
<version>0.20.1</version>
```

- Goals: **`docker:X`**, X= **`stop`**, **`build`**, **`push`**, …

Maven - Configuration

```xml
<plugin>
    <groupId>io.fabric8</groupId>
    <artifactId>docker-maven-plugin</artifactId>
    <version>0.20.1</version>
    <configuration>
        <images>
            <image>
                <name>hellojava</name>
                <build>
                    <from>openjdk:latest</from>
                    <assembly>
                        <descriptorRef>artifact</descriptorRef>
                    </assembly>
                    <cmd>java -jar maven/${project.name}-${project.version}.jar</cmd>
                </build>
                <run>
                    <wait>
                        <log>Hello World!</log>
                    </wait>
                </run>
            </image>
        </images>
    </configuration>
```

https://github.com/arun-gupta/docker-java-sample

Maven - Execution

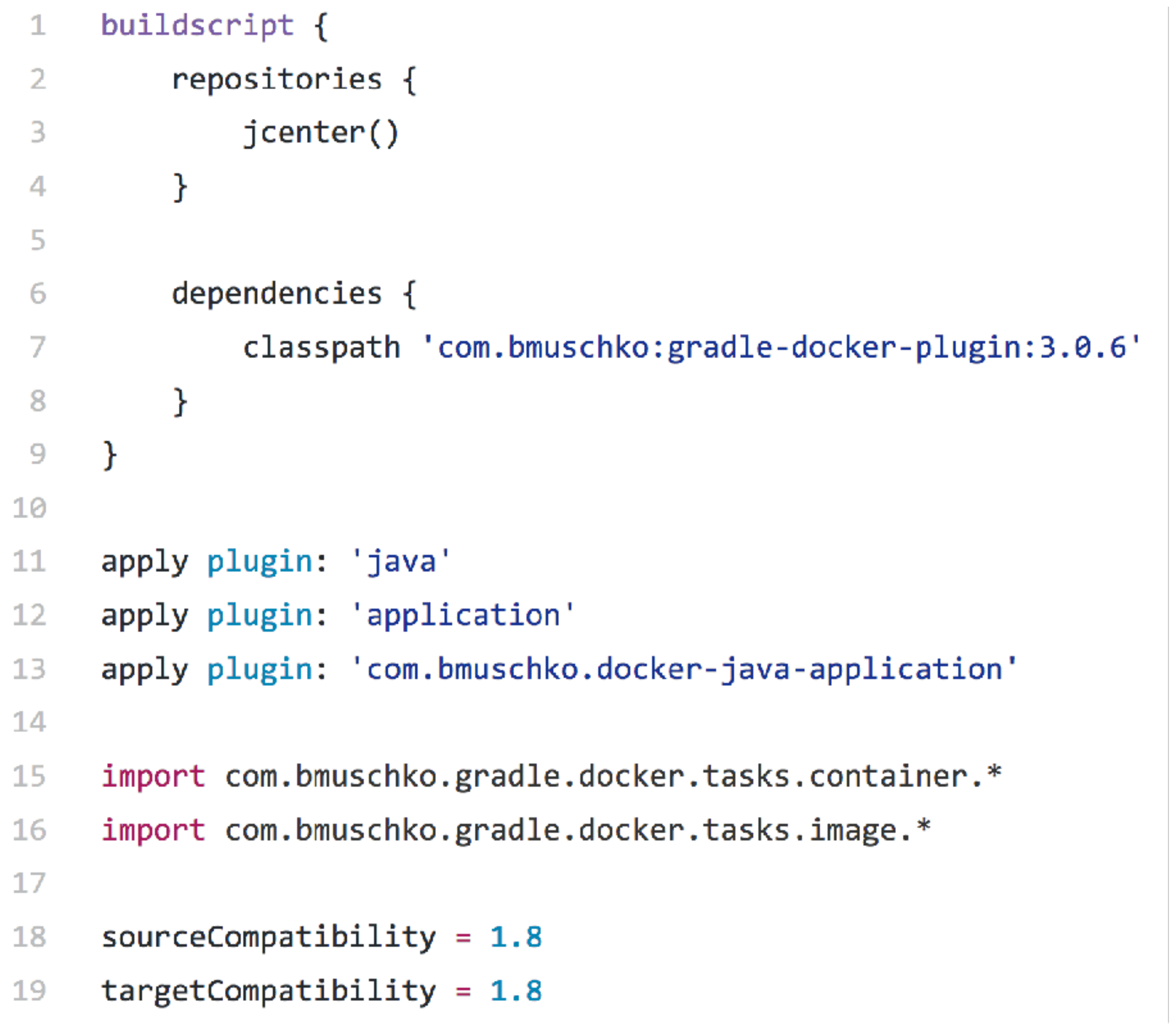
```xml
            <executions>
                <execution>
                    <id>docker:build</id>
                    <phase>package</phase>
                    <goals>
                        <goal>build</goal>
                    </goals>
                </execution>
                <execution>
                    <id>docker:start</id>
                    <phase>install</phase>
                    <goals>
                        <goal>run</goal>
                        <goal>logs</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
```

Gradle Plugin

- Plugin: **`com.bmuschko:gradle-docker-plugin:3.0.6`**
- General purpose Docker Remote API
  - **`DockerXImage`**, X = **`Build`**, **`Push`**, **`Remove`**, …
  - **`DockerXContainer`**, X = **`Create`**, **Start**, **`Stop`**, **`Kill`**, …
- Opinionated Java application plugin
  - Extension properties: **`baseImage`**, **`tag`**, **`port`**, …

https://github.com/bmuschko/gradle-docker-plugin

Gradle - Configuration



```
buildscript {
    repositories {
        jcenter()
    }

    dependencies {
        classpath 'com.bmuschko:gradle-docker-plugin:3.0.6'
    }
}

apply plugin: 'java'
apply plugin: 'application'
apply plugin: 'com.bmuschko.docker-java-application'

import com.bmuschko.gradle.docker.tasks.container.*
import com.bmuschko.gradle.docker.tasks.image.*

sourceCompatibility = 1.8
targetCompatibility = 1.8
```

Gradle - Execution

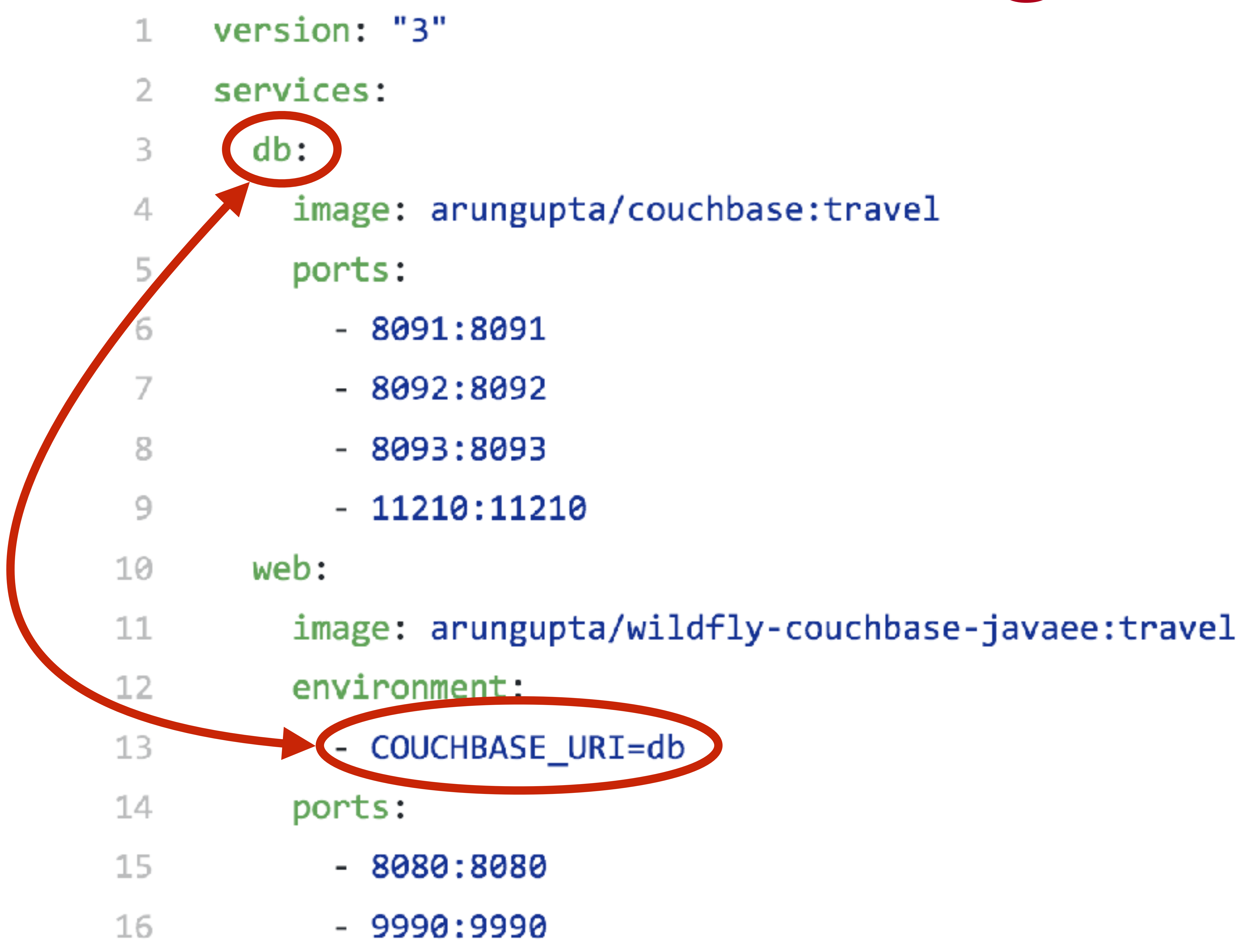
```
docker {
    javaApplication {
        baseImage = 'openjdk:latest'
        tag = 'hellojava'
    }
}

task createContainer(type: DockerCreateContainer) {
    dependsOn dockerBuildImage
    targetImageId { dockerBuildImage.getImageId() }
}

task startContainer(type: DockerStartContainer) {
    dependsOn createContainer
    targetContainerId { createContainer.getContainerId() }
}
```

Multi-Container Application

Docker Compose

- Define and run multi-container applications
- Configuration defined in one or more files
  - `docker-compose.yml` (default)
  - `docker-compose.override.yml` (default)
  - Multiple files specified using `-f`
- Single command to manage all services
- Great for dev, staging, and CI

Multi-container on single host



```yaml
version: "3"
services:
  db:
    image: arungupta/couchbase:travel
    ports:
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
  web:
    image: arungupta/wildfly-couchbase-javaee:travel
    environment:
      - COUCHBASE_URI=db
    ports:
      - 8080:8080
      - 9990:9990
```

```
docker-compose up
```

Multiple Files - Image and Ports

docker-compose.db.yml

```
version: '3'
services:
  web:

    ports:
     - 80:8080
  db:
    image: couchbase:prod
    ports:
      - 8091:8091
```

Run

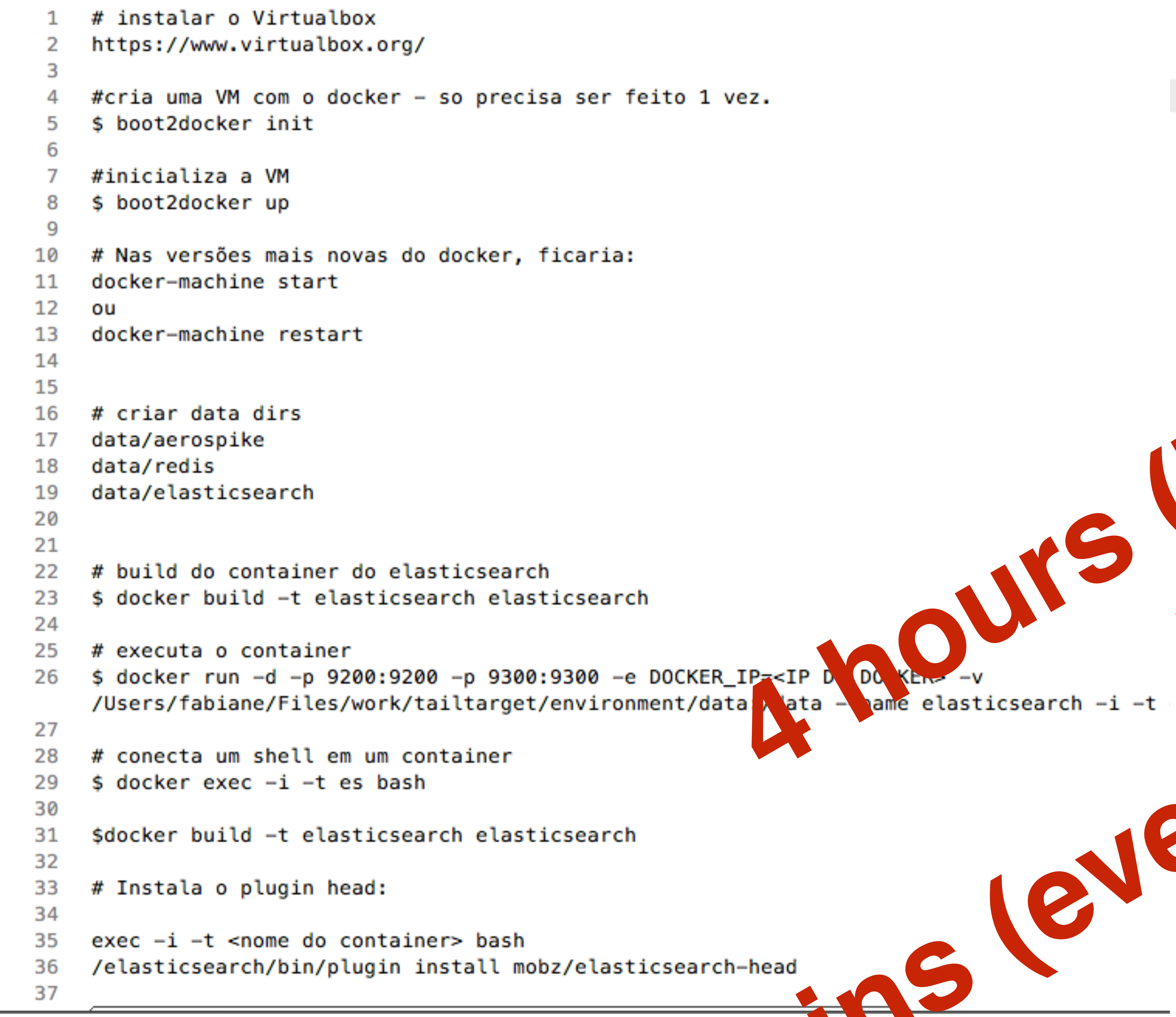
```
docker-compose \
-f docker-compose.yml \
-f docker-compose.db.yml \
up -d
```

Services

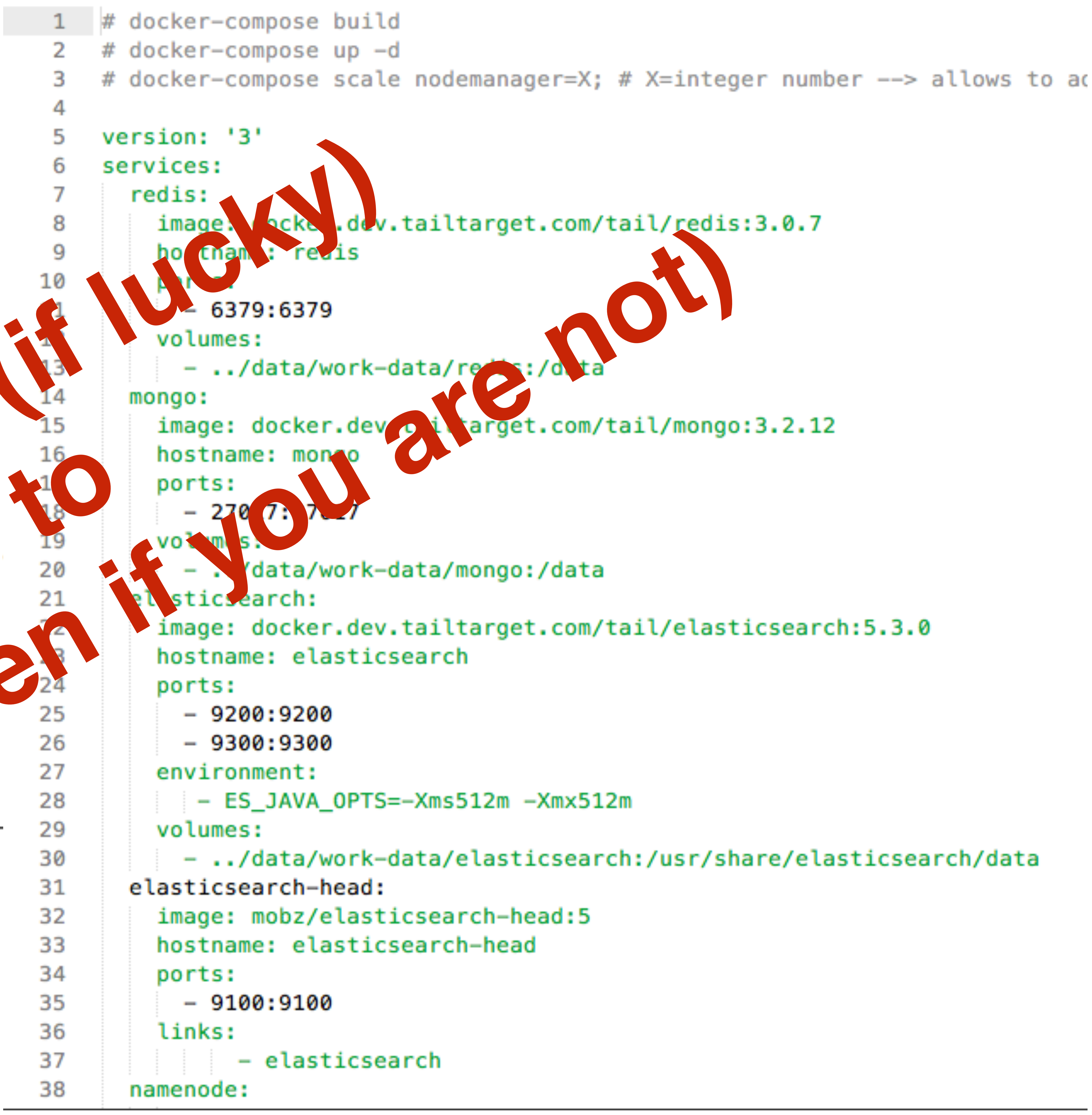
```
docker-compose \
-f docker-compose.yml \
-f docker-compose.db.yml \
ps
```

Shutdown

```
docker-compose \
-f docker-compose.yml \
-f docker-compose.db.yml \
down
```



```
# instalar o Virtualbox
https://www.virtualbox.org/

#cria uma VM com o docker - so precisa ser feito 1 vez.
$ boot2docker init

#inicializa a VM
$ boot2docker up

# Nas versões mais novas do docker, ficaria:
docker-machine start
ou
docker-machine restart


# criar data dirs
data/aerospike
data/redis
data/elasticsearch


# build do container do elasticsearch
$ docker build -t elasticsearch elasticsearch

# executa o container
$ docker run -d -p 9200:9200 -p 9300:9300 -e DOCKER_IP=<IP DO DOCKER> -v
/Users/fabiane/Files/work/tailtarget/environment/data:/data --name elasticsearch -i -t

# conecta um shell em um container
$ docker exec -i -t es bash

$docker build -t elasticsearch elasticsearch

# Instala o plugin head:

exec -i -t <nome do container> bash
/elasticsearch/bin/plugin install mobz/elasticsearch-head
```



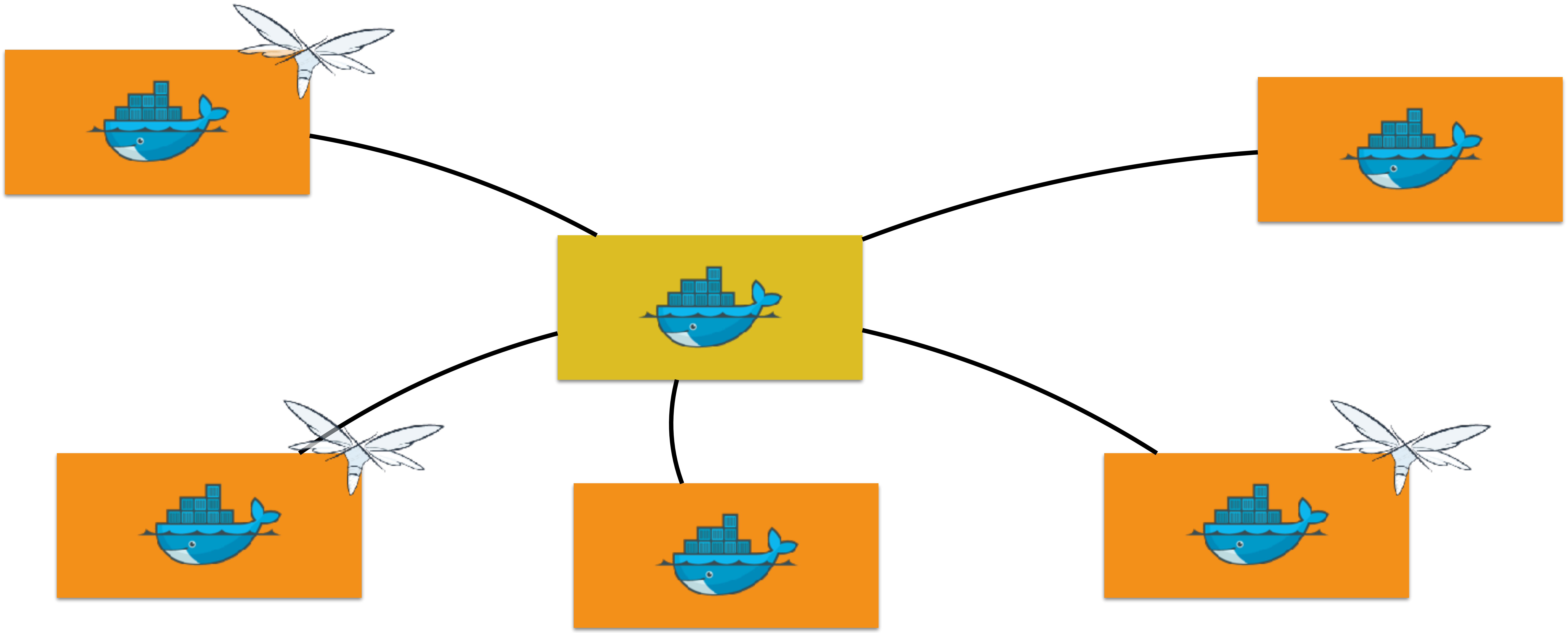
```
# docker-compose build
# docker-compose up -d
# docker-compose scale nodemanager=X; # X=integer number --> allows to ac

version: '3'
services:
  redis:
    image: docker.dev.tailtarget.com/tail/redis:3.0.7
    hostname: redis
    ports:
      - 6379:6379
    volumes:
      - ../data/work-data/redis:/data
  mongo:
    image: docker.dev.tailtarget.com/tail/mongo:3.2.12
    hostname: mongo
    ports:
      - 27017:27017
    volumes:
      - ../data/work-data/mongo:/data
  elasticsearch:
    image: docker.dev.tailtarget.com/tail/elasticsearch:5.3.0
    hostname: elasticsearch
    ports:
      - 9200:9200
      - 9300:9300
    environment:
      - ES_JAVA_OPTS=-Xms512m -Xmx512m
    volumes:
      - ../data/work-data/elasticsearch:/usr/share/elasticsearch/data
  elasticsearch-head:
    image: mobz/elasticsearch-head:5
    hostname: elasticsearch-head
    ports:
      - 9100:9100
    links:
        - elasticsearch
  namenode:
```

4 hours (if lucky)
to
2 mins (even if you are not)

Swarm Mode

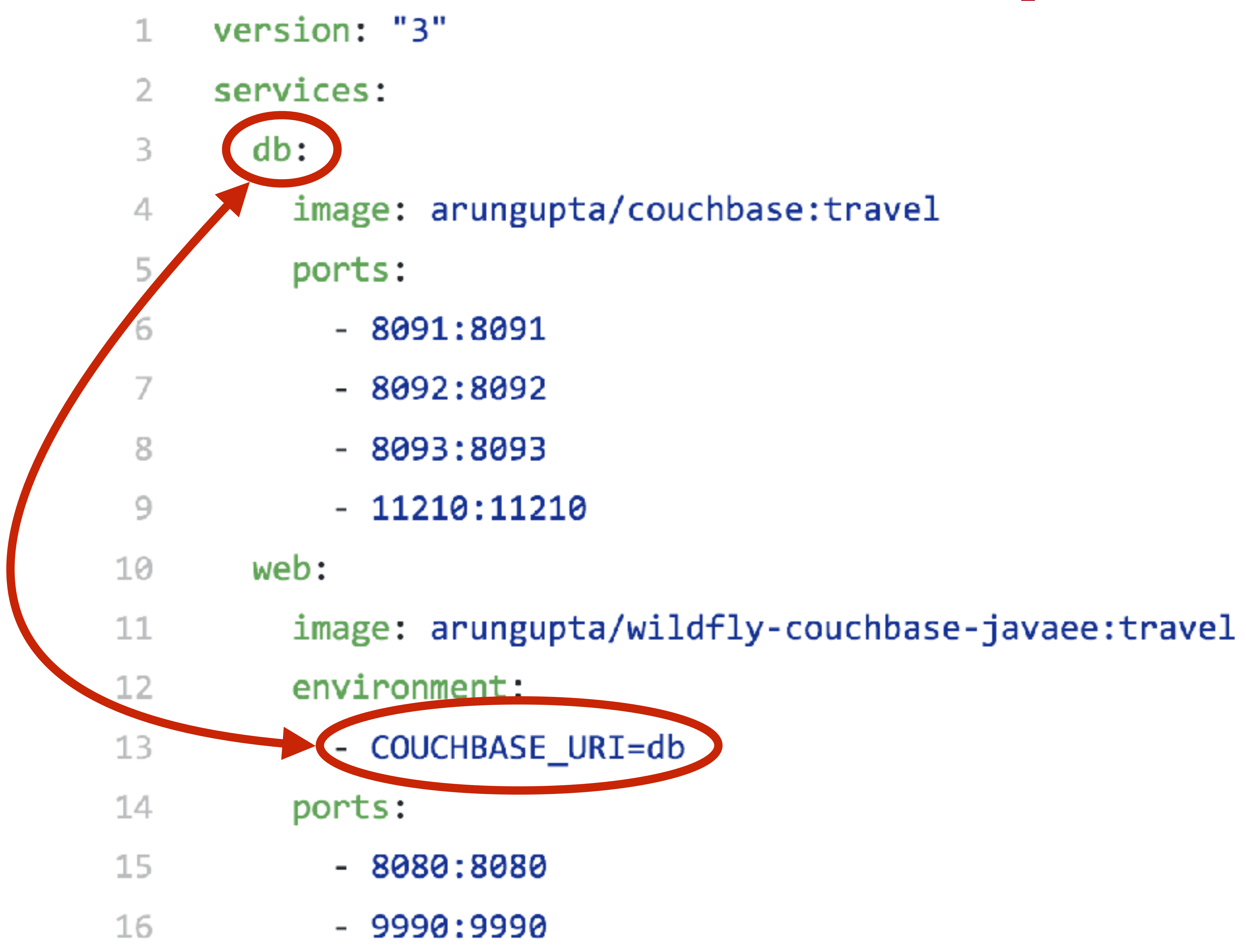
- Natively managing a cluster of Docker Engines called a Swarm
- Docker CLI to create a swarm, deploy apps, and manage swarm
  - Optional feature, need to be explicitly enabled
- No Single Point of Failure (SPOF)
- Declarative state model
- Self-organizing, self-healing
- Service discovery, load balancing and scaling
- Rolling updates

Swarm Mode: Replicated Service



```
docker service create --replicas 3 --name web jboss/wildfly
```

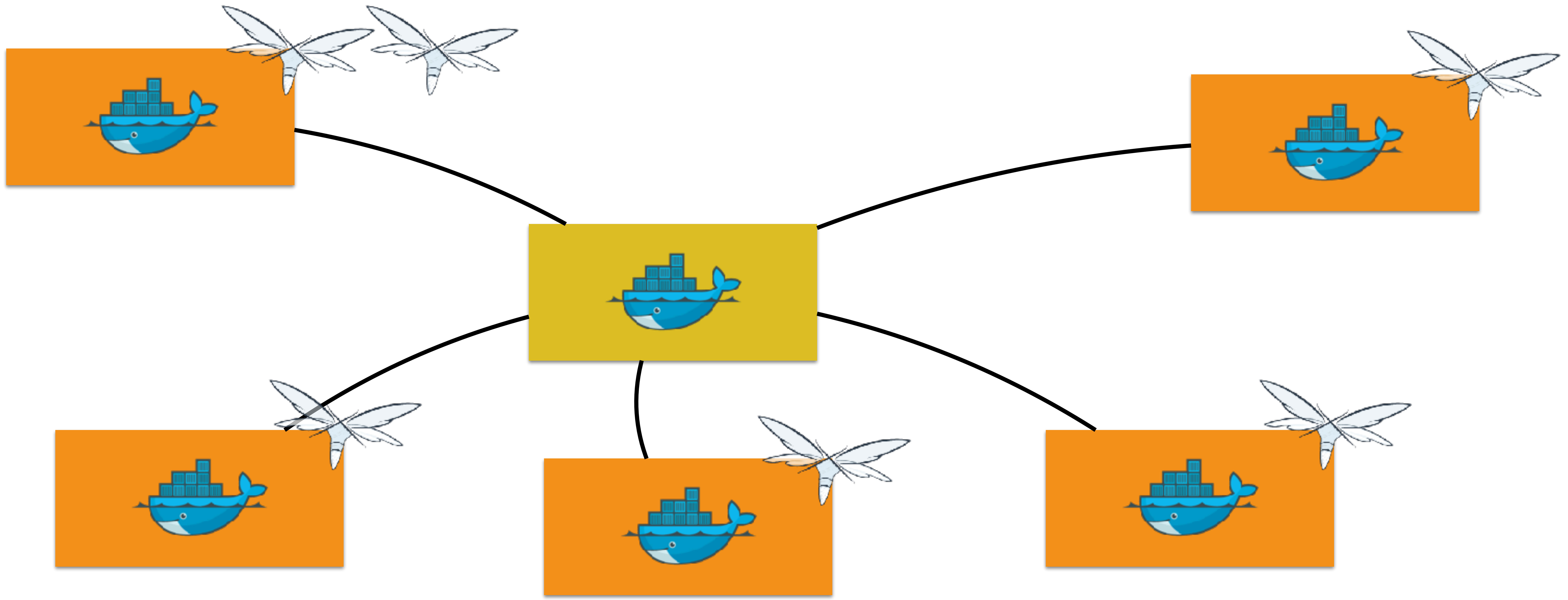
Multi-container on multiple hosts



```yaml
version: "3"
services:
  db:
    image: arungupta/couchbase:travel
    ports:
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
  web:
    image: arungupta/wildfly-couchbase-javaee:travel
    environment:
      - COUCHBASE_URI=db
    ports:
      - 8080:8080
      - 9990:9990
```

```
docker stack deploy --compose-file=docker-compose.yml webapp
```

Swarm Mode: Scale



```
docker service scale web=6
```

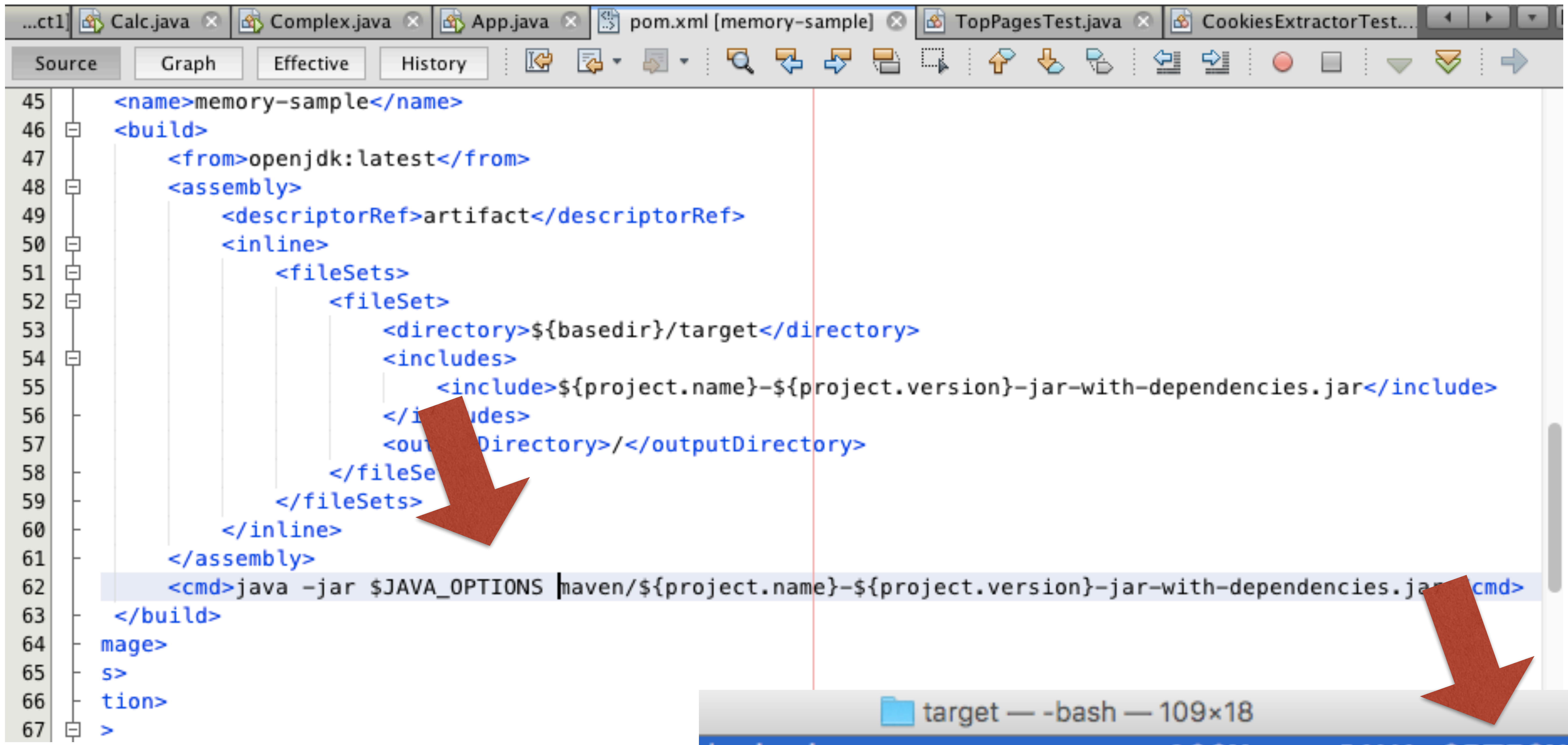
Scaling Apps on AWS or Azure

Docker for AWS/Azure

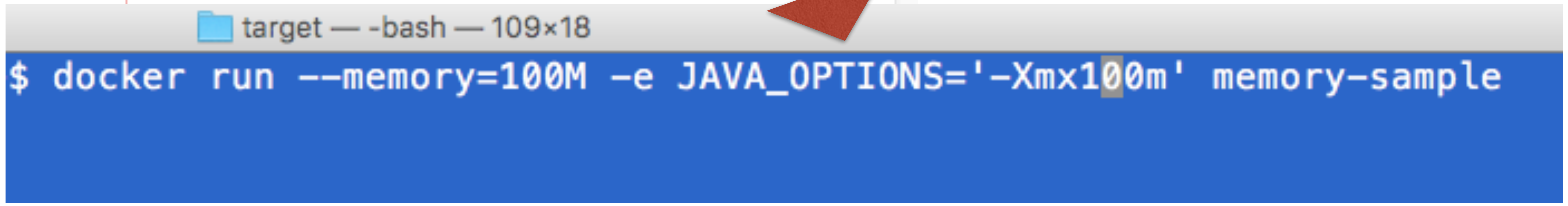
- Amazon Web Services
  - Amazon CloudFormation templates
  - Integrated with Autoscaling, ELB, and EBS
- Azure
  - Integrated with VM Scale Sets for autoscaling, Azure Load Balancer, Azure Storage
- Available in Docker CE and Docker EE

Memory Management for Java Applications

How much memory is available for containers?



```
    <name>memory-sample</name>
    <build>
        <from>openjdk:latest</from>
        <assembly>
            <descriptorRef>artifact</descriptorRef>
            <inline>
                <fileSets>
                    <fileSet>
                        <directory>${basedir}/target</directory>
                        <includes>
                            <include>${project.name}-${project.version}-jar-with-dependencies.jar</include>
                        </i   udes>
                        <ou   Directory>/</outputDirectory>
                    </fileSe
                </fileSets>
            </inline>
        </assembly>
        <cmd>java -jar $JAVA_OPTIONS maven/${project.name}-${project.version}-jar-with-dependencies.ja   cmd>
    </build>
  mage>
  s>
  tion>
  >
```



```
$ docker run --memory=100M -e JAVA_OPTIONS='-Xmx100m' memory-sample
```

How much memory is available for containers?

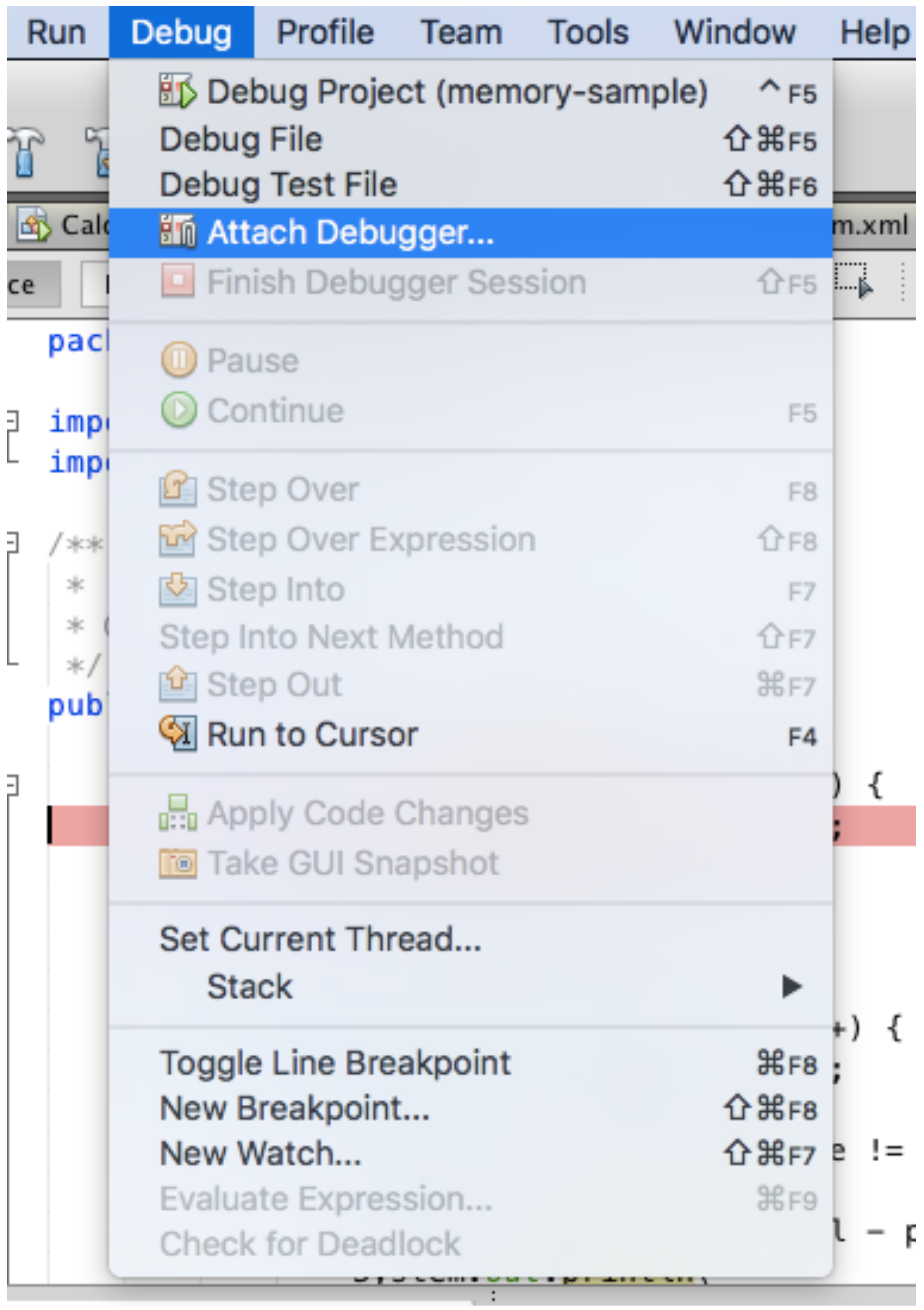
- By default, container will use as much memory and swap
- Can be restricted
  - `--memory`
  - `--memory-reservation`
  - `--memory-swap`
- Today, JDK unaware of container's limited resources
  - For example, memory or CPU using `--cpus`, `--cpu-shares`
- JDK 9 has experimental support for *cgroup* memory limits

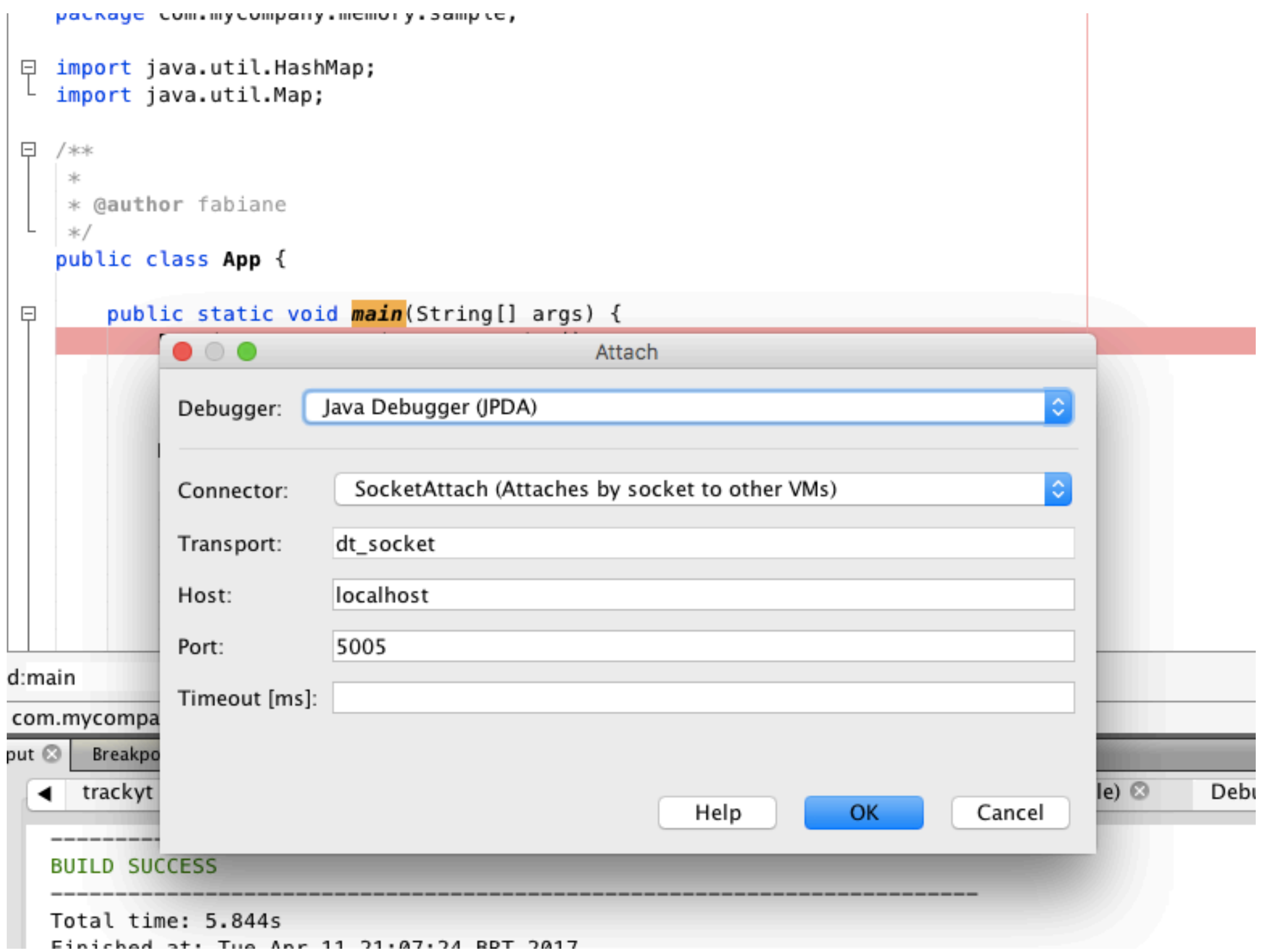
Debugging Java Applications

Running in debug mode

```
docker run -p5005:5005 \
   -e JAVA_OPTIONS= \
   '-Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=5005' \
   memory-sample
```

Attaching the IDE





```
package com.mycompany.memory.sample;

import java.util.HashMap;
import java.util.Map;

/**
 *
 * @author fabiane
 */
public class App {

    public static void main(String[] args) {
```

Attach

Debugger: Java Debugger (JPDA)

Connector: SocketAttach (Attaches by socket to other VMs)

Transport: dt_socket

Host: localhost

Port: 5005

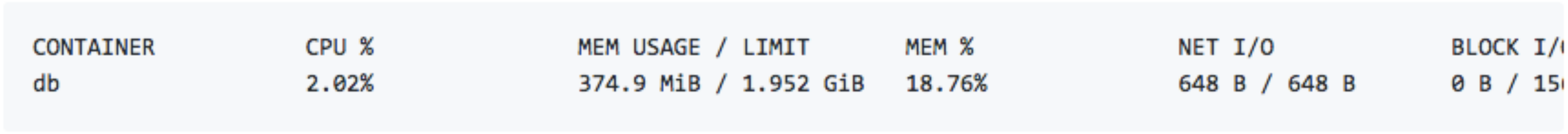
Timeout [ms]:

```
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 5.844s
Finished at: Tue Apr 11 21:07:24 BRT 2017
```

Monitor Java Applications

Monitoring Docker Containers

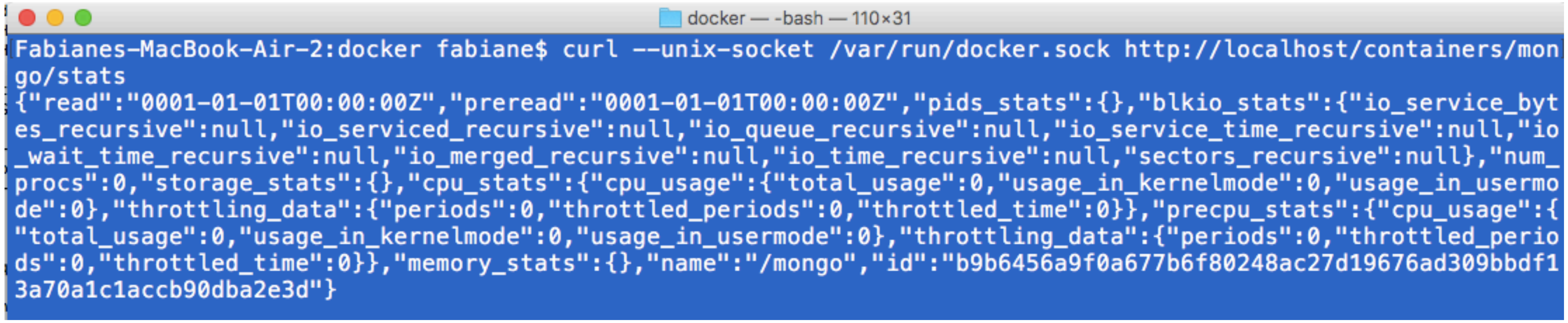
- `docker stats` command



```
CONTAINER           CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/
db                  2.02%               374.9 MiB / 1.952 GiB   18.76%              648 B / 648 B       0 B / 15
```

Monitoring Docker Containers

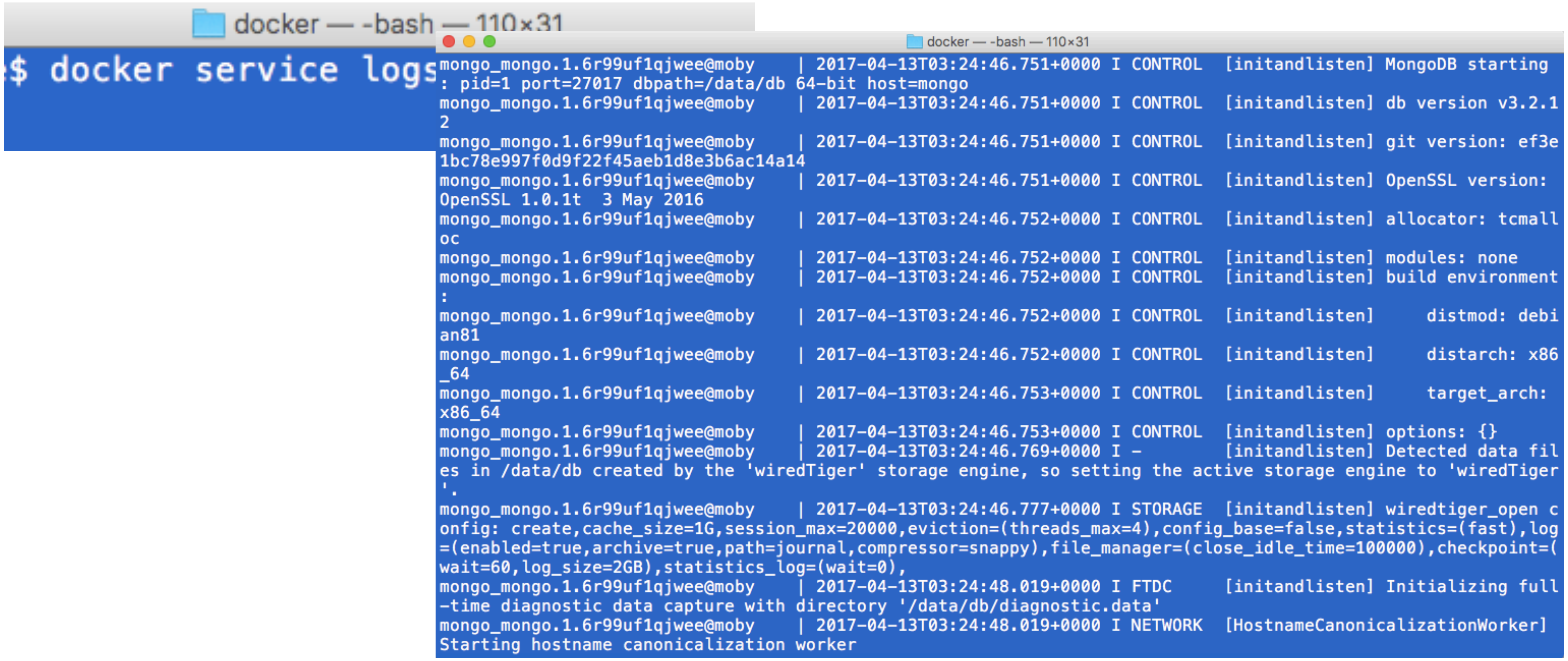
- Docker Remote API: `/container/{container-name|cid}/stats`



```
Fabianes-MacBook-Air-2:docker fabiane$ curl --unix-socket /var/run/docker.sock http://localhost/containers/mon
go/stats
{"read":"0001-01-01T00:00:00Z","preread":"0001-01-01T00:00:00Z","pids_stats":{},"blkio_stats":{"io_service_byt
es_recursive":null,"io_serviced_recursive":null,"io_queue_recursive":null,"io_service_time_recursive":null,"io
_wait_time_recursive":null,"io_merged_recursive":null,"io_time_recursive":null,"sectors_recursive":null},"num_
procs":0,"storage_stats":{},"cpu_stats":{"cpu_usage":{"total_usage":0,"usage_in_kernelmode":0,"usage_in_usermo
de":0},"throttling_data":{"periods":0,"throttled_periods":0,"throttled_time":0}},"precpu_stats":{"cpu_usage":{
"total_usage":0,"usage_in_kernelmode":0,"usage_in_usermode":0},"throttling_data":{"periods":0,"throttled_perio
ds":0,"throttled_time":0}},"memory_stats":{},"name":"/mongo","id":"b9b6456a9f0a677b6f80248ac27d19676ad309bbdf1
3a70a1c1accb90dba2e3d"}
```

Monitoring Docker Containers

- Service logs `docker service logs <service>`

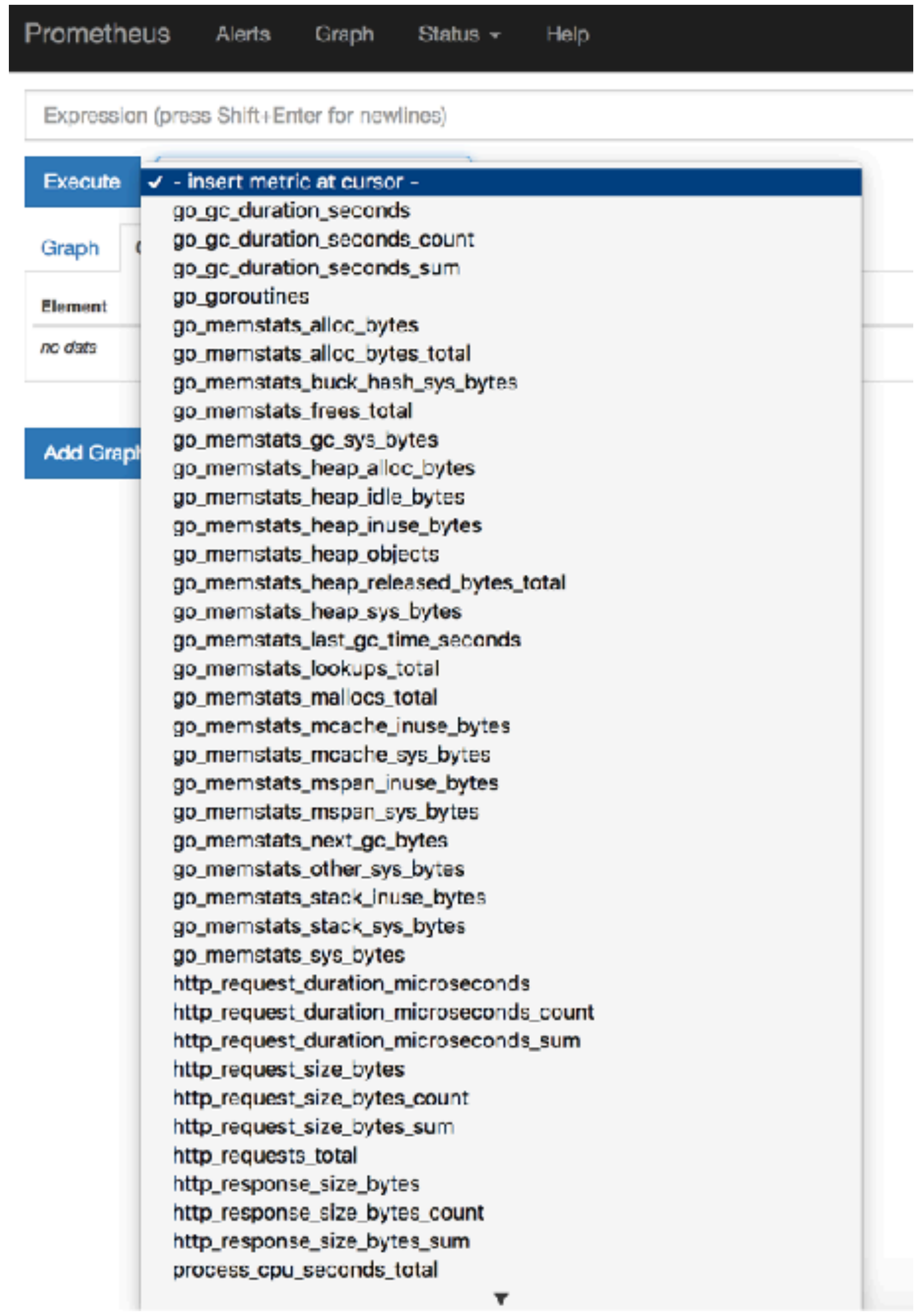


```
$ docker service logs
```

```
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.751+0000 I CONTROL  [initandlisten] MongoDB starting
: pid=1 port=27017 dbpath=/data/db 64-bit host=mongo
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.751+0000 I CONTROL  [initandlisten] db version v3.2.1
2
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.751+0000 I CONTROL  [initandlisten] git version: ef3e
1bc78e997f0d9f22f45aeb1d8e3b6ac14a14
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.751+0000 I CONTROL  [initandlisten] OpenSSL version:
OpenSSL 1.0.1t  3 May 2016
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.752+0000 I CONTROL  [initandlisten] allocator: tcmall
oc
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.752+0000 I CONTROL  [initandlisten] modules: none
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.752+0000 I CONTROL  [initandlisten] build environment
:
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.752+0000 I CONTROL  [initandlisten]     distmod: debi
an81
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.752+0000 I CONTROL  [initandlisten]     distarch: x86
_64
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.753+0000 I CONTROL  [initandlisten]     target_arch:
x86_64
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.753+0000 I CONTROL  [initandlisten] options: {}
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.769+0000 I -        [initandlisten] Detected data fil
es in /data/db created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger
'.
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:46.777+0000 I STORAGE  [initandlisten] wiredtiger_open c
onfig: create,cache_size=1G,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log
=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(
wait=60,log_size=2GB),statistics_log=(wait=0),
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:48.019+0000 I FTDC     [initandlisten] Initializing full
-time diagnostic data capture with directory '/data/db/diagnostic.data'
mongo_mongo.1.6r99uf1qjwee@moby    | 2017-04-13T03:24:48.019+0000 I NETWORK  [HostnameCanonicalizationWorker]
Starting hostname canonicalization worker
```

Monitoring Docker Containers

- Prometheus endpoint - New in 1.13

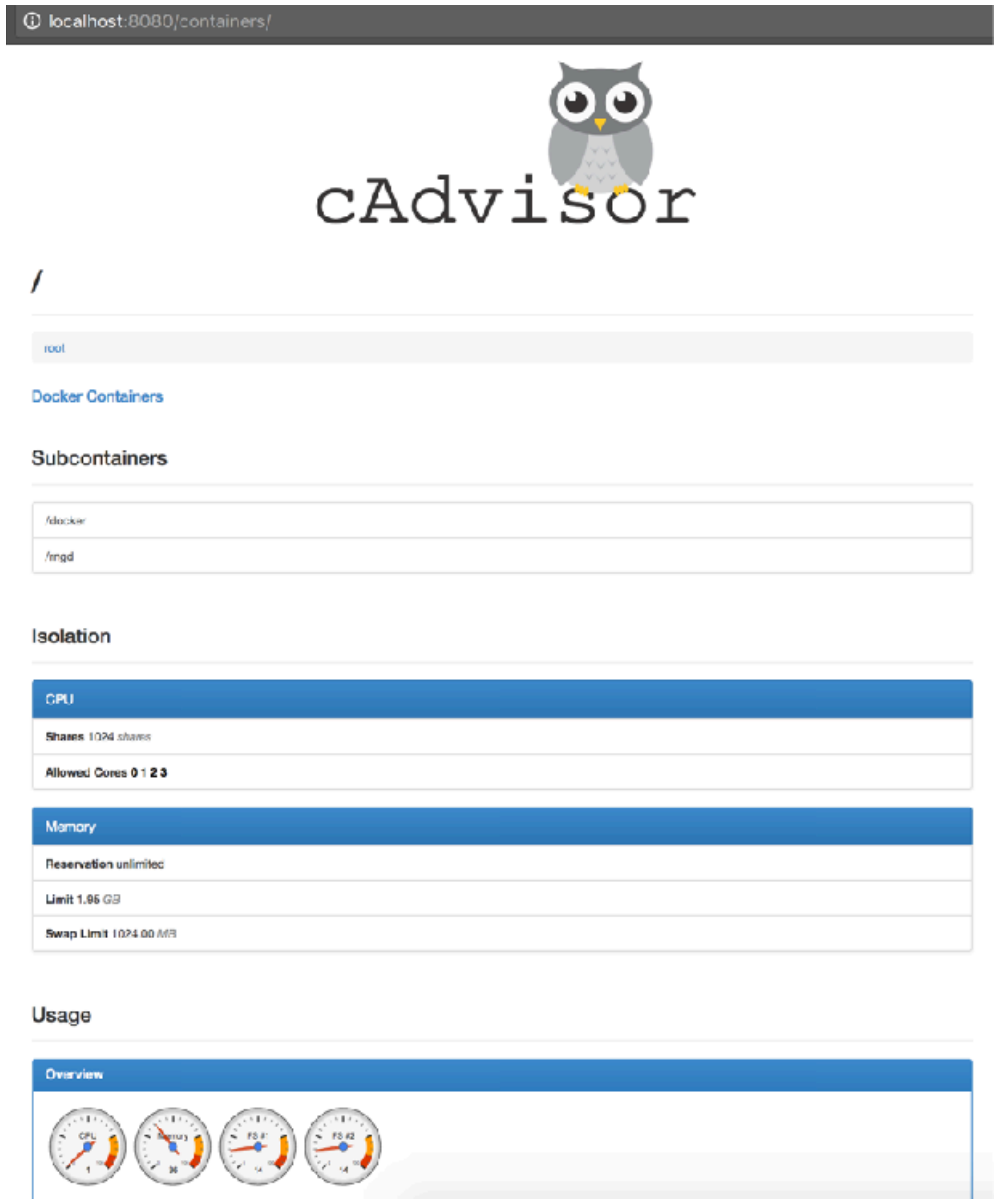


```
✓ - insert metric at cursor -
go_gc_duration_seconds
go_gc_duration_seconds_count
go_gc_duration_seconds_sum
go_goroutines
go_memstats_alloc_bytes
go_memstats_alloc_bytes_total
go_memstats_buck_hash_sys_bytes
go_memstats_frees_total
go_memstats_gc_sys_bytes
go_memstats_heap_alloc_bytes
go_memstats_heap_idle_bytes
go_memstats_heap_inuse_bytes
go_memstats_heap_objects
go_memstats_heap_released_bytes_total
go_memstats_heap_sys_bytes
go_memstats_last_gc_time_seconds
go_memstats_lookups_total
go_memstats_mallocs_total
go_memstats_mcache_inuse_bytes
go_memstats_mcache_sys_bytes
go_memstats_mspan_inuse_bytes
go_memstats_mspan_sys_bytes
go_memstats_next_gc_bytes
go_memstats_other_sys_bytes
go_memstats_stack_inuse_bytes
go_memstats_stack_sys_bytes
go_memstats_sys_bytes
http_request_duration_microseconds
http_request_duration_microseconds_count
http_request_duration_microseconds_sum
http_request_size_bytes
http_request_size_bytes_count
http_request_size_bytes_sum
http_requests_total
http_response_size_bytes
http_response_size_bytes_count
http_response_size_bytes_sum
process_cpu_seconds_total
```

Monitoring Docker Containers

- cAdvisor

```
docker container run \
  --volume=/:/rootfs:ro \
  --volume=/var/run:/var/run:rw \
  --volume=/sys:/sys:ro \
  --volume=/var/lib/docker/:/var/lib/docker:ro \
  --publish=8080:8080 \
  --detach=true \
  --name=cadvisor \
  google/cadvisor:latest
```


localhost:8080/containers/

cAdvisor

/

root

Docker Containers

Subcontainers

/docker

/rngd

Isolation

CPU

Shares 1024 *shares*

Allowed Cores 0 1 2 3

Memory

Reservation unlimited

Limit 1.95 *GB*

Swap Limit 1024.00 *MB*

Usage

Overview

Integration Testing

Integration tests with Docker

- Start services with `docker-compose.yml` for tests
  - no volumes mapped (no data will be stored when the test is over)
  - no published ports (allows simultaneous tests)
- Run the application
- Run integration tests
- Stop services - clean environment for the next test

Running simultaneous tests

```
docker-compose -p app-$BUILD_NUMBER up
```

References

- Slides: github.com/docker/labs/tree/master/slides
- Workshop: github.com/docker/labs/tree/master/java
- Docs: docs.docker.com