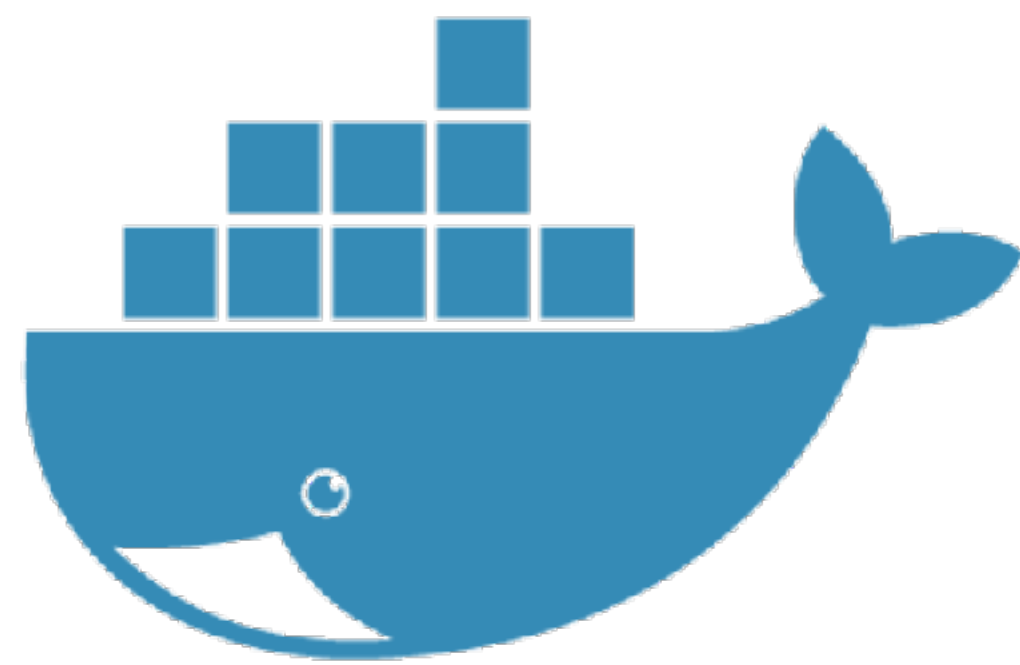


Docker for Java Developers



Fabiane Nardon, @fabianenardon
Arun Gupta, @arungupta

Java Champion
Duke's Choice Award Winner (2 years)
SouJava Founder
Data Scientist / Big Data expert



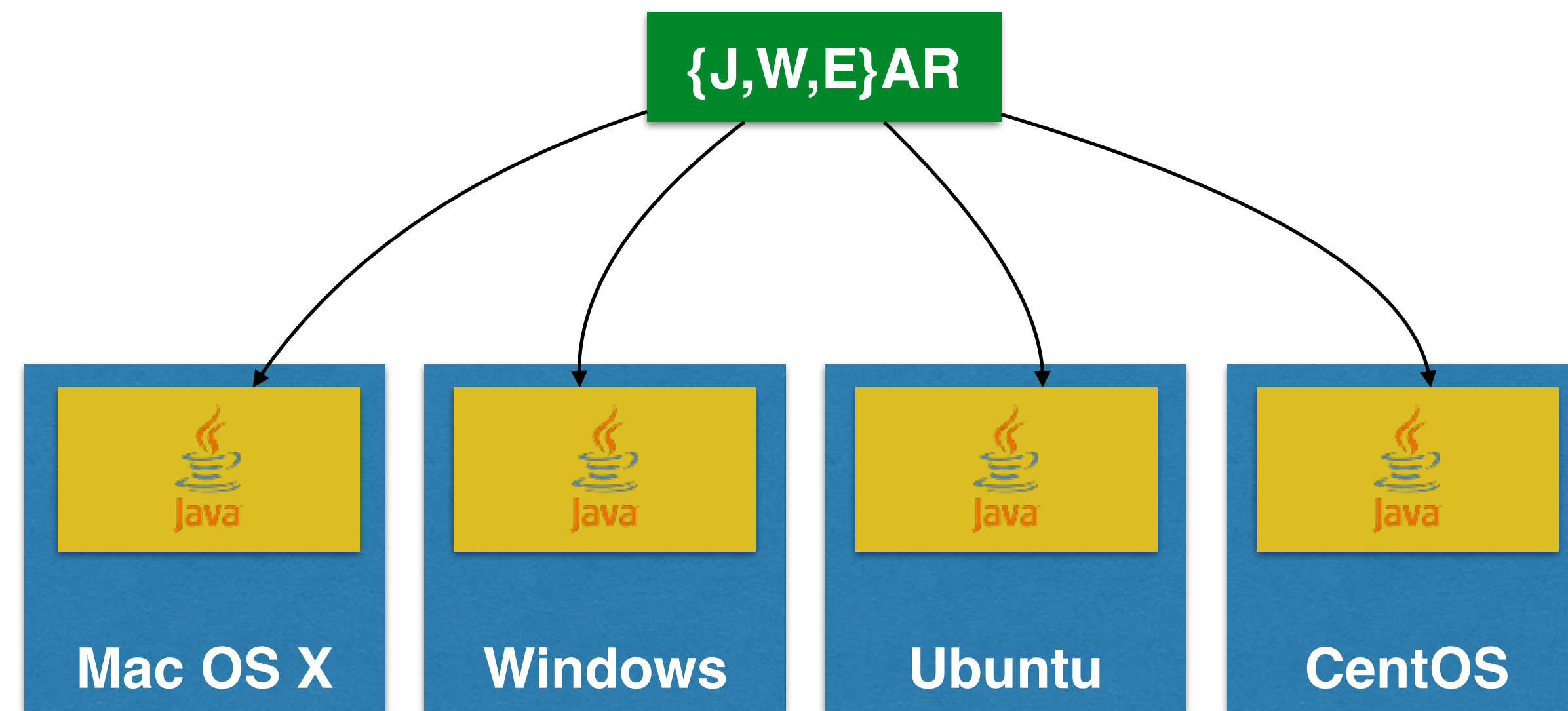
Docker Captain
Java Champion
JavaOne Rock Star (4 years)
NetBeans Dream Team
Silicon Valley JUG Leader
Author
Runner
Lifelong learner



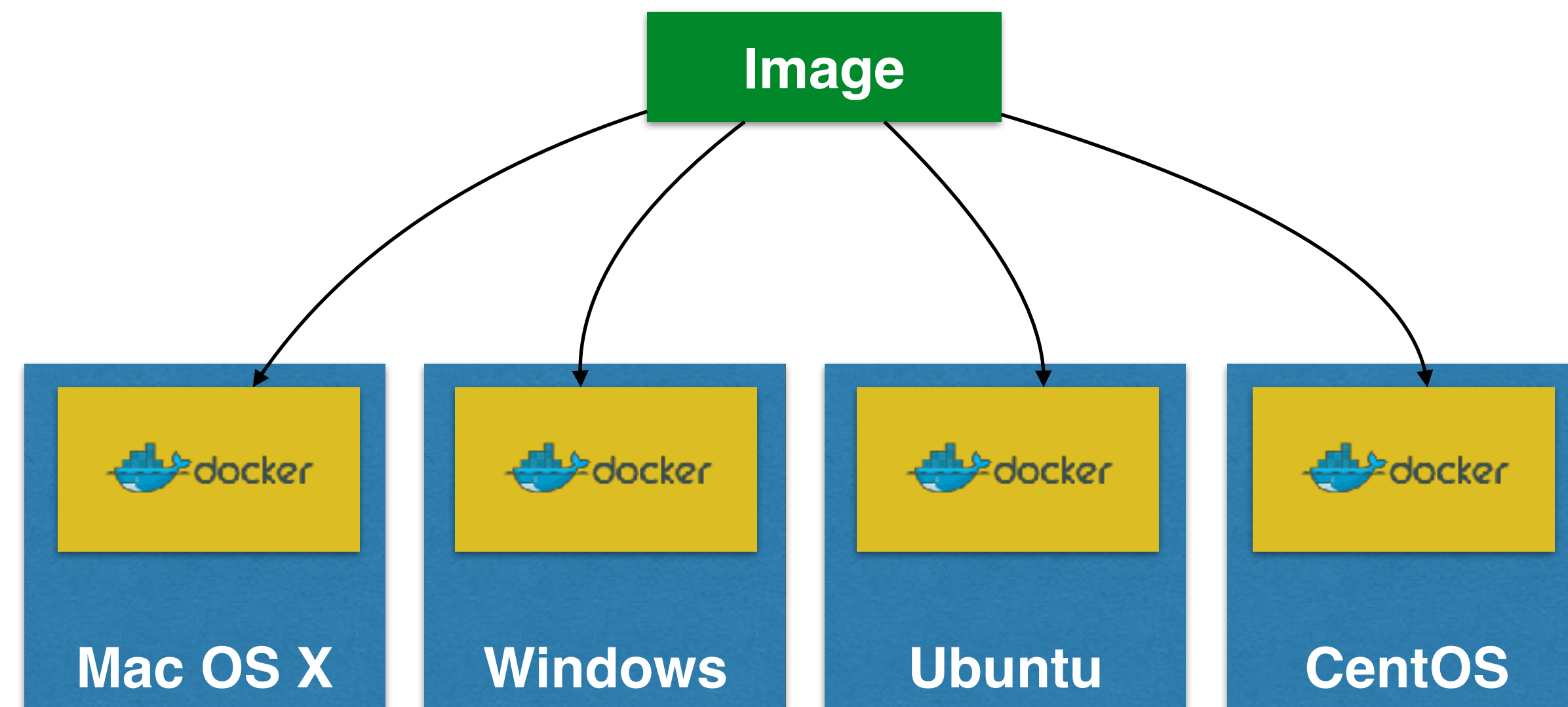
What we plan to cover?

- Java Base Image
- Package Java application with Maven and Gradle
- Multi-container application on single/multiple host(s)
- Scaling apps on AWS or Azure
- Memory management
- Debugging Java applications
- Monitor Java application
- Deployment pipeline for Java applications with Docker

Java Base Image



WORA = Write Once Run Anywhere




PODA = Package Once Deploy Anywhere


Java base image #1

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

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



OFFICIAL REPOSITORY

java 

Last pushed: 17 days ago

Repo Info [Tags](#)

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/



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-jdk/windowsservercore/Dockerfile](#))



```
38     cd /tmp && unzip /tmp/jce_policy-${JAVA_VERSION_MAJOR}.zip && \
39     cp -v /tmp/UnlimitedJCEPolicyJDK8/*.jar /opt/jdk/jre/lib/security; \
40     fi && \
41     sed -i s/#networkaddress.cache.ttl=-1/networkaddress.cache.ttl=10/ $JAVA_HOME/jre/lib/security/java.security && \
42     apk del curl glibc-i18n && \
43     rm -rf /opt/jdk/*src.zip \
44         /opt/jdk/lib/missioncontrol \
45         /opt/jdk/lib/visualvm \
46         /opt/jdk/lib/*javafx* \
47         /opt/jdk/jre/plugin \
48         /opt/jdk/jre/bin/javaws \
49         /opt/jdk/jre/bin/jjs \
50         /opt/jdk/jre/bin/orbd \
51         /opt/jdk/jre/bin/pack200 \
52         /opt/jdk/jre/bin/policytool \
53         /opt/jdk/jre/bin/rmid \
54         /opt/jdk/jre/bin/rmiregistry \
55         /opt/jdk/jre/bin/servertool \
56         /opt/jdk/jre/bin/tnameserv \
57         /opt/jdk/jre/bin/unpack200 \
58         /opt/jdk/jre/lib/javaws.jar \
59         /opt/jdk/jre/lib/deploy* \
60         /opt/jdk/jre/lib/desktop \
61         /opt/jdk/jre/lib/*javafx* \
62         /opt/jdk/jre/lib/*jfx* \
63         /opt/jdk/jre/lib/amd64/libdecora_sse.so \
64         /opt/jdk/jre/lib/amd64/libprism_*.so \
65         /opt/jdk/jre/lib/amd64/libfxplugins.so \
66         /opt/jdk/jre/lib/amd64/libglass.so \
67         /opt/jdk/jre/lib/amd64/libgstreamer-lite.so \
68         /opt/jdk/jre/lib/amd64/libjavafx*.so \
69         /opt/jdk/jre/lib/amd64/libjfx*.so \
70         /opt/jdk/jre/lib/ext/jfxrt.jar \
71         /opt/jdk/jre/lib/ext/nashorn.jar \
72         /opt/jdk/jre/lib/oblique-fonts \
73         /opt/jdk/jre/lib/plugin.jar \
74         /tmp/* /var/cache/apk/* && \
75     echo 'hosts: files mdns4_minimal [NOTFOUND=return] dns mdns4' >> /etc/nsswitch.conf
76
77 # EOF
```

Java base image #3


Add snapshot from container-
registry.oracle.com

Only in US/UK/Australia

https://hub.docker.com/_/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-jdk/windowsservercore/Dockerfile](#))


Java base image #4

Snapshot from EC2 Container Registry
Also available in Artifactory

https://hub.docker.com/_/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-jdk/windowsservercore/Dockerfile](#))


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

[Repo Info](#) [Tags](#) [Dockerfile](#) [Build Details](#)

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 and verified builds of the latest versions of the OpenJDK 8, 7, and 6 platforms. Zulu is available for 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

First Java Application

```
FROM openjdk:jdk-alpine
```

```
CMD java -version
```

First Java Web Application

```
FROM jboss/wildfly:10.1.0.Final
```

```
COPY target/webapp.war /opt/jboss/wildfly/  
standalone/deployments/webapp.war
```

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

```
63     <plugin>
64         <groupId>io.fabric8</groupId>
65         <artifactId>docker-maven-plugin</artifactId>
66         <version>0.20.1</version>
67         <configuration>
68             <images>
69                 <image>
70                     <name>hellojava</name>
71                     <build>
72                         <from>openjdk:latest</from>
73                         <assembly>
74                             <descriptorRef>artifact</descriptorRef>
75                         </assembly>
76                         <cmd>java -jar maven/${project.name}-${project.version}.jar</cmd>
77                     </build>
78                     <run>
79                         <wait>
80                             <log>Hello World!</log>
81                         </wait>
82                     </run>
83                 </image>
84             </images>
85         </configuration>
```

Maven - Execution

```
86         <executions>
87             <execution>
88                 <id>docker:build</id>
89                 <phase>package</phase>
90                 <goals>
91                     <goal>build</goal>
92                 </goals>
93             </execution>
94             <execution>
95                 <id>docker:start</id>
96                 <phase>install</phase>
97                 <goals>
98                     <goal>run</goal>
99                     <goal>logs</goal>
100                 </goals>
101             </execution>
102         </executions>
103     </plugin>
104 </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`, ...

Gradle - Configuration

```
1  buildscript {
2      repositories {
3          jcenter()
4      }
5
6      dependencies {
7          classpath 'com.bmuschko:gradle-docker-plugin:3.0.6'
8      }
9  }
10
11  apply plugin: 'java'
12  apply plugin: 'application'
13  apply plugin: 'com.bmuschko.docker-java-application'
14
15  import com.bmuschko.gradle.docker.tasks.container.*
16  import com.bmuschko.gradle.docker.tasks.image.*
17
18  sourceCompatibility = 1.8
19  targetCompatibility = 1.8
```

Gradle - Execution

```
30  docker {
31      javaApplication {
32          baseImage = 'openjdk:latest'
33          tag = 'hellojava'
34      }
35  }
36
37  task createContainer(type: DockerCreateContainer) {
38      dependsOn dockerBuildImage
39      targetImageId { dockerBuildImage.getImageId() }
40  }
41
42  task startContainer(type: DockerStartContainer) {
43      dependsOn createContainer
44      targetContainerId { createContainer.getContainerId() }
45  }
```

Multi-Container Application

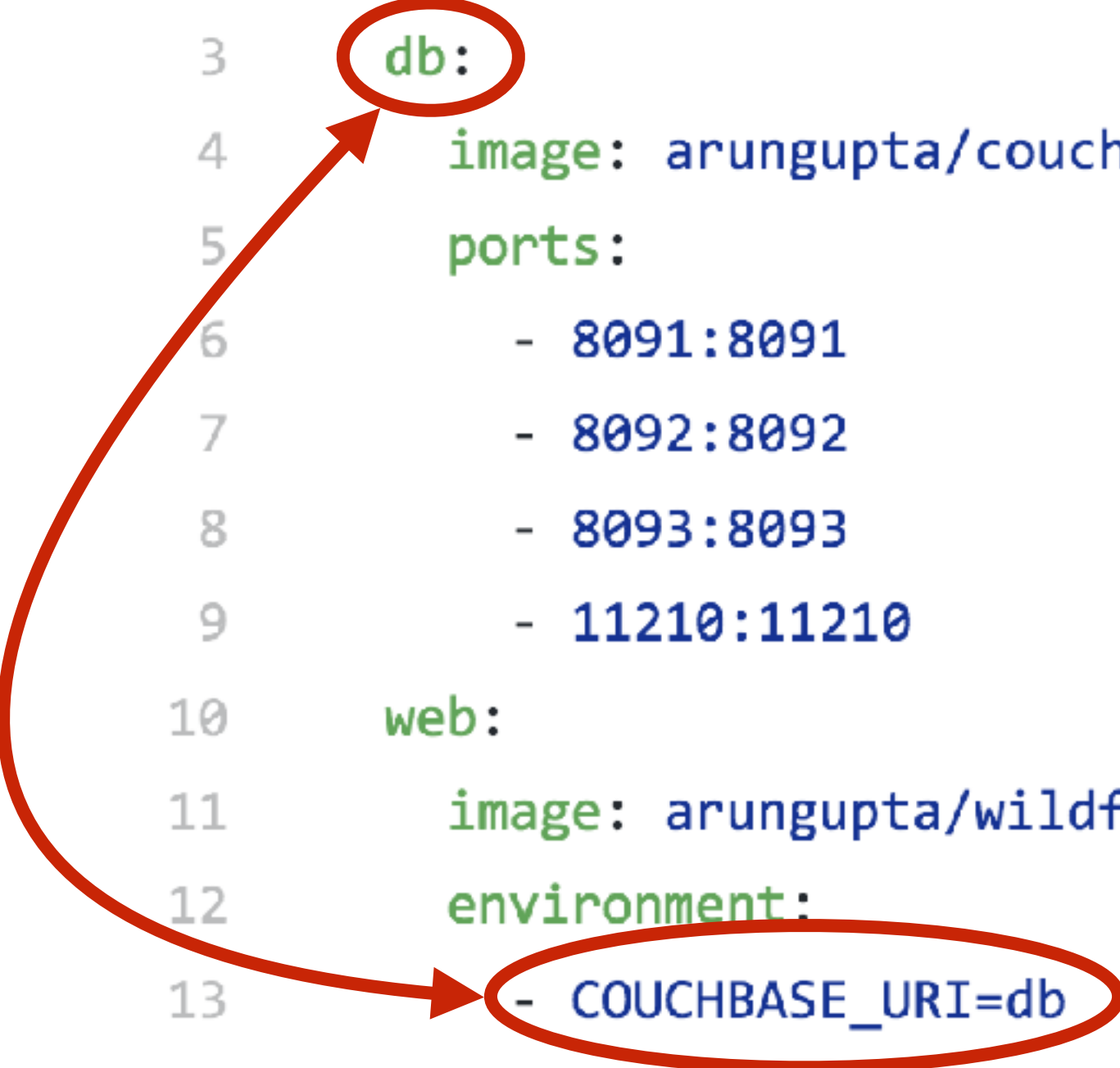


Docker Compose

- 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

```
1  version: "3"
2  services:
3    db:
4      image: arungupta/couchbase:travel
5      ports:
6        - 8091:8091
7        - 8092:8092
8        - 8093:8093
9        - 11210:11210
10   web:
11     image: arungupta/wildfly-couchbase-javaee:travel
12     environment:
13       - COUCHBASE_URI=db
14     ports:
15       - 8080:8080
16       - 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
```

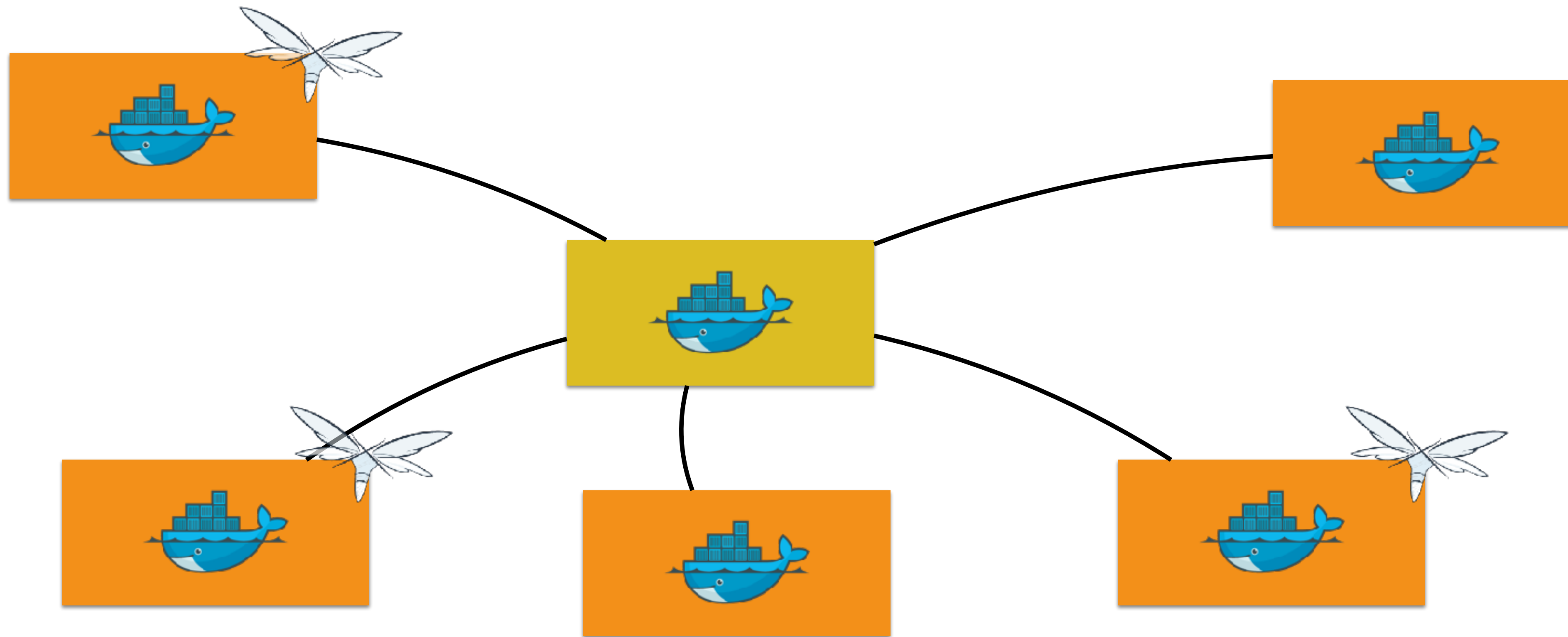
Multi-host using Swarm-mode



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

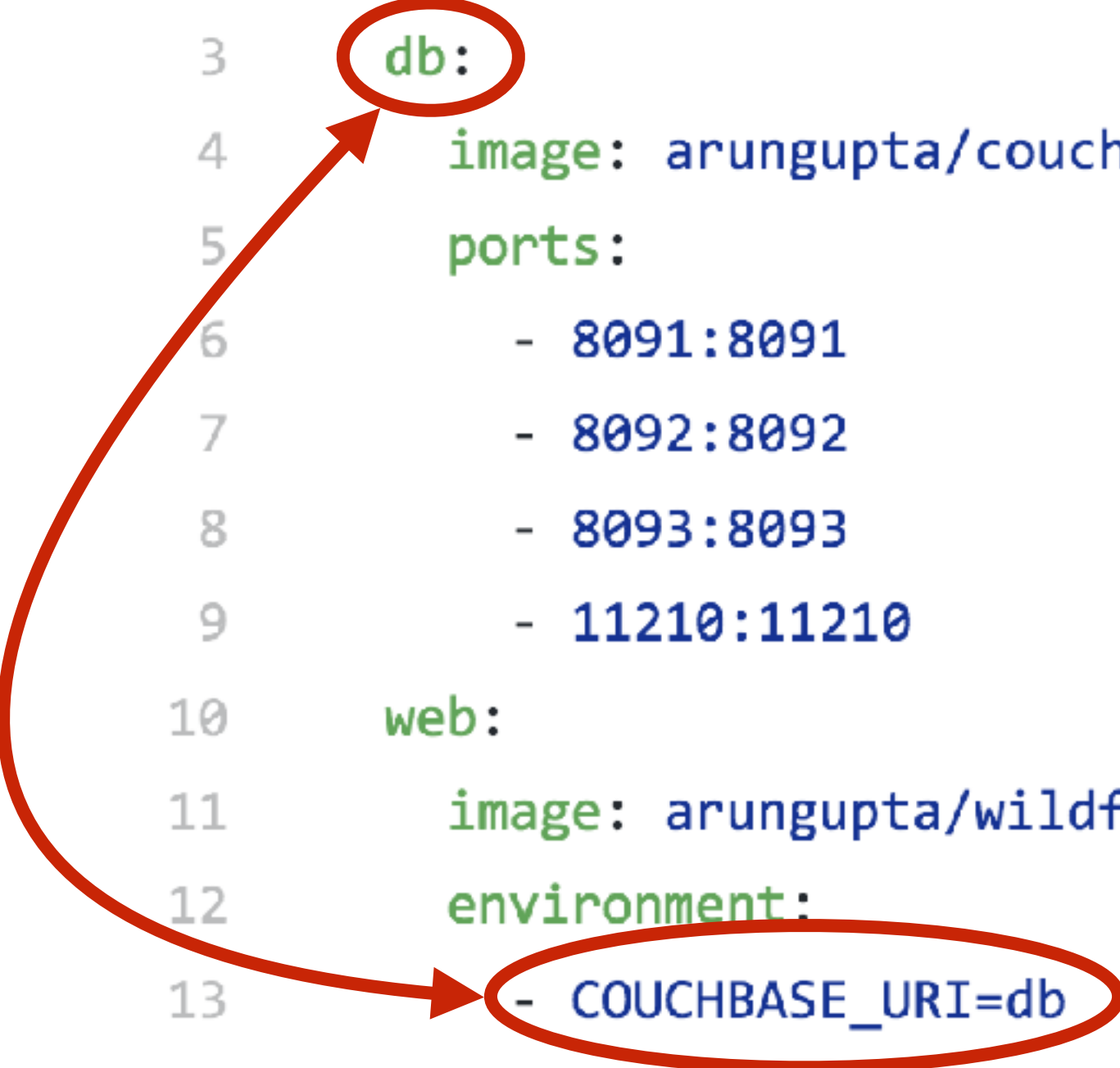
Swarm Mode: Replicated Service



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

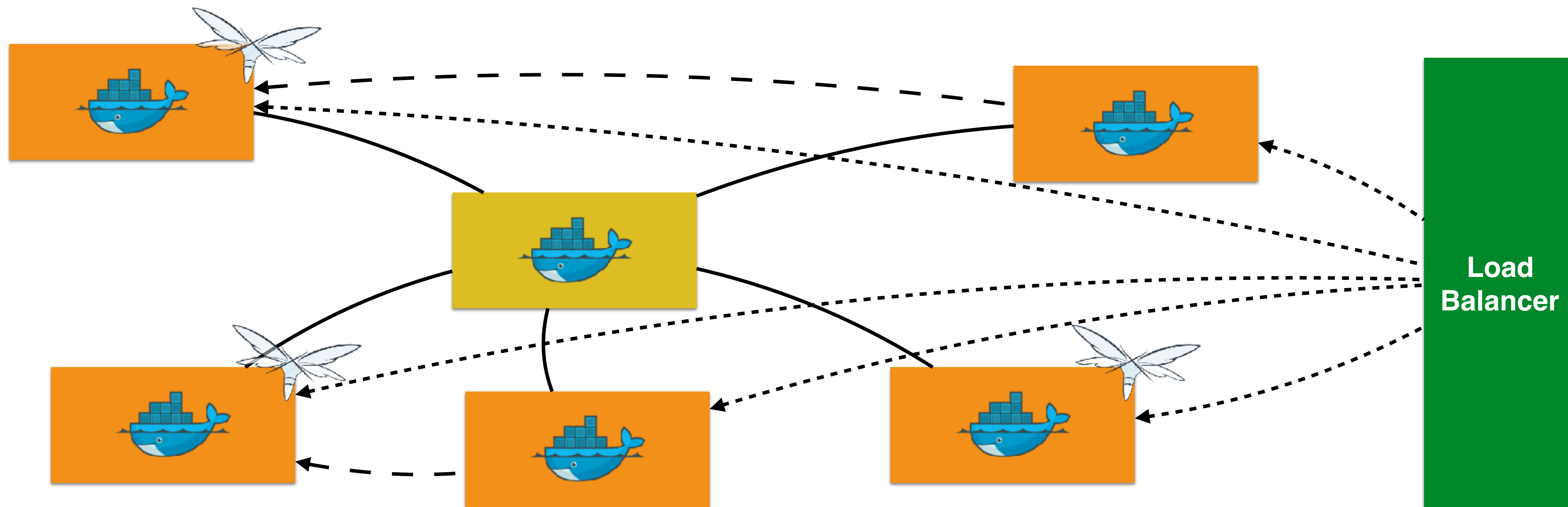

Multi-container on multiple hosts

```
1  version: "3"
2  services:
3    db:
4      image: arungupta/couchbase:travel
5      ports:
6        - 8091:8091
7        - 8092:8092
8        - 8093:8093
9        - 11210:11210
10   web:
11     image: arungupta/wildfly-couchbase-javaee:travel
12     environment:
13       - COUCHBASE_URI=db
14     ports:
15       - 8080:8080
16       - 9990:9990
```



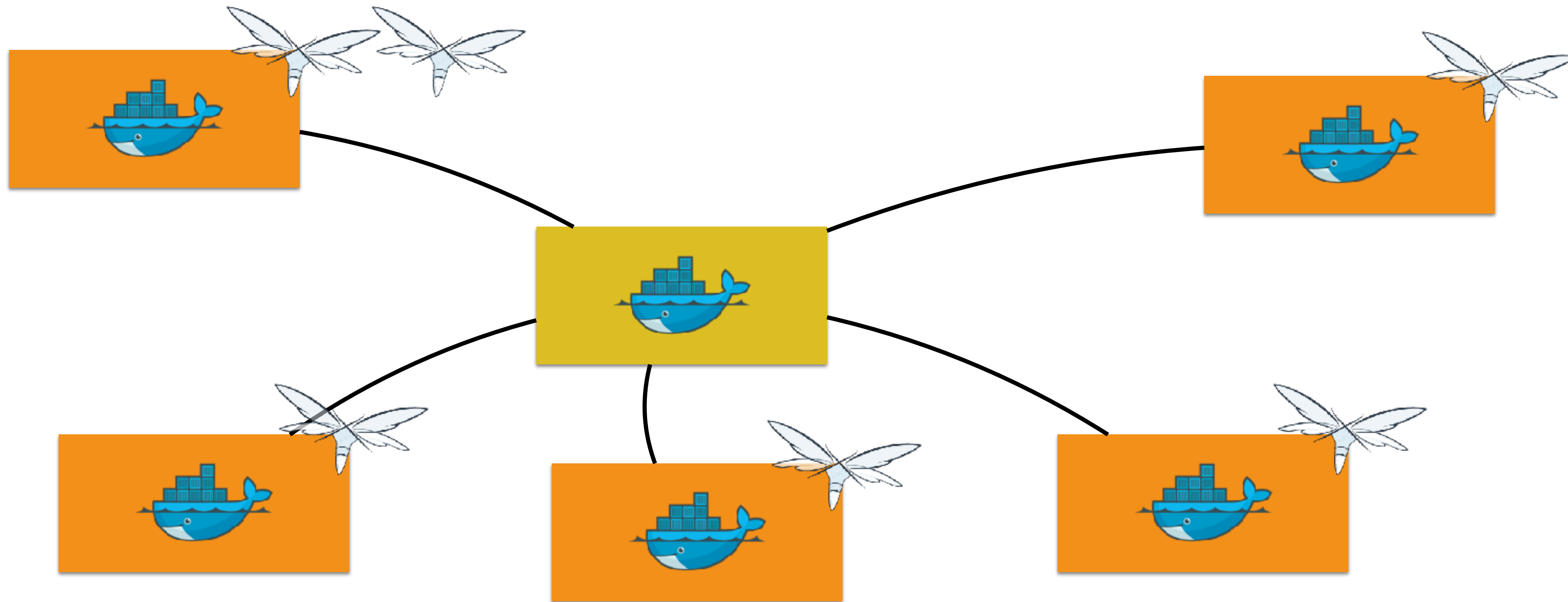
```
docker stack deploy --compose-file=docker-compose.yml webapp
```

Swarm Mode: Routing Mesh



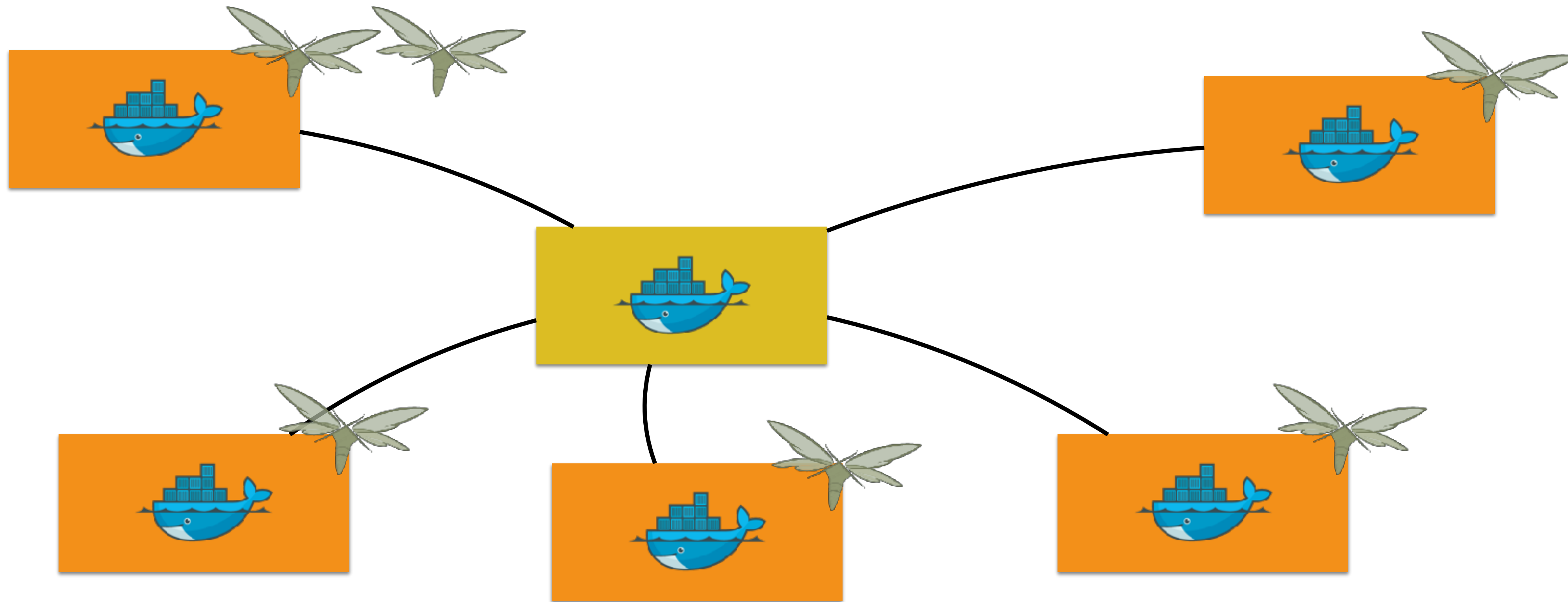
```
docker service create --replicas 3 --name web -p 8080:8080 jboss/wildfly
```

Swarm Mode: Scale



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

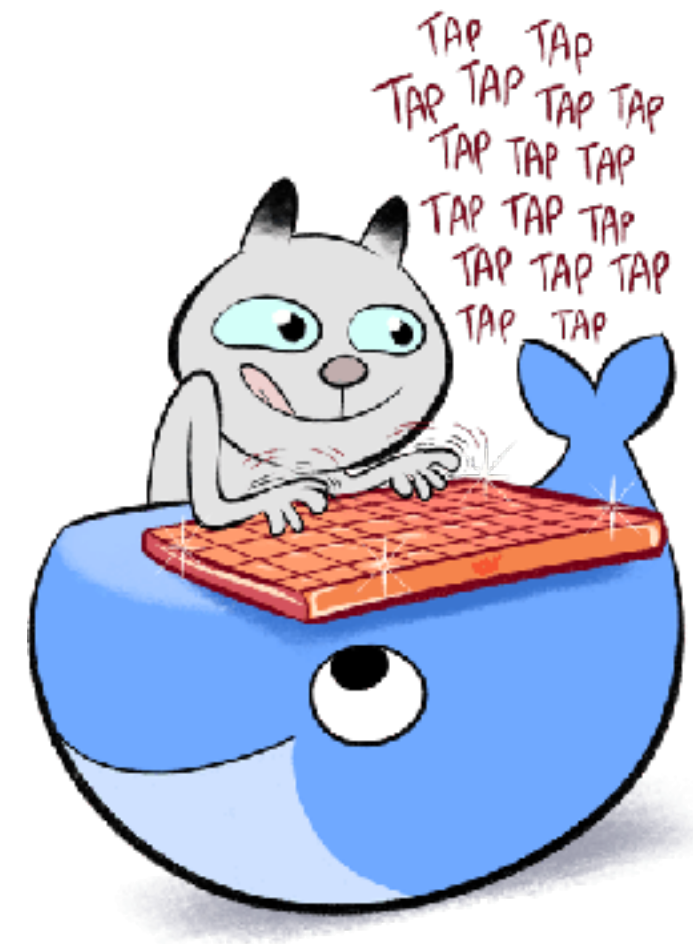
Swarm Mode: Rolling Updates



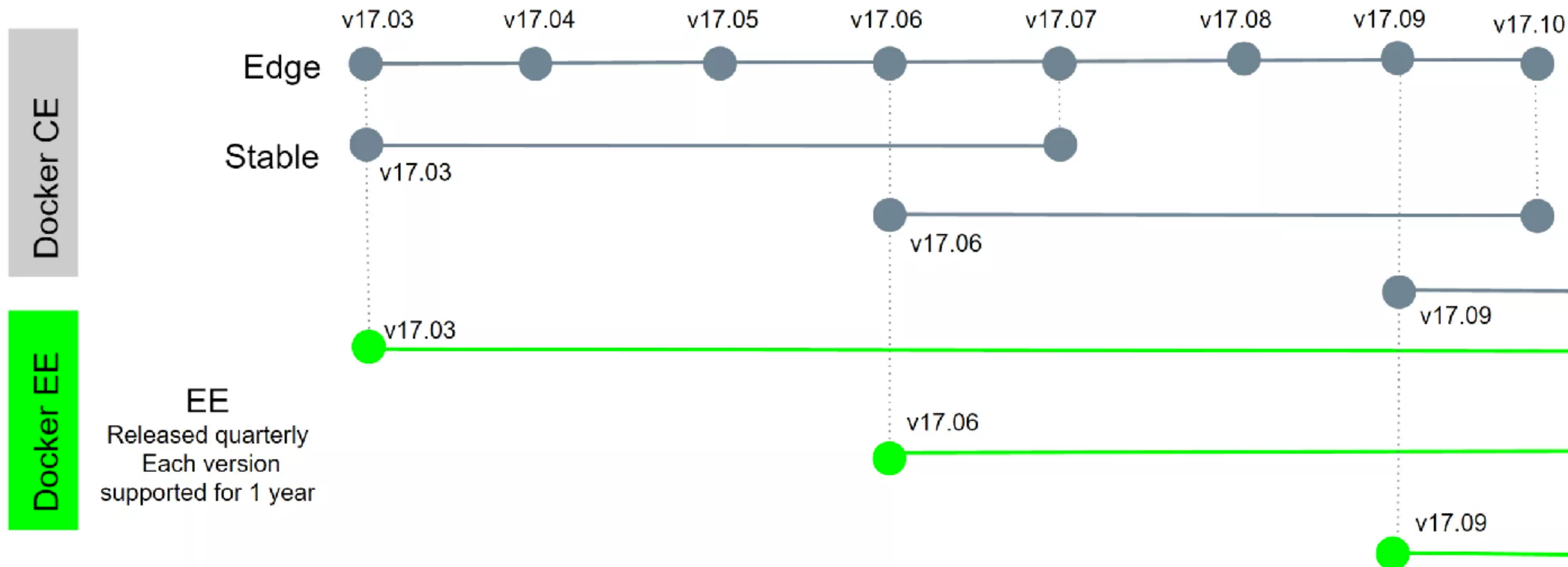
```
docker service update web --image wildfly:2 --update-parallelism  
2 --update-delay 10s
```

Scaling Apps on AWS or Azure

Docker Community Edition



- Docker for Mac/Windows/Linux
- Native desktop or cloud provider experience
- Monthly edge and quarterly stable release

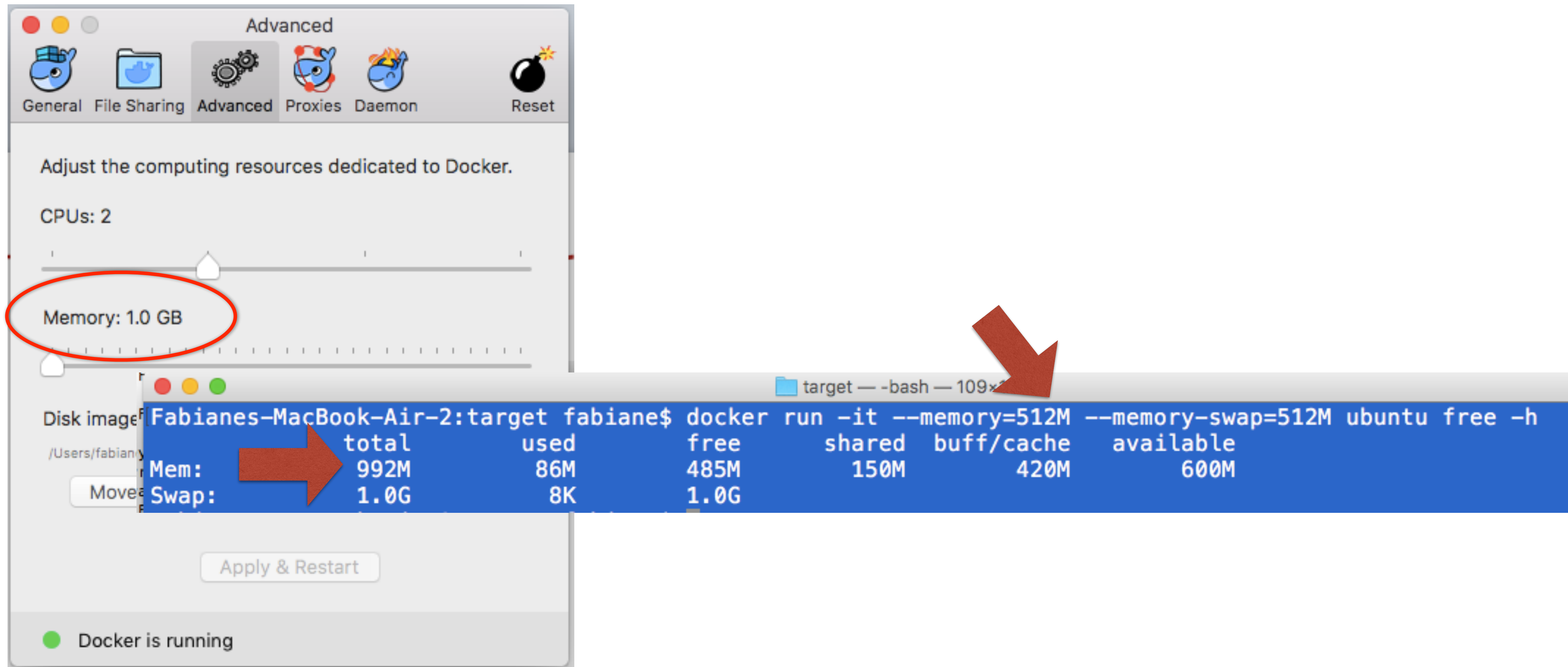


Docker for AWS/Azure

- 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

How much memory is available for containers?



The image shows the Docker Desktop 'Advanced' settings window on a Mac. The 'Memory' slider is set to 1.0 GB, which is circled in red. A red arrow points from this setting to a terminal window. The terminal window shows the output of the `free -h` command, with a red arrow pointing to the 'Mem' row. The terminal output is as follows:


```
Fabianes-MacBook-Air-2:target fabiane$ docker run -it --memory=512M --memory-swap=512M ubuntu free -h
```

	total	used	free	shared	buff/cache	available
Mem:	992M	86M	485M	150M	420M	600M
Swap:	1.0G	8K	1.0G			

The terminal window title is 'target — -bash — 109x2'. The Docker Desktop window shows 'Apply & Restart' and 'Docker is running'.


How much memory is available for containers?

```
target — -bash — 109x18  
$ docker run --memory=100M memory-sample
```



docker memory switches
instruct linux to kill the
process if limits are
exceeded

```
target — -bash — 109x18  
#2698842, Total: 253427712, Used: 148020032, Free: 105407680  
#2727631, Total: 253427712, Used: 149402536, Free: 104025176  
#2756421, Total: 253427712, Used: 150785036, Free: 102642688  
#2785210, Total: 253427712, Used: 152167536, Free: 101260184  
#2814000, Total: 253427712, Used: 153550036, Free: 99877696  
#2842789, Total: 253427712, Used: 154932536, Free: 98495192  
#2871579, Total: 253427712, Used: 156315036, Free: 97112704  
#2900368, Total: 253427712, Used: 157697536, Free: 95730200  
#2929158, Total: 253427712, Used: 159080036, Free: 94347712  
#2957947, Total: 253427712, Used: 160462536, Free: 92965208  
#2986737, Total: 253427712, Used: 161845036, Free: 91582720  
#3015526, Total: 253427712, Used: 163227496, Free: 90200216  
#3044316, Total: 253427712, Used: 164609984, Free: 88817728  
#3073105, Total: 253427712, Used: 165992488, Free: 87435224  
#3101895, Total: 253427712, Used: 167374976, Free: 86052736  
#3130684, Total: 253427712, Used: 168757480, Free: 84670232  
Killed
```



—memory means RAM
plus Swap

How much memory is available for containers?

The screenshot shows an IDE window with a Maven POM file named 'memory-sample'. The POM file contains the following XML structure:

```
<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>
          </includes>
          <outputDirectory>./</outputDirectory>
        </fileSet>
      </fileSets>
    </inline>
  </assembly>
  <cmd>java -jar $JAVA_OPTIONS maven/${project.name}-${project.version}-jar-with-dependencies.jar</cmd>
</build>
```

A red arrow points from the `<fileSet>` section of the POM file to a terminal window. The terminal window shows the command:

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

How much memory is available for containers?

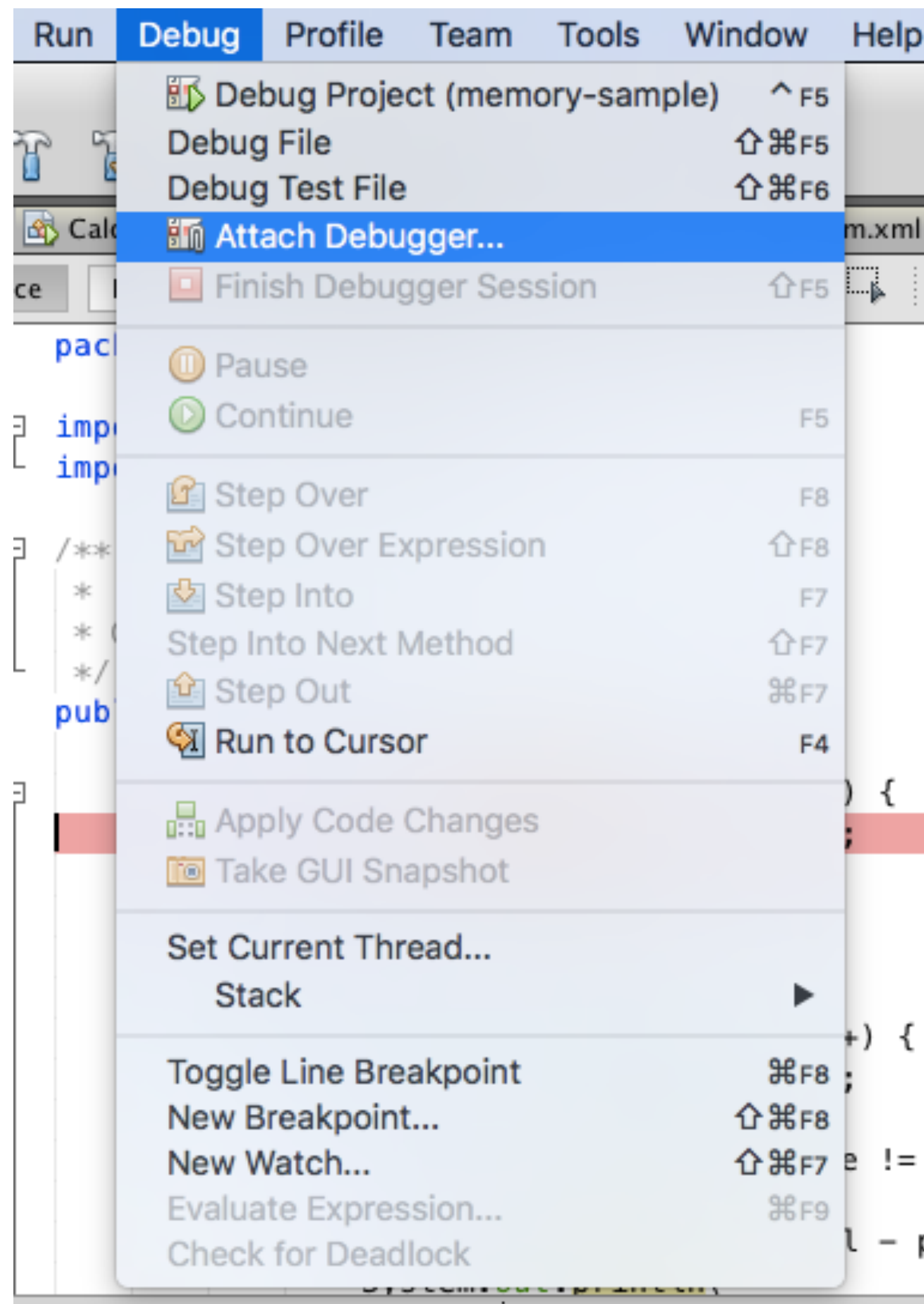
- By default, container will use as much memory and swap
- Can be restricted using `-memory`, `-memory-reservation`, `-memory-swap`
- Today, JDK unaware of container's limited resources
 - For example, memory or CPU using `-cpus`, `-cpu-shares`
- JDK 9 has experimental support for cgroup memory limits

Debugging Java Applications

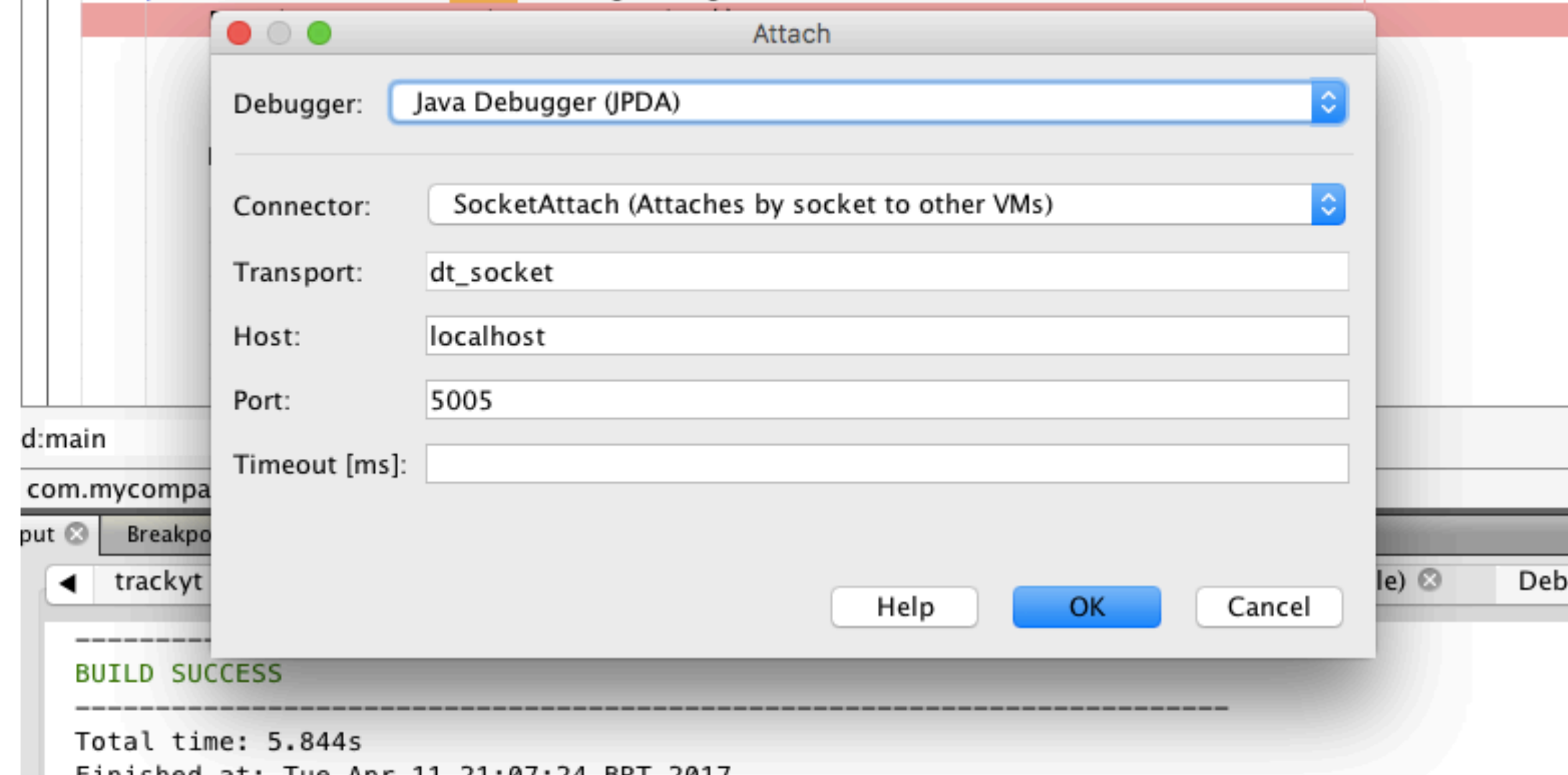
Running in debug mode

```
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) {
```



Monitor Java Applications

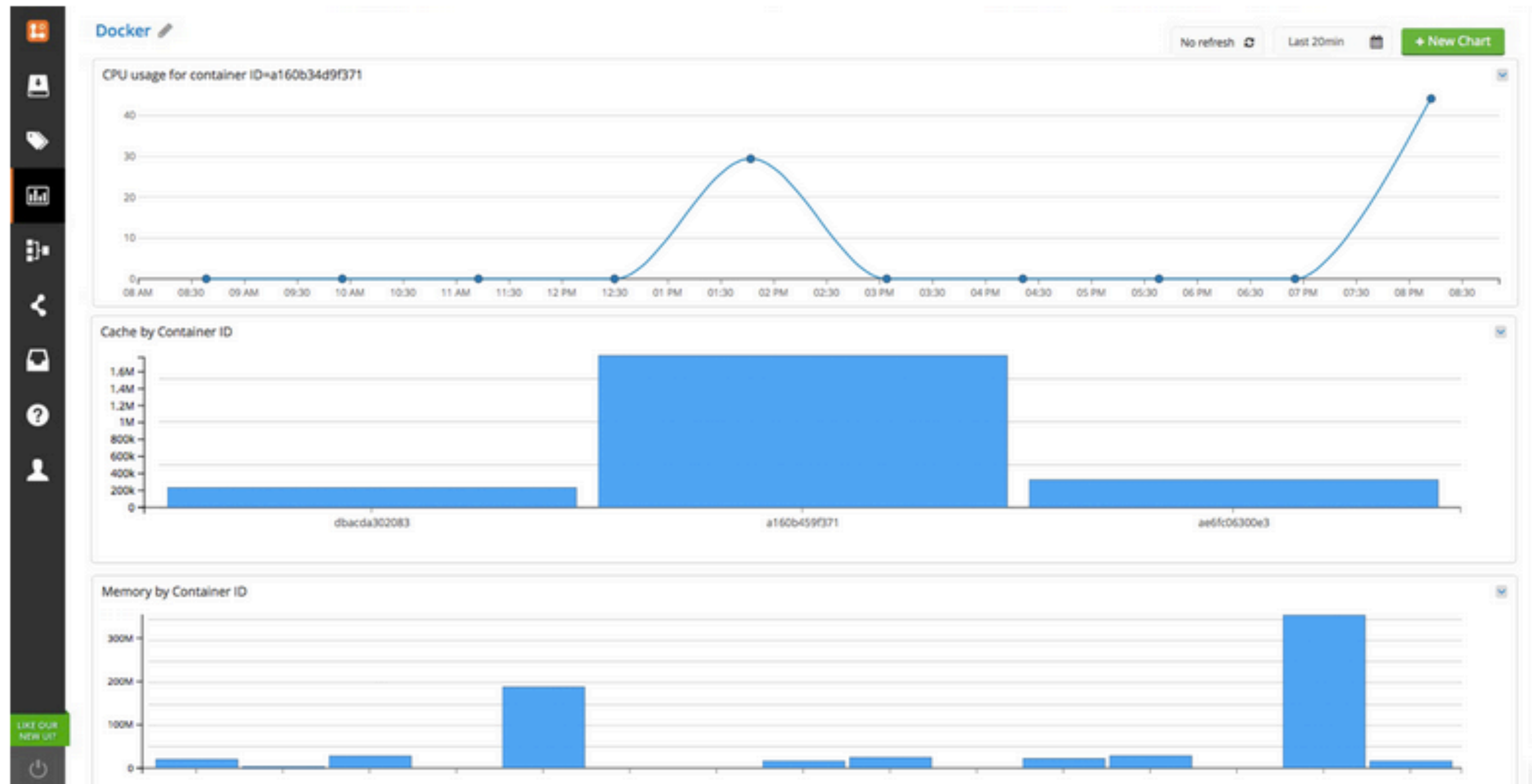
Monitoring Docker Containers

- `docker stats` command

CONTAINER	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O
db	2.02%	374.9 MiB / 1.952 GiB	18.76%	648 B / 648 B	0 B / 150

Monitoring Docker Containers

- LogEntries



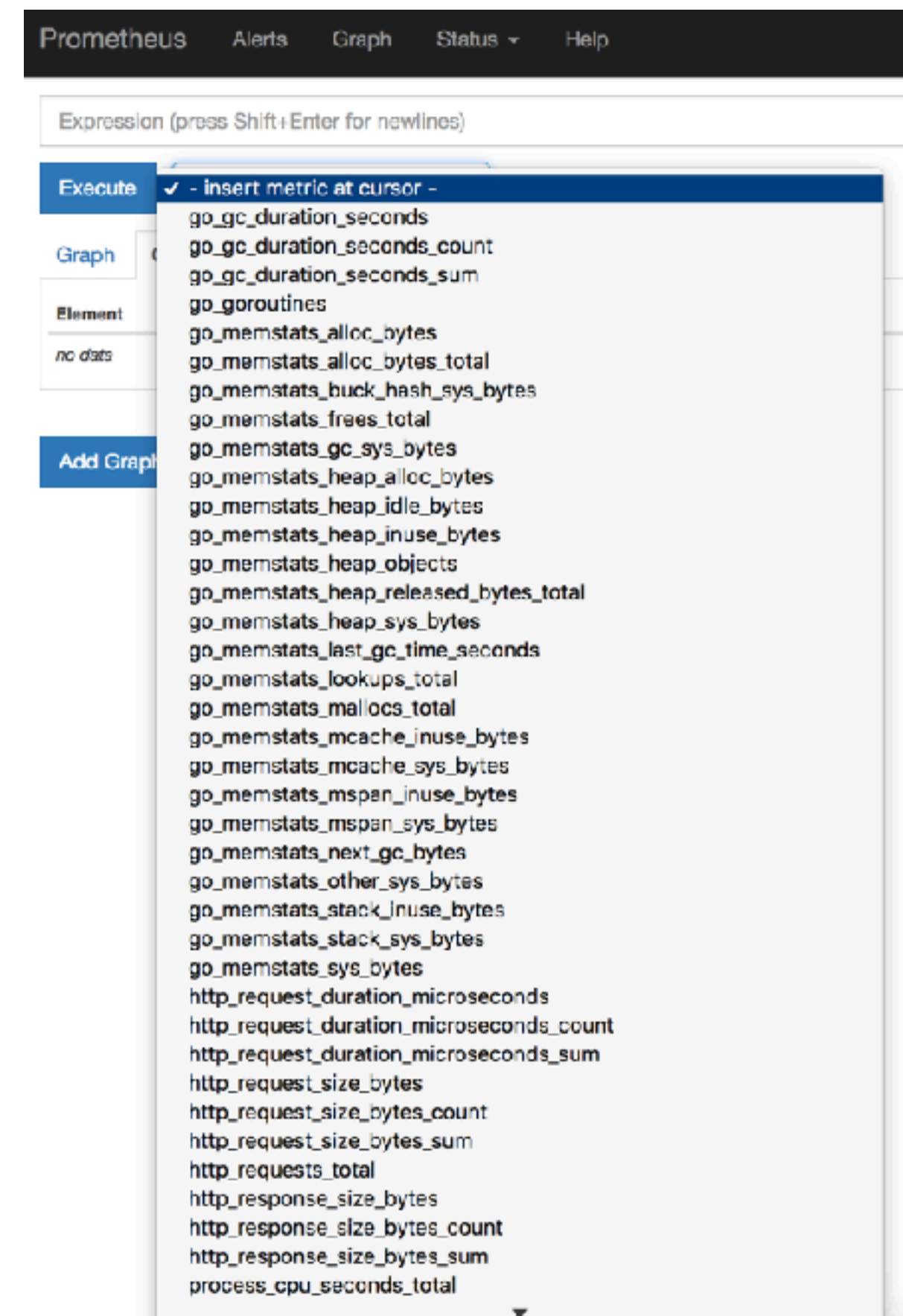
Monitoring Docker Containers

- Service logs `docker service logs <service>`

```
docker — -bash — 110x31
$ docker service logs mongo_mongo.1.6r99uf1qjwee@moby
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
```

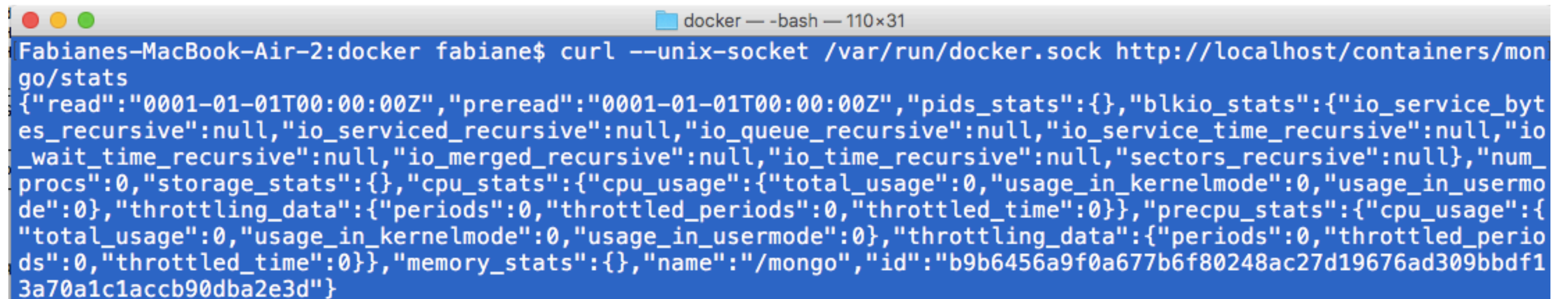

Monitoring Docker Containers

- Prometheus endpoint - New in 1.13



Monitoring Docker Containers

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

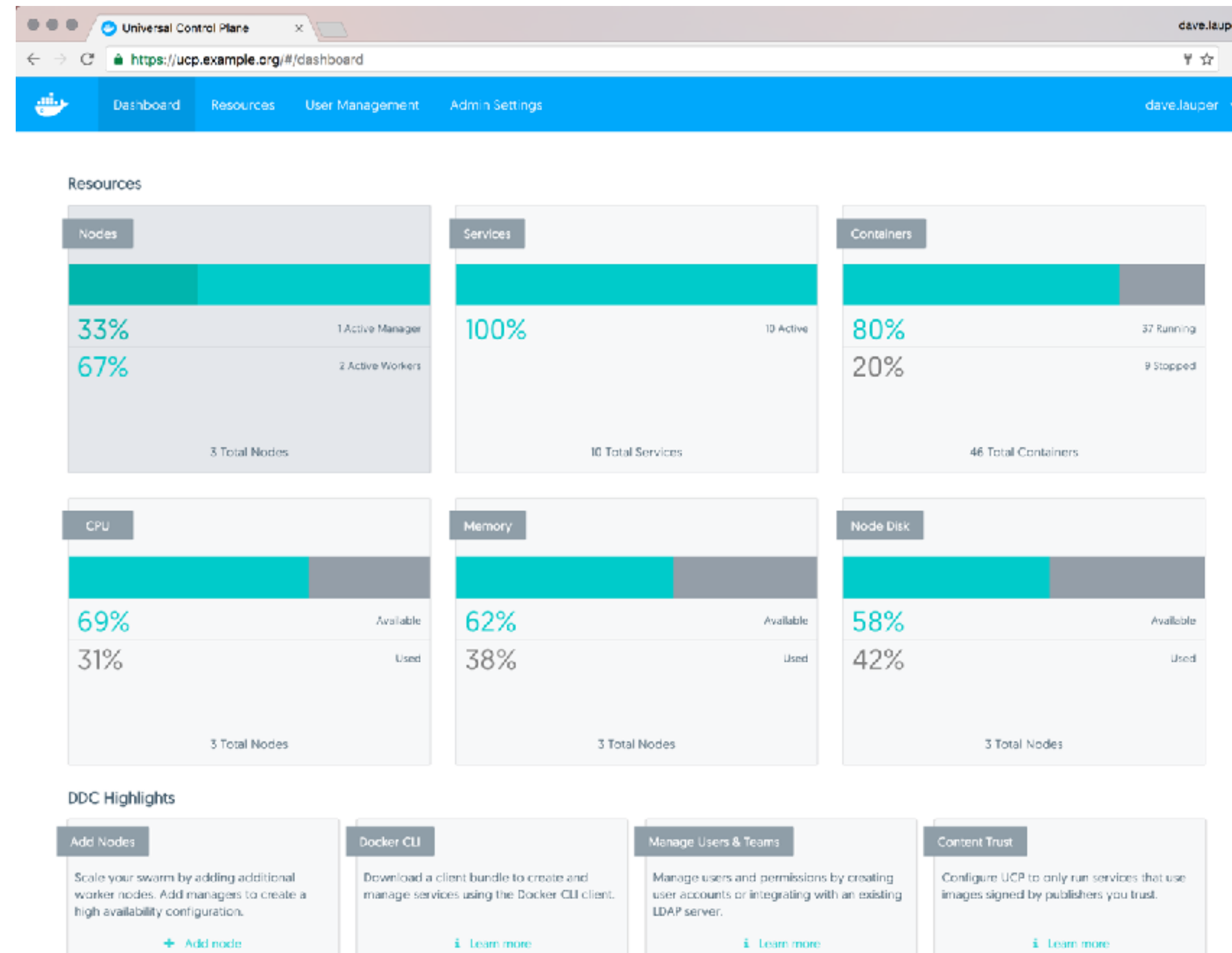


A terminal window titled "docker — -bash — 110x31" shows the command `curl --unix-socket /var/run/docker.sock http://localhost/containers/mongo/stats` being executed. The output is a JSON object representing the container's statistics.

```
Fabianes-MacBook-Air-2:docker fabiane$ curl --unix-socket /var/run/docker.sock http://localhost/containers/mongo/stats
{"read":"0001-01-01T00:00:00Z","preread":"0001-01-01T00:00:00Z","pids_stats":{},"blkio_stats":{"io_service_bytes_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_usermode":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_periods":0,"throttled_time":0}},"memory_stats":{},"name":"/mongo","id":"b9b6456a9f0a677b6f80248ac27d19676ad309bbdf13a70a1c1accb90dba2e3d"}
```

Monitoring Docker Containers

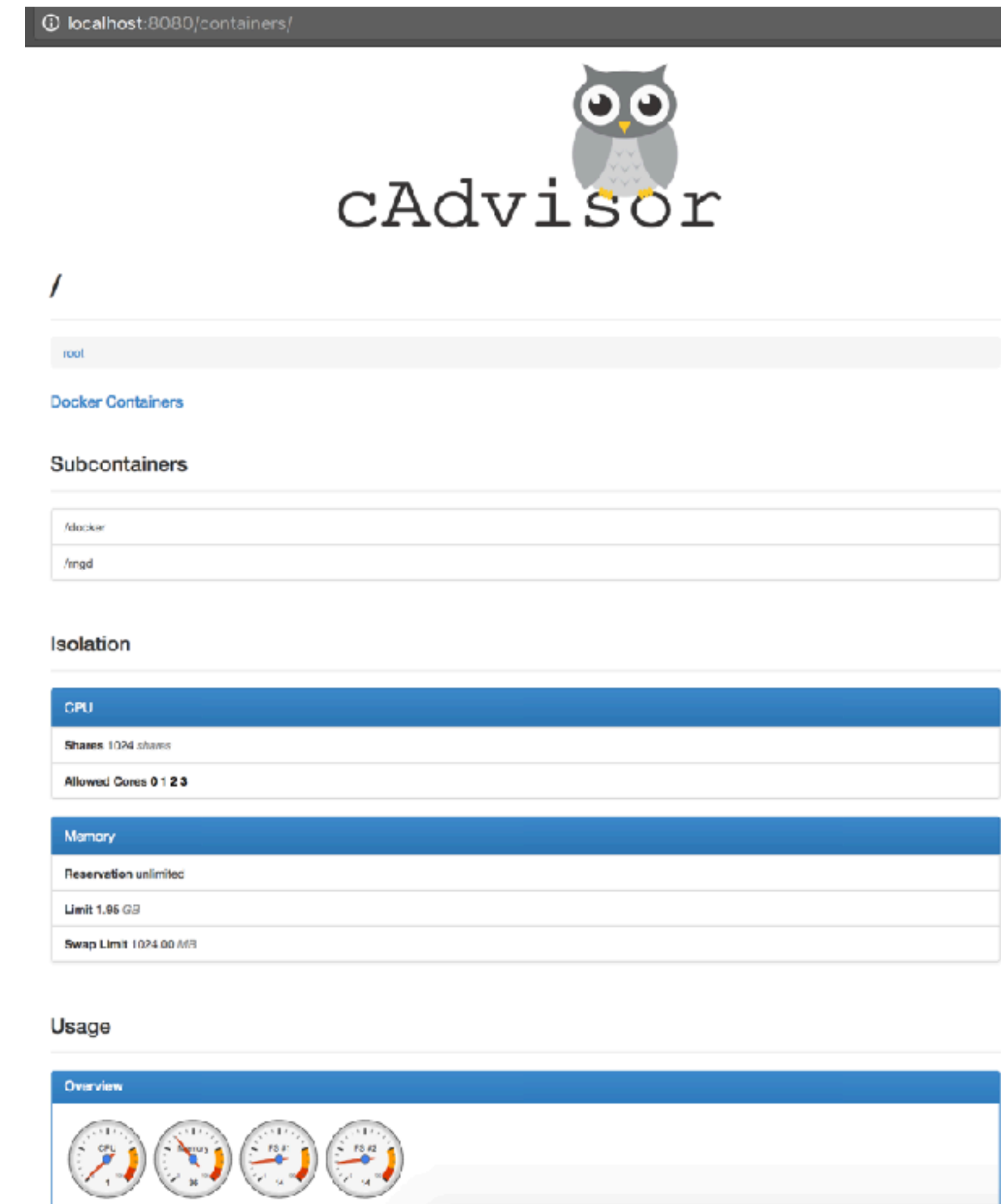
■ Docker Universal Control Pane



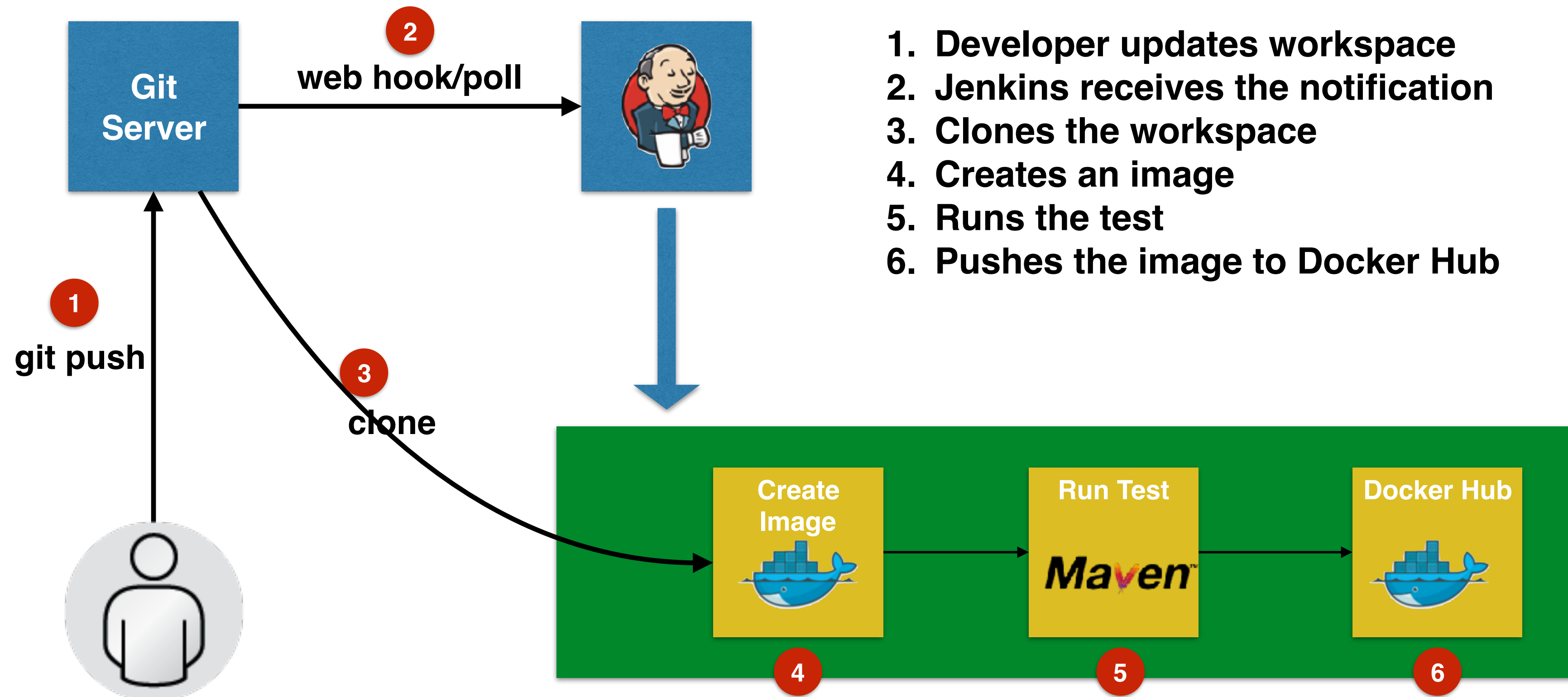
Monitoring Docker Containers

■ cAdvisor

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



CI/CD with Docker + Jenkins



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 - you'll have a 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