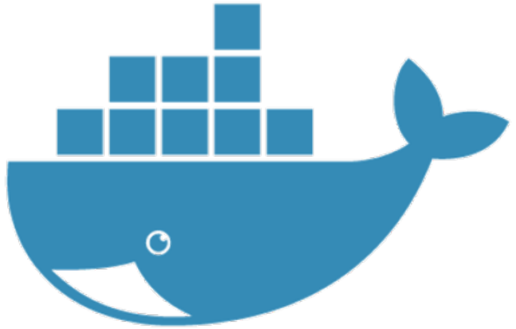
Docker for Java Developers



Fabiane Nardon, @fabianenardon
Arun Gupta, @arungupta

Fabiane’s introduction

Docker Captain

Java Champion

JavaOne Rock Star (4 years)

NetBeans Dream Team

Silicon Valley JUG Leader

Author

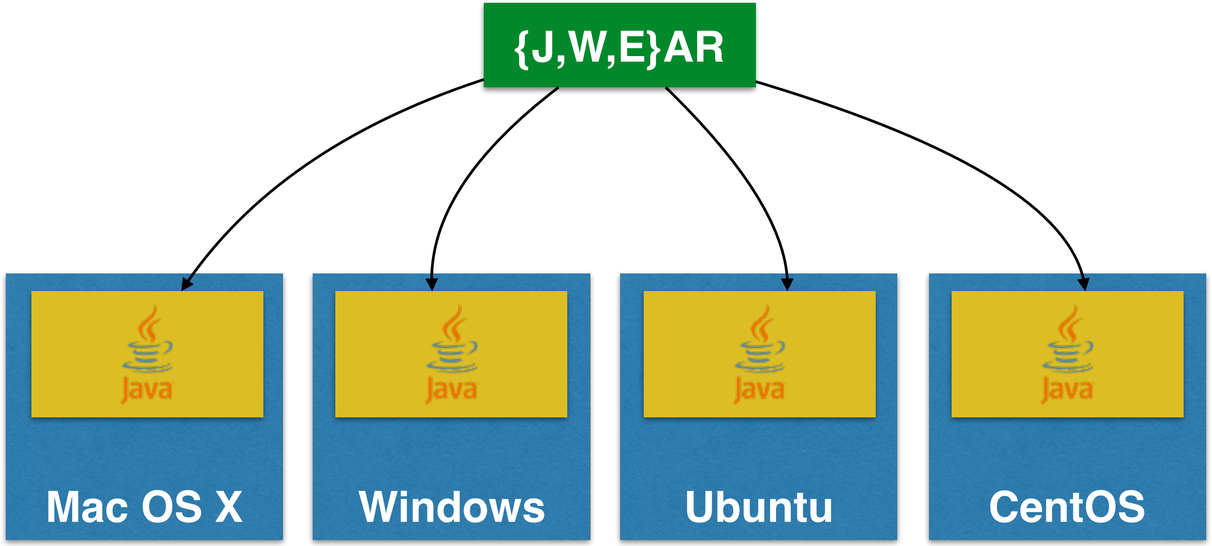
Runner

Lifelong learner

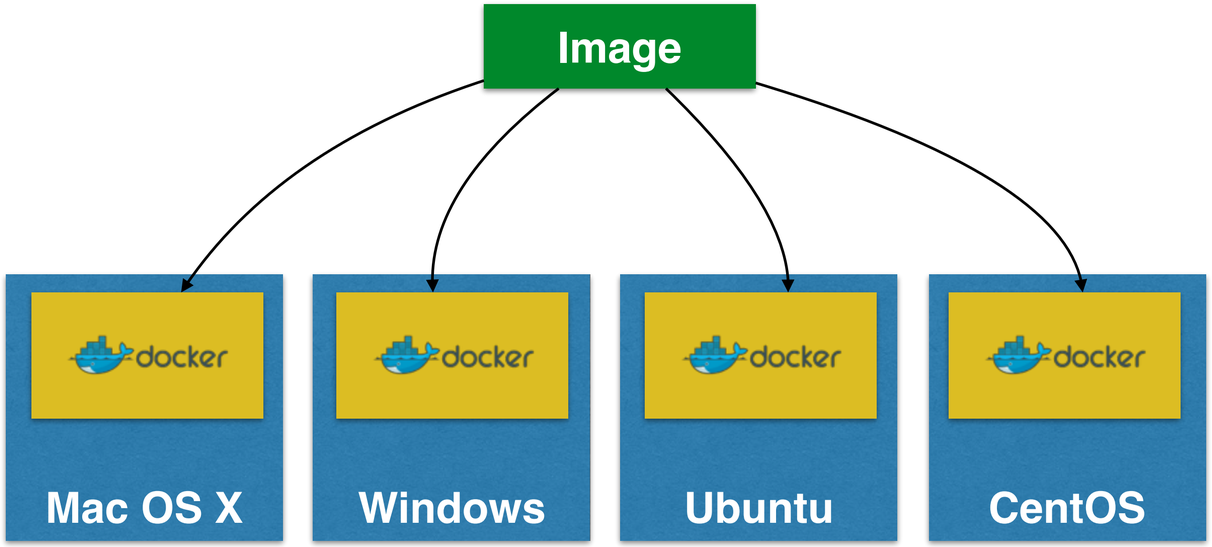
What we plan to cover?

- 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 Dock

java

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/

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

```
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/

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 #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

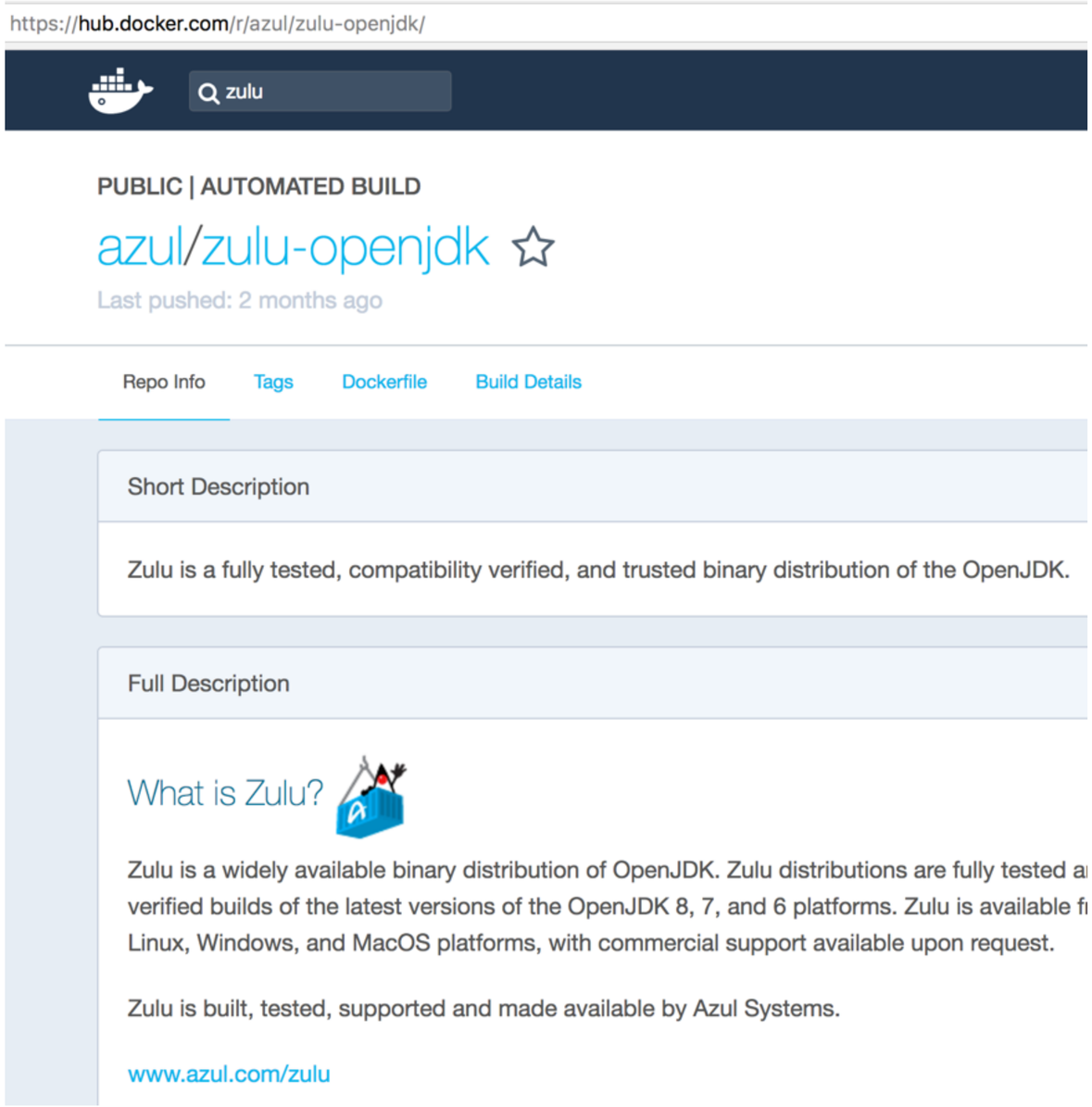
Full Description

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



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>
```

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

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`, …

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

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

```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

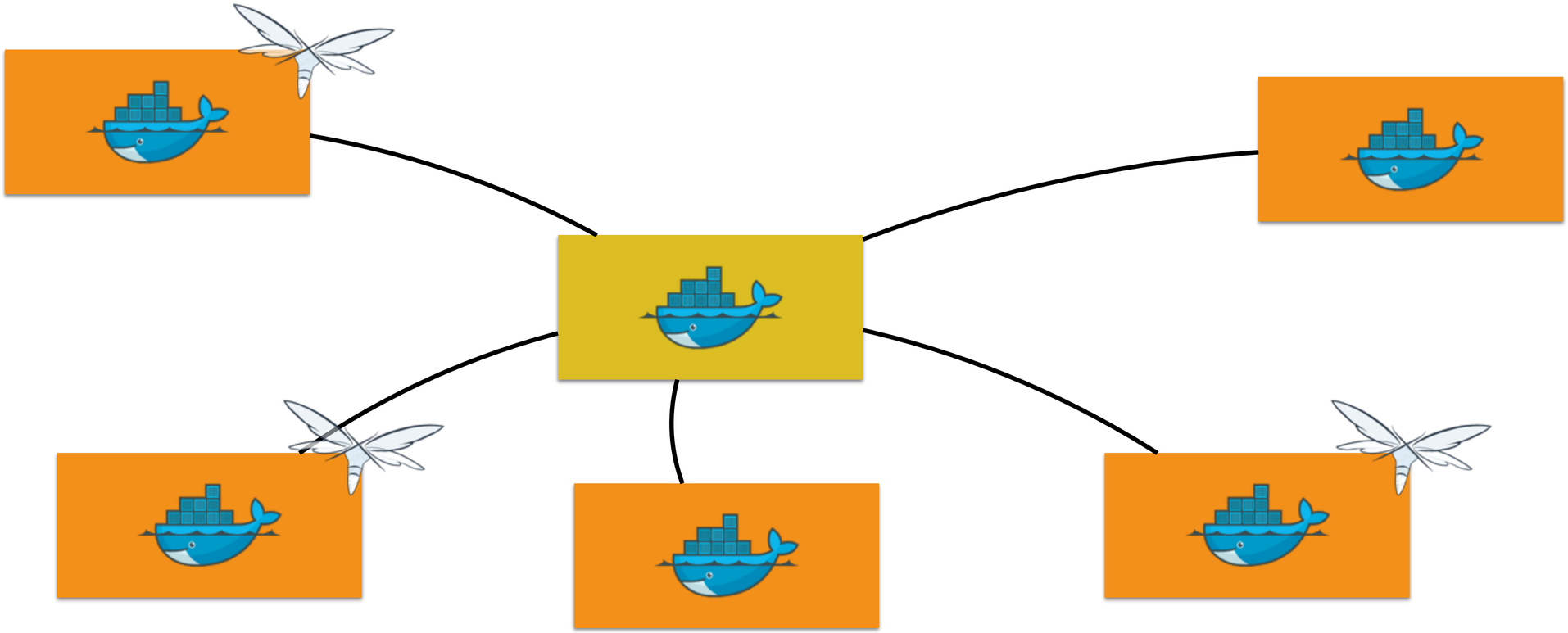
```
docker-compose \
-f docker-compose.yml \
-f docker-compose.db.yml \
down
```

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



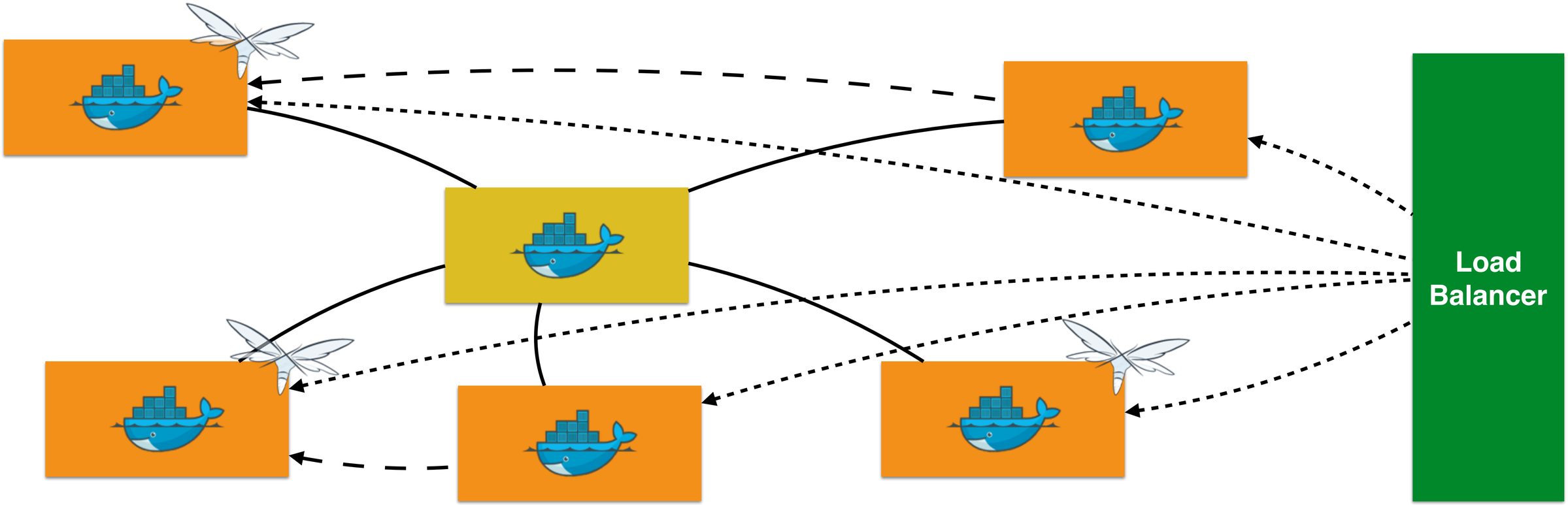
```
docker service create --replicas 3 --name web jboss/wildfly
```

Multi-container on multiple hosts

```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
```

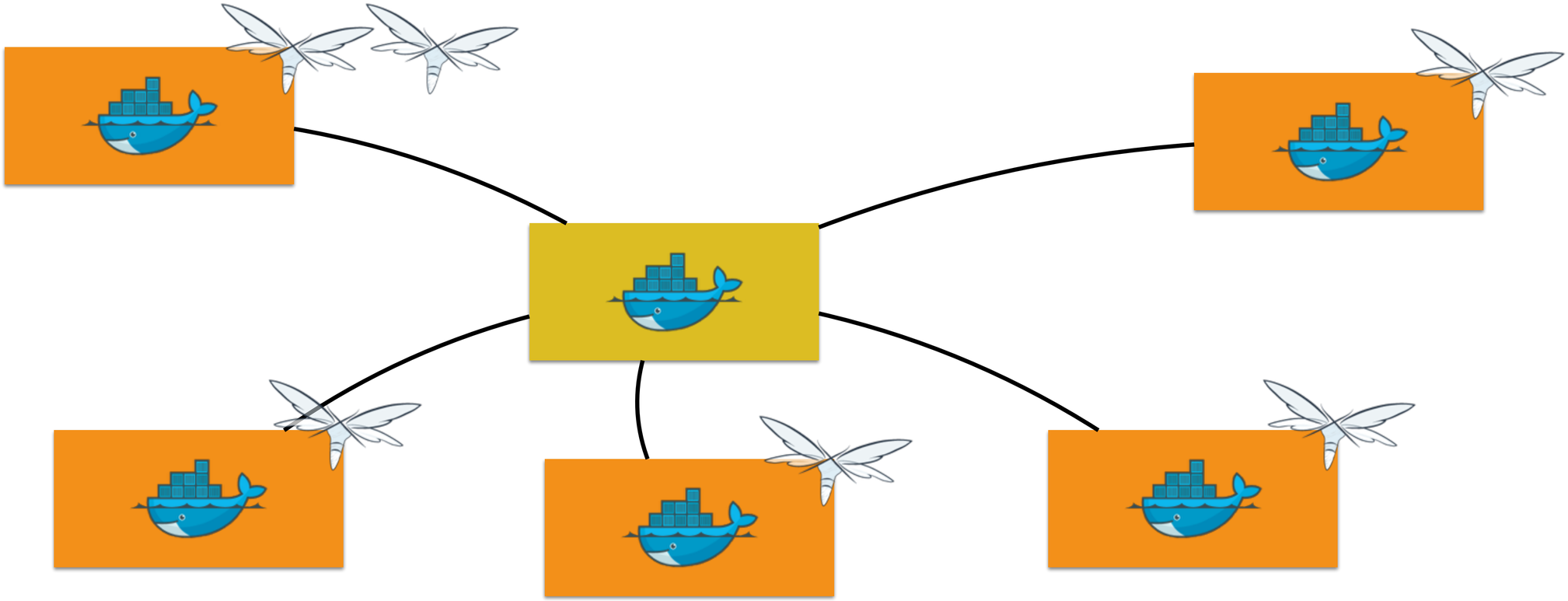
```
docker stack deploy --compose-file=docker-compose.yml webapp
```

Swarm Mode: Routing Mesh



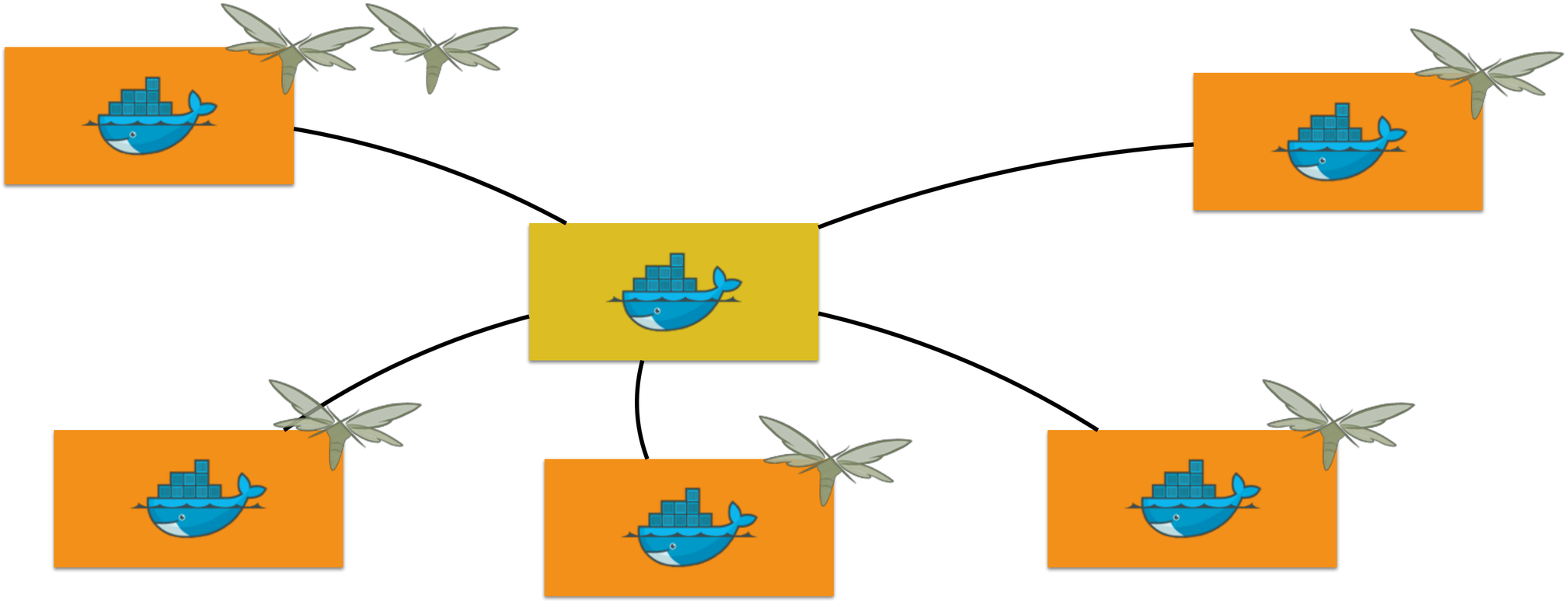

```
docker service create --replicas 3 --name web -p 8080:8080 jboss/
wildfly
```

Swarm Mode: Scale



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

Swarm Mode: Rolling Updates

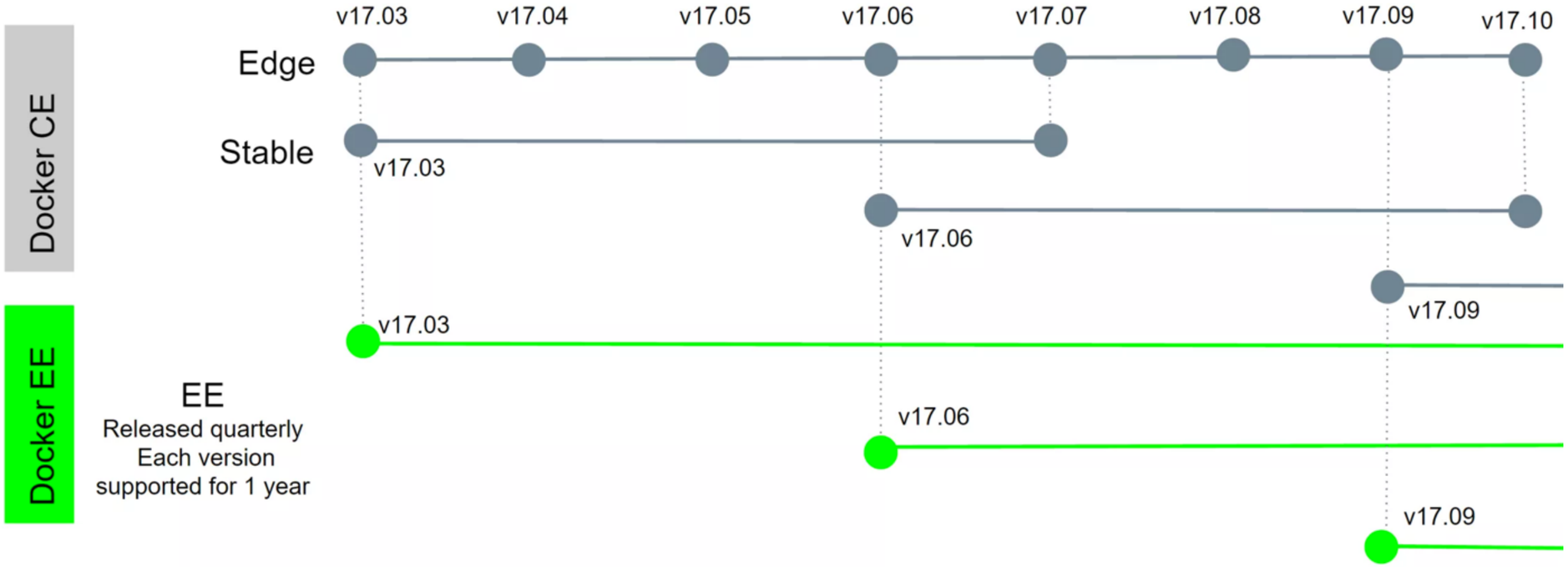


```
docker service update web --image wildfly:2 --update-parallelism
2 --update-delay 10s
```

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?

Fabine, thi si your section. Feel free to rewrite any way you like.

https://developers.redhat.com/blog/2017/03/14/java-inside-docker/ has good resources.

- By default, container will use as much memory an
- Can be restricted using **`-memory`**, **`-memory-res`** **`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

Runtime constraints on containers

-m or -memory	Memory limit on container	-m50M	Container terminated if exceeds the memory

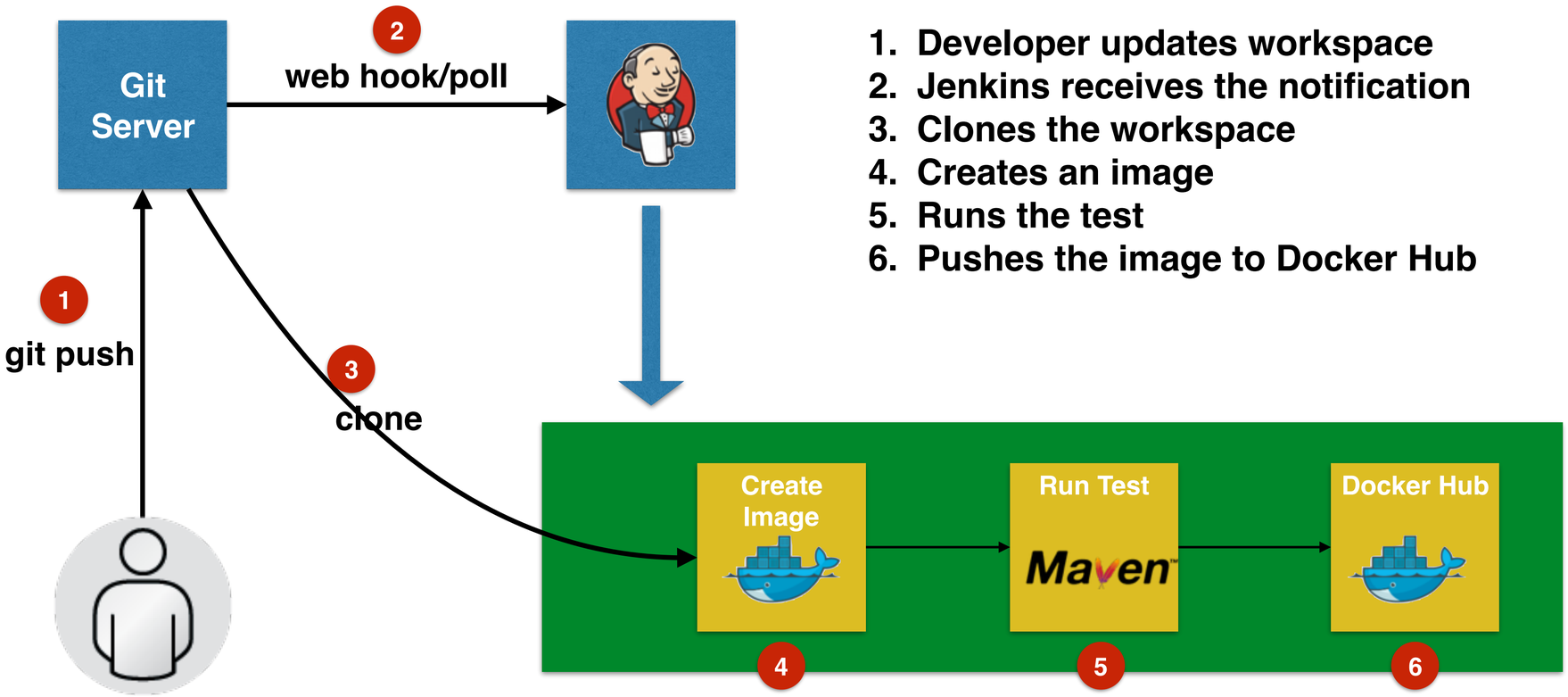
Debugging Java Applications

Monitor Java Applications

Monitoring Docker Containers

- `docker stats` command
  - LogEntries
- Service logs: `docker service logs <service>`
- Prometheus endpoint - New in 1.13
- Docker Remote API: `/container/{container-name|cid}/stats`
- Docker Universal Control Plane
- cAdvisor
  - Prometheus
  - InfluxDB

CI/CD with Docker + Jenkins



1. Developer updates workspace
2. Jenkins receives the notification
3. Clones the workspace
4. Creates an image
5. Runs the test
6. Pushes the image to Docker Hub

http://blog.couchbase.com/2016/september/deployment-pipeline-docker-jenkins-java-couchbase

References

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