Docker Deploy an Application

A FRAMEWORK FOR DATA INTENSIVE COMPUTING

Agenda

Intro / Prep Environments

Day 1: Docker Deep Dive

Deploy Apache Tomcat in Docker

- Now let's run a JVM based application like Apache Tomcat:

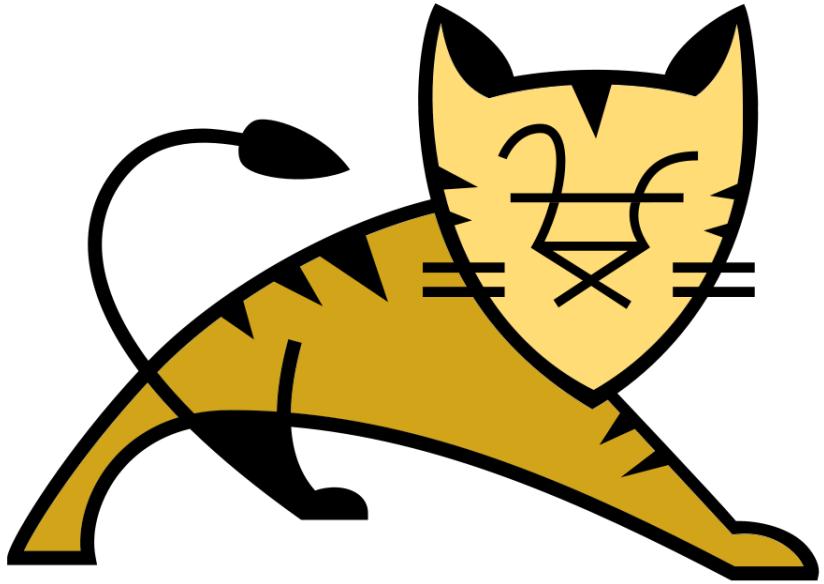
```
docker run --rm -p 8888:8080 tomcat:8.0
```

- Output:

```
16-Oct-2016 18:30:51.541 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment
of web application directory /usr/local/tomcat/webapps/manager has finished in 28 ms

16-Oct-2016 18:30:51.542 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying
web application directory /usr/local/tomcat/webapps/examples

16-Oct-2016 18:30:52.108 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment
of web application directory /usr/local/tomcat/webapps/examples has finished in 566 ms

16-Oct-2016 18:30:52.117 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deploying
web application directory /usr/local/tomcat/webapps/ROOT

16-Oct-2016 18:30:52.161 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory Deployment
of web application directory /usr/local/tomcat/webapps/ROOT has finished in 45 ms

16-Oct-2016 18:30:52.176 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-nio-8080"]

16-Oct-2016 18:30:52.206 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-nio-8009"]

16-Oct-2016 18:30:52.208 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 1589 ms
```

Understanding Tomcat

- Apache Tomcat (or simply Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF).
- Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Oracle.
- It provides a "pure Java" HTTP web server environment for Java code to run in.



Understanding Tomcat

- In the simplest config Tomcat runs in a single operating system process.
- The process runs a Java virtual machine (JVM).
- Every single HTTP request from a browser to Tomcat is processed in the Tomcat process in a separate thread.

Deploy Apache Tomcat

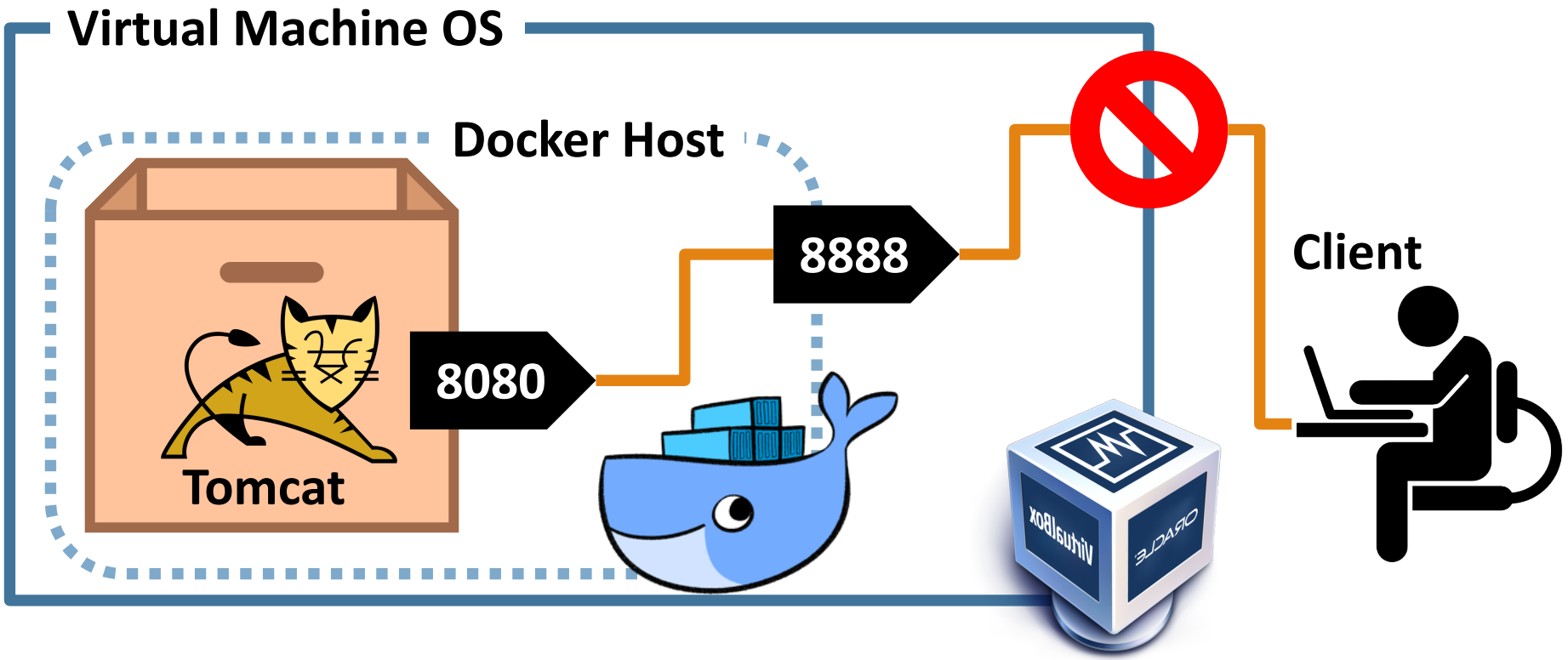
- Let's explore that command for a quick sec:

```
docker run --rm -p 8888:8080 tomcat:8.0
```

- The option (`--rm`) tells Docker we want to remove the container when it's done running.
- We are exposing ports `8888:8080` with `(-p)`, which tells Docker we want to map the container's port 8080 to the host port of 8888
- If we were to connect to `http://localhost:8888` we should be able to reach our tomcat server.

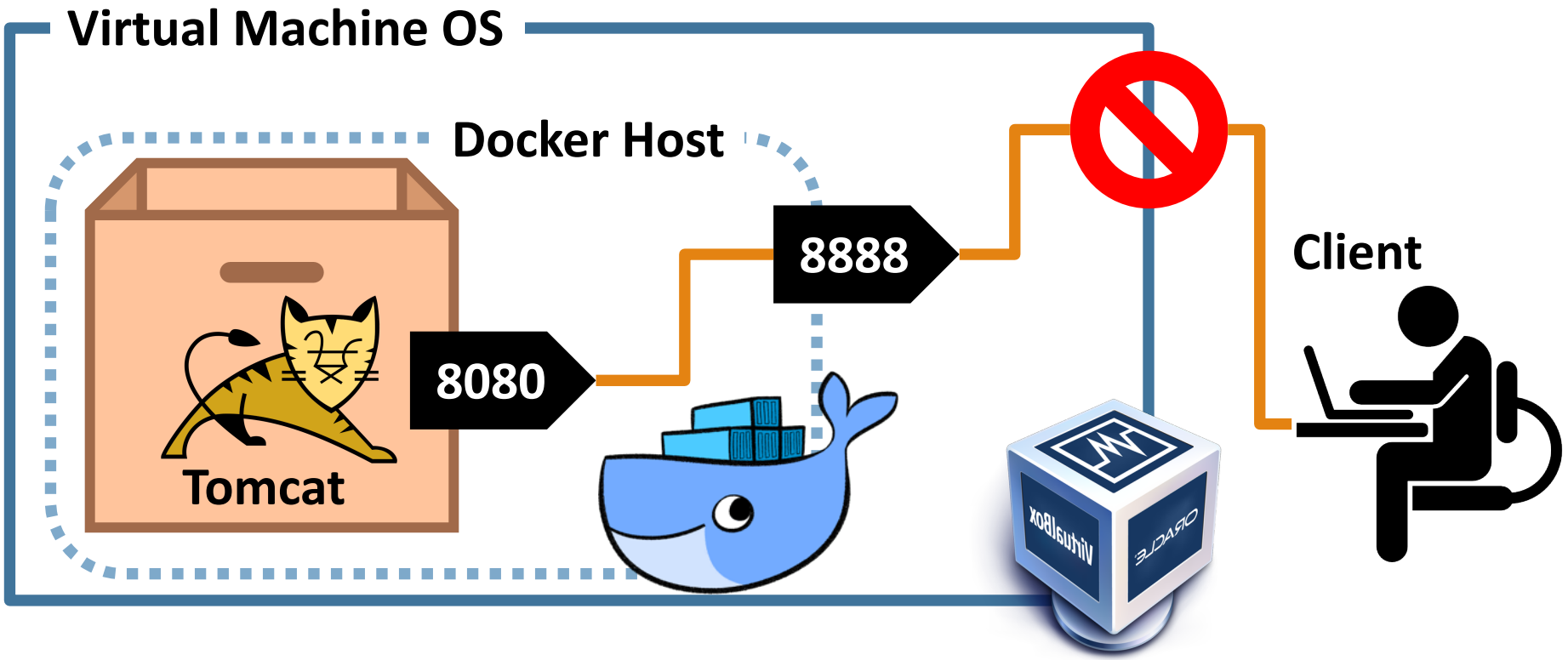
Deploy Apache Tomcat VM

- If we are operating Docker from inside a VM image, there is some additional work to be done. Here's a look at the problem:



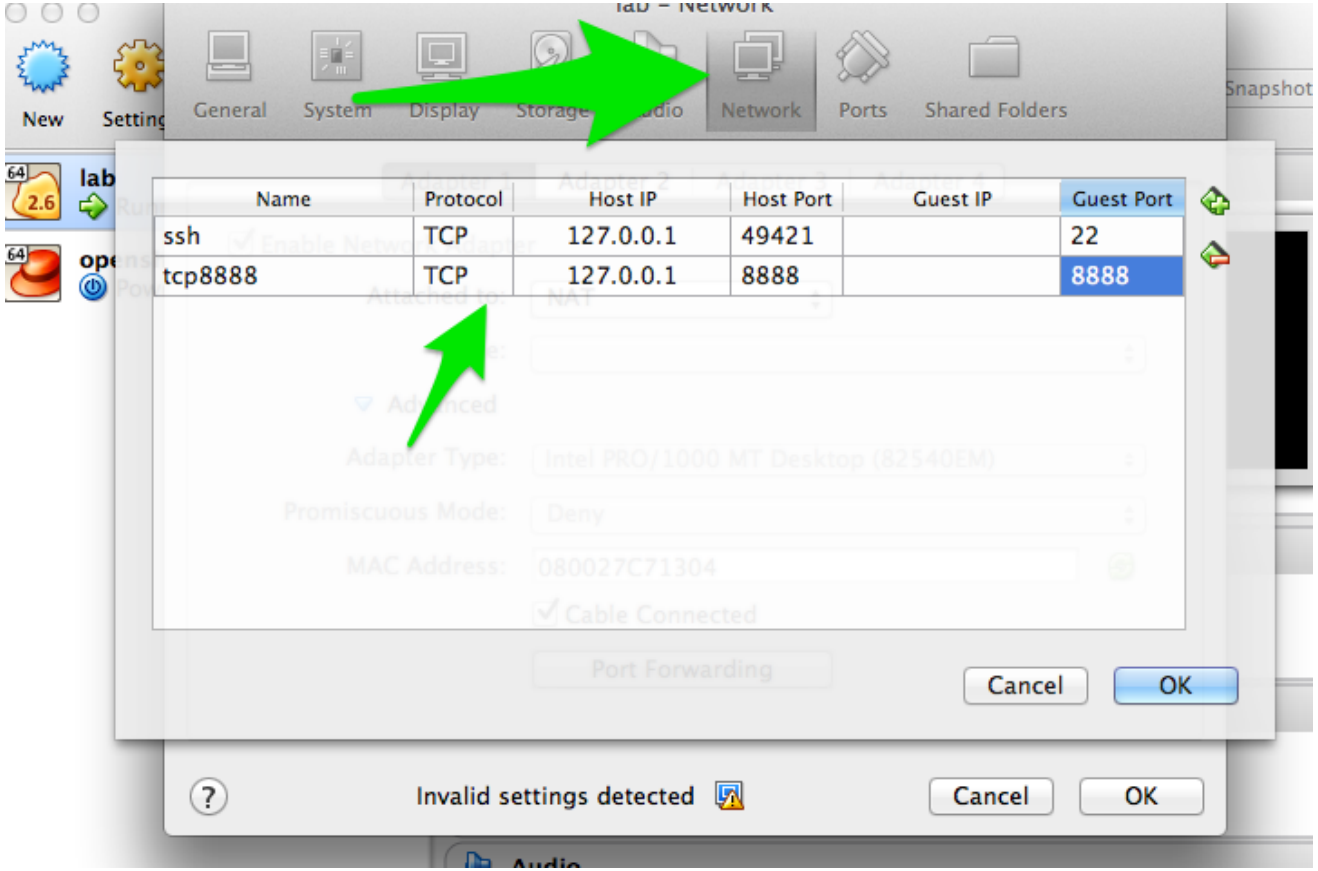
Deploy Apache Tomcat VM

- Our Docker Host has been mapped properly, but we cannot reach it from our host (Windows/MacOSX) because the VM does not expose those ports.



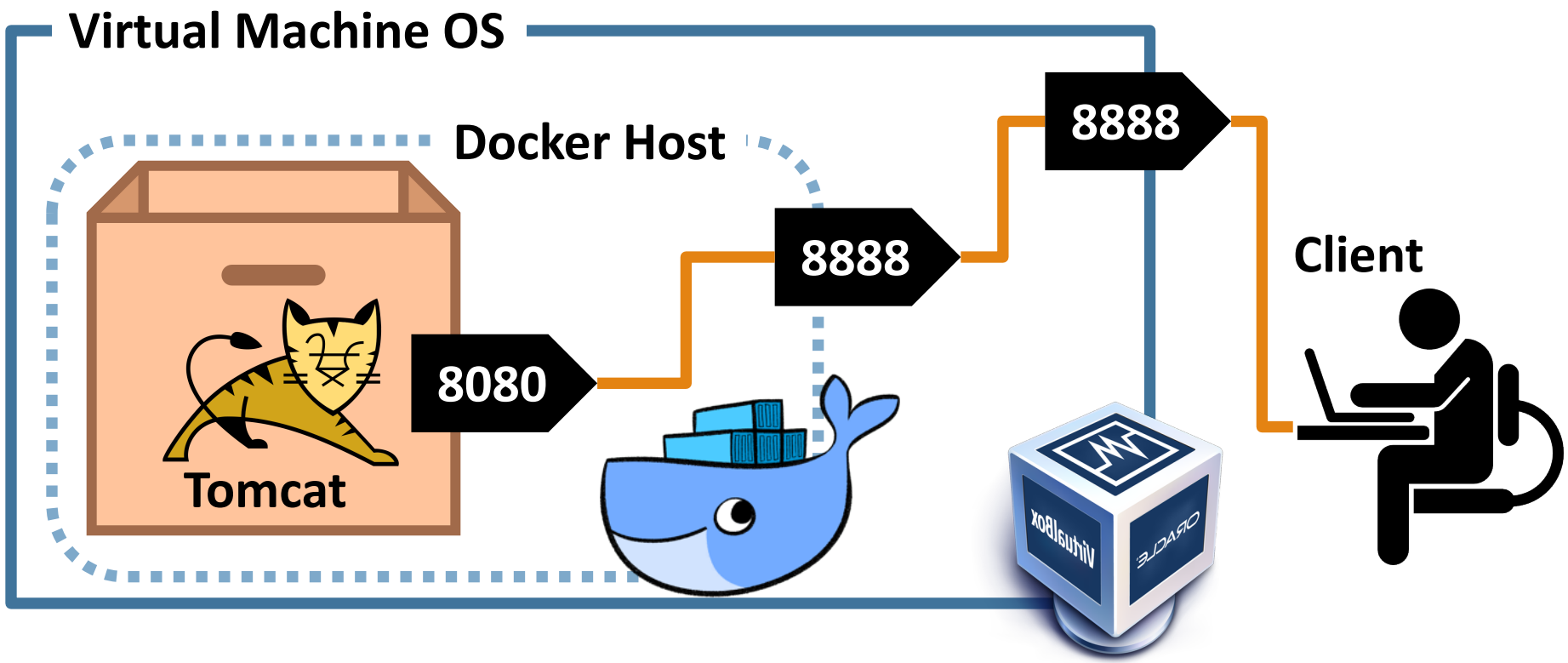
Map Ports for Tomcat VM

- Enable port forwarding between the VM Host (windows/Mac) and the VM Guest (Docker host):



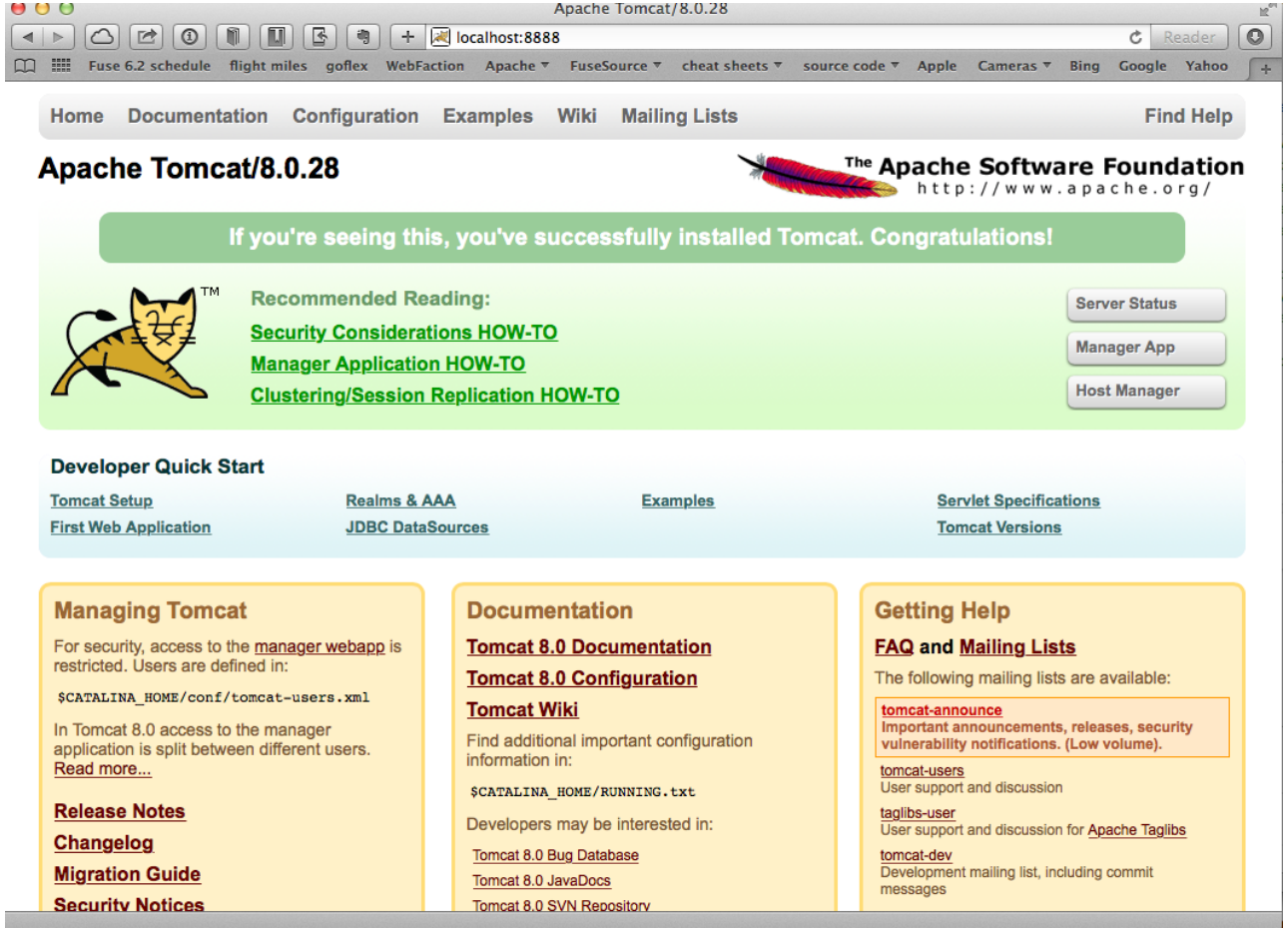
Map Ports for Tomcat

- The outcome is represented in the diagram below:



Deploy Apache Tomcat

- Now, when we navigate to `http://localhost:8888` in our browser, we see:



Deploy Apache Tomcat

- We now have a running container that has tomcat in it.
- Let's explore the tomcat container really quick.
- Let's fire up a new shell window and observe Tomcat's processes, like this:

```
docker ps
```

- Output:

```
CONTAINER ID    IMAGE          COMMAND

xxxxxxxxxxxx       tomcat:8.0   ...
```

Explore Inside Tomcat Container

- Let’s log into the container to explore:

```
docker exec -it <container_id> bash
```

- We should now be at the bash prompt for the tomcat container.
- We can navigate around like we would normally.

Deploy Apache Tomcat

- We can exit out of the Tomcat container, like this:

```
# exit
```

- If we then switch back to the other window where Tomcat is running, we can exit by issuing the `CTR+C` command.
- We then have no containers running and we can verify that, with:

```
docker ps -a
```

- NOTE: Because of the (`--rm`) option, the container removed itself when finished.

Other Useful Flags

- Here are some other useful Docker `run` flags:

`--name`	Gives containers a unique name
`-d`	Runs containers in daemon mode (in the background)
`--dns`	Gives containers a different name server from the host
`-it`	Interactive with tty (caution using this with -d)
`-e`	Passes in environment variables to the container
`--expose`	Exposes ports from the Docker container
`-P`	Exposes all published ports on containers
`-p`	Map a specific port from the container to the host (host:container)

Deploy Apache Tomcat as a Daemon

- Let's use some of those previous run command-line flags and start tomcat in the background:

```
docker run -d --name="tomcat8" -p 8888:8080 tomcat:8.0
```

  - NOTE: We also gave this container a name, so we can refer to it by name instead of container id

Print Apache Tomcat Logs

- With Tomcat running as a daemon (-d) in the background, we can print it's logs, like this:

```
docker logs tomcat8
```

- Output:

```
16-Oct-2016 19:19:20.441 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory
Deployment of web application directory /usr/local/tomcat/webapps/examples has finished in 526 ms

16-Oct-2016 19:19:20.447 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory
Deploying web application directory /usr/local/tomcat/webapps/ROOT

16-Oct-2016 19:19:20.507 INFO [localhost-startStop-1] org.apache.catalina.startup.HostConfig.deployDirectory
Deployment of web application directory /usr/local/tomcat/webapps/ROOT has finished in 60 ms

16-Oct-2016 19:19:20.515 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["http-
nio-8080"]

16-Oct-2016 19:19:20.527 INFO [main] org.apache.coyote.AbstractProtocol.start Starting ProtocolHandler ["ajp-nio-
8009"]

16-Oct-2016 19:19:20.547 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in 1497 ms
```

Print Apache Tomcat Processes

- Let's use a couple of interesting Docker commands with our tomcat8 container:

```
docker top tomcat8
```

- [Output] Instead of the normal Linux top container, it just displays the processes running in the container:

```
PID                 USER                COMMAND

5301                root                /usr/bin/java -
Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties -
Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager -
Djava.endorsed.dirs=/usr/local/tomcat/endorsed -classpath
/usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar -
Dcatalina.base=/usr/local/tomcat -Dcatalina.home=/usr/local/tomcat -
Djava.io.tmpdir=/usr/local/tomcat/temp org.apache.catalina.startup.Bootstrap
start
```

Inspect Apache Tomcat Daemon

- Let's say we want to "inspect" tomcat8, we can do that like this:

```
docker inspect tomcat8
```

- We can also use a --format template to pick out specific info from that output :

```
docker inspect --format='{{.Config.Env}}' tomcat8
```

- Output:

```
root@server(~) $ docker inspect --format='{{.Config.Env}}' tomcat8

[PATH=/usr/local/tomcat/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
LANG=C.UTF-8 JAVA_VERSION=7u79 JAVA_DEBIAN_VERSION=7u79-2.5.6-1~deb8u1
CATALINA_HOME=/usr/local/tomcat TOMCAT_MAJOR=8 TOMCAT_VERSION=8.0.28
TOMCAT_TGZ_URL=https://www.apache.org/dist/tomcat/tomcat-8/v8.0.28/bin/apache-tomcat-
8.0.28.tar.gz]
```

Stop and Remove Container

- When we finished, we stop the container, like this:

```
docker stop tomcat
```

- If we ran `docker ps` we wouldn't see the container running any more.
- However, `docker ps -a` will show all containers, even those stopped.
- We can remove a container from that (`-a`) list, with:

```
docker rm tomcat8
```

Lab

End of Chapter