Deploying Java EE Applications

A FRAMEWORK FOR DATA INTENSIVE COMPUTING

Agenda

Intro / Prep Environments

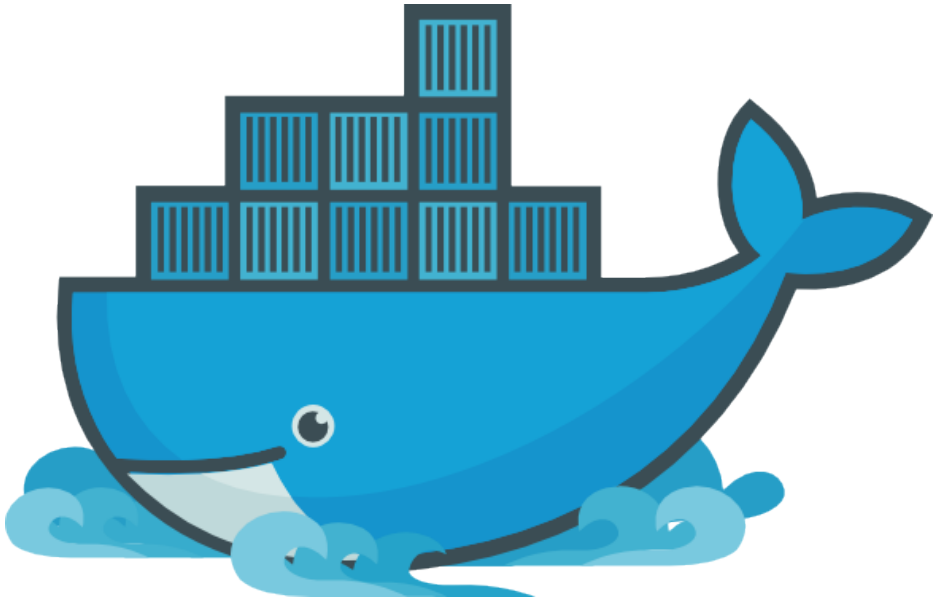
Day 1: Docker Deep Dive

Understanding Java EE App Server

- Wikipedia says: “An **application server** is a software framework that provides both facilities to create web applications and a server environment to run them.”
- Java Platform, Enterprise Edition or Java EE (was J2EE) defines the core set of API and features of Java Application Servers.
- Java application servers behave like an extended virtual machine for running applications.
- It transparently handles connections to the database on one side, and, often, connections to the Web client on the other.
- Open source Java application servers that support Java EE include: JOnAS from Object Web, **WildFly** (formerly JBoss AS) from JBoss, TomEE from Apache, and more.

Understanding Dockerizing

- Dockerizing an application is the process of converting an application to run within a Docker container.
- While Dockerizing most applications is straight-forward, there are a few problems that need to be worked around each time.



Stay Mindful of the Cons

- Two common problems that occur during Dockerization are:
  - Making an application use environment variables when it relies on configuration files.
  - Sending application logs to STDOUT/STDERR when it defaults to files in the container's file system.

- Remember both of these as we make our way through this course!

Deployment Preparation

- First we map the ports `8080` and `9990` from the VM to your host following the steps previously discussed in the previous chapter, if needed.
- We will follow the similar steps we used to deploy Tomcat and will re-use the port mappings we had earlier.
- Verify your Tomcat server is no longer running on `8888`.
- Any containers are running, especially those that share same ports need to be stopped prior to starting more.

Deploying a Java EE App Server

- First, we pull our image from Docker Hub:

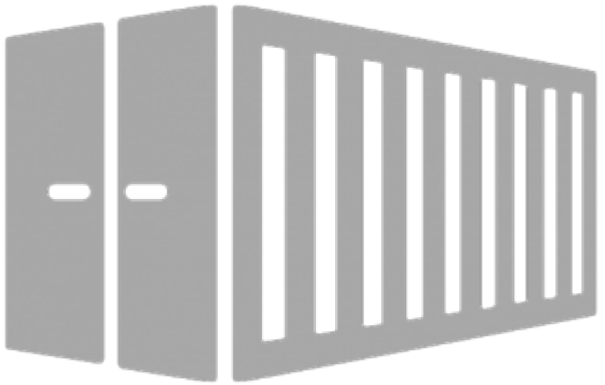
```
docker pull arungupta/wildfly-management
```

- Now, we have our image, that's get it started and assign ports and a name:

```
docker run -d -p 9990:9990 -p 8080:8080 --name wildfly
arungupta/wildfly-management
```

- To confirm our container is running, a name has been assigned, and the appropriate ports are functional:

```
docker ps
```

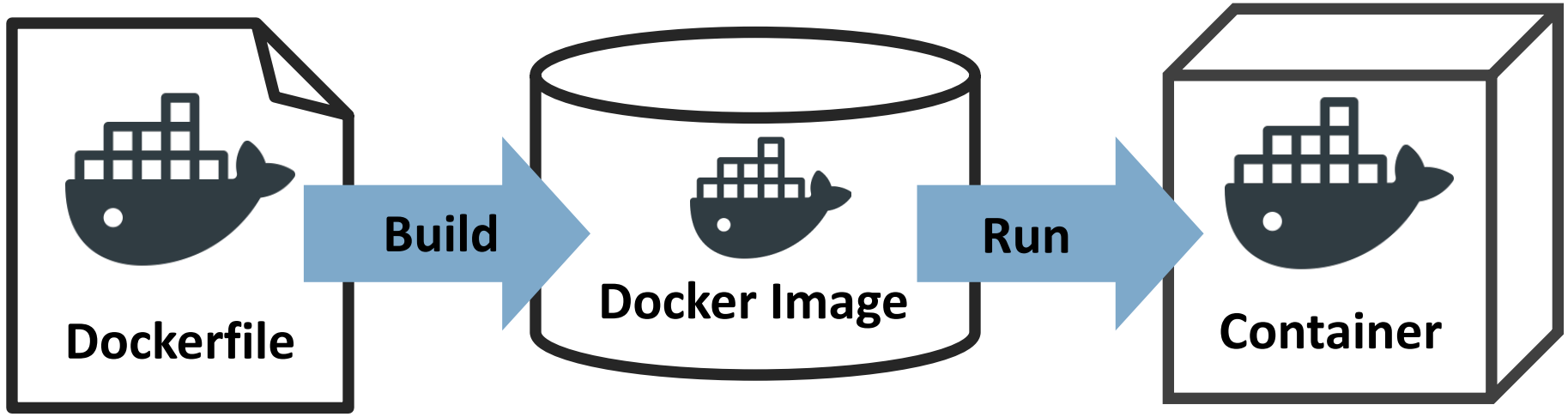


Deploying a Java EE App Server

- Navigate to `http://localhost:8080` to see the main page of the server you've deployed.
- Click "Administration Console" and login to the management console, with:
  - User: `admin`
  - Password: `docker#admin`
- We can easily create a deployment, define datastorage and more.
- But what's the downsides, and is this The Docker Way?

The Docker Way

- Now, you can deploy applications using a platform like we just saw, but that's not the Docker way.
- The Docker way is to package up your app as a new layer on top of the app server Docker image.
- Have we talked about a way to package all our components together?
  - We have...



WildFly

- Now, with the WildFly server you can deploy your application in multiple ways:
  - You can use CLI
  - You can use the web console
  - You can use the management API directly
  - You can use the deployment scanner

Dockerfile WildFly Deployment

- The most popular way of deploying an application is using the deployment scanner.
- In WildFly this method is enabled by default and the only thing you need to do is to place your application inside of the deployments/ directory.
- Pick `/opt/jboss/wildfly/standalone/deployments/` or `/opt/jboss/wildfly/domain/deployments/` depending on which mode you choose.
- Remember, standalone is default in the `jboss/wildfly` image -- see above.

Dockerfile Wildfly Deployment

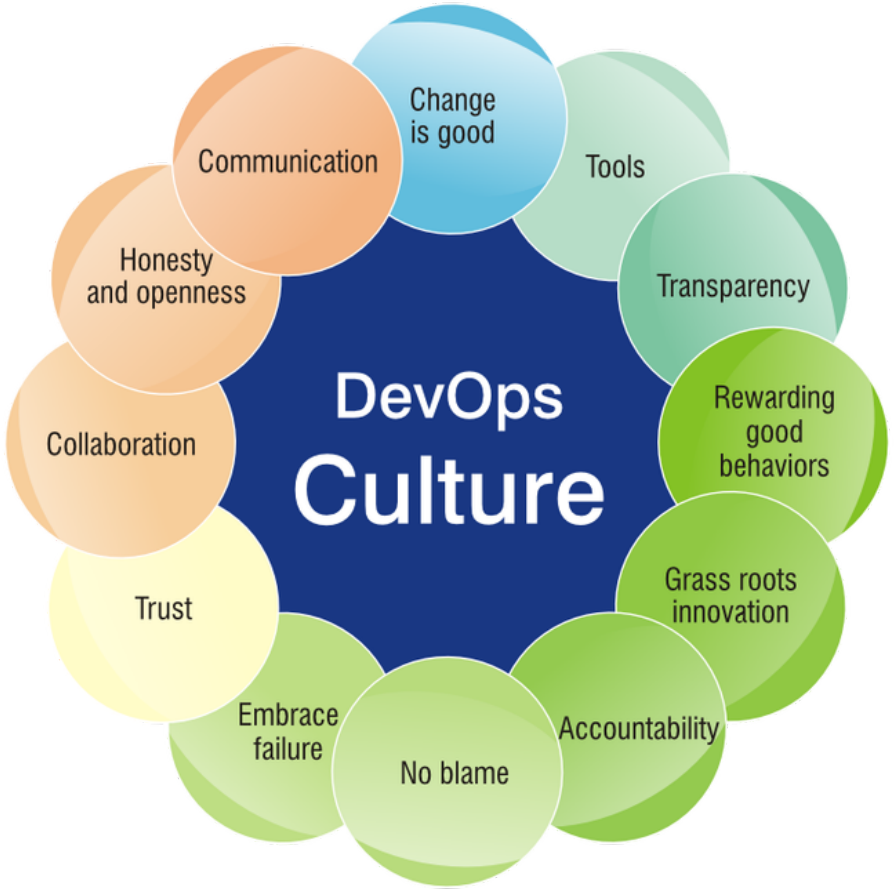
- The simplest and cleanest way to deploy an application to WildFly, is to use the deployment scanner method.
- To do this you just need to extend the `jboss/wildfly` image by creating a new one.
- Place your application inside the `deployments/` directory with the `ADD` command.
  - NOTE: You can also do the changes to the configuration (if any) as additional steps (`RUN` command).

Dockerfile Wildfly Deployment

- A simple example of how this might look is:
  - Create Dockerfile with following content:
    - `FROM jboss/wildfly`
    - `ADD your-awesome-app.war /opt/jboss/wildfly/standalone/deployments/`
  - Place your `your-awesome-app.war` file in the same directory as your Dockerfile.
  - Run the build with `docker build --tag=wildfly-app .`
  - Run the container with `docker run -it wildfly-app`.
  - Application will be deployed on the container boot.

Dockerizing Benefits

- This way of **deployment** is great because of a few things:
  - It utilizes Docker as the build tool providing stable builds
  - Rebuilding image this way is very fast
  - You only need to do changes to the base WildFly image that are required to run your application
- So much of is DevOps, and DevOps is very **philosophical** in nature.



Ticket Monster Running on WildFly

- Let's look at an example that puts it all together:

```
# Use latest jboss/wildfly
FROM jboss/wildfly:latest

#Create admin user
RUN /opt/jboss/wildfly/bin/add-user.sh -u admin -p docker#admin --silent

# Add customization folder
COPY customization /opt/jboss/wildfly/customization/

USER root

# Run customization scripts as root
RUN chmod +x /opt/jboss/wildfly/customization/execute.sh
RUN /opt/jboss/wildfly/customization/execute.sh standalone standalone-ha.xml
...
```

Ticket Monster Running on WildFly

```
...
ADD ticket-monster.war /opt/jboss/wildfly/standalone/deployments/
# Fix for Error:
# Could not rename /opt/jboss/wildfly/standalone/configuration/standalone_xml_history/current
RUN rm -rf /opt/jboss/wildfly/standalone/configuration/standalone_xml_history
RUN chown -R jboss:jboss /opt/jboss/wildfly/
USER jboss

# Expose the ports we're interested in
EXPOSE 8080 9990

# Set the default command to run on boot
# This will boot WildFly in the standalone mode and bind to external interface and enable HA
CMD /opt/jboss/wildfly/bin/standalone.sh -b `hostname -i` -bmanagement `hostname -i`
       -c standalone-ha.xml
```

Deploying Ticket Monster

- Deploy the application. First stop and remove the previous wildfly deployment:

```
docker stop wildfly

docker rm wildfly
```

- Now run the new app:

```
docker run -d -p 9990:9990 -p 8080:8080 --name wildfly
arungupta/javaee7-hol
```

- Now you could navigate to `http://localhost:8080/movieplex7` to see the Java EE application

Extending an Image

- To create a management user to access the administration console create a Dockerfile with the following content:

```
FROM jboss/wildfly

RUN /opt/jboss/wildfly/bin/add-user.sh  admin Admin#70365 --
silent

CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0",
"-bmanagement", "0.0.0.0"]
```

- Then you can build the image:

```
docker build --tag=jboss/wildfly-admin
```

- Run it:

```
docker run -it -p 9990:9990 jboss/wildfly-admin
```

Lab

End of Chapter