Week 7 – Getting Started with Docker Part 1

Objective:

- Learn how to use Docker for development purposes
- We will set up Jenkins (CI/CD) in a Docker Container to build, test, deploy our app
- Use Maven build system
- We will build a Java application (revision)

Before you begin. Download Java, Maven, Docker, Git. When installing Maven ensure the bin folder is part of your PATH.

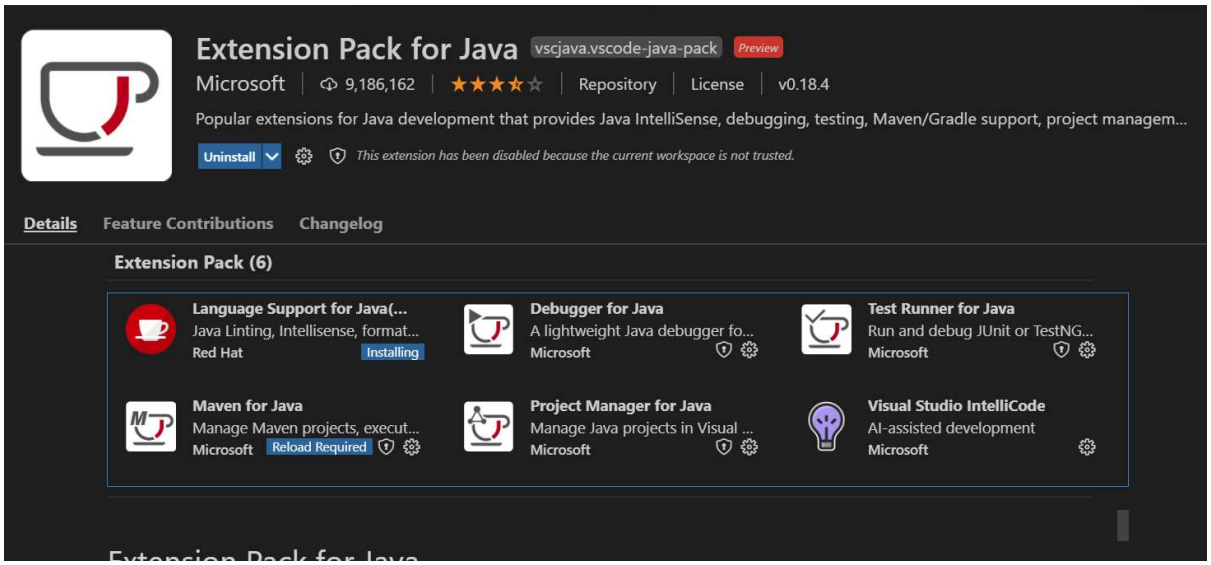
Once you have installed the above, launch a terminal and create a directory for the workshop.

Type mvn --version and you should get an output as per below, we are ready!!!

```
C:\Users\Bone\PRB_DockerExample>mvn --version
Apache Maven 3.8.2 (ea98e05a04480131370aa0c110b8c54cf726c06f)
Maven home: C:\Users\Bone\Downloads\apache-maven-3.8.2
Java version: 11.0.11, vendor: AdoptOpenJDK, runtime: C:\Program Files\AdoptOpenJDK\jdk-11.0.11.9-hotspot
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

1. Create the Java Application

We will use MS Visual Code to code our application, as a pre-req we need to update and install the "Extension Pack for Java". This will give us the Maven build system support along with other features for developing Java applications.



Download it from:
https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-java-pack

For background on Maven in VS Code
https://gorkem1.gitbooks.io/visual-studio-code-for-java/content/chapter-1/Maven-Create.html

In your terminal let's create a Maven project

```
mvn archetype:generate -DgroupId=au.scott -DartifactId=PRB-APP -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

This will take some time to build! Note: change the ID's to your particular name etc.

From the terminal start MS Visual Studio Code

code ./<NAME_OF_YOUR_APP> in my case code ./PRB-APP

If asked, trust the authors to allow full functionality. This will take some time initially to load.

Edit the POM file (this is the project config) to confirm the project settings, mine looks like

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.or
g/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
g/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>au.scott</groupId>
  <artifactId>PRB-APP</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>PRB-APP</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

You will need to add a properties block to the end of the project i.e. prior to the </project>

```
<properties>
    <maven.compiler.source>1.6</maven.compiler.source>
    <maven.compiler.target>1.6</maven.compiler.target>
  </properties>
```

In your src folder navigate to the App.java file and edit it.

Let's add a method that we can test with JUnit.

```
public static boolean isDivisibleByFive(int number){
        return number % 5 == 0;
    }
```

Modify your App.java to call this method from the main.

```
package au.scott;

/**
 * PRB Example
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println(isDivisibleByFive(50));

    }

    public static boolean isDivisibleByFive(int number){
        return number % 5 == 0;
    }
}
```

Run the code  and you should see the output 'true'

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Bone\PRB_DockerExample\PRB-APP>  & 'c:\Users\Bone\.vscode\extensions\vscjava.vscode-java-debug-0.35.0
\scripts\launcher.bat' 'C:\Program Files\AdoptOpenJDK\jdk-11.0.11.9-hotspot\bin\java.exe' '-Dfile.encoding=UTF-8'
 '-cp' 'C:\Users\Bone\PRB_DockerExample\PRB-APP\target\classes' 'au.scott.App'
true
```

Let write a test.

In the test folder edit the AppTest.java file

Add this method

```
public void testInputIsDivByFive(){
        assertTrue(App.isDivisibleByFive(14)); // Assertion
}
```

This will clearly fail the assertion.

Let's confirm by executing the test

Within the root of your project file execute

mvn test

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running au.scott.AppTest
Tests run: 2, Failures: 1, Errors: 0, Skipped: 0, Time elapsed: 0.015 sec <<< FAILURE!
testInputIsDivByFive(au.scott.AppTest)  Time elapsed: 0.009 sec  <<< FAILURE!
junit.framework.AssertionFailedError
        at junit.framework.Assert.fail(Assert.java:47)
        at junit.framework.Assert.assertTrue(Assert.java:20)
        at junit.framework.Assert.assertTrue(Assert.java:27)
        at au.scott.AppTest.testInputIsDivByFive(AppTest.java:41)


Results :

Failed tests:   testInputIsDivByFive(au.scott.AppTest)

Tests run: 2, Failures: 1, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  18.651 s
[INFO] Finished at: 2021-09-08T00:02:23+10:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test (default-test) on project PRB
APP: There are test failures.
```

Now change the test input to a multiple of 5 and re-run the test. It should pass.

2. Setup GitHub

Go to the root of your project

```
git init -b main //To initialize the local repository
```

```
git add .
```

Head over to github.com and create a repo, copy out the repo address

In my case https://github.com/scott-d-mann/PRBJavaDockerDemo.git

Now let's sync to the remote repo.

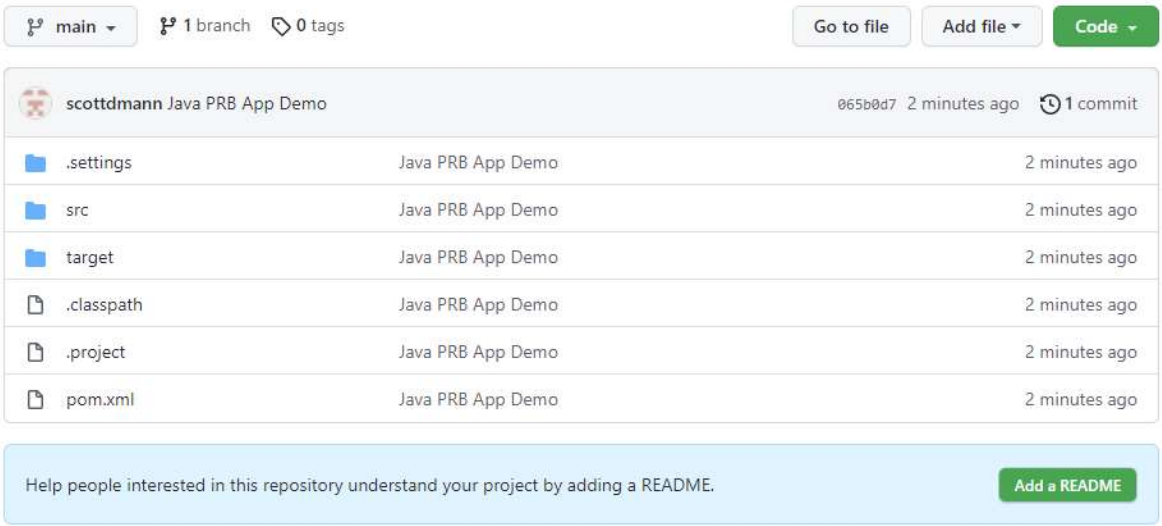
```
git remote add origin https://github.com/scott-d-mann/PRBJavaDockerDemo.git
```

```
git remote -v
```

```
git commit -m "Java PRB App Demo"
```

```
git push origin main
```

You may be asked to authenticate here, refer to the previous workshop if you want to use a PAT else auth via the browser.



3. Docker Setup

Install the VS Code extensions for Docker

https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-docker

First we need to create a network interface to access our container.

docker network create jenkins

Lets use docker to create an image (this may take a while)

```
docker run --name jenkins-docker --rm --detach --privileged --network jenkins --network-alias docker --env DOCKER_TLS_CERTDIR=/certs  --volume jenkins-docker-certs:/certs/client  --volume jenkins-data:/var/jenkins_home docker:dind
```

Let's create a Dockerfile to customise the image, remember this is the instructions for building an image, this is run in a container.

In your project root for example, create a Dockerfile with no extension, use the below instructions.

```
FROM jenkins/jenkins:2.303.1-jdk11
USER root
RUN apt-get update && apt-get install -y apt-transport-https \
    ca-certificates curl gnupg2 \
    software-properties-common
RUN curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
RUN apt-key fingerprint 0EBFCD88
RUN add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/debian \
    $(lsb_release -cs) stable"
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean:1.24.7 docker-workflow:1.26"
```

Save it as "Dockerfile"

```
09/08/2021  09:34 AM    <DIR>          .
09/08/2021  09:34 AM    <DIR>          ..
09/08/2021  09:34 AM               567 Dockerfile
09/08/2021  12:28 AM    <DIR>          PRB-APP
               1 File(s)            567 bytes
               3 Dir(s)   5,388,816,384 bytes free

C:\Users\Bone\PRB_DockerExample>
```

On your terminal let's build the image.

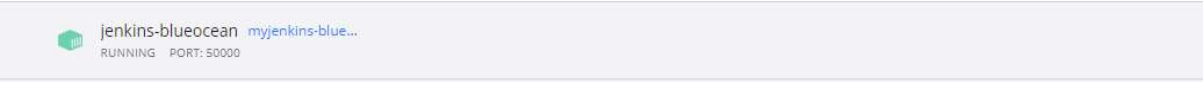
docker build -t myjenkins-blueocean:1.1 . (this may take a while to download components)

```
Command Prompt - docker build -t myjenkins-blueocean:1.1 .

09/08/2021  09:34 AM    <DIR>          .
09/08/2021  09:34 AM    <DIR>          ..
09/08/2021  09:34 AM               567 Dockerfile
09/08/2021  12:28 AM    <DIR>          PRB-APP
               1 File(s)            567 bytes
               3 Dir(s)   5,388,816,384 bytes free

C:\Users\Bone\PRB_DockerExample>docker build -t myjenkins-blueocean:1.1 .
[+] Building 73.6s (3/10)
 => [1/7] FROM docker.io/jenkins/jenkins:2.303.1-jdk11@sha256:a942c30fc3bcf269a1c32ba27eb4a470148eff9aba08691132  69.9s
 => => sha256:4c25b3090c2685271afcffc2a4db73f15ab11a0124bfcde6085c934a4e6f4a51 26.21MB / 54.92MB                 69.9s
 => => sha256:750d566fdd606a5e91b543dace36f953e149cd71f521310dcbb926e880e574b7 26.21MB / 52.74MB                 69.9s
 => => sha256:2625c929bb0e004e7d6748765ef08b95386536a778c51a2cbf8dddf83cd07224 12.58MB / 72.07MB                 69.9s
```

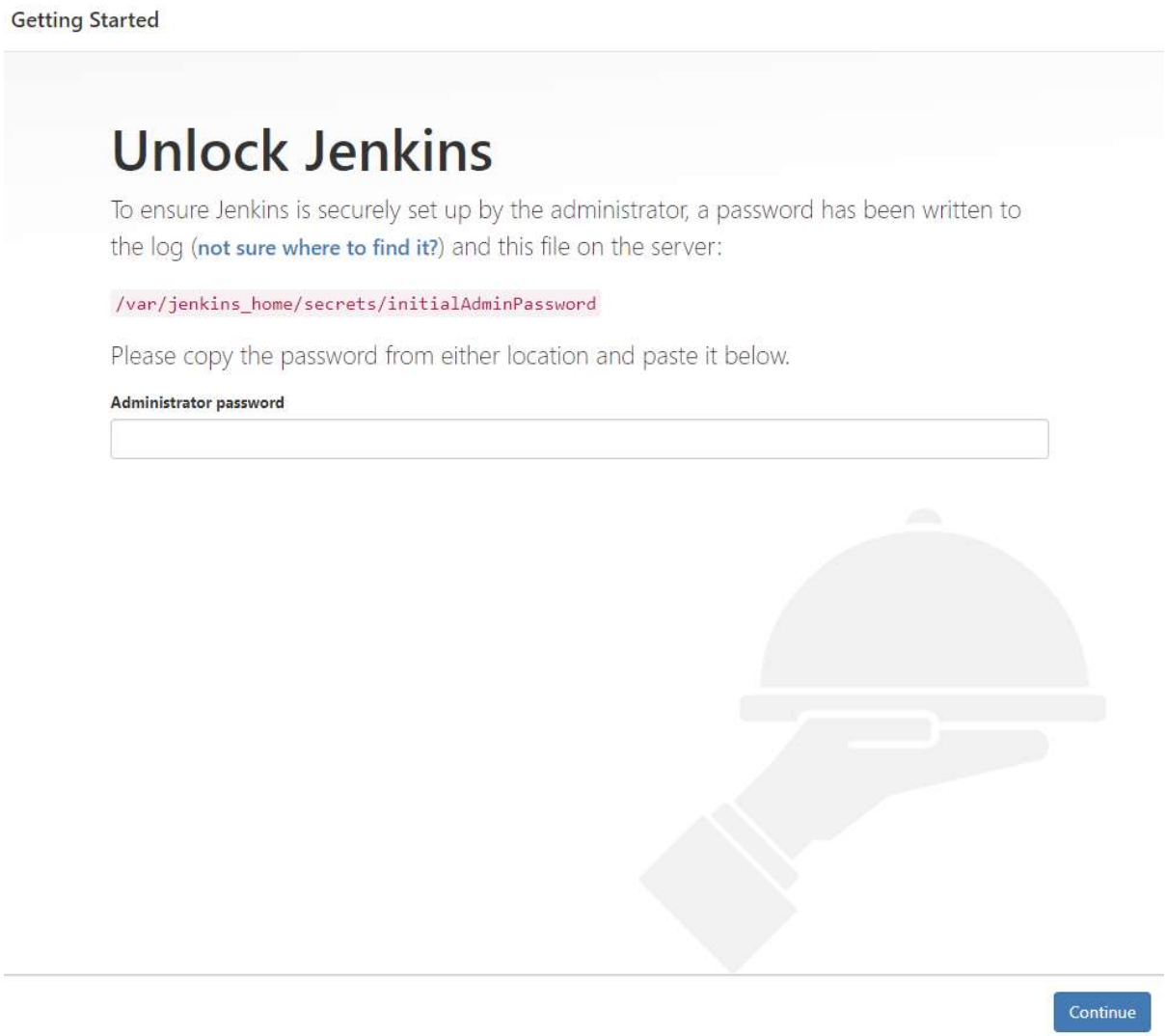
Let's run the image as a container.

docker run --name jenkins-blueocean --rm --detach --network jenkins --env DOCKER_HOST=tcp://docker:2376 --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 --volume jenkins-data:/var/jenkins_home --volume jenkins-docker-certs:/certs/client:ro --publish 8080:8080 --publish 50000:50000 myjenkins-blueocean:1.1

In your Docker Desktop App you will see the container status



In your browser navigate to http://localhost:8080



If you see the Jenkins admin page, you have completed the workshop, Part 2 next week !!

We will get Jenkins to build, test, dockerise and deploy the image to Docker Hub on code commits to the GitHub repository.