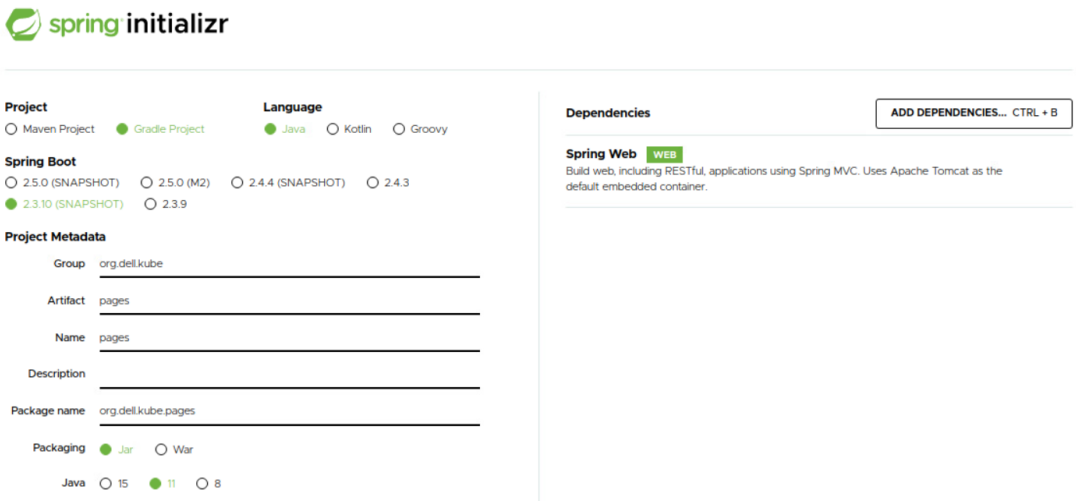
# Building and Dockerizing a Spring Boot Application

## Creating Spring Boot application

1. Create the spring boot application using spring initializer

   Refer the below example snapshot for creating the application along with its dependencies - `Spring Web` . You can choose the appropriate version of java and springboot after checking with the instructor.




2. Click on `Generate` after adding the dependencies and entering other fields to download the codebase.
3. Extract the codebase to `~/workspace` directory.
4. Navigate to `~/workspace/pages` directory in terminal

## Build and Run the application

1. Build the application

```
./gradlew clean build
```

2. Run the application

```
./gradlew bootRun
```

3. Access your application

```
Browse to http://localhost:8080
```

You will witness `White Label Error` . This is because, you do not have any endpoints configured which can serve the request. Let us resolve this by adding a `HomeController` class. Stop the running process by entering `CTRL-C` in the terminal.

4. Create the controller

```
cat > src/main/java/org/dell/kube/pages/HomeController.jav
a << EOF

package org.dell.kube.pages;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapp
ing;
import org.springframework.web.bind.annotation.RestControl
ler;
@RestController
@RequestMapping("/")
public class HomeController {
    @GetMapping
    public String getPage(){
        return "Hello Kubernetes!";
    }
}

EOF
```

5. Build the application

```
./gradlew clean build
```

6. Run the application

```
./gradlew bootRun
```

7. Access your application

```
Browse to http://localhost:8080
```

# Dockerizing

1. Create a new file named `Dockerfile` inside root project folder & add instructions to download the base image. In order to run the Java application using JDK 11, use the image- `adoptopenjdk:11-jre-openj9` . Add instructions to copy the

dependencies & build artifacts(jar/war) from the local directory into the docker image. Finally, Provide a command or an entrypoint to start the application within the docker container

a. Create Dockerfile

```
cd ~/workspace/pages

touch Dockerfile
```

b. Update Dockerfile

```
FROM adoptopenjdk:11-jre-openj9
ARG JAR_FILE=build/libs/page*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","/app.jar"]
```

2. Build the docker image

```
docker build -t pages .
```

Warning: If you get an error containing `Permission Denied while trying to connect to the docker daemon socket` you will need to execute the command `sudo chmod 666 /var/run/docker.sock`

3. Verify the image exists

```
docker images
```

4. Run the image as a container

```
docker run -p 8080:8080 pages
```

In the run command, we have specified that the port 8080 on the container should be mapped to the port 8080 on the Host OS.

Once the application is started, you should be able to access it at http://localhost:8080

The container runs in the foreground. You can run the container in the background using -d option.

Stop the process by pressing `CTRL + C` Pressing `CTRL + C` sometimes might not stop the process. You will need to manually terminate the container.

a. Use `docker ps` and fetch the container id.

b. `docker kill <container-id>`

# Pushing the docker image to docker hub

1. Login with your Docker ID to push or pull images from Docker Hub.

   If you don't have a Docker ID, head over to docker hub to create one, before proceeding futher.

   ```
   docker login
   ```

2. Tag the image using the notation `docker-username/repository:tag`

   ```
   docker tag pages [docker-username]/pages:1.0
   ```

   Make sure to replace username with your docker id in the above command.

3. Verify the newly created tagged image

   ```
   docker images
   ```

4. Push the image to docker hub

   ```
   docker push [docker-username]/pages:1.0
   ```

5. Pull the image from docker hub and test it on local machine. Stop the process after you test it.

   ```
   docker run -p 8080:8080 [docker-username]/pages:1.0
   ```