How-to: Set up a Microservices Landscape

1. Create a root directory for your project in bash (or file explorer).
   ```
   mkdir  some-project
   ```

2. Update the create-projects.bash script file
   Make sure there is a section for each of the microservices in your microservices landscape. Make sure the name, package-name and name at the end of each section correspond.
   ```
   spring init \
   --boot-version=3.0.2 \
   --build=gradle \
   --type=gradle-project \
   --java-version=17 \
   --packaging=jar \
   --name=clients-service \
   --package-name=com.cardealership.clientservice \
   --groupId=com.cardealership.clientservice \
   --dependencies=web \
   --version=1.0.0-SNAPSHOT \
   clients-service
   ```

   (repeat for all your microservices)

3. In your project root directory, execute the create-projects.bash script.
   ```
   ./create-projects.bash
   ```

4. To see if the script worked, find the files in one of your microservices.

```
ls -la (look for the name of the microservices you
created)
```

5. Build all your microservices in order to generate the Gradle files and directories we will need.
   ```
   cd some-service1; ./gradlew build; cd -
   cd some-servcie2; ./gradlew build; cd -
   ```
   (repeat for all your microservices)

   Note: there is a space between cd and –
   Note: there is no / after the name of the service i.e. the service directory
   Make sure the build succeeds for each of your microservices.

6. Now we are going to create a multi-project setup so that we can build all of the microservices with one command.
   Note: this is only to make it easier for us since we are in school. In real-life, we'd want to keep each microservice's build, release, run pipeline separate.

   a) In your project's top-level directory, create the settings.gradle file for your multi-project setup.
      ```
      cat <<EOF > settings.gradle
      include 'some-service1'
      include 'some-service2'
      EOF
      ```

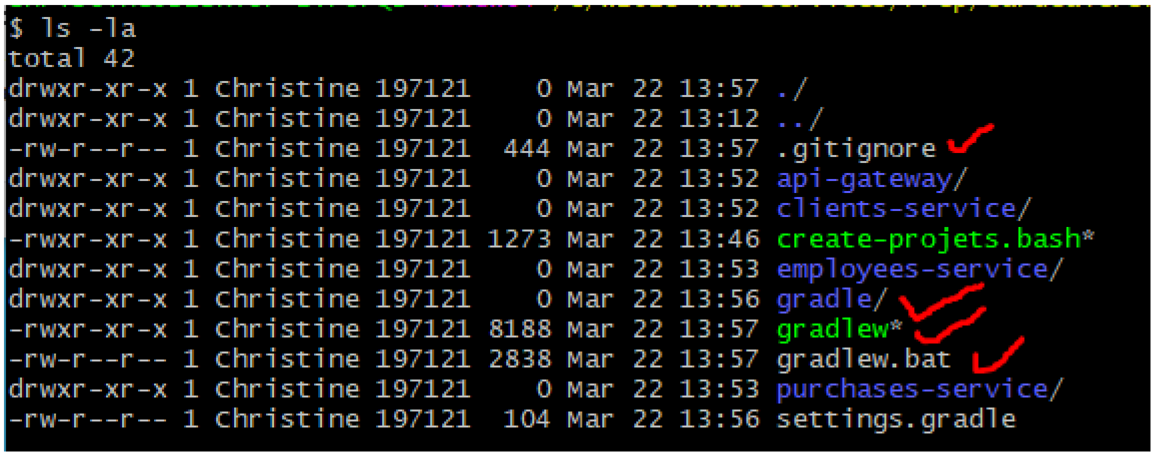
      (include all your microservices)

   b) Copy the Gradle executable files that we generated for one of the microservices to your project's root directory so we can have multi-project builds.
      ```
      cp -r some-service1/gradle .
      cp some-service1/gradlew .
      cp some-service1/gradlew.bat .
      cp some-service1/.gitignore .
      ```

c) Check that all the files have been copied to your project’s top-level directory.

```
ls -la
```

You should see:



```
$ ls -la
total 42
drwxr-xr-x 1 Christine 197121    0 Mar 22 13:57 ./
drwxr-xr-x 1 Christine 197121    0 Mar 22 13:12 ../
-rw-r--r-- 1 Christine 197121  444 Mar 22 13:57 .gitignore
drwxr-xr-x 1 Christine 197121    0 Mar 22 13:52 api-gateway/
drwxr-xr-x 1 Christine 197121    0 Mar 22 13:52 clients-service/
-rwxr-xr-x 1 Christine 197121 1273 Mar 22 13:46 create-projets.bash*
drwxr-xr-x 1 Christine 197121    0 Mar 22 13:53 employees-service/
drwxr-xr-x 1 Christine 197121    0 Mar 22 13:56 gradle/
-rwxr-xr-x 1 Christine 197121 8188 Mar 22 13:57 gradlew*
-rw-r--r-- 1 Christine 197121 2838 Mar 22 13:57 gradlew.bat
drwxr-xr-x 1 Christine 197121    0 Mar 22 13:53 purchases-service/
-rw-r--r-- 1 Christine 197121  104 Mar 22 13:56 settings.gradle
```

d) Delete the Gradle executable files in each microservice since we don’t need them anymore. **Be sure NOT to delete the files we just copied to the project’s root directory!**

```
find some-service1 -depth -name “gradle” -exec rm -rfv “{}”
find some-service1 -depth -name “gradlew*” -exec rm -fv “{}”
```

(repeat for all your microservices)

e) Build all your microservices with one command.

```
./gradlew build
```