COMP30220 Distributed Systems Practical

Lab 1: Maven, Docker & Sockets

This laboratory is basically an introduction to maven & docker that is based on socket programming. The laboratory will not be formally assessed, but you will need to be able to use maven and docker to complete the rest of the laboratories on this module.

Initial Steps:

- Start by setting up a command shell (Windows Subsystem for Linux, Windows PowerShell or Command Prompt is fine, but I recommend a bash-based shell).

- Install support for git, maven and a version of docker desktop (or docker toolkit). Try to check that everything works before you download the sample code.

- Download the sample socket project from the module Gitlab repository: https://gitlab.com/comp30220/sockets

- This repository contains two projects that implement a client and a server using TCP sockets. For the initial tasks, you should attempt to follow the instructions on the project page (the README.md file).

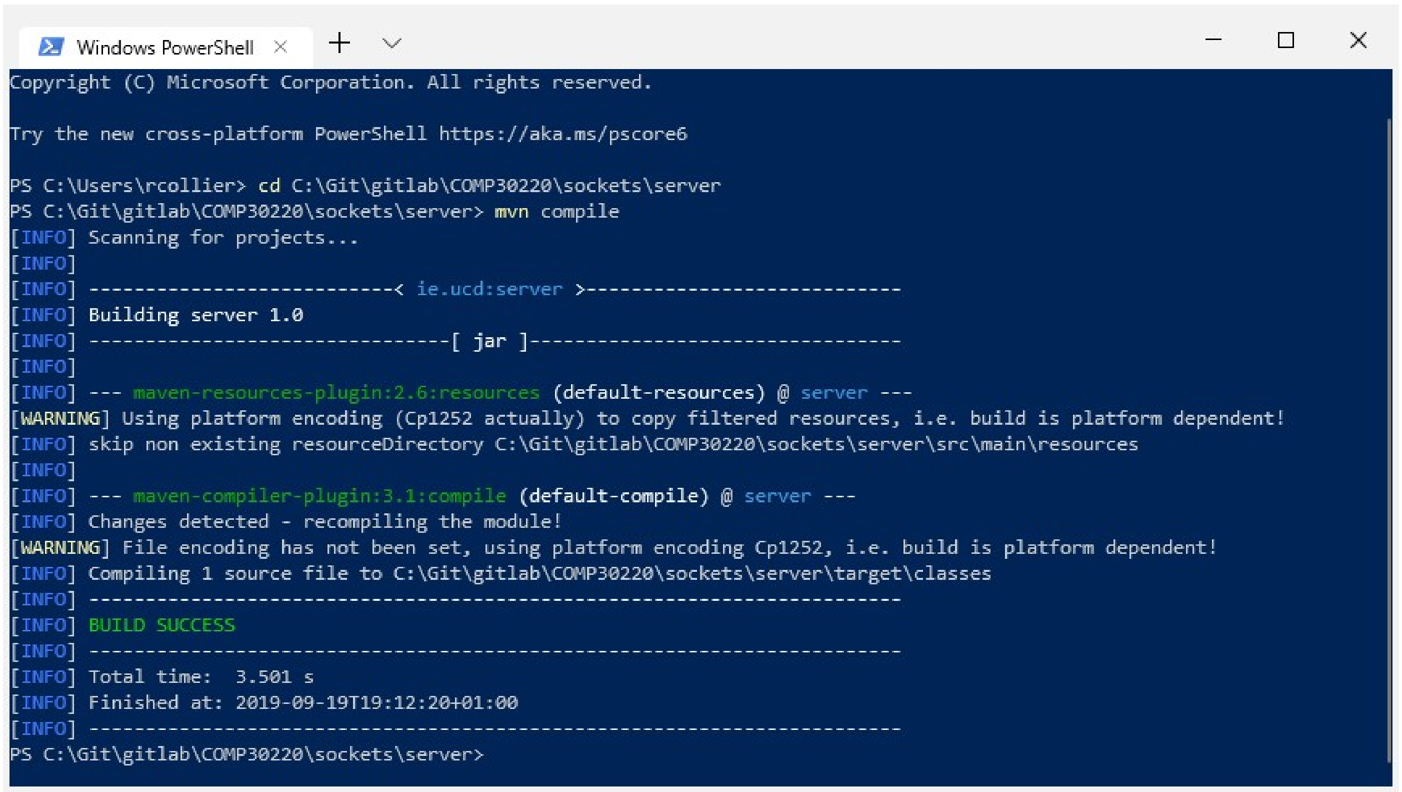
Task 1: Compiling and Running the Server code using Maven

This lab requires that you run 2 programs: a server program and a client program. To do this from a command shell, you should change the current folder to the root folder of the project containing the program (in maven, this folder is always the folder that contains the pom.xml file).

When in this folder, you can compile the code by typing in:

```
$ mvn compile
```

The output should look something like this:

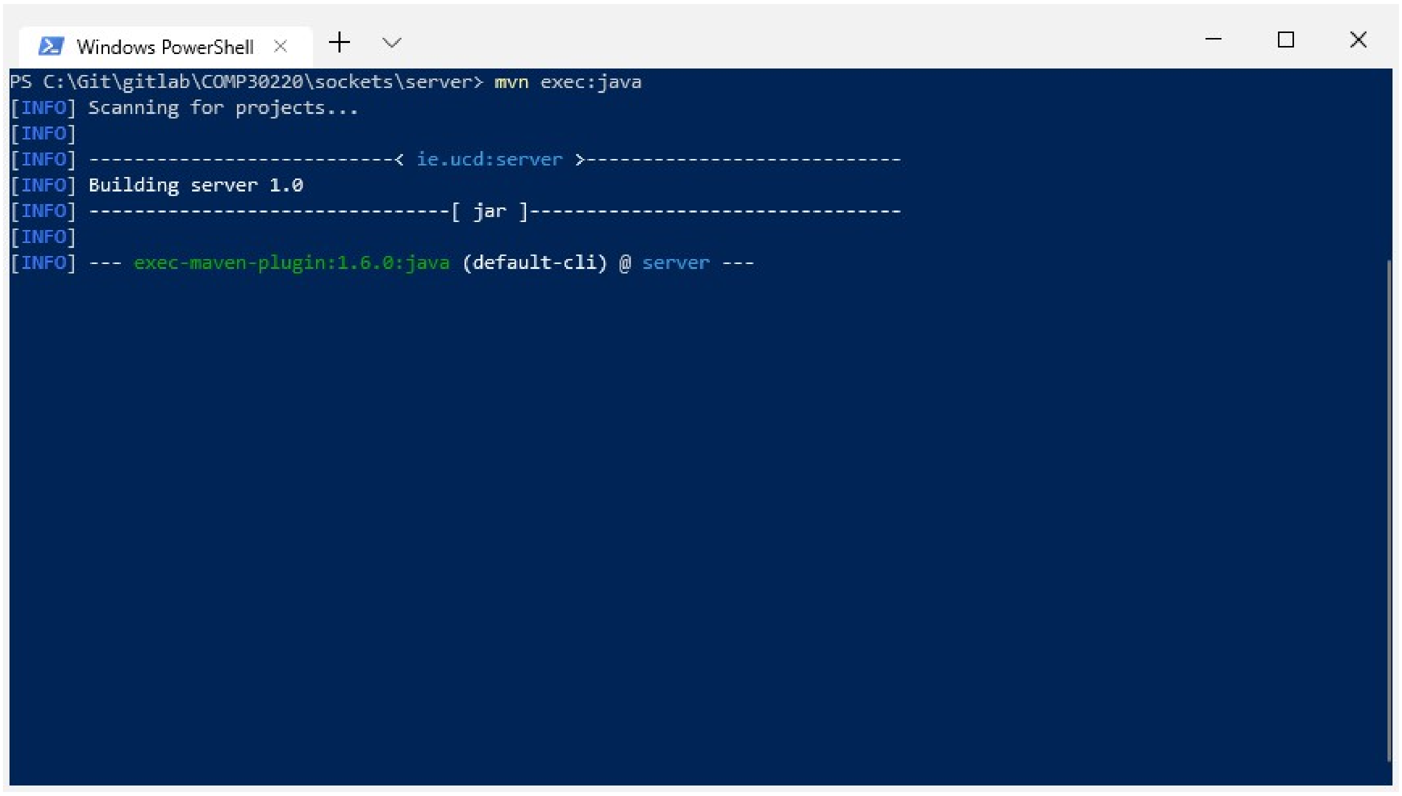


```
Copyright (C) Microsoft Corporation. All rights reserved.

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

PS C:\Users\rcollier> cd C:\Git\gitlab\COMP30220\sockets\server
PS C:\Git\gitlab\COMP30220\sockets\server> mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------------< ie.ucd:server >---------------------------
[INFO] Building server 1.0
[INFO] --------------------------------[ jar ]--------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ server ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Git\gitlab\COMP30220\sockets\server\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ server ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 1 source file to C:\Git\gitlab\COMP30220\sockets\server\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.501 s
[INFO] Finished at: 2019-09-19T19:12:20+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Git\gitlab\COMP30220\sockets\server>
```

To run the code, you need to use the codehaus "exec" plugin (you can find this in the pom.xml files). You simply type:

$ mvn exec:java

Running it should result in something like this:

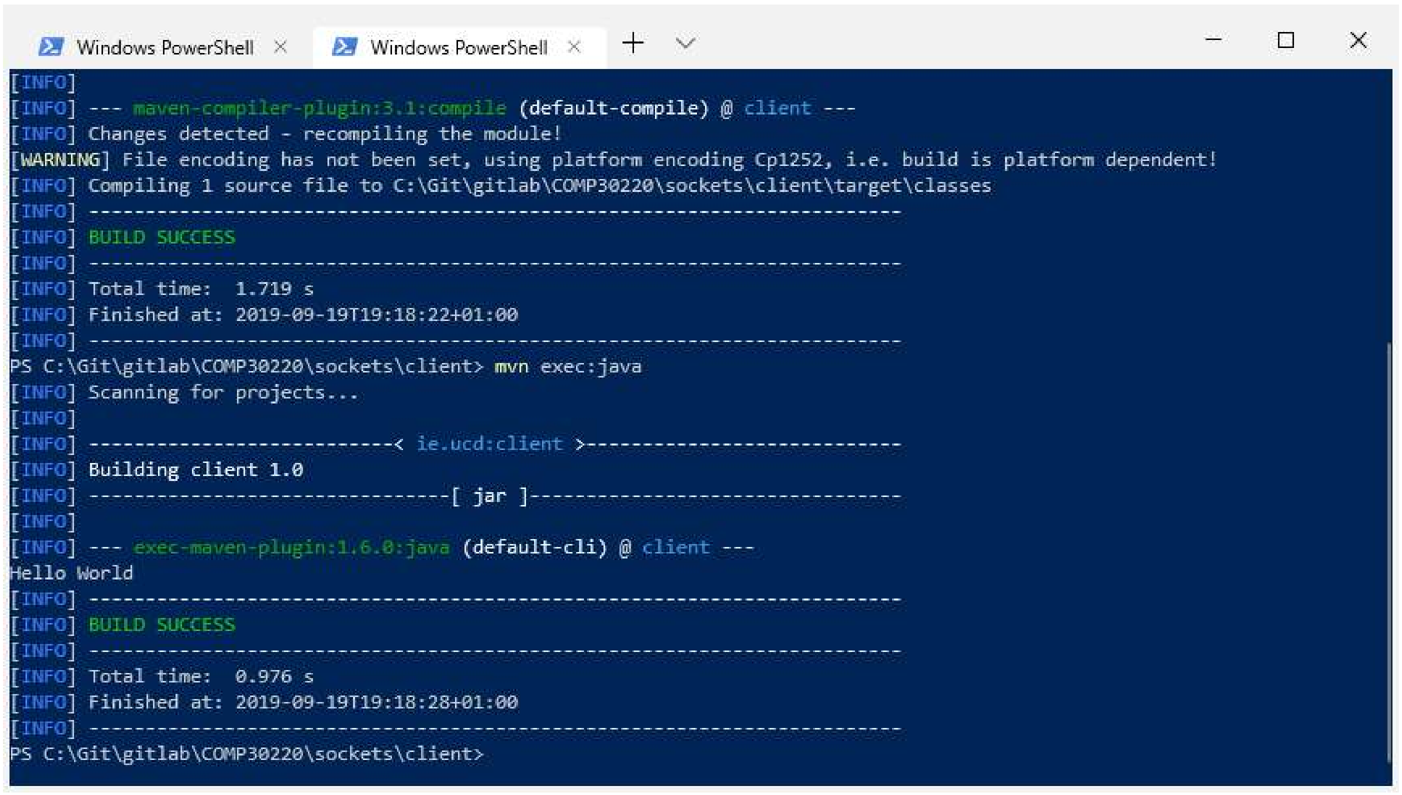


```
PS C:\Git\gitlab\COMP30220\sockets\server> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------------< ie.ucd:server >---------------------------
[INFO] Building server 1.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ server ---
```

Nothing is happening because the server has been started, it is listening on port 7788, and is blocking while it waits for a client socket connection.

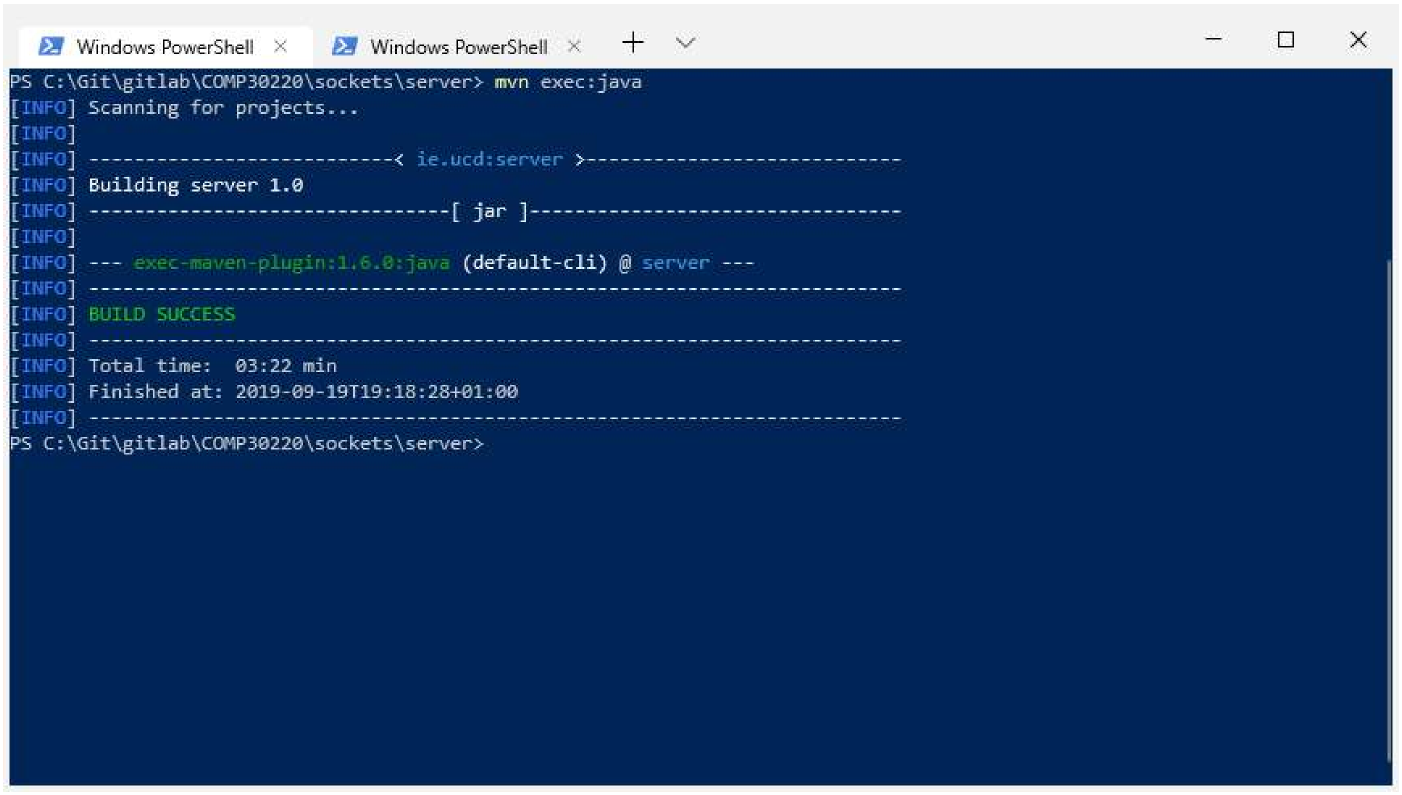
Task 2: Compiling and Running the Client code using Maven

Follow the same instructions as for task 1 but start in the client folder instead of the server folder. The final output should look like this:



```
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ client ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1252, i.e. build is platform dependent!
[INFO] Compiling 1 source file to C:\Git\gitlab\COMP30220\sockets\client\target\classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.719 s
[INFO] Finished at: 2019-09-19T19:18:22+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Git\gitlab\COMP30220\sockets\client> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------------< ie.ucd:client >---------------------------
[INFO] Building client 1.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ client ---
Hello World
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.976 s
[INFO] Finished at: 2019-09-19T19:18:28+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Git\gitlab\COMP30220\sockets\client>
```

Notice that the server code has now stopped. This is because the behaviour of the server was to read a message from the client, write that message back to the client and then to shutdown.

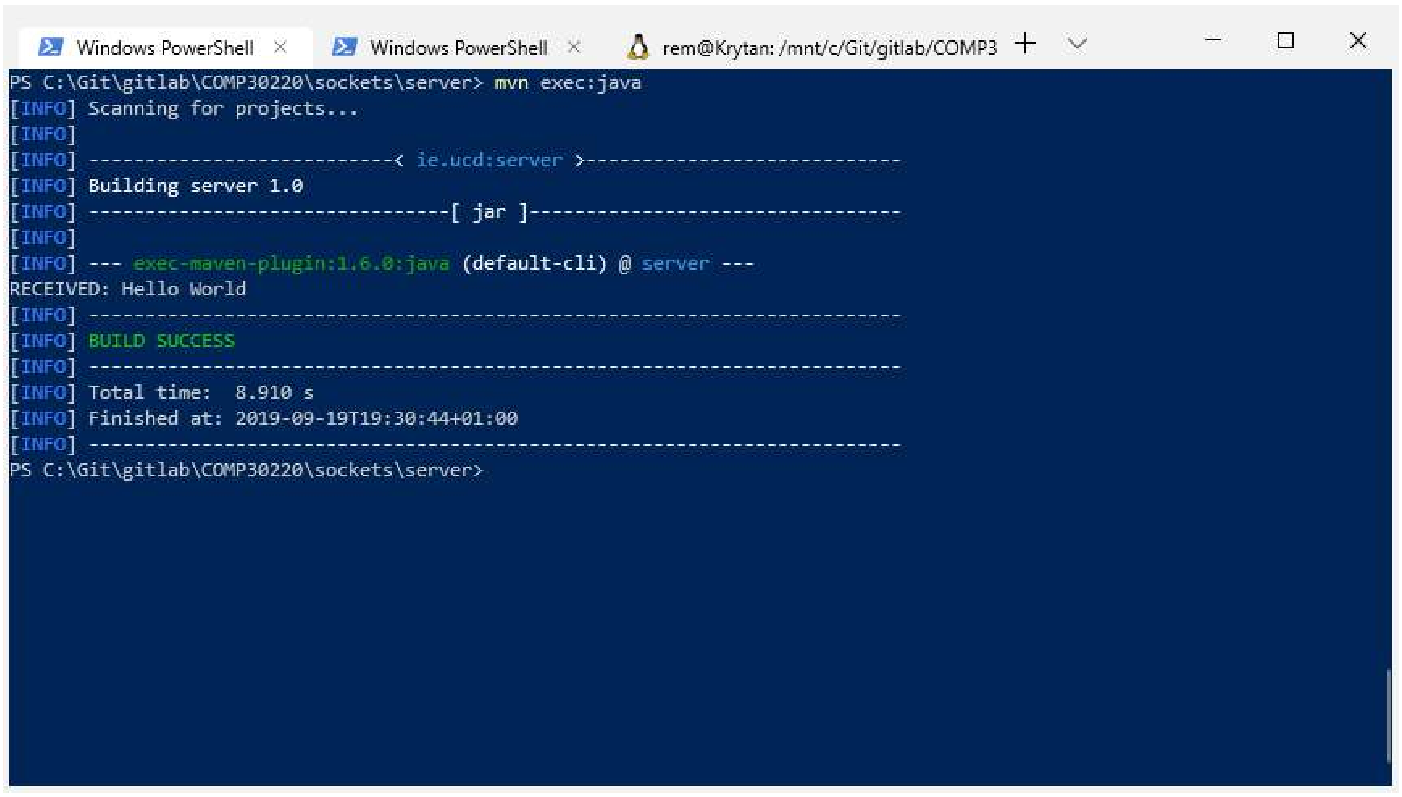


```
PS C:\Git\gitlab\COMP30220\sockets\server> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------------< ie.ucd:server >---------------------------
[INFO] Building server 1.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ server ---
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  03:22 min
[INFO] Finished at: 2019-09-19T19:18:28+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Git\gitlab\COMP30220\sockets\server>
```

Task 3: Changing the Server code

Modify the EchoServer code to print out the message “RECEIVED: <message>” where <message is that actual message received from the client. Recompile the server code and run it again using Maven. Run the client again, and see what has changed.

HINT: The server command prompt should look like this at the end:



```
PS C:\Git\gitlab\COMP30220\sockets\server> mvn exec:java
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------------< ie.ucd:server >---------------------------
[INFO] Building server 1.0
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ server ---
RECEIVED: Hello World
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  8.910 s
[INFO] Finished at: 2019-09-19T19:30:44+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Git\gitlab\COMP30220\sockets\server>
```

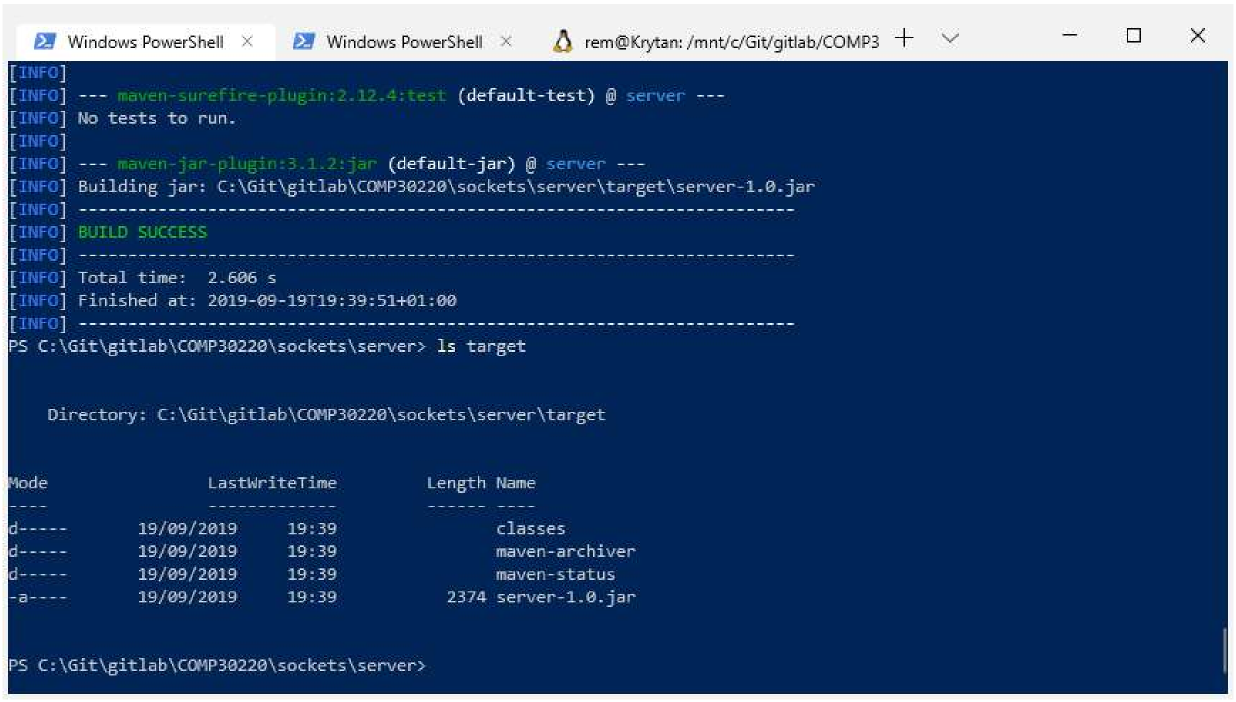
Task 4: Creating and Deploying the Server as a Docker image

Next, we will try to create and deploy the server code in a docker container. To do this, we must first create a jar file for our project (the `mvn compile` command does not do this).

To create a jar file using Maven, you need to run the package command:

```
$ mvn package
```

If you look in the target subfolder before / after running the package command, you should see that this command produces the server-1.0.jar file.

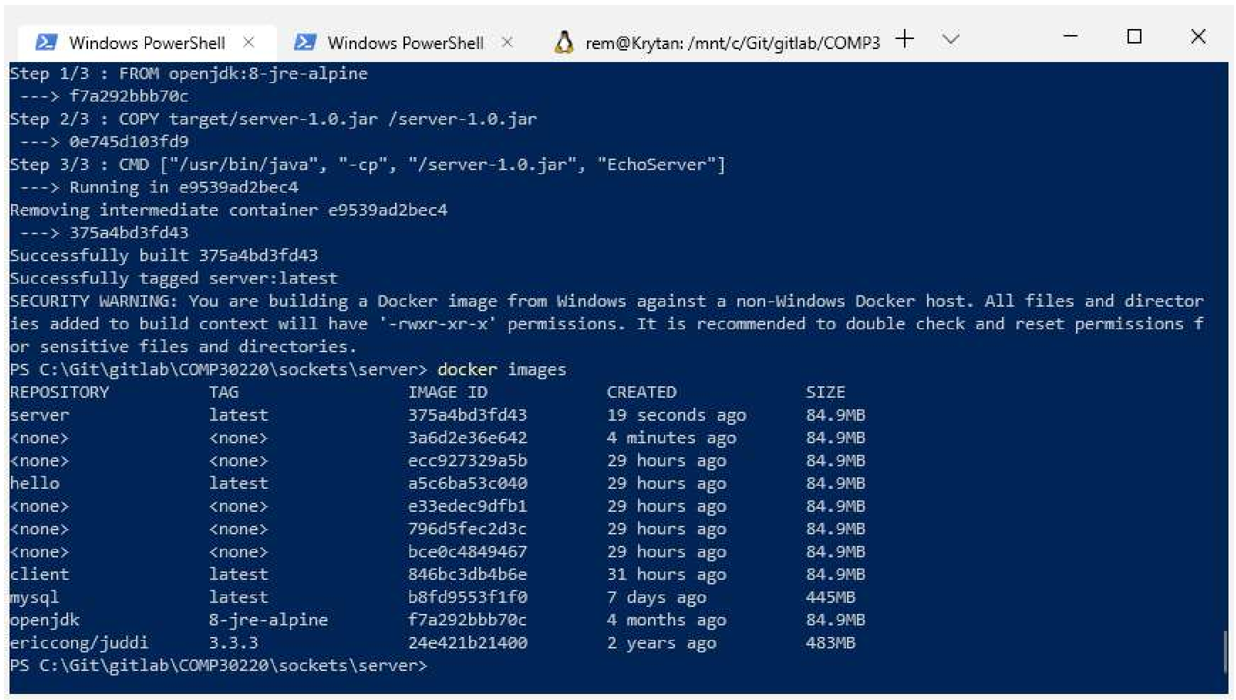


```
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ server ---
[INFO] No tests to run.
[INFO]
[INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ server ---
[INFO] Building jar: C:\Git\gitlab\COMP30220\sockets\server\target\server-1.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.606 s
[INFO] Finished at: 2019-09-19T19:39:51+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Git\gitlab\COMP30220\sockets\server> ls target


    Directory: C:\Git\gitlab\COMP30220\sockets\server\target


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----       19/09/2019     19:39                classes
d-----       19/09/2019     19:39                maven-archiver
d-----       19/09/2019     19:39                maven-status
-a----       19/09/2019     19:39           2374 server-1.0.jar


PS C:\Git\gitlab\COMP30220\sockets\server>
```

To create a docker image, we can now run the docker build command. This uses he Dockerfile file in the same folder to create the image:

```
$ docker build -t server:latest .
```

If you run the docker images command just after this, you should see it listed and its “CREATED” time should be a few seconds…

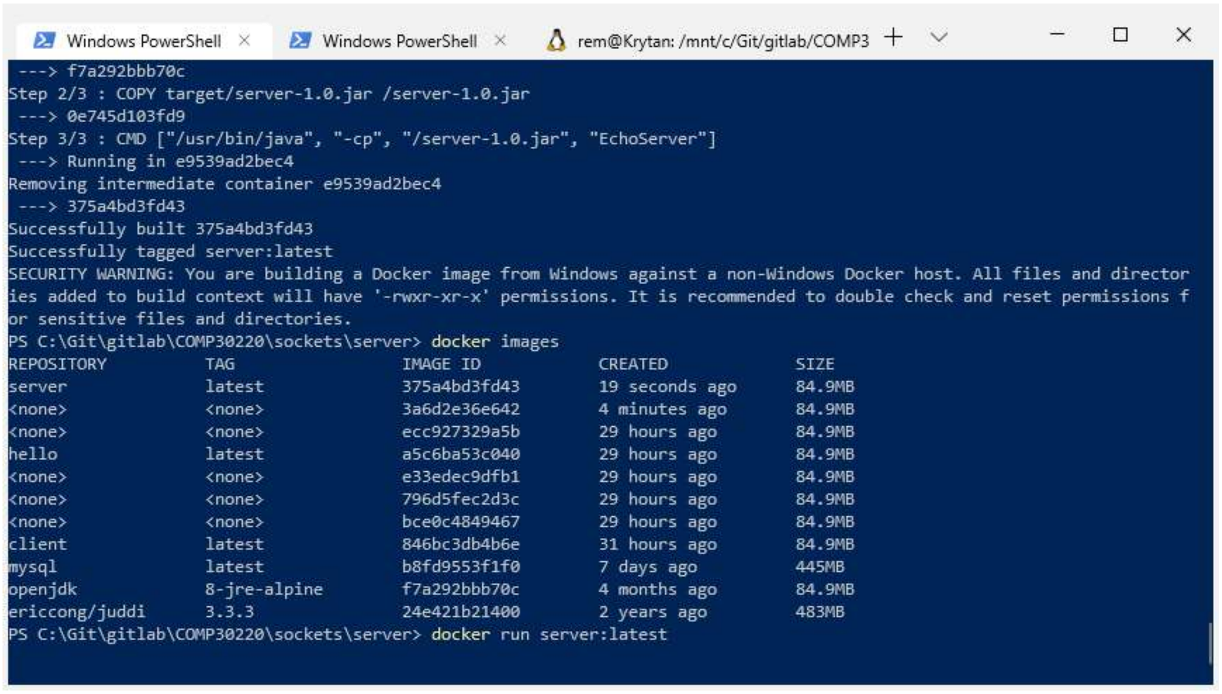


```
Step 1/3 : FROM openjdk:8-jre-alpine
 ---> f7a292bbb70c
Step 2/3 : COPY target/server-1.0.jar /server-1.0.jar
 ---> 0e745d103fd9
Step 3/3 : CMD ["/usr/bin/java", "-cp", "/server-1.0.jar", "EchoServer"]
 ---> Running in e9539ad2bec4
Removing intermediate container e9539ad2bec4
 ---> 375a4bd3fd43
Successfully built 375a4bd3fd43
Successfully tagged server:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and director
ies added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions f
or sensitive files and directories.
PS C:\Git\gitlab\COMP30220\sockets\server> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
server              latest              375a4bd3fd43        19 seconds ago      84.9MB
<none>              <none>              3a6d2e36e642        4 minutes ago       84.9MB
<none>              <none>              ecc927329a5b        29 hours ago        84.9MB
hello               latest              a5c6ba53c040        29 hours ago        84.9MB
<none>              <none>              e33edec9dfb1        29 hours ago        84.9MB
<none>              <none>              796d5fec2d3c        29 hours ago        84.9MB
<none>              <none>              bce0c4849467        29 hours ago        84.9MB
client              latest              846bc3db4b6e        31 hours ago        84.9MB
mysql               latest              b8fd9553f1f0        7 days ago          445MB
openjdk             8-jre-alpine        f7a292bbb70c        4 months ago        84.9MB
ericcong/juddi      3.3.3               24e421b21400        2 years ago         483MB
PS C:\Git\gitlab\COMP30220\sockets\server>
```

To run this image, type:

```
$ docker run server:latest
```

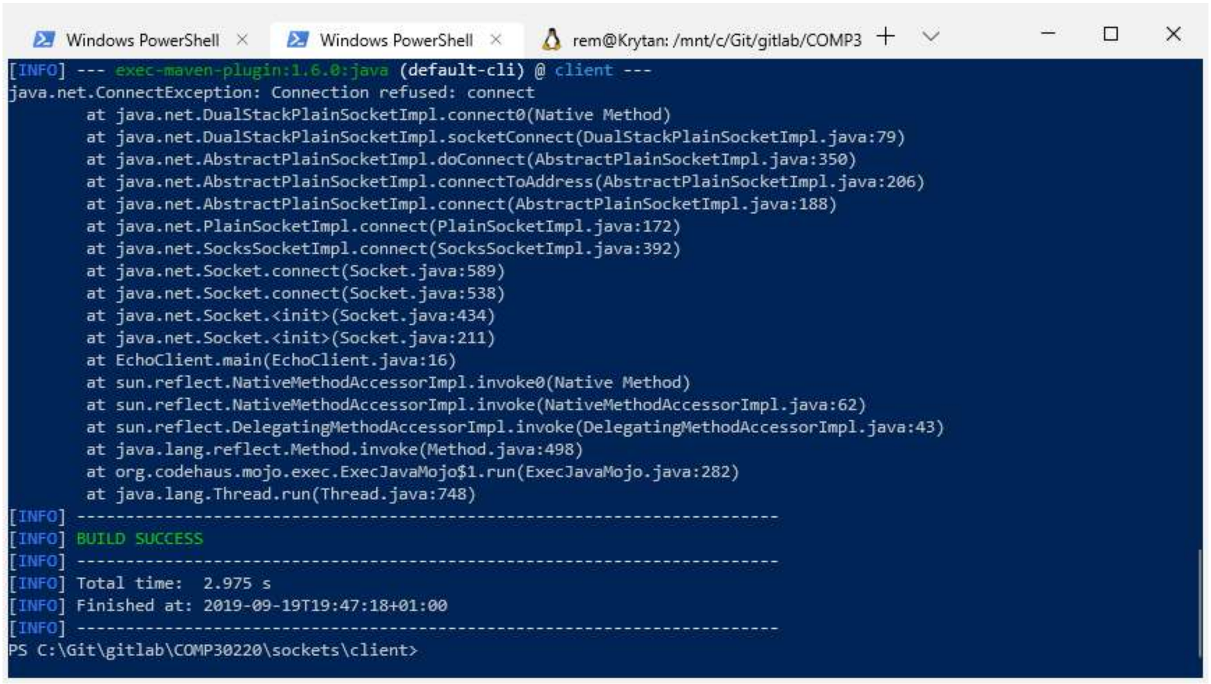
The output should be a flashing cursor:



Windows PowerShell | Windows PowerShell | rem@Krytan: /mnt/c/Git/gitlab/COMP3

```
 ---> f7a292bbb70c
Step 2/3 : COPY target/server-1.0.jar /server-1.0.jar
 ---> 0e745d103fd9
Step 3/3 : CMD ["/usr/bin/java", "-cp", "/server-1.0.jar", "EchoServer"]
 ---> Running in e9539ad2bec4
Removing intermediate container e9539ad2bec4
 ---> 375a4bd3fd43
Successfully built 375a4bd3fd43
Successfully tagged server:latest
SECURITY WARNING: You are building a Docker image from Windows against a non-Windows Docker host. All files and director
ies added to build context will have '-rwxr-xr-x' permissions. It is recommended to double check and reset permissions f
or sensitive files and directories.
PS C:\Git\gitlab\COMP30220\sockets\server> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
server              latest              375a4bd3fd43        19 seconds ago      84.9MB
<none>              <none>              3a6d2e36e642        4 minutes ago       84.9MB
<none>              <none>              ecc927329a5b        29 hours ago        84.9MB
hello               latest              a5c6ba53c040        29 hours ago        84.9MB
<none>              <none>              e33edec9dfb1        29 hours ago        84.9MB
<none>              <none>              796d5fec2d3c        29 hours ago        84.9MB
<none>              <none>              bce0c4849467        29 hours ago        84.9MB
client              latest              846bc3db4b6e        31 hours ago        84.9MB
mysql               latest              b8fd9553f1f0        7 days ago          445MB
openjdk             8-jre-alpine        f7a292bbb70c        4 months ago        84.9MB
ericcong/juddi      3.3.3               24e421b21400        2 years ago         483MB
PS C:\Git\gitlab\COMP30220\sockets\server> docker run server:latest
```

If you run the client code using maven, you should get a “connection refused” runtime error:



Windows PowerShell | Windows PowerShell | rem@Krytan: /mnt/c/Git/gitlab/COMP3

```
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ client ---
java.net.ConnectException: Connection refused: connect
        at java.net.DualStackPlainSocketImpl.connect0(Native Method)
        at java.net.DualStackPlainSocketImpl.socketConnect(DualStackPlainSocketImpl.java:79)
        at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
        at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
        at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)
        at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:172)
        at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
        at java.net.Socket.connect(Socket.java:589)
        at java.net.Socket.connect(Socket.java:538)
        at java.net.Socket.<init>(Socket.java:434)
        at java.net.Socket.<init>(Socket.java:211)
        at EchoClient.main(EchoClient.java:16)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.codehaus.mojo.exec.ExecJavaMojo$1.run(ExecJavaMojo.java:282)
        at java.lang.Thread.run(Thread.java:748)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.975 s
[INFO] Finished at: 2019-09-19T19:47:18+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Git\gitlab\COMP30220\sockets\client>
```

This happens because the server socket is running inside a container and is not accessible from outside that container (remember – each docker container has its own network interface). To overcome this, we use port mapping to expose the server socket to the host machine:

```
$ docker run -p 7788:7788 server:latest
```

The program should run as before…

Task 5: Running the Client code as a Docker image

For the penultimate task, you will deploy the client code as a docker image. To create the image, perform the same steps as in task 4, but from the client folder.

As before, run the server docker image (with port mappings), and then run the new docker client image. Again, it does not work – you should get a connection refused exception. This is because the container with the client code contains its own network interface, and by default, it cannot access sockets in other containers using localhost.

You can make use of the port mapping to overcome this. First you must find the machines IP address, and then you must modify the CMD statement in the client Dockerfile as follows:

```
CMD ["/usr/bin/java", "-cp", "/client-1.0.jar", "EchoClient", "<my ip address>"]
```

If you put the correct IP address in and rebuild the docker image, the code should work again.

This approach is not great, because it depends on the IP address of the machine that the server is running on. You would have to rebuild the image for each machine it is deployed on…

A better solution is to use a user defined docker network. To achieve this, we will need to still make the same type of modification we made above, but instead, we specify an alias:

CMD ["/usr/bin/java", "-cp", "/client-1.0.jar", "EchoClient", "**myserver**"]

We then create a custom docker network and expose the server container using the given alias:

```
$ docker network create mynet
```

```
$ docker run -p 7788:7788 --network-alias myserver --network mynet -it
server:latest
```

Similarly, you can expose the client container using the following command:

```
$ docker run --network mynet -it client:latest
```

As a final point: because I am still including the port mapping, I can now access my server by running the client docker image (on the same docker network) or by running the code from maven (mvn exec:java)…

Task 6: The final step(s)

If you have completed up to task 5 then you have the basic skills needed for the module. However, there are a couple of challenges I want to set for you:

- If you look in the pom.xml files, you will see commented out configuration for the spotify maven docker plugin. Uncomment this code and run “mvn package” – see what happens.
- Modify the code to include a server that can handle multiple client connections (initially sequentially)
- Can you modify the code to handle client connections in parallel?