Installation Manual

Aquadine

Version	1.0
Date	4th June 2019
Team	Aquadis team 3
Authors	500775493 – Yasmine Ben Youb 500718509 – Brian Dekker
Client	Aquadis
Product owner	Okechukwu Onwunli

Index

Introduction

In this document you can find the requirements and steps to successfully install Aquadine. This app works with a MySQL database, a back end API in java and a front end in Angular. Once all the required software has been installed, and the setup has been completed it will be possible to run the app.

Requirements.

All the software needed to run the application is listed below

The back-end

The required software and the dependencies that are needed to get the server up and running are the following:

- The back-end git project: https://github.com/Youby/GIT-Aquadine-BackEnd
- JDK1.8: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
- Maven Java.
- Hibernate (Maven).
- Jax RS (Maven).
- Glassfish or any other Java server that you would like to use.
- Database running on MySQL

The front-end

The required software and dependencies that are needed for the front end are:

- The front-end git project: https://gitlab.fdmci.hva.nl/youby/aquadine/
- Github Desktop or any other FTP of your choice.
- Node Package Manager (NPM): https://node.js
- Angular CLI: https://angular.io/guide/quickstart

The Database

To make full use of the application which is able to send and retrieve data you will need a working database. The database is configured with MySQL and will be able to communicate with the local Java server. For our project we used MySQL, Hibernate and Jax RS.

Setup

First you will need to open the database in MySQL you can do this by downloading the database following this git link: https://github.com/Youby/GIT-Aquadine-BackEnd
After opening the project run the database.sql in your own MySQL workbench.

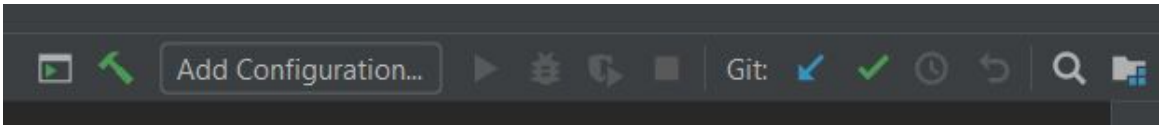
After this Open the “resources” folder, then the ”META-INF” folder and then open the persistence.xml. In the persistence XML change the following:

1: If needed change the localhost depending on the device used.
2: Change the value to your own username.
3: Change the value to your own password.

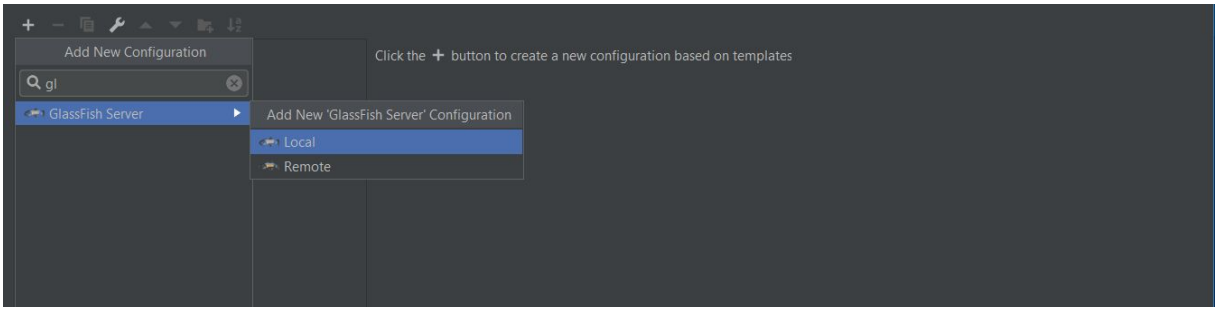
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="2.1"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="aquadineDB">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
        <properties>
            <property name="javax.persistence.schema-generation.database.action" value="update"/>

            <property name="hibernate.dialect" value="org.hibernate.dialect.MySQLDialect" />
            <property name="javax.persistence.jdbc.driver" value="com.mysql.jdbc.Driver" />
            <property name="javax.persistence.jdbc.url" value="jdbc:mysql://localhost:3306/aquadinedb?createDatabaseIfNotExist=true" />

            <property name="javax.persistence.jdbc.user" value="root" />
            <property name="javax.persistence.jdbc.password" value="admin" />
        </properties>
    </persistence-unit>
</persistence>
```

Deploying the back end server

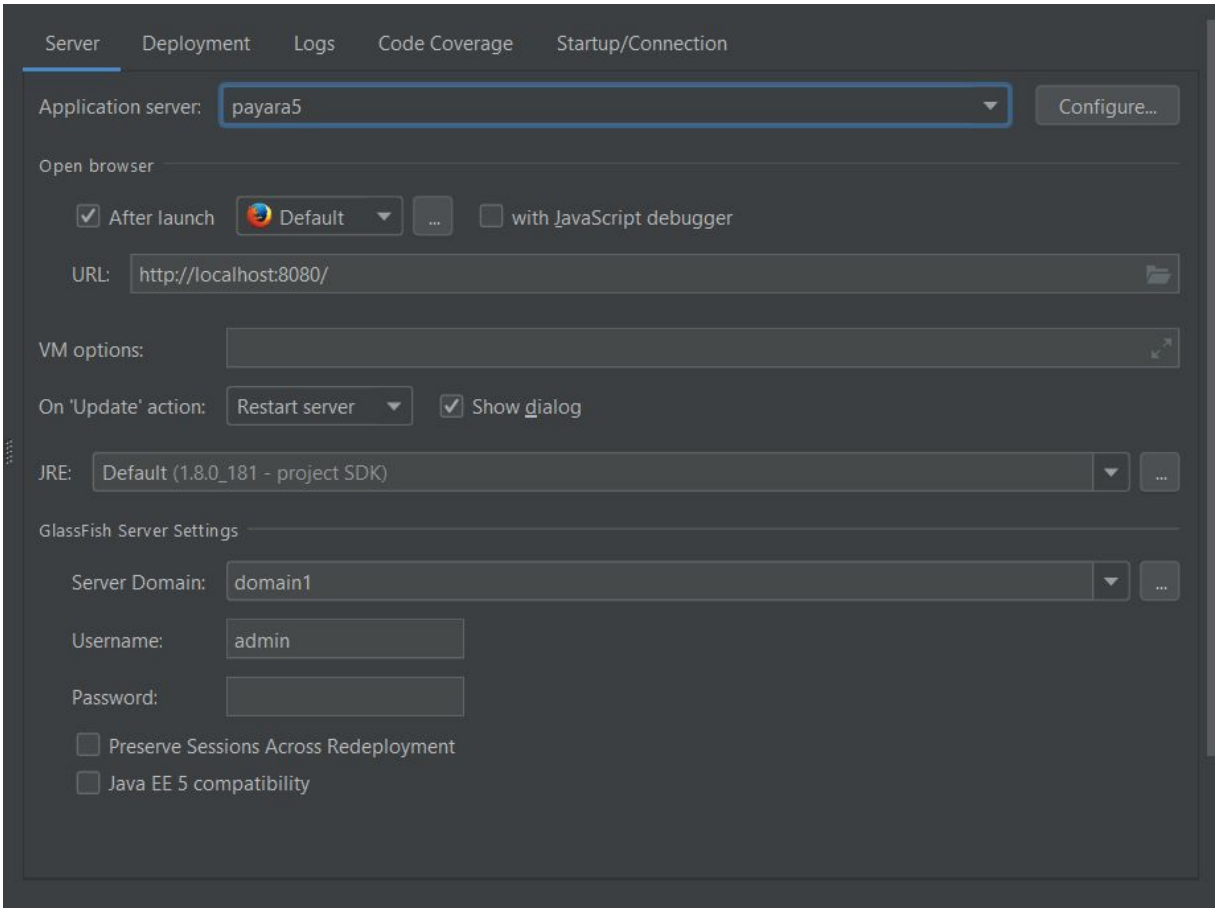
To deploy the server configurations must be added.
It’s possible to add the server configuration by clicking: “add configuration”.



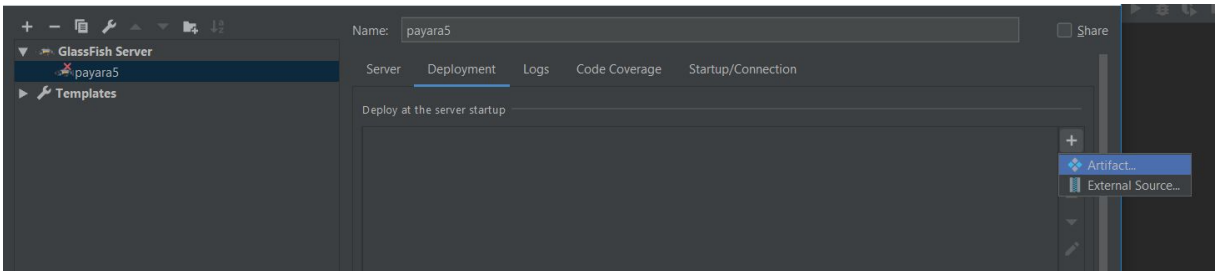
To add a configuration click on the plus button in the top left of the pop-up. Then search for glassfish server and select “local”.



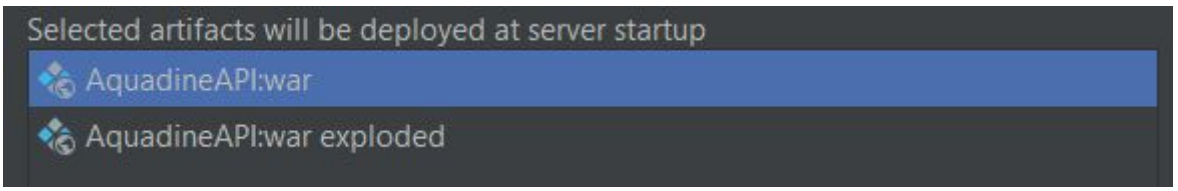
In the server tab, add the Payara server to the “application server”. Make sure the JRE is using the correct Java version. Select “domain1” from the available server domains and enter “admin” as username.



After this go to the deployment tab.
And click on the “+” button, click on artifact.



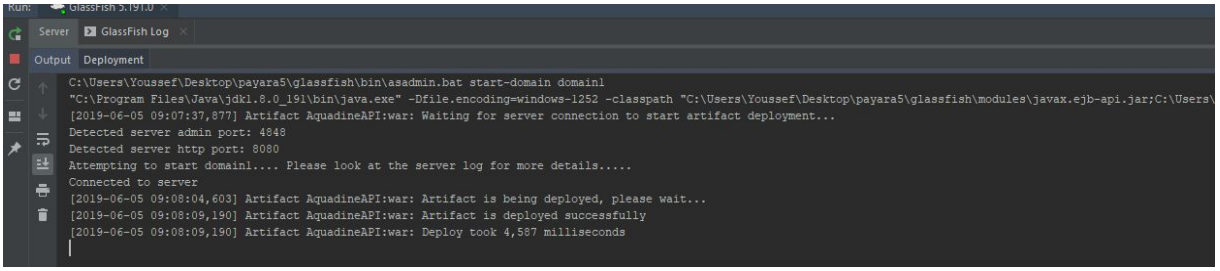
Select AquadineAPI:war.



After this press ok to add the artifact, and then press ok once more to add the configurations.

The server can now be deployed. To deploy the server press the run button right next to the configurations.

If the server has successfully deployed the log will display this:



The website

After setting up the back end, the front end can be setup.

Cors extension

To make full use of the app you will need to install a CORS extension you can use any CORS extension on google chrome and firefox.

Chrome:
https://chrome.google.com/webstore/detail/allow-control-allow-origi/nlfbmbojpeacfghkpbjhddihlkkiljbi

Firefox: https://addons.mozilla.org/nl/firefox/addon/cors-everywhere/

Front end setup.

First download the git project for the front end application
Link: https://gitlab.fdmci.hva.nl/youby/aquadine/

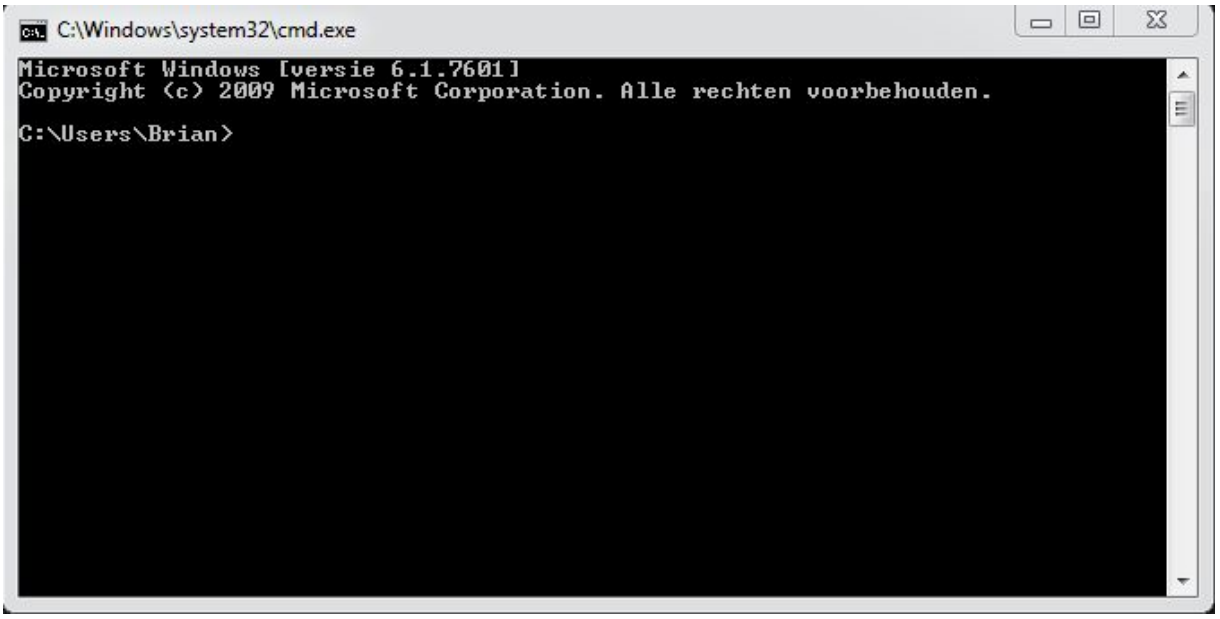
After downloading the application you will need to install Angular CLI and Node package manager, you can find these in the requirements chapter.

When Node package manager and Angular CLI are both installed you can open your CMD and access the app files where you stored them by using the **cd** command. When you have accessed the file with the src code simply run the **npm install** command to install the node modules for the application. After this run the command **ng serve -o** and the application will open in your default browser.

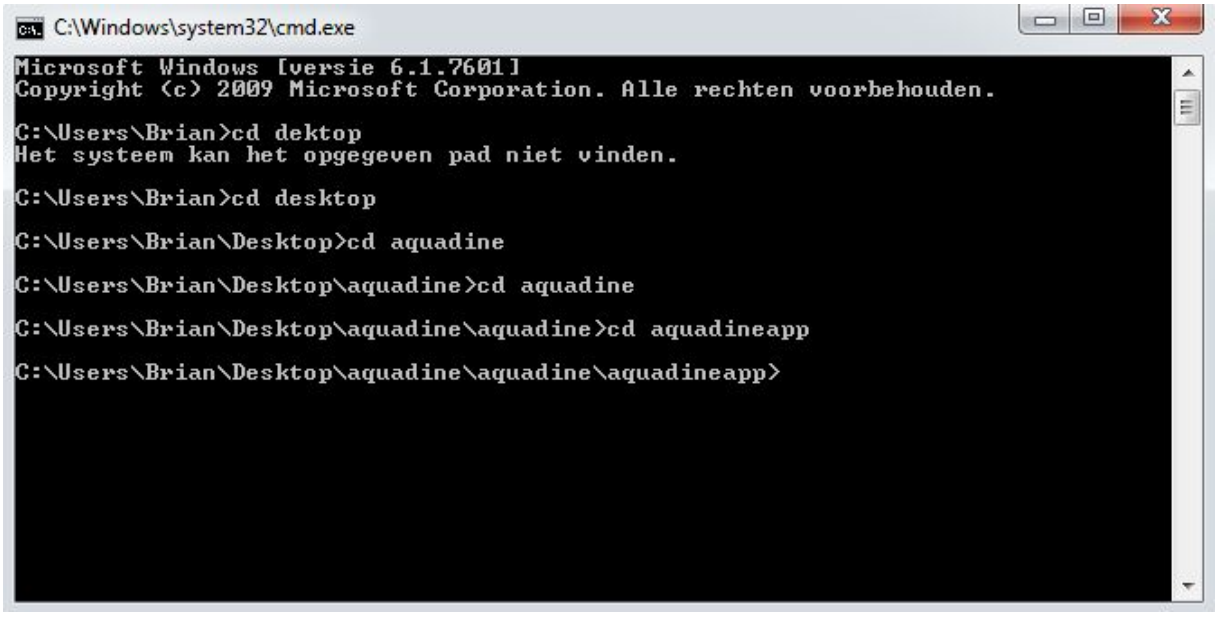
You can find step by step instructions in the images below.

step by step front end setup

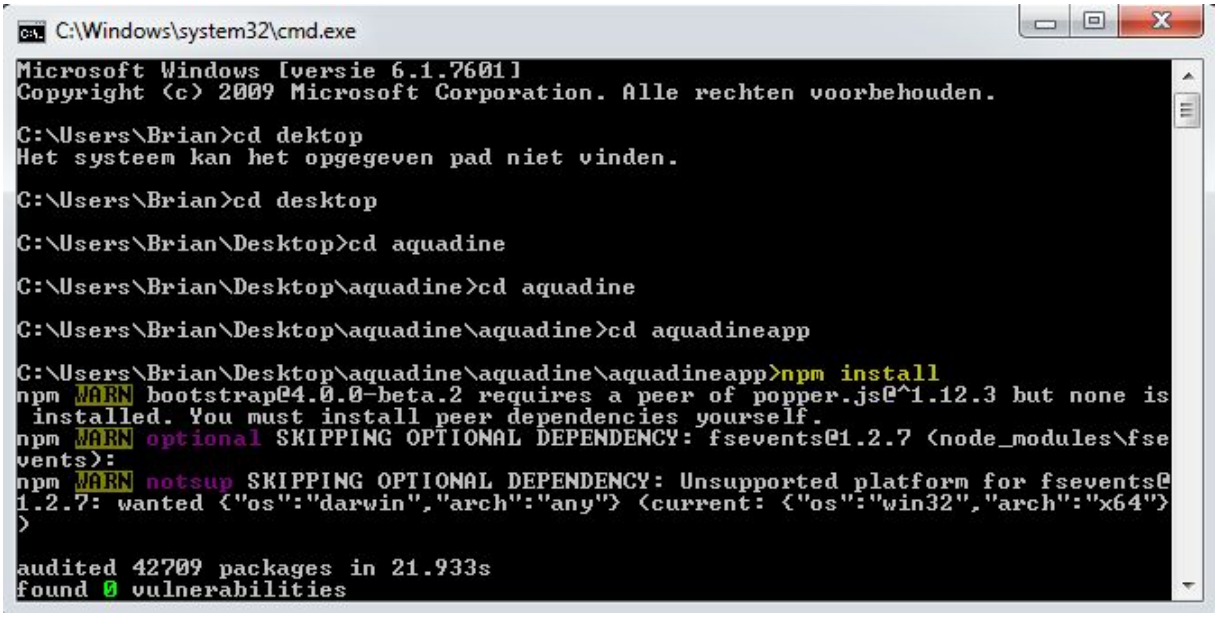
step1: Open the CMD and install the node modules.



Use the **cd** command to reach the location where the application has been saved.
This is different depending on where you saved it.



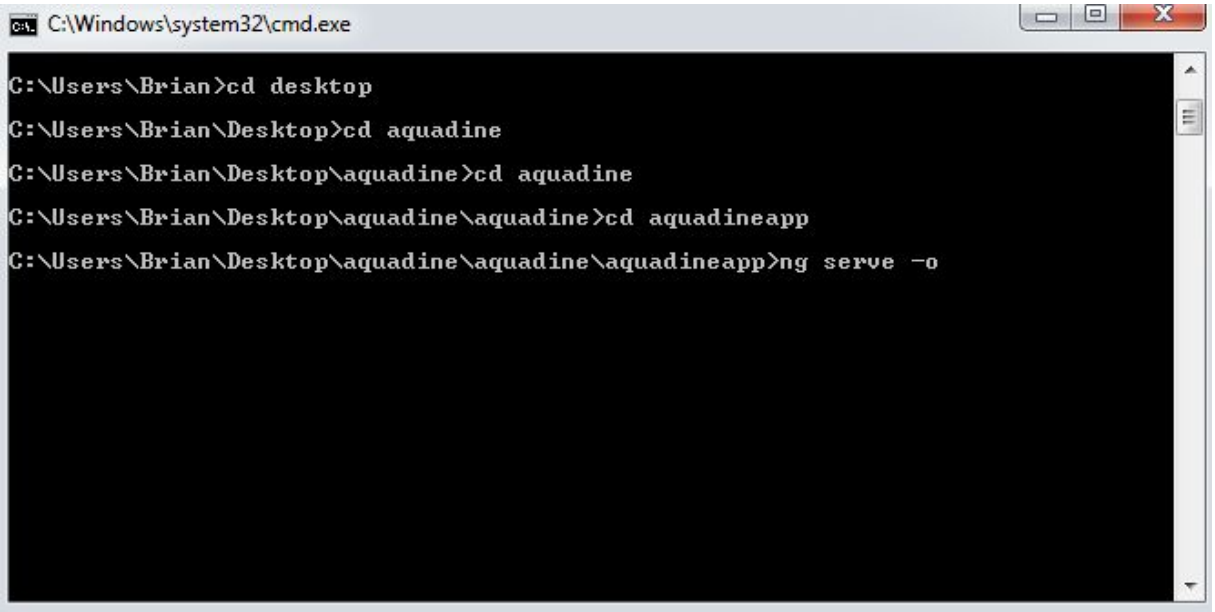
Install the node modules by running **npm install.**



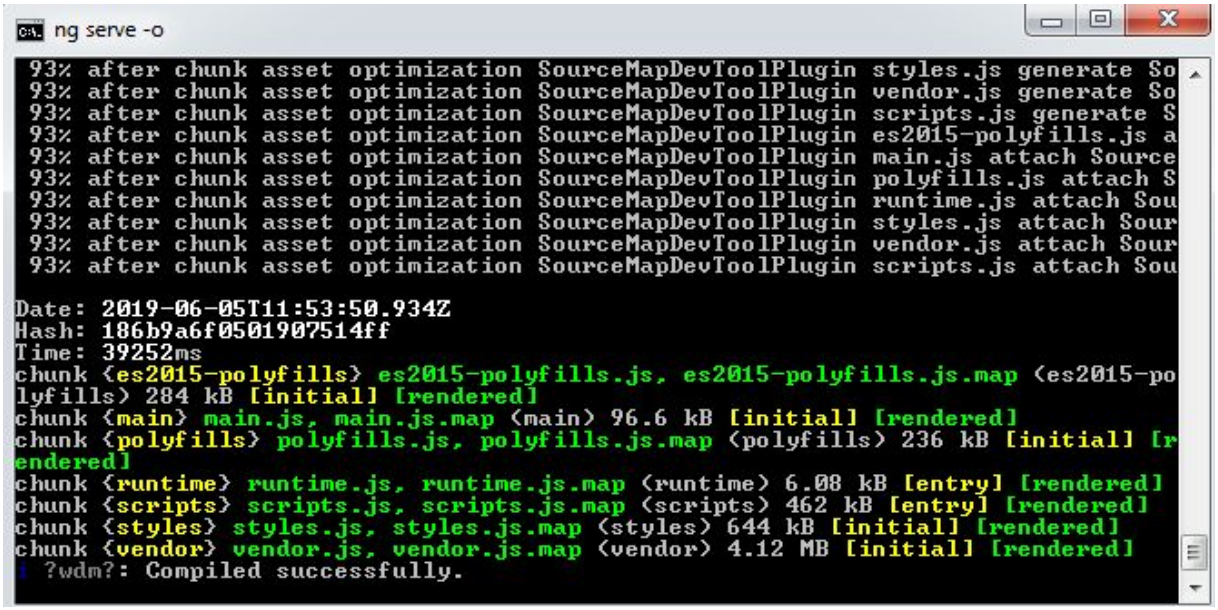
This can take a few minutes.

Step 2: Running the app on the local server

run the command **ng serve -o** to run the local server for the app.



This process can take a few minutes.



When the process is finished it should look something like this with **compiled successfully** at the bottom. And your default browser should open with the app running.