A picture containing text, sign, clipart

Description automatically generated

Microservices

Assignment 5

Mammon Ahmad Alshamali

14/5/2022

Outline

Introduction……………………………………..…….…3

System Details...………..……………………………..…5

Docker File……….…………………...……….……….13

Docker-Compose………………...……...……………...17

Introduction

Diagram

Description automatically generated

The system microservices can be described as follows:

* Enter data (web app) is used to collect data (any data: grades, temperatures, ...etc.). Users are allowed to enter data after validating their credentials through the Authentication Service. Any entered data will be written to the MySQL DB service.
* Show results (web app) is used to present simple analytics (max, min, average …etc.). Users are allowed to see the analytics after validating their credentials through the Authentication Service. The Show Results service reads data from the Mongo DB service.
* The Authentication Service is a simple service to validate users’ credentials.
* The Analytics Service read data from the MySQL DB service and gets simple statistics like Max, Min, Avg…etc. and write that to Mongo DB Service.

From the requirements, the solution will have two web apps, four rest API and, two databases. After that six docker images and finally a docker-compose to run all of them together.

All the projects are built using spring boot, and for web apps, thyme leaf is used because the spring boot environment is helpful to deal with it.

System Details

The system was developed to be used on localhost for testing purposes. After ensuring the system was working correctly in the localhost, the next stage was replacing these localhost URLs with DNS which is created in the docker network.

* Enter Data (web app)

This app will have two main pages and one common error page. The first page is the login page to authenticate the user. After the user inserts the login details, the data will be sent to the authentication service to validate the user. The communication method used in the code:

Text

Description automatically generated

After login, a session will be created for the user. The user will be moved to enter the data page and can enter numbers on this page.

After entering the numbers (data), the number will be sent to the MySQL service. The communication method used in the code:

Text

Description automatically generated

For errors, there is a common error handler to deal with the errors and sent the user to the error page.

The web app pages:

A picture containing application

Description automatically generated

Figure login page

A picture containing graphical user interface

Description automatically generated

Figure enter numbers page

Graphical user interface, text

Description automatically generated

Figure error page

* Show Data (web app)

Most of the things which exist in the enter-data web app exist in this app too. After login, the user will be moved to the show data page, every few seconds this page sends a request to the Mongo service to get the results. The method to communicate is:

Text

Description automatically generated

Table

Description automatically generated

Figure show data page

* Authentication- service:

The authentication service is a rest API, it has a static database. The user details are received by post method:

Text

Description automatically generated

* MySQL Service

The MySQL service is a rest API, it is connected to a MySQL database. The rest API uses spring data JPA to store and retrieve data in a relational database. The duties for this rest API are to count the number of the received input from the user, store the input from the user, and get the list of the stored data.

Text, application

Description automatically generated

* Mongo Service

The Mongo service is a rest API, it is connected to MongoDB. The rest API uses spring data MongoDB to store data and retrieves it from MongoDB. The rest API duties are receiving the data from the analyzer and storing the data in MongoDB. What is more, retrieve the data from MongoDB and send it to the show data web app.

Text

Description automatically generated

* The Analyzer

From the requirements of the assignment, it shows the Analyzer should send the request to the MySQL service and MongoDB service. For this, the Awaitility 3.1.2 Dependency enables scheduling the tasks. The task which has been scheduled is checking the number of inputs at the MySQL service.

Text

Description automatically generated

The Analyzer’s job is to get data from the MySQL service, find statistics, and pass them to the MongoDB service

Text

Description automatically generated

* Clean Code and SOLID

The code in the project is simple and follows clean code and SOLID principles. These principles are noticeable in many things like the naming convention for the classes, variables, constants, and functions. Most of the functions do one thing and don’t have a lot of arguments. Comments are avoided; there is almost no single comment in the code. There is no return null or pass null in the total system.

Classes are small, encapsulated, and follow Uncle Bob’s format; the private function should be after the first use, the local variable before the first use. So, to find a private function in the code just downward from the first usage. Constant is not inherited as well.

No dead code and no duplicated code in the same project. But for microservices, it is common to see the same class being used many times in different projects as an example the user class exists in three projects the two web apps and the authentication rest API.

Docker file

All the six images are the same except the last line in each docker file. Many things are considered while building the images and some fallbacks happened like using windows to build the images cannot enable the build kit feature. But the image can contain that code in case the image will rebuild in the future on another OS. Another solution is using “docker buildx” to build the images.

All the docker images are built according to <https://spring.io/guides/topicals/spring-boot-docker/>

This is one of the docker files used to build the images:

Graphical user interface, text, application, email

Description automatically generated

The base image is openjdk:8-jdk-alpine. The alpine images are smaller than the standard OpenJDK library images from the Docker hub. What is more, “20 MB can be saved in the base image by using the JRE label instead of JDK and this can be done in the next base image. Not all applications work with a JRE (as opposed to a JDK), but most do. The advantage of working with a JRE is avoiding the risk of misuse of some of the JDK features.”

“A Spring Boot fat JAR naturally has “layers” because of the way that the JAR itself is packaged. If we unpack it first, it is already divided into external and internal dependencies. To do this in one step in the docker build, we need to unpack the JAR first.”

The following commands will do the job:



Text

Description automatically generated

“After advancing, the multistage was added to the docker file it allows to build JAR file and copying the result from one image to another. The first image is labeled build, and it is used to run Maven, build the fat JAR, and unpack it. the source code has been split into four layers. The later layers contain the build configuration and the source code for the application, and the earlier layers contain the build system itself (the Maven wrapper). This is a small optimization, and it also means that we do not have to copy the target directory to a docker image, even a temporary one used for the build.”

“Every build where the source code changes are slow because the Maven cache must be re-created in the first RUN section. But we have a completely standalone build that anyone can run to get our application running as long as they have docker. That can be quite useful in some environments — for example, where you need to share your code with people who do not know Java.”

The following commands do the job:



Text

Description automatically generated

The first line makes a “way to cache build dependencies. The RUN directive accepts a new flag: --mount for more details” <https://github.com/moby/buildkit/blob/master/frontend/dockerfile/docs/syntax.md>

The following commands show do the job:





“For security aspects, the processes should not be run with root permissions. Instead, the image should contain a non-root user that runs the application.”

“In a Docker file, this can be achieved by adding another layer that adds a (system) user and group and set it as the current user.”

The following commands do the job:

Graphical user interface, text, application

Description automatically generated

“In case someone manages to break out of your application and run system commands inside the container, this precaution limits their capabilities”

To build the image this command was used in the terminal:



The details of resulted images:

A picture containing text

Description automatically generated

Docker Compose

Docker-compose is important to run all the images together, this is also can be done without docker-compose but docker-compose makes things easier for the users.

In our Docker-compose file, there are 8 images which are the previous six images with MySQL image and mongo image. The next picture shows the order of creating containers for the 8 images:

Text

Description automatically generated with medium confidence

Using docker PS command we can see the following:

A screenshot of a computer

Description automatically generated with medium confidence

* Docker-Compose code

Mongo service and MongoDB:

Text

Description automatically generated

MySQL service and MySQL:

Text

Description automatically generated

* Authentication

A picture containing text

Description automatically generated

* Analyzer

Text

Description automatically generated

* Enter data web app and Show data web app

Text

Description automatically generated

* Volumes

Text

Description automatically generated

* Network

A picture containing text

Description automatically generated