ATYPON

Containerization

Written by: Moayad AL-SALEH

Contents

What is the problem? 3
Methodology 4
Docker-Compose: 4
Container Enter data: 4
Container Authentication Service: 5
Container MySQL: 6
Container Analytics Service: 6
Container MongoDB: 7
Container Show Results: 8

What is the problem?

Build a containerized microservices data collection and analytics system as shown in Fig.1 You need to write a docker file for each image (service), and docker compose file to run the system.

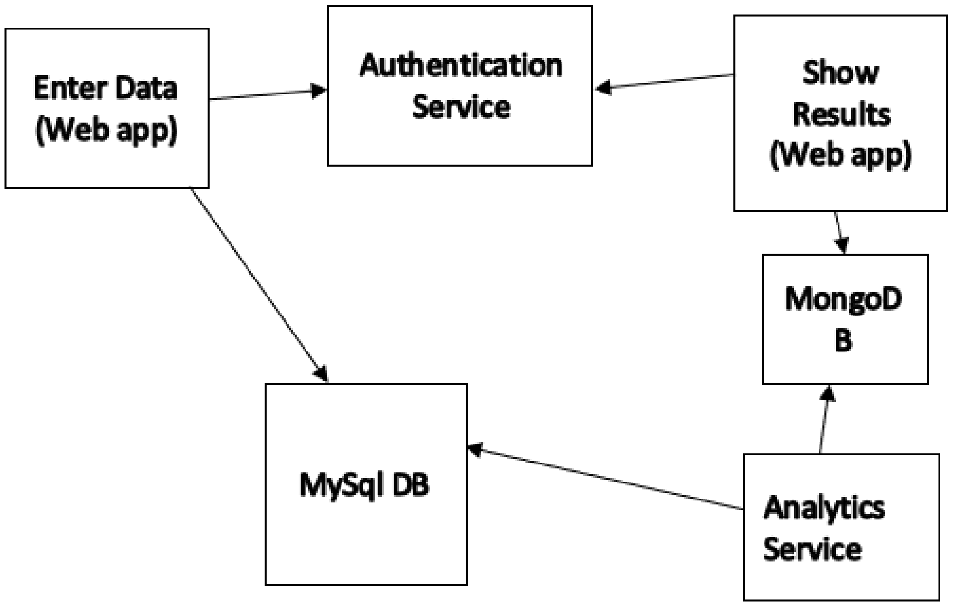


Fig. 1 The Data collection and Analytics System Architecture

What is the purpose of the System?

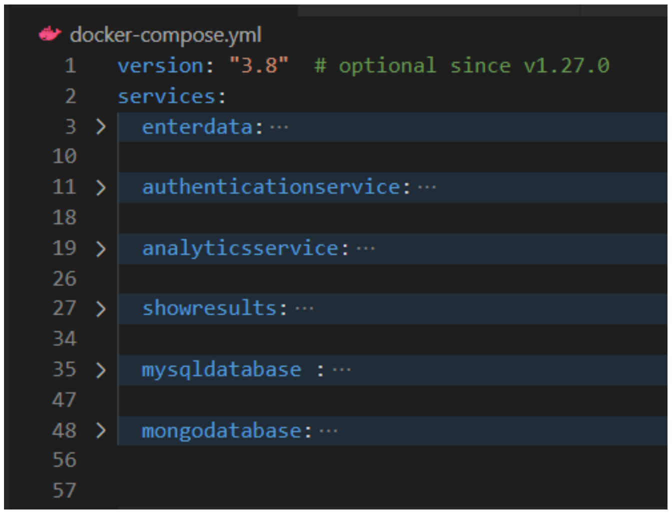
The Microservices find the Maximum number entered by the user and displays it.

Methodology

I have built these six main containers that I collected in a Docker Compose file, and I will explain each container separately:

Docker-Compose:

It is a file that contains other containers to facilitate the process of building them and make them interconnected by building a default network for all containers.

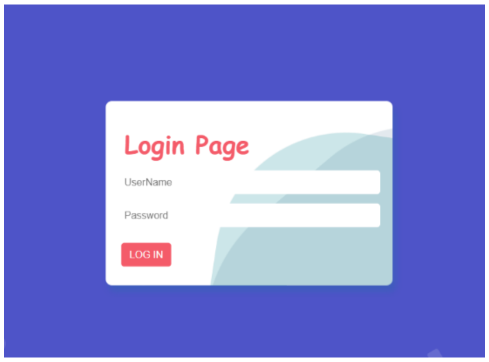


docker-compose.yml

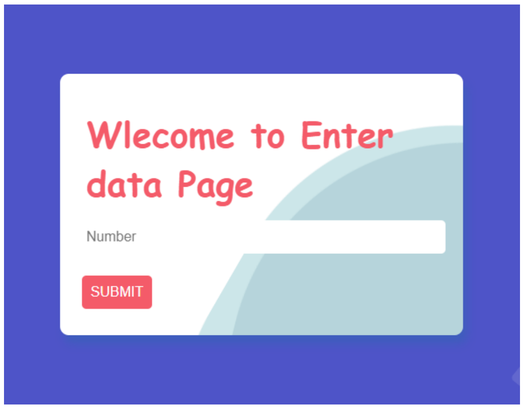
```
version: "3.8"  # optional since v1.27.0
services:
  enterdata:…

  authenticationservice:…

  analyticsservice:…

  showresults:…

  mysqldatabase :…

  mongodatabase:…

```

Container Enter data:

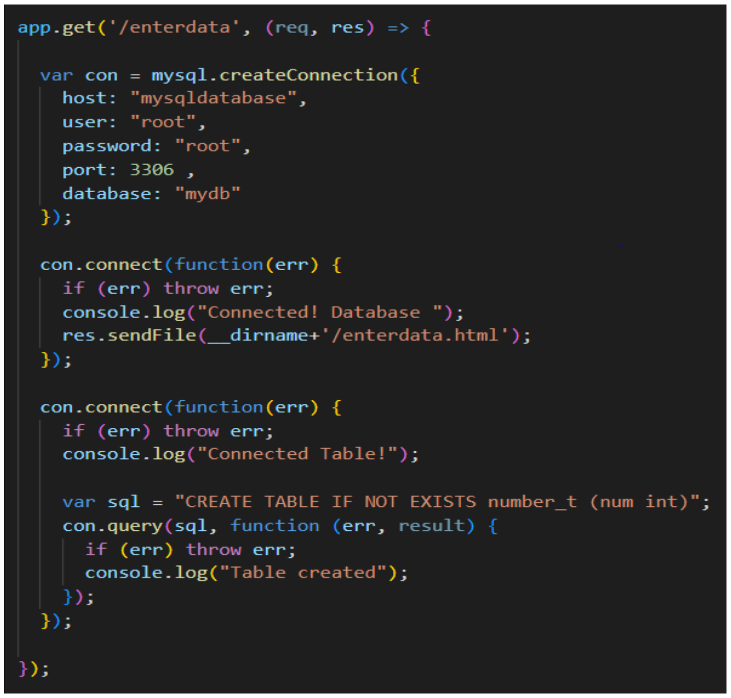
it is a container where the user enters the name and password to be transferred to the Authentication Service container.



After that the user enters the number:



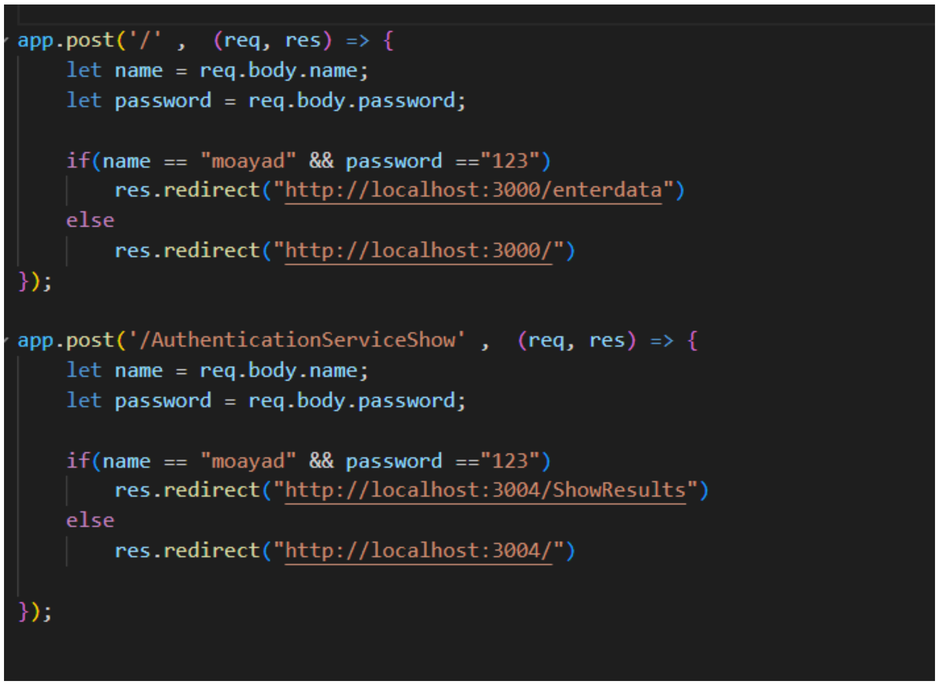
The number is sent to a controller in the node js, which receives it and inserts it in the Database container:


```
app.get('/enterdata', (req, res) => {

  var con = mysql.createConnection({
    host: "mysqldatabase",
    user: "root",
    password: "root",
    port: 3306 ,
    database: "mydb"
  });

  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected! Database ");
    res.sendFile(__dirname+'/enterdata.html');
  });

  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected Table!");

    var sql = "CREATE TABLE IF NOT EXISTS number_t (num int)";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("Table created");
    });
  });

});
```

Container Authentication Service:

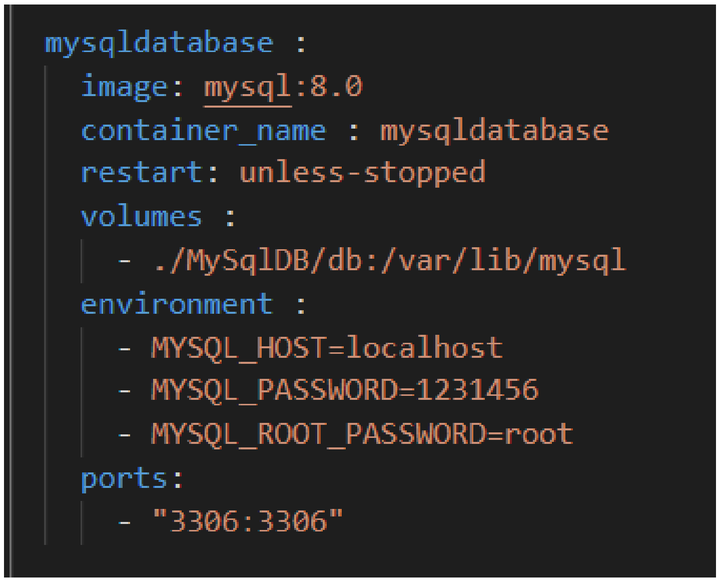
It is a container that does not have front end,

it takes the name and password and checks them:


```
app.post('/' ,  (req, res) => {
    let name = req.body.name;
    let password = req.body.password;

    if(name == "moayad" && password =="123")
        res.redirect("http://localhost:3000/enterdata")
    else
        res.redirect("http://localhost:3000/")
});

app.post('/AuthenticationServiceShow' ,  (req, res) => {
    let name = req.body.name;
    let password = req.body.password;

    if(name == "moayad" && password =="123")
        res.redirect("http://localhost:3004/ShowResults")
    else
        res.redirect("http://localhost:3004/")

});
```

Container MySQL:

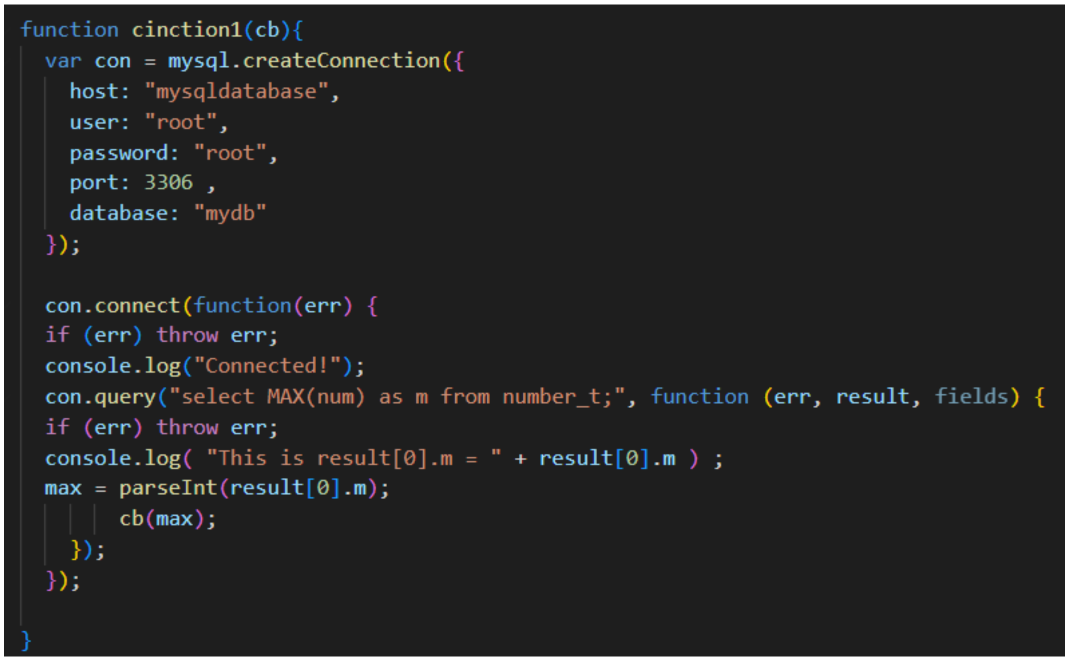
It is a container in which the numbers are stored .



```
mysqldatabase :
  image: mysql:8.0
  container_name : mysqldatabase
  restart: unless-stopped
  volumes :
    - ./MySqlDB/db:/var/lib/mysql
  environment :
    - MYSQL_HOST=localhost
    - MYSQL_PASSWORD=1231456
    - MYSQL_ROOT_PASSWORD=root
  ports:
    - "3306:3306"
```

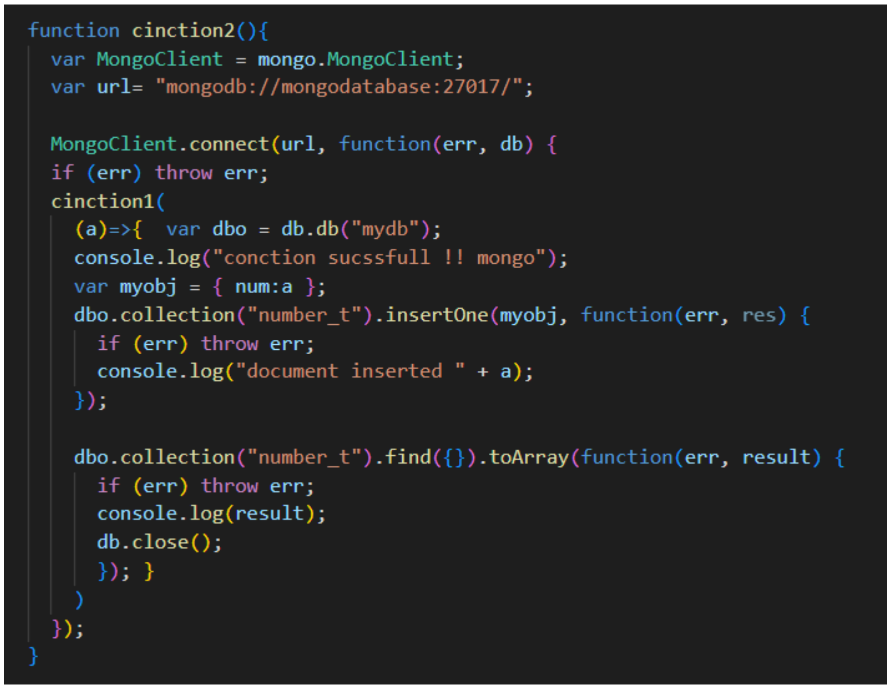
Container Analytics Service:

It is a container that does not have front end, it reads and find the maximum Number from MySQL DB.



```
function cinction1(cb){
  var con = mysql.createConnection({
    host: "mysqldatabase",
    user: "root",
    password: "root",
    port: 3306 ,
    database: "mydb"
  });

  con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query("select MAX(num) as m from number_t;", function (err, result, fields) {
  if (err) throw err;
  console.log( "This is result[0].m = " + result[0].m ) ;
  max = parseInt(result[0].m);
        cb(max);
    });
  });

}
```

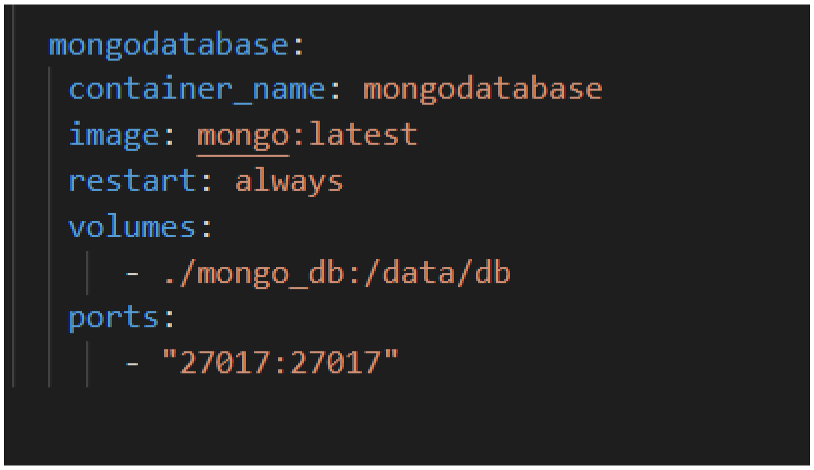
After that, it communicates with Mongo DB to store the result of the analysis:



```
function cinction2(){
  var MongoClient = mongo.MongoClient;
  var url= "mongodb://mongodatabase:27017/";

  MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  cinction1(
    (a)=>{  var dbo = db.db("mydb");
    console.log("conction sucssfull !! mongo");
    var myobj = { num:a };
    dbo.collection("number_t").insertOne(myobj, function(err, res) {
      if (err) throw err;
      console.log("document inserted " + a);
    });

    dbo.collection("number_t").find({}).toArray(function(err, result) {
      if (err) throw err;
      console.log(result);
      db.close();
      }); }
    )
  });
}
```

Container MongoDB:

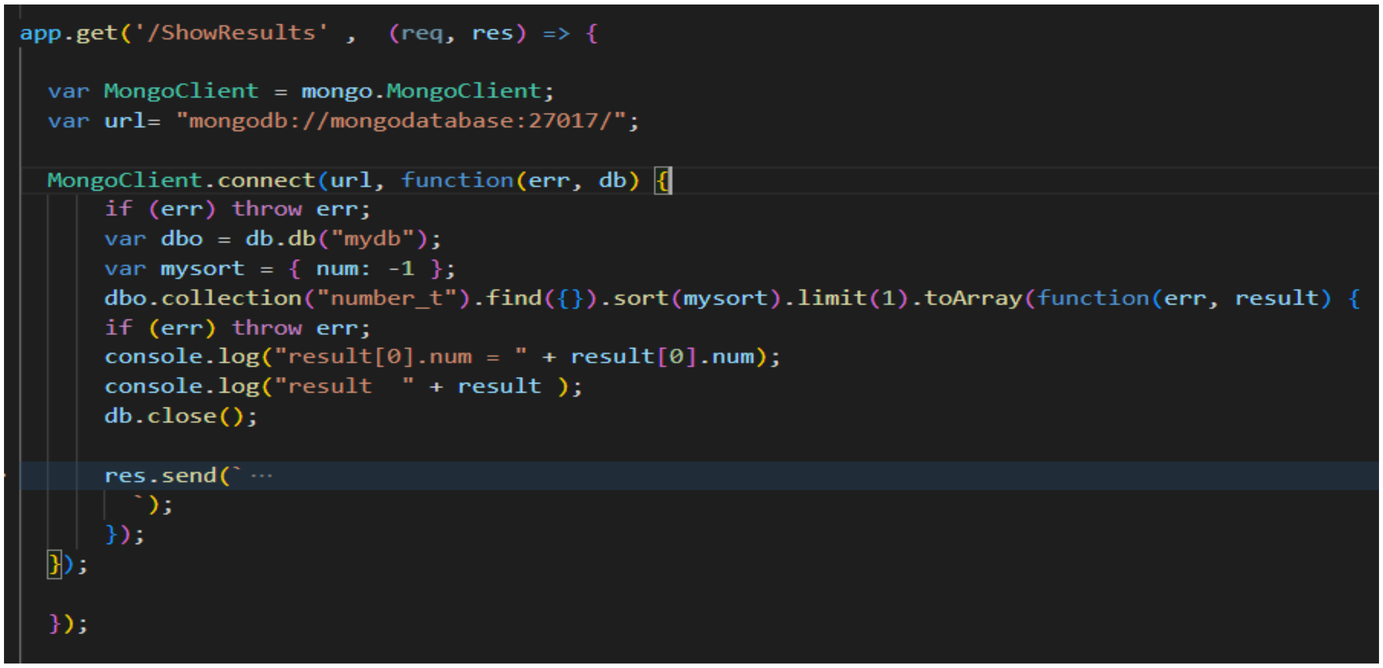
A container that stores the result that the analysis container sends.



```
mongodatabase:
  container_name: mongodatabase
  image: mongo:latest
  restart: always
  volumes:
    - ./mongo_db:/data/db
  ports:
    - "27017:27017"
```

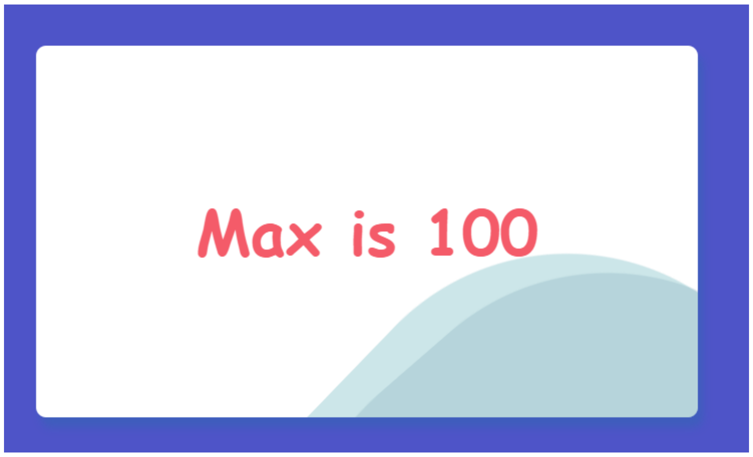
Container Show Results:

The user enters has name and password in a page in this container, then the container send them to Authentication Service to verify the user’s identity.



```
app.get('/ShowResults' ,  (req, res) => {

  var MongoClient = mongo.MongoClient;
  var url= "mongodb://mongodatabase:27017/";

  MongoClient.connect(url, function(err, db) {
      if (err) throw err;
      var dbo = db.db("mydb");
      var mysort = { num: -1 };
      dbo.collection("number_t").find({}).sort(mysort).limit(1).toArray(function(err, result) {
      if (err) throw err;
      console.log("result[0].num = " + result[0].num);
      console.log("result  " + result );
      db.close();

      res.send(` ···
        `);
      });
  });

  });
```

After verification, the user will be redirected to show results page.



Max is 100