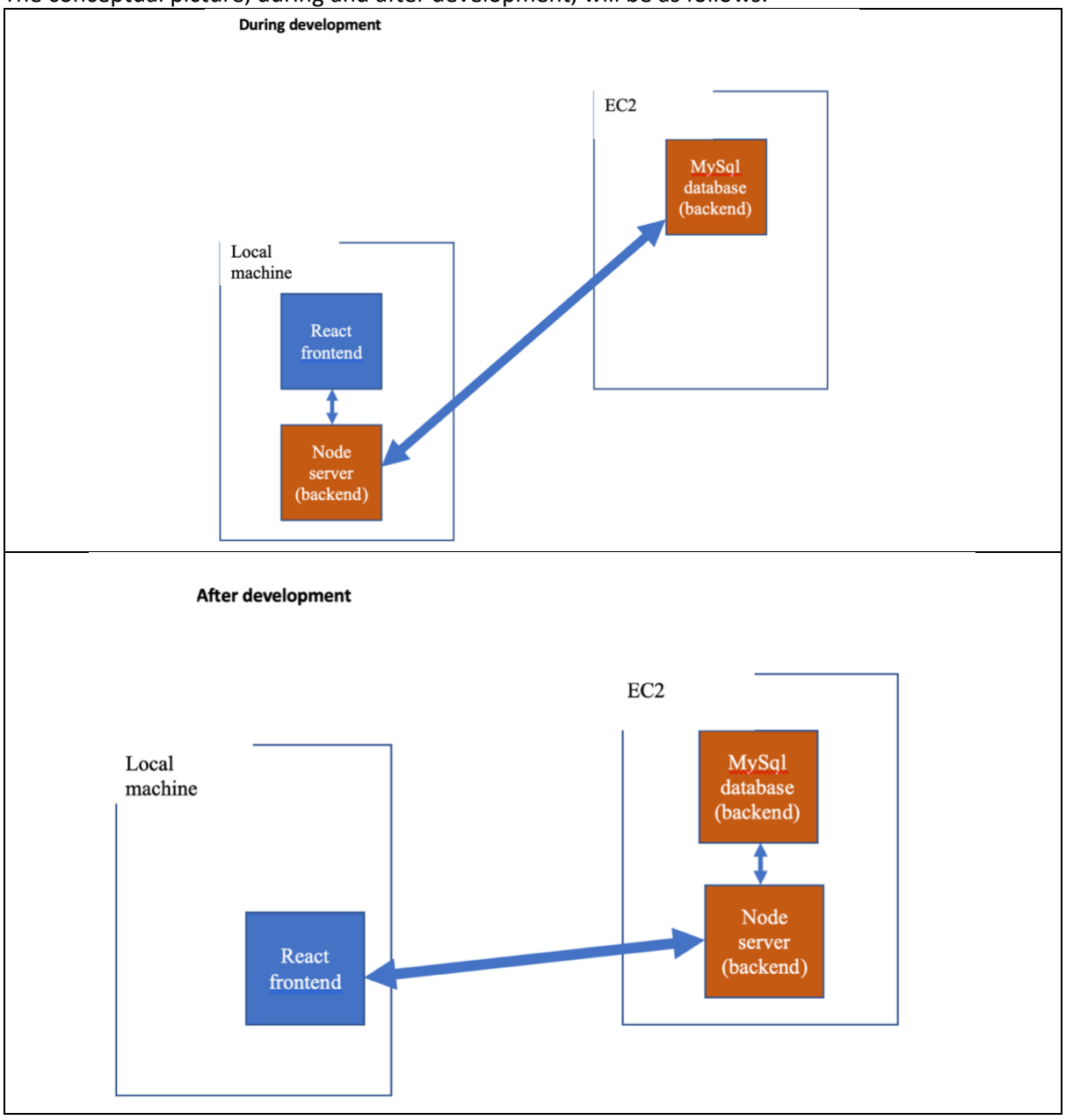
Part 5

MySQL with Node.js / React

In this guide we will do the following:

Step 1 Set up a MySQL database on ec2 and create a user account
Step 2 Connect and make queries to the database from a development Node server (backend)
Step 3 Query the backend server from React frontend for database access

The conceptual picture, during and after development, will be as follows:

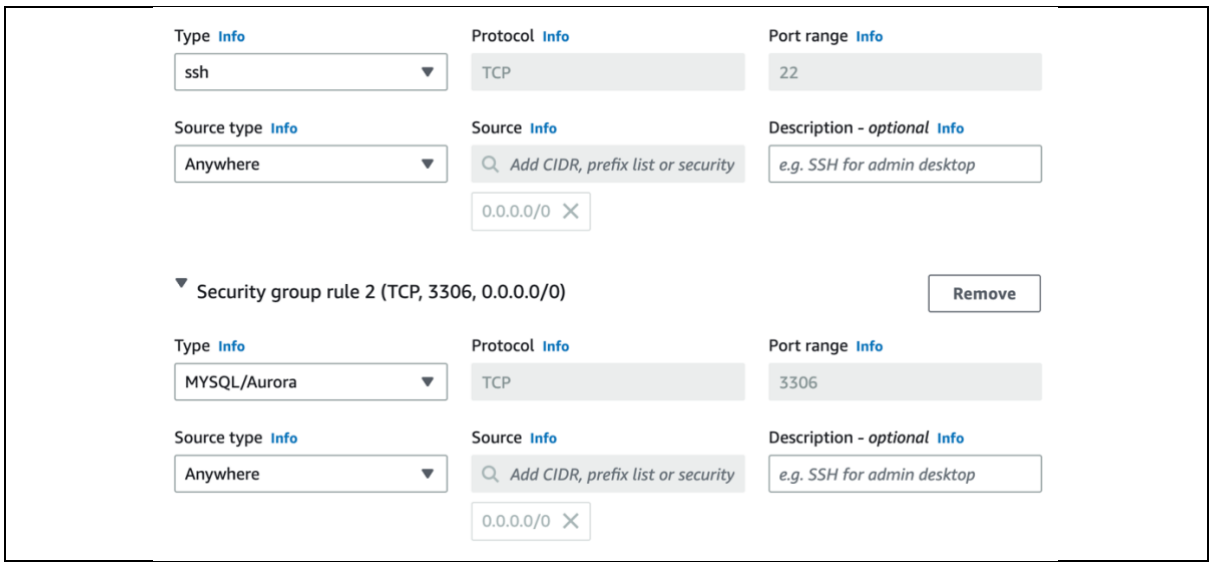


Sarim Baig
May 2022

Step 1 Set up a MySQL database on ec2 and create a user account

We're running Ubuntu 22.04 on t2.micro.

When instantiating the ec2 instance, add at least two security group rules as follows (note: this is for the development phase, the backend server is on the local machine in this case and only connecting to ec2 for mysql.)



ssh into your ec2 instance.

Run the following commands:

~$ sudo apt-get update

~$ sudo apt install mariadb-server -y

~$ sudo systemctl start mariadb

~$ sudo systemctl enable mariadb

(I have chosen empty password for root)

MySQL is the largest open-source database community. MariaDB is a fork from MySQL and is 100% compatible with prior versions of MySQL.

The enable command makes sure that the db keeps running when you restart ec2.

~$ sudo mysql -u root -p

You have entered the RDBMS as root user.

We would like to create a user (other than root) who can access a database on the RDBSM remotely from Node.js code.

Run the following SQL commands to create the user:
(you may write a username and password of your own choice)

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'usrpwd';

CREATE USER 'username'@'%' IDENTIFIED BY 'usrpwd';
```

'username'@'%' allows user called username to connect to mysql database from any client host using the password usrpwd.

Let's also create a database and add a table with a row of data in it.

```
CREATE DATABASE Persondb;
USE PersonDB;

CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);

INSERT INTO Persons VALUES (1, 'Woo', 'John', 'Imperial College London',
'London');
```

Let's grant the user called username privileges to use any database:

```
GRANT ALL PRIVILEGES ON *.* to 'username'@'%';

exit
```

Finally, but importantly, we need to make a change in one of the configuration files to allow the backend_server on the local machine to connect with MySQL on ec2. To have MySQL listen on all interfaces, we need to set the attribute bind-address to 0.0.0.0

In my case, this attribute was found in /etc/mysql/mariadb.conf.d/50-server.cnf

Change to the home directory:

~$ cd ..

Now run the following command to find the file in /etc/:

~$ sudo grep -rnw '/etc/' -e 'bind-address'

Assuming the attribute was found in /etc/mysql/mariadb.conf.d/50-server.cnf

~$ sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf

Change **from** bind-address = 127.0.0.1 **to** bind-address = 0.0.0.0

Save the file.

Restart the RDBMS:

~$ sudo systemctl restart mariadb

Step 2 Connect and make queries to the database from a development Node server (backend)

To start with, add the following code to index.js (backend_server)

In host add the IP address of the EC2 instance.

user and password should be the username and password you created in the database above.

```
var mysql = require('mysql');

var con = mysql.createConnection(
{
  host: "44.202.243.84",
  user: "username",
  password: "usrpwd",
  database: "Personsdb"
});

con.connect(function(err) {
  if (err) throw err;
```

```
  console.log("Successfully connected to the database...\n");
});
```

We'll now add an end point that fetches all data from the Persons table. Add the following code to index.js somewhere before app.get('*', …

```
app.get("/personQuery", (req, res) => {
   con.query("SELECT * FROM Persons", function (err, result, fields) {
     if (err) throw err;
     res.json(result)
   });
 });
```

Run the server, and test is by entering http://localhost:3001/personQuery in the browser.

Step 3 Query the backend server from React frontend for database access

Queries originate at the frontend. The frontend should *fetch* appropriate endpoint at the server to satisfy the query. We'll add some code to App.js to test this process. For this step, once again we're running two development servers, at port 3000 for the react app and at 3001 for the backend.

In App.js, replace the code in useEffect with the following:

```
React.useEffect(() => {
   ///See CORS
   fetch("http://localhost:3001/personQuery/")
     .then((res) => res.json())
     .then((data) => alert(JSON.stringify(data)))
     .catch((err) => alert(err)
      );
 }, []);
```

This is simply displaying the returned data in a dialog bog.

You can use the post method with fetch to send query related data to the server, for example, for tasks such as fetching a specific user's data from the database.

Obviously, this fetching code can be added to any function, not just useEffect.

*