**Employee Management is the application which enables you to add new employee or delete, update, search existing employees.**

* **For Backend: https://github.com/BobbySingla/EmployeManagement.git**

**Tech Stack Used:** Spring Rest Services, Spring JPA, Spring Actuator, Prometheus Library (which helps to monitor metric data of Spring Boot apllication), Database: MySql.

* **For Frontend: https://github.com/BobbySingla/EmployeeManagementApp.git**

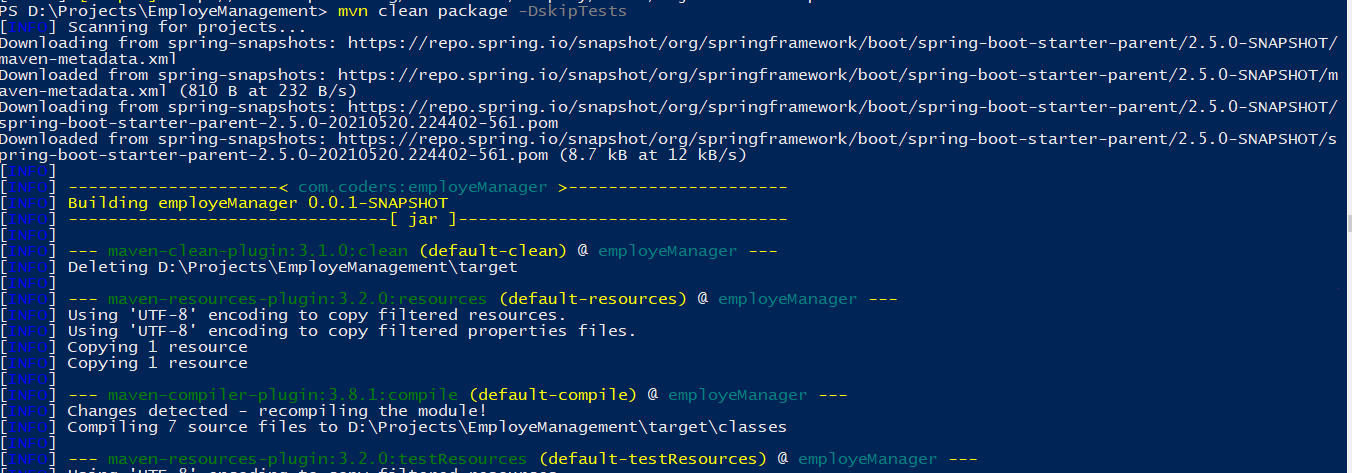
**Tech Stack Used**: HTML, CSS, BootStrap, JavaScript, TypeScript, Angular

* **Docker** is used to expose ports of backend and frontend applications to each other, so that applications can communicate to each other.

**Prerequisites:** java 16, docker, maven, node.js(npm), typescript and angular must be installed in your system.

1. **Setup guide for backend Application:**
2. git clone <https://github.com/BobbySingla/EmployeManagement.git>
3. Open cmd and locate to root directory of the github downloaded project and execute following command to build the jar of backend spring boot application.

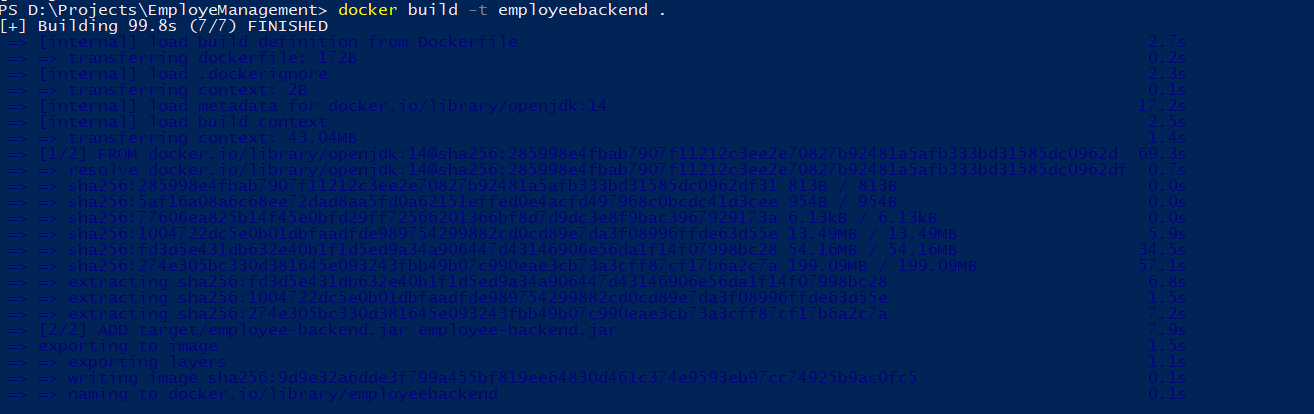
**Mvn clean package –DskipTests**





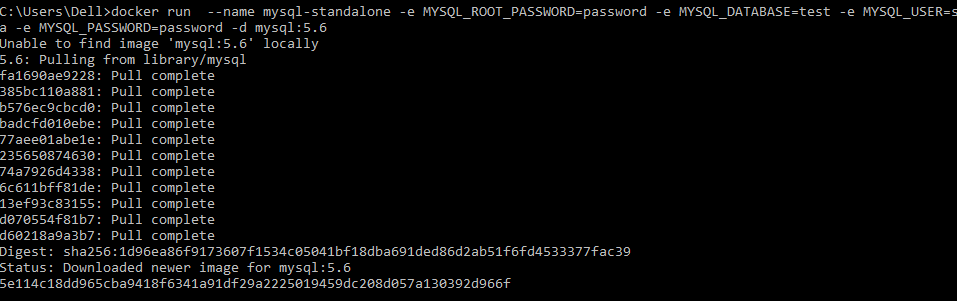
1. Build the docker image of the application using Docker file. Execute cmd command :

**docker build –t employeebackend .**



1. Download mysql docker images, run it as container and set its required parameters.

**Docker run --name mysql-standalone –e MYSQL\_ROOT\_PASSWORD=password –e MYSQL\_DATABASE=test –e MYSQL\_USER=sa –e MYSQL\_PASSWORD=password –d mysql:5.6**



1. Link Docker image with of backend application with MySql docker container.

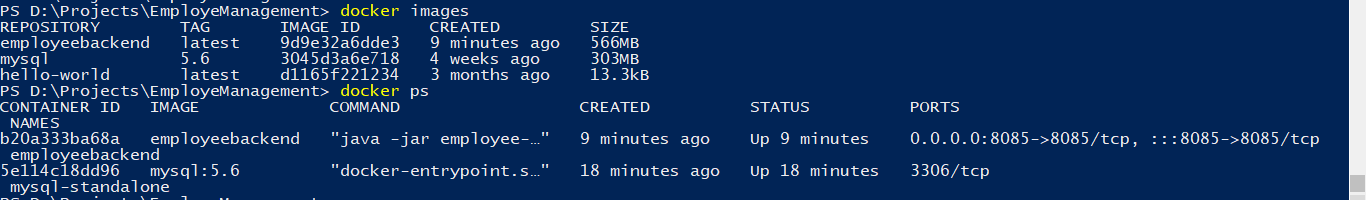
**Docker run –d –p 8085:8085 –name employeebackend --link mysql-standalone employeebackend**



* **The metric data of Spring actuator will be produced at endpoint :**

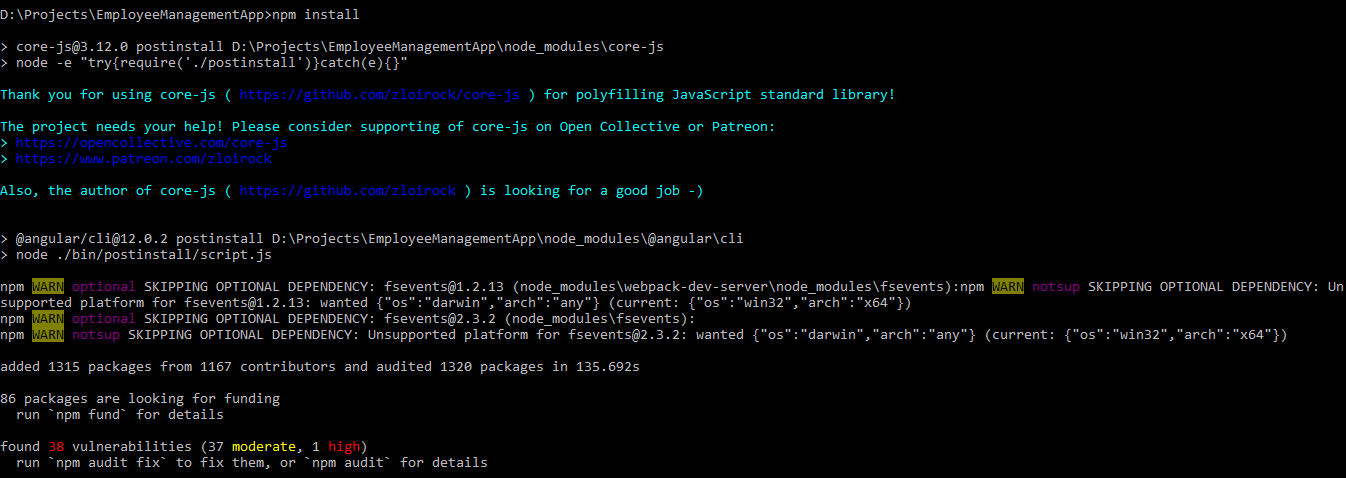
[**http://localhost:8085/actuator/prometheus**](http://localhost:8085/actuator/prometheus)

1. **To check docker images and containers are up:**



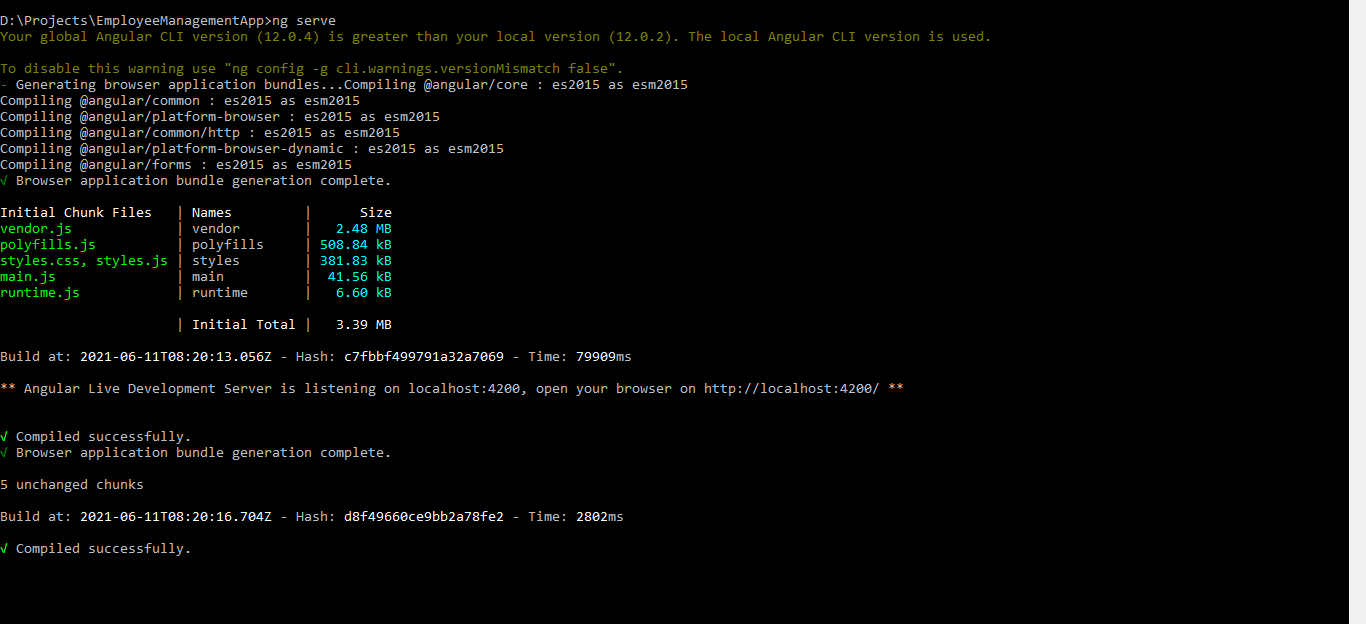
1. **Setup guide for FrontEnd Angular Application:**
2. Git clone : <https://github.com/BobbySingla/EmployeeManagementApp.git>
3. Open cmd and locate to root directory of the github downloaded project and execute following command to serve frontend angular Application

**npm install - used to download all required third party packages**

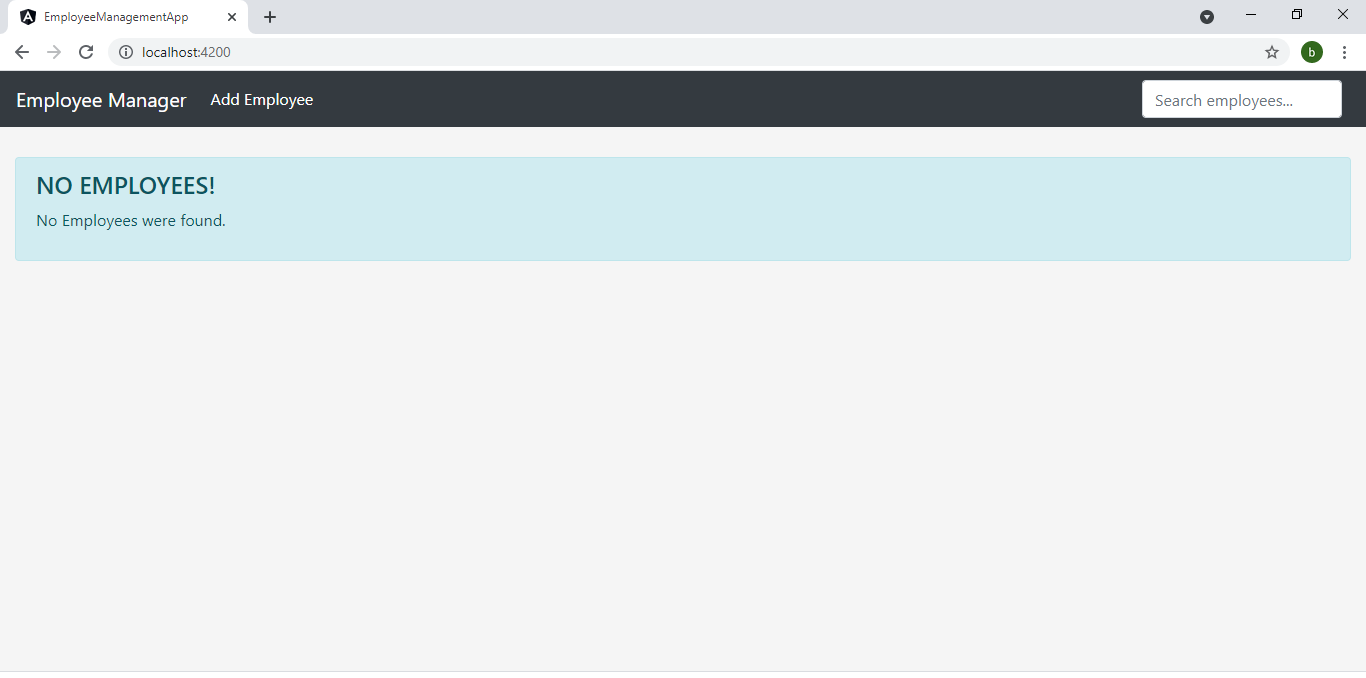


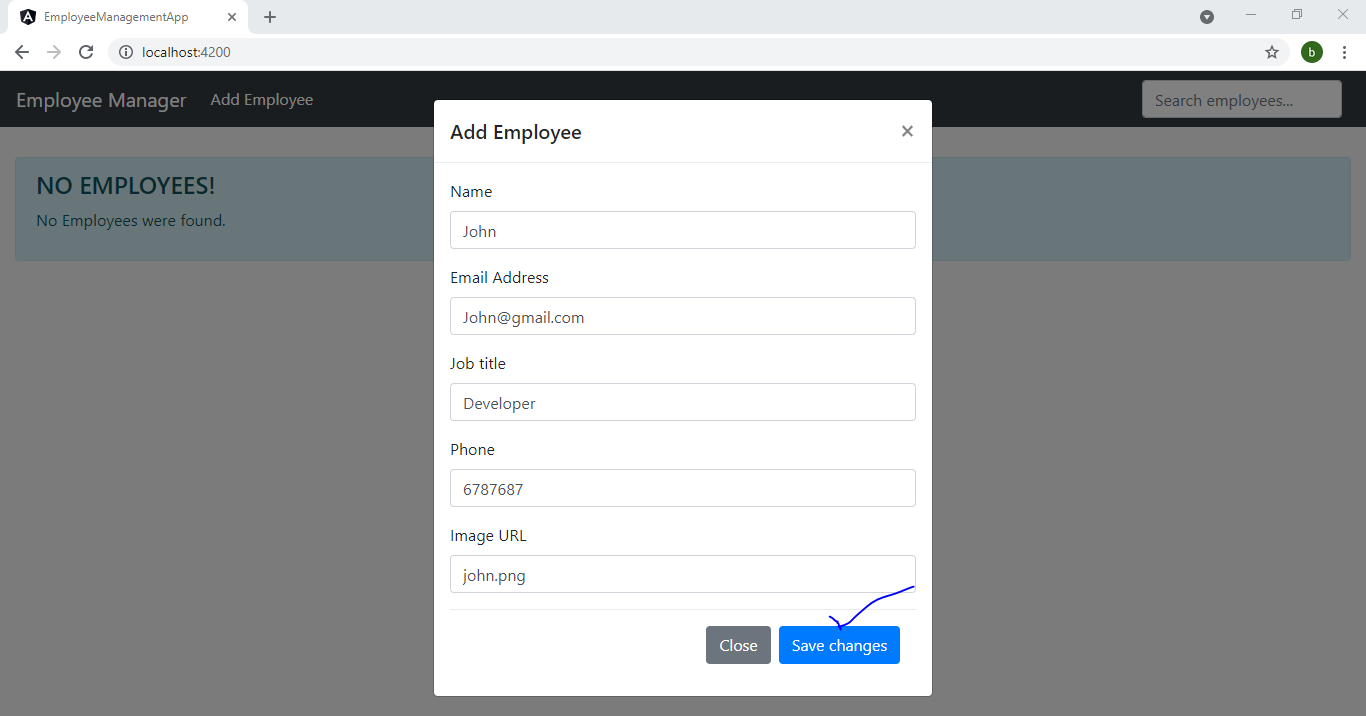
1. Execute following command to server the application on URL: ttp://localhost:4200/

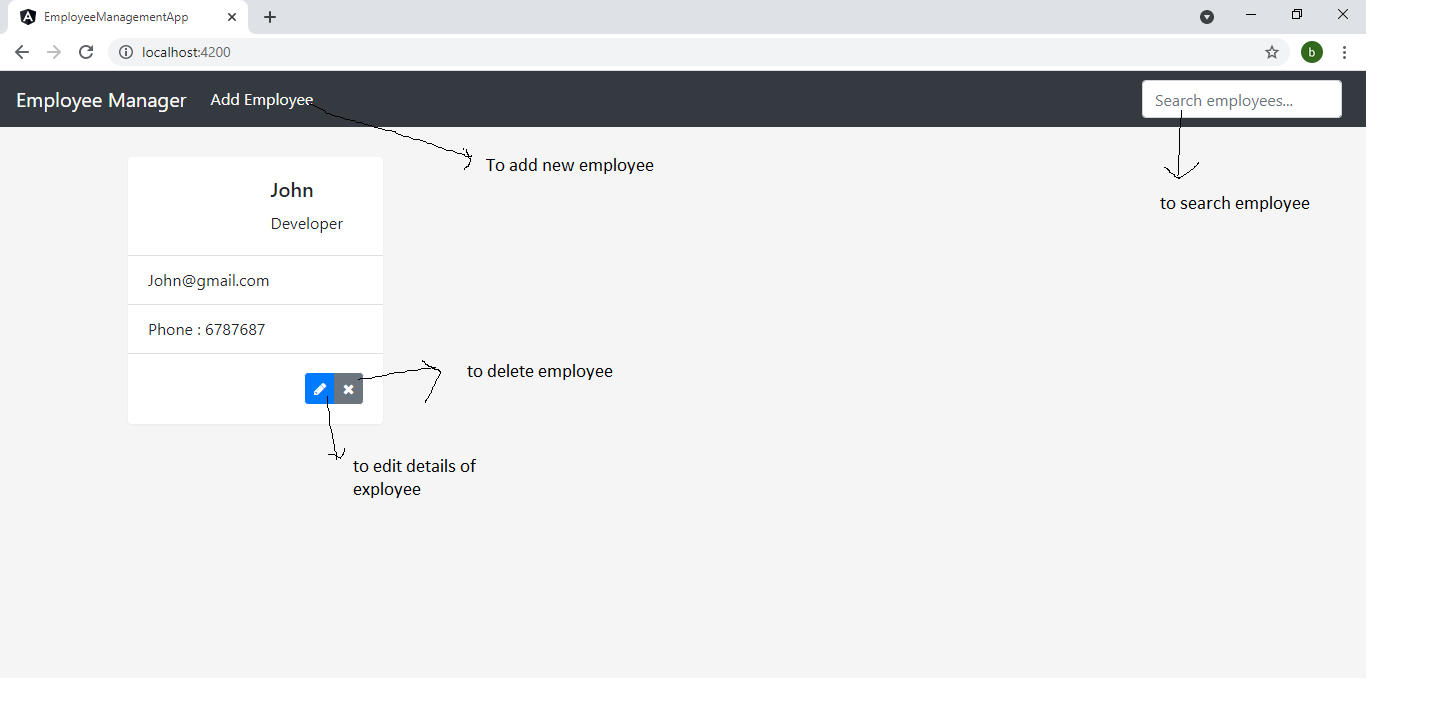
**ng serve**



Employee Management Application UI(Screenshots):

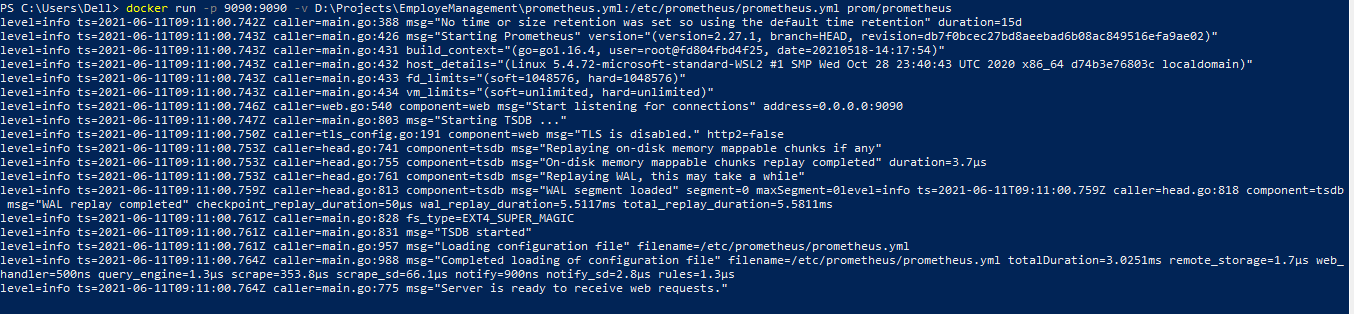




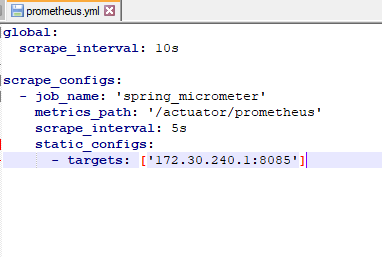


1. **How to generate metric graphs of Spring Boot Apllication using Prometheus tool:**
2. Download docker image of Prometheus and run it is container on port 9090 using following command:

**docker run -p 9090:9090 -v (your local location of Prometheus.yml file):/etc/prometheus/prometheus.yml prom/Prometheus**

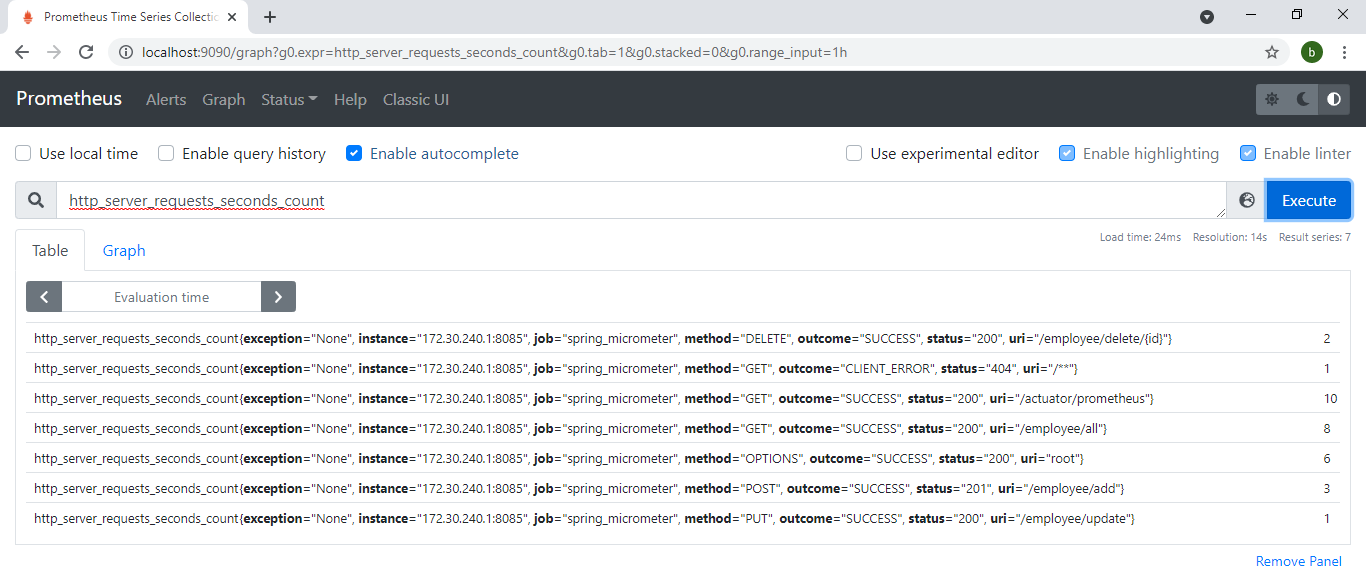


**Prometheus.yml file:** This file helps expose all metric data of spring boot backend application to Prometheus tool.



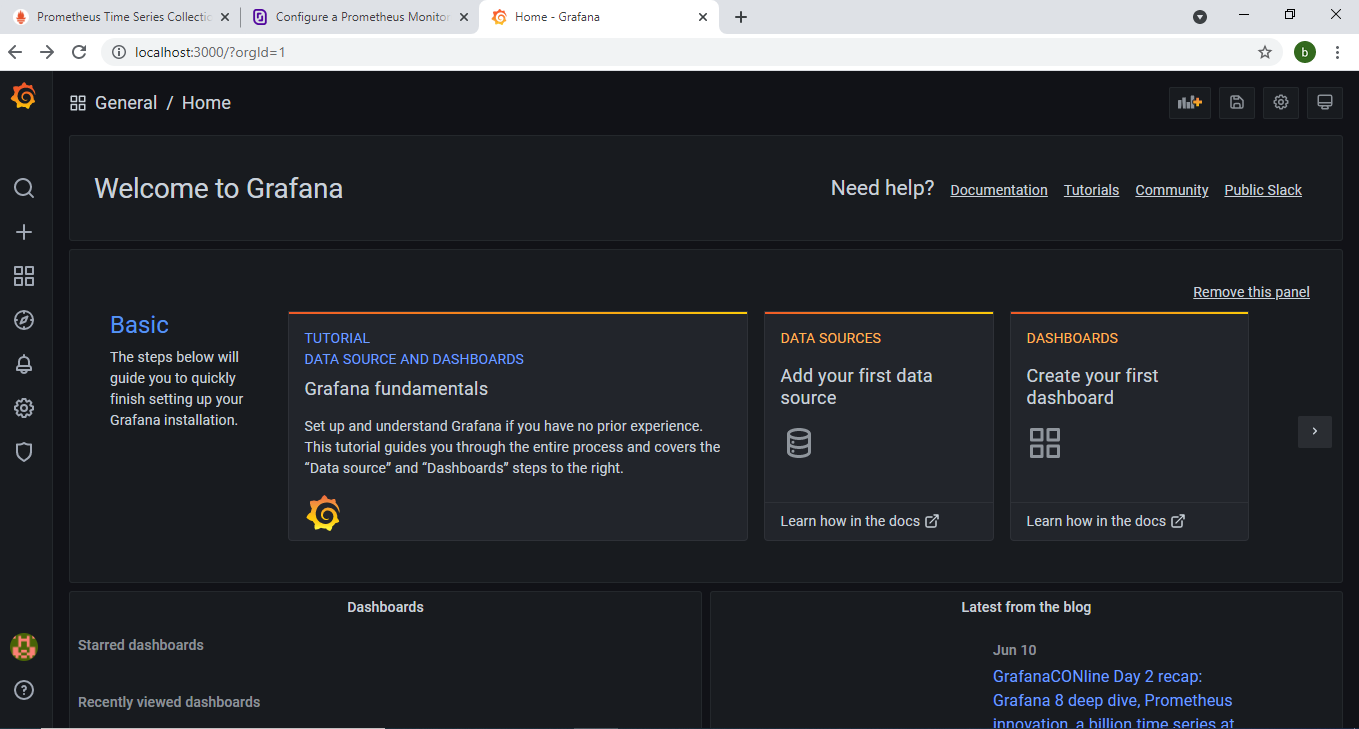
//172.30.240.1 Your local system IP address

//8085 port on which spring backend application is running



1. **How to setup Grafana with Prometheus:**
2. Run docker command to download grafana image and run its container on port 3000.

**docker run -d -p 3000:3000 grafana/grafana**



**Use ID/password: {admin/admin}**

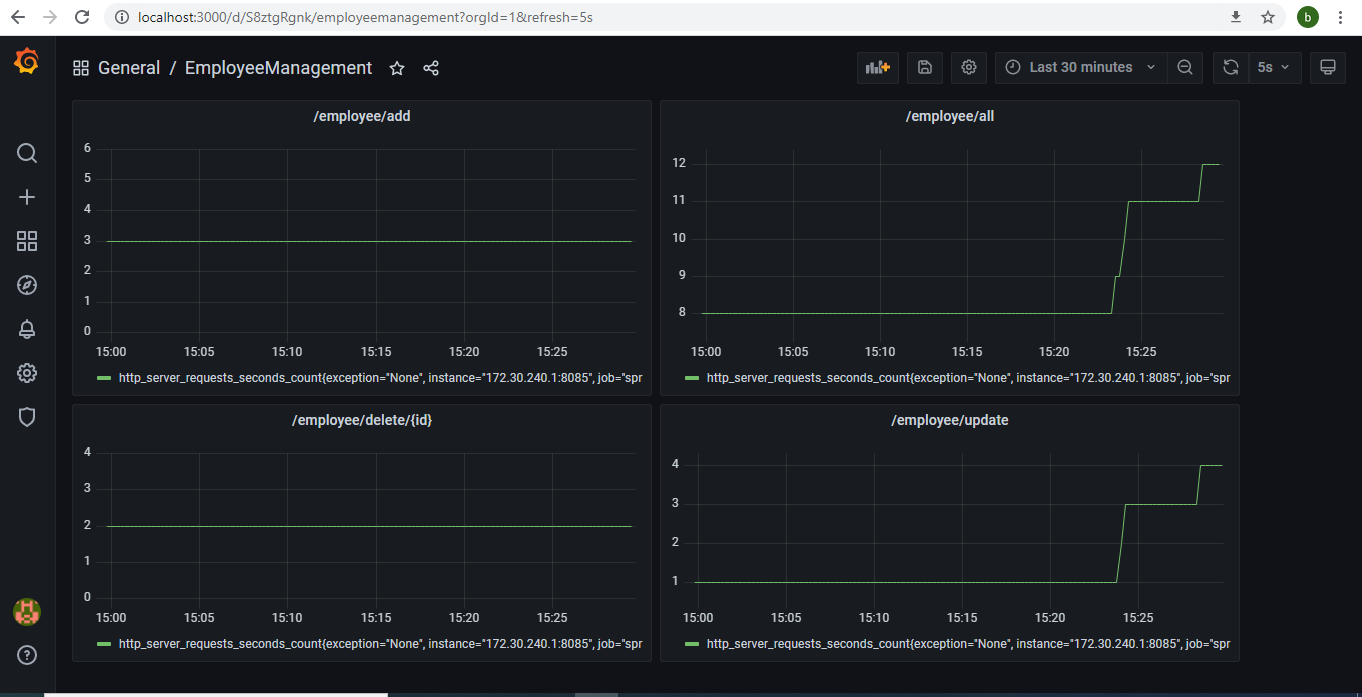
* **Setup Prometheus with Grafana:**

(<https://prometheus.io/docs/visualization/grafana/>)

1. To create a Prometheus data source in Grafana:
2. Click on the "cogwheel" in the sidebar to open the Configuration menu.
3. Click on "Data Sources".
4. Click on "Add data source".
5. Select "Prometheus" as the type.
6. Set the appropriate Prometheus server URL (for example, http://localhost:9090/)
7. Adjust other data source settings as desired (for example, choosing the right Access method).
8. Click "Save & Test" to save the new data source.

* Start creating custom dashboards as per requirement:

Sample:



**Useful Docker commands:**

docker images - to list all docker images

docker logs <container\_name> - to print logs of specific container

docker ps -a - to list all conatainers(running & stopped containers)

docker ps - to list all running containers