**Application Deployment**

**(Deployed the react app application to a production ready state)**

**Application:**

Cloned the Github Repository using git clone command.

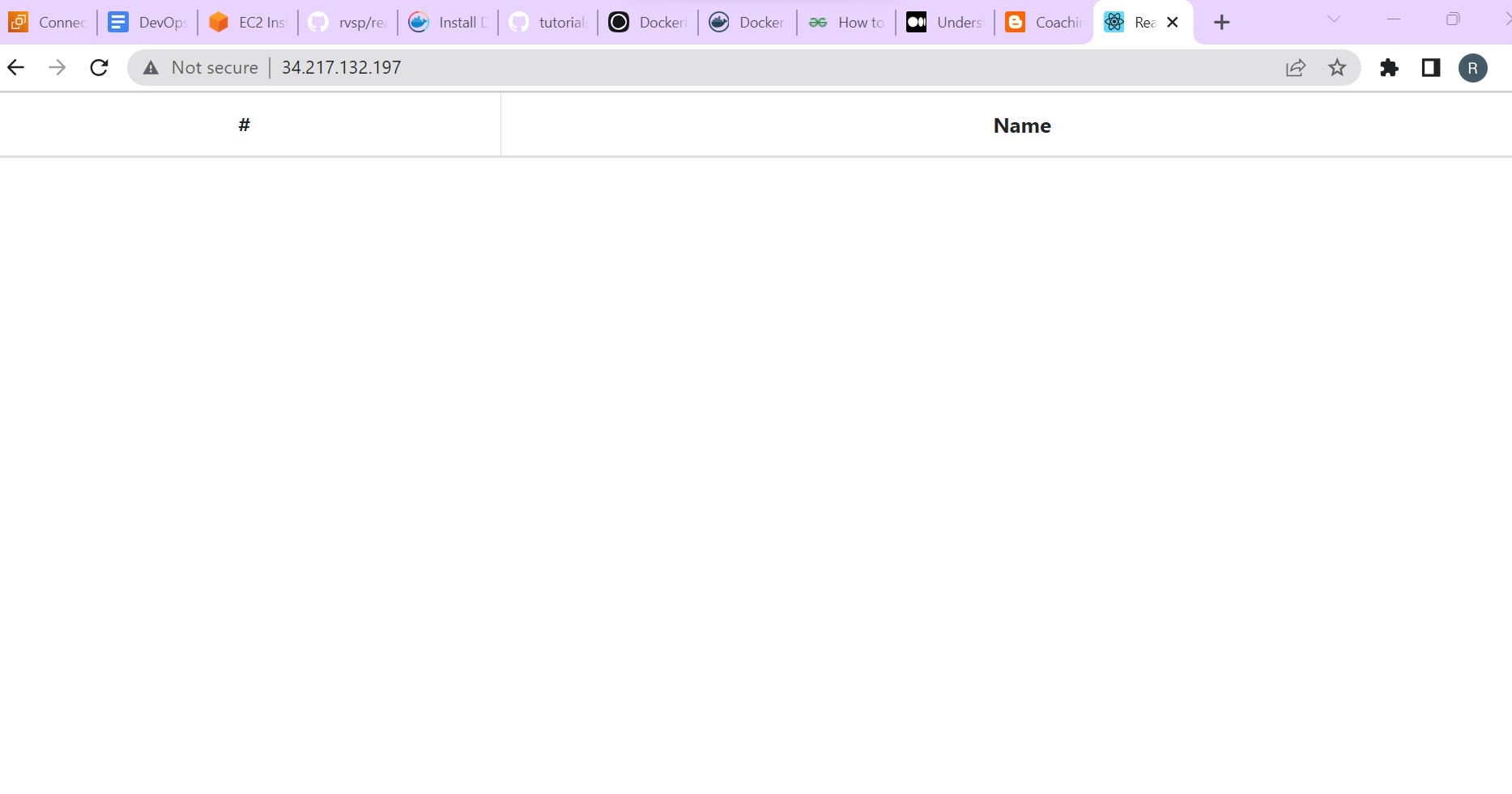
**Docker:**

Firstly, I have created a EC2 instance and installed the Docker on top of it.

[To Dockerize a React app, I have created three files.](https://www.simplilearn.com/tutorials/docker-tutorial/what-is-docker-react-know-to-dockerize-a-react-app) i.e

* **Dockerfile**: A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Docker can build images automatically by reading the instructions from a Dockerfile. The Dockerfile contains all the commands that are needed to build an image. For example, Node.js is needed for a React project.
* .**dockerignore**: This file specifies which files and directories should be ignored when building the Docker image.
* **docker-compose.yml:** This file is used to define and run multi-container Docker applications.
* To build the docker image the command is (docker build . -t dockerized-react .)
* Map the port 80 inside the container with 3000 on current host
  + docker run -p 80:3000 -d dockerized-react
  + start up your application by running
  + docker compose up

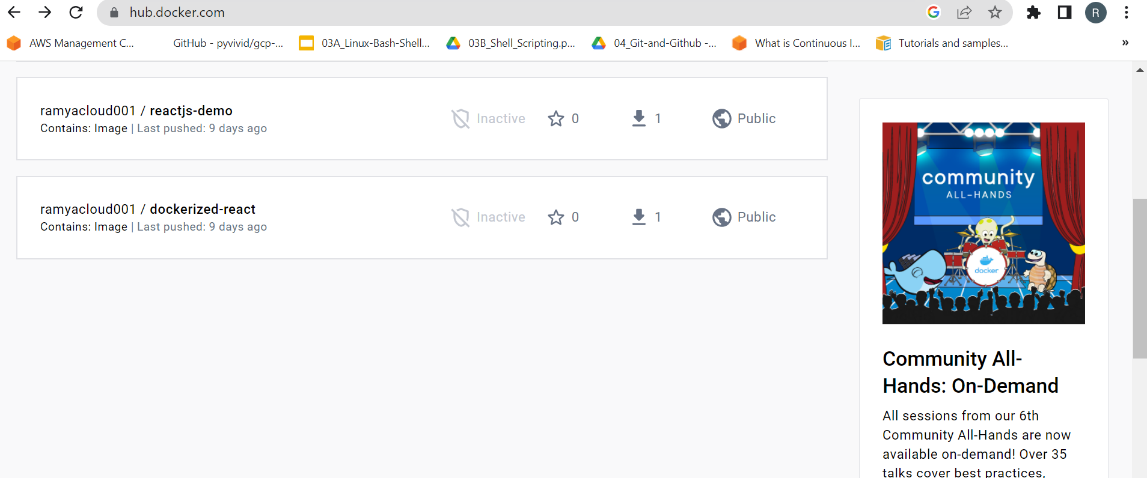
Please refer the **Deployed URL** in below image.



**Bash scripting**

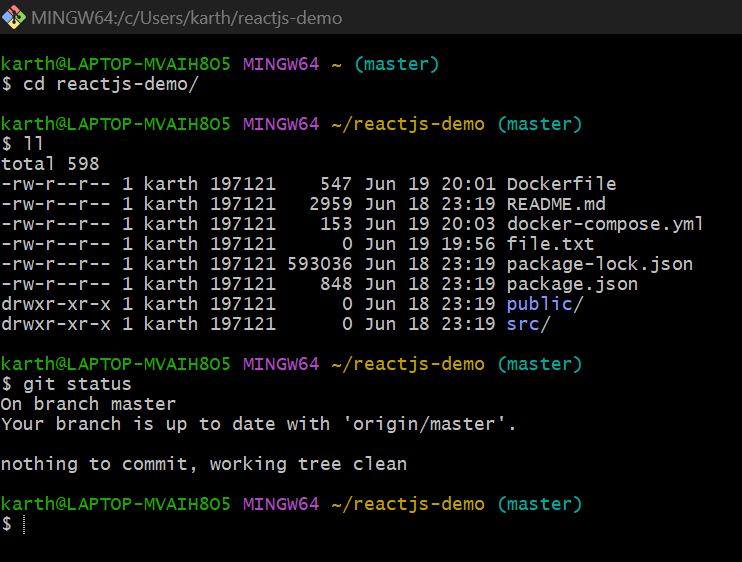
I have included the build.sh(For Building the docker images) and deploy.sh(For deploying the image to server) in a single bash file.

Created a bash script **deploy1.sh** for building and deploying docker images to server.



**Version control**

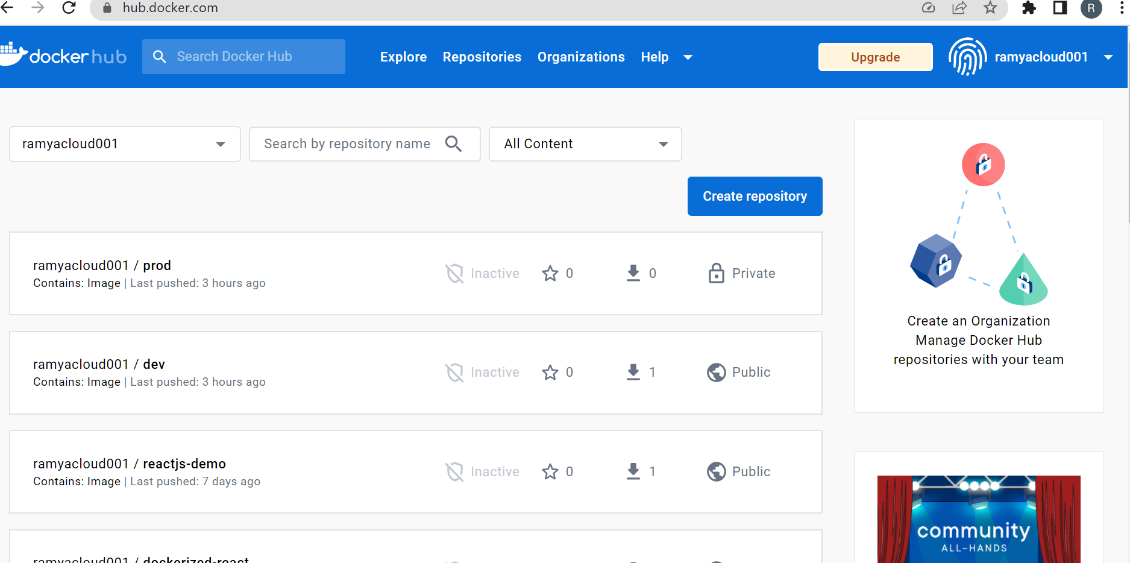
Used git bash to push the code to dev branch.

****

**Docker Hub**

Created two repositories dev and prod in my docker hub container registry(ramyacloud001) to push images.

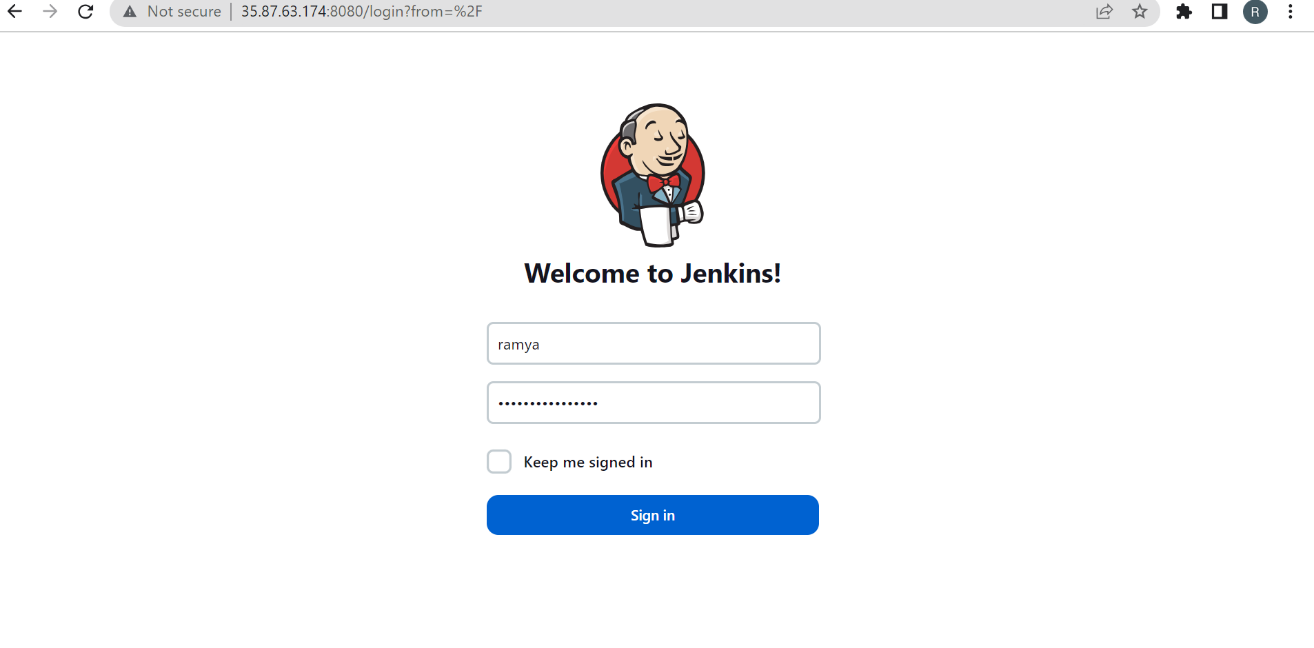
* Prod repo is in private.
* Dev repo is in public.



**Jenkins**

* Created a ec2 instance with Ubuntu AMI.
* Installed and configured Jenkins server for build and deploy the application.

Jenkins server login page



# Setting up the Jenkins CI/CD pipeline

Click on New Item, then enter a job name and select Pipeline from the options.

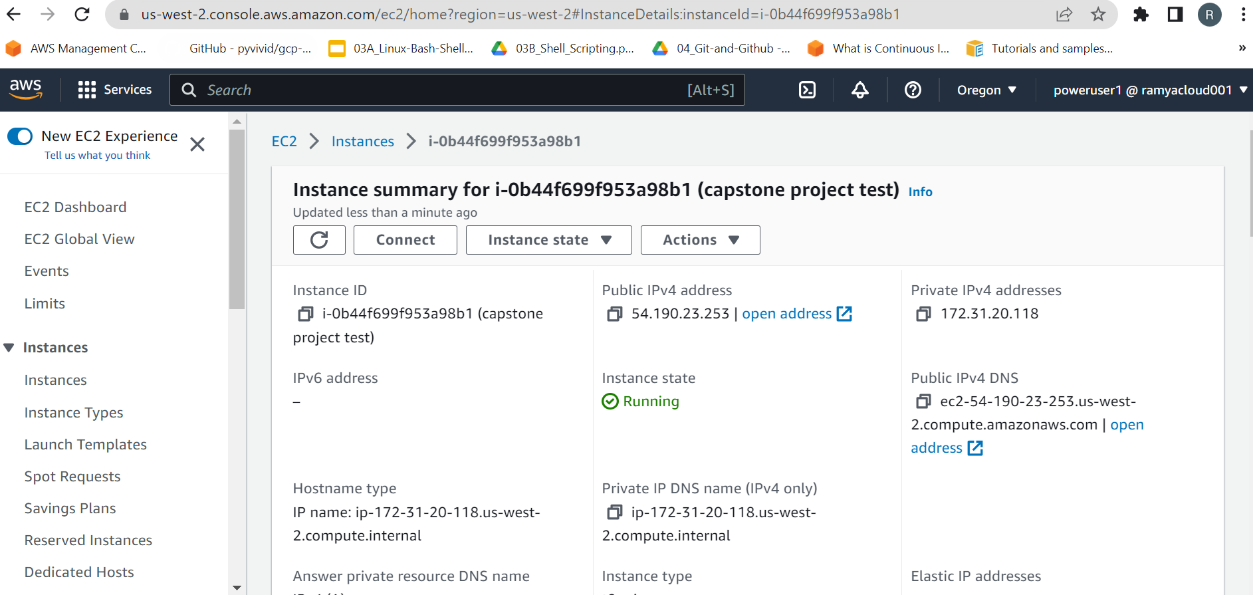
After creating the Pipeline, under Build Triggers tick the GitHub hook trigger for GIT Scm polling option. Under the Pipeline section, fill the repository URL and add the credentials if it is a private repository. Enter the branch which we want to build and also set the Script Path as Jenkins file. Finally, apply and save changes.

* Build Triggers: GitHub hook trigger for GITScm polling
* Pipeline: pipeline script from scm
* Scm : git
* Repositories: repository [url: github](url:github) URL
* Branch: master
* Script path: jenkinsv2
* Build now
* Docker image is pushed to prod repo in the docker hub.

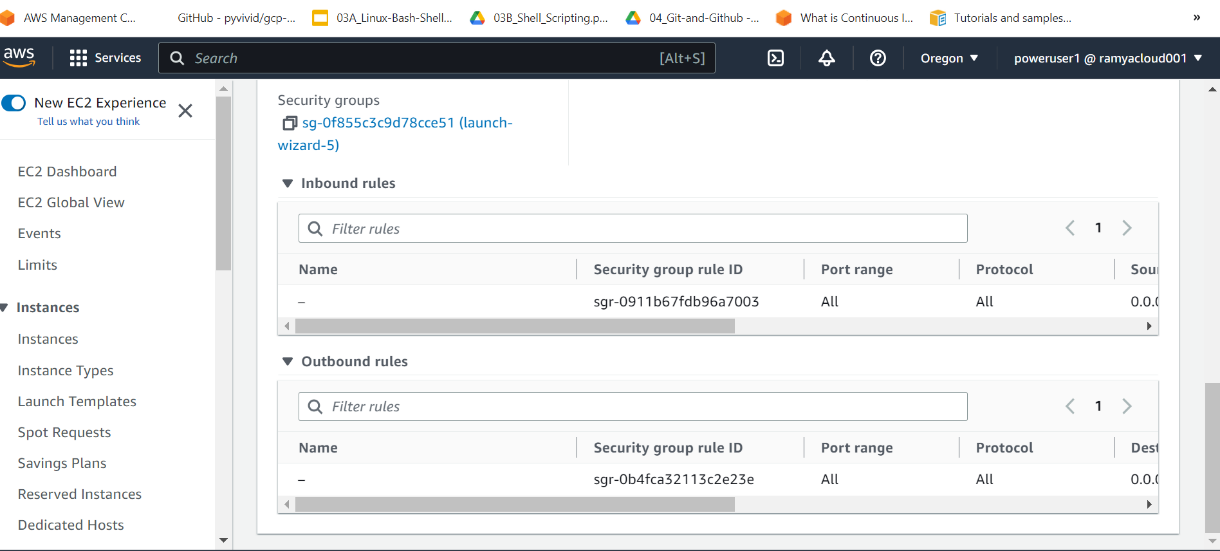
**AWS**

Created ec2 instance named capstone project test and security group with whoever has ip address can access the application.

**EC2 Instance**



**Security group.**



* **Created an amazon S3 bucket and deployed the react application.**

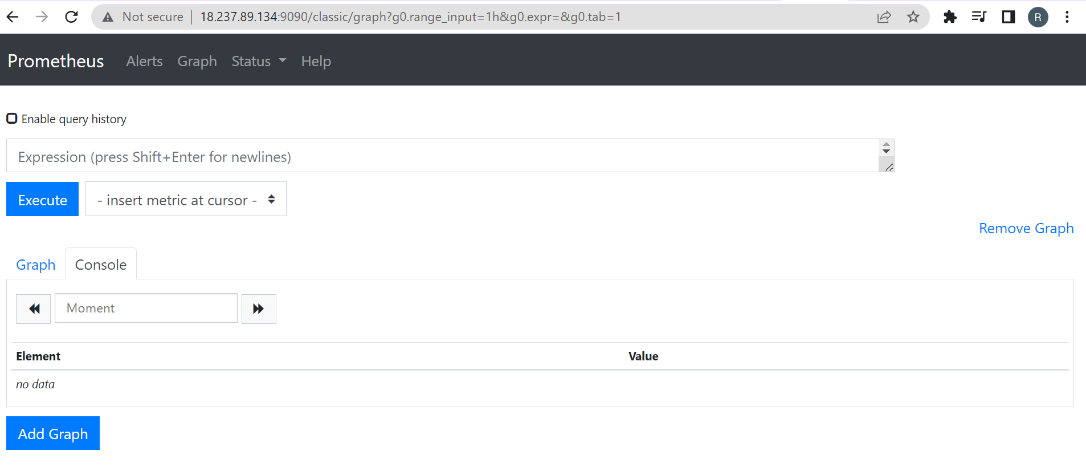
**Output of the React app in S3 bucket**



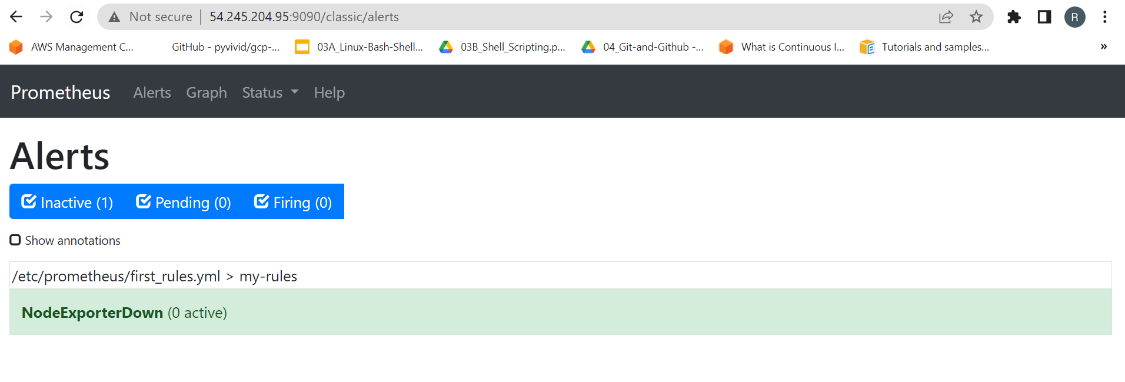
**Monitoring**

* Installed Prometheus in ec2 instance
* Gave the ip address of react app ec2 instance in prometheus.yml file.

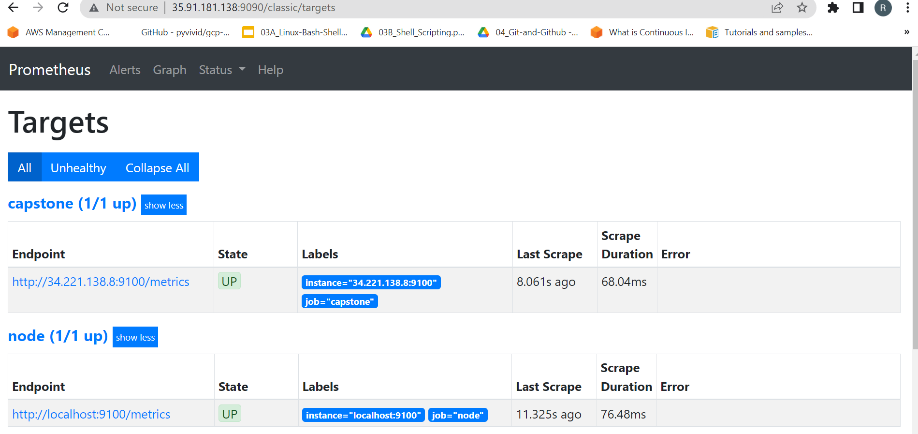
**Metrics of Prometheus**



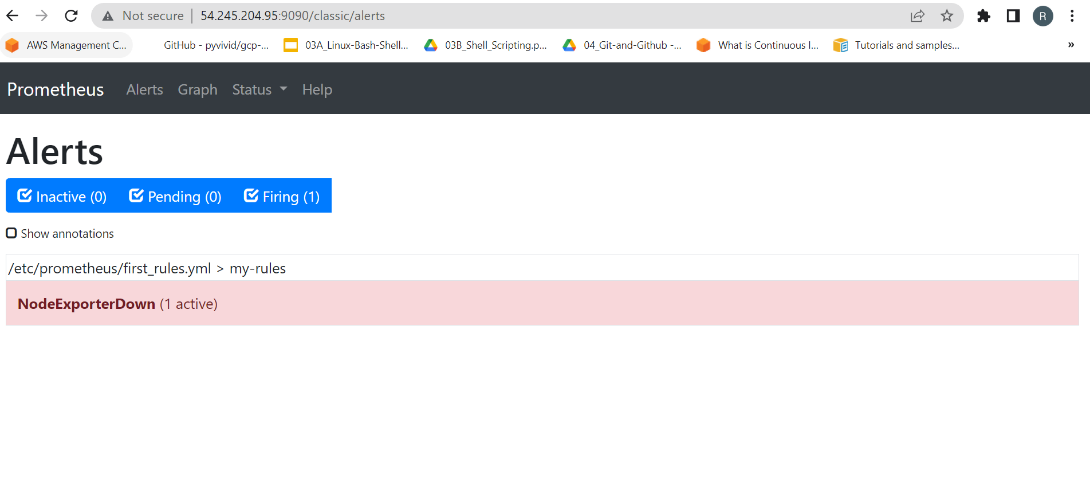
**Alerts in Prometheus**



**Targets in Prometheus**

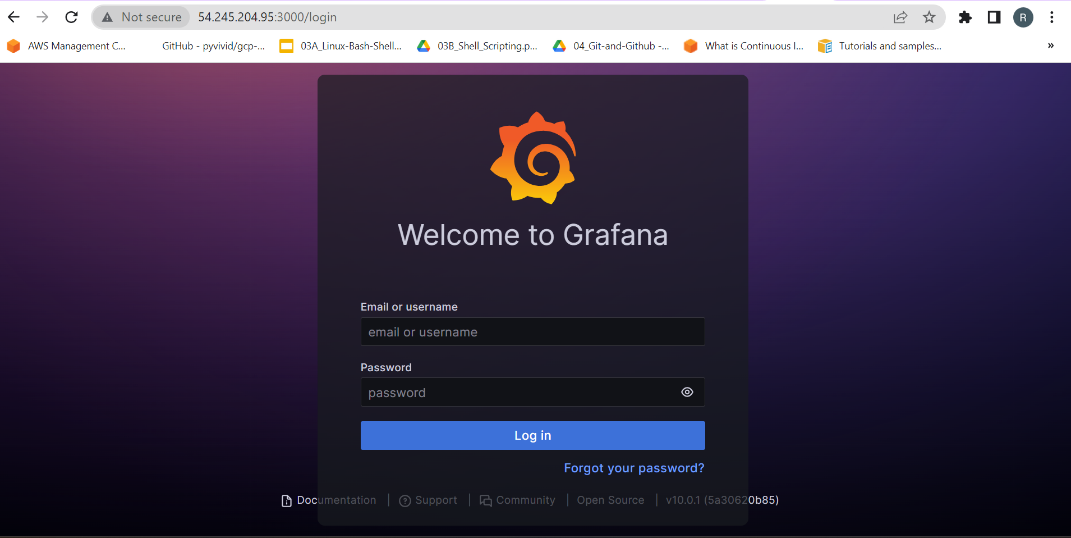


**Alert in prometheus when react app goes down**

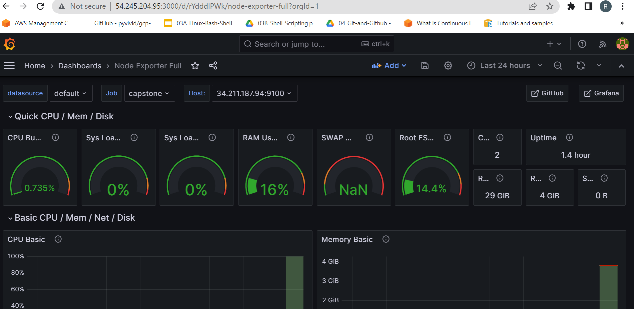
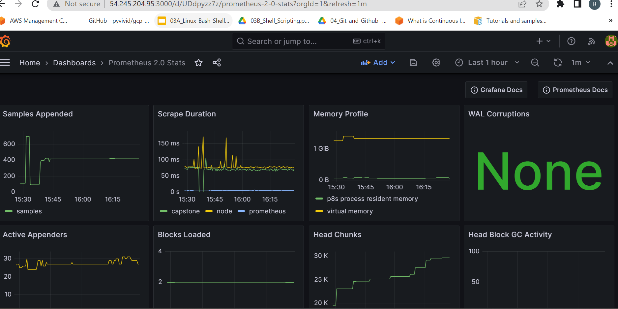


* Installed Grafana in ec2 instance connected with Prometheus.

**Login page of Grafana**



**Below are the Graphical representation image references in Grafana Monitoring tool.**

Submission :

Github link : <https://github.com/ramyacloud001/reactjs-demo>

Please refer the deployed URL under Docker(1st Image).

Docker Images tags screenshot:

