Application Deployment

**Clone the below mentioned repo and deploy the application. (Run the application in port 80 [HTTP] )**

**Repo URL :** [**https://github.com/rvsp/reactjs-demo**](https://github.com/rvsp/reactjs-demo)

Updated the Launched Ec2 instances and installed docker and docker-compose on the server

**Installed Docker on the server**

# Update the package index

sudo apt update

# Install prerequisites

sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

# Add the official Docker GPG key

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Add the Docker repository

echo "deb [signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update the package index again

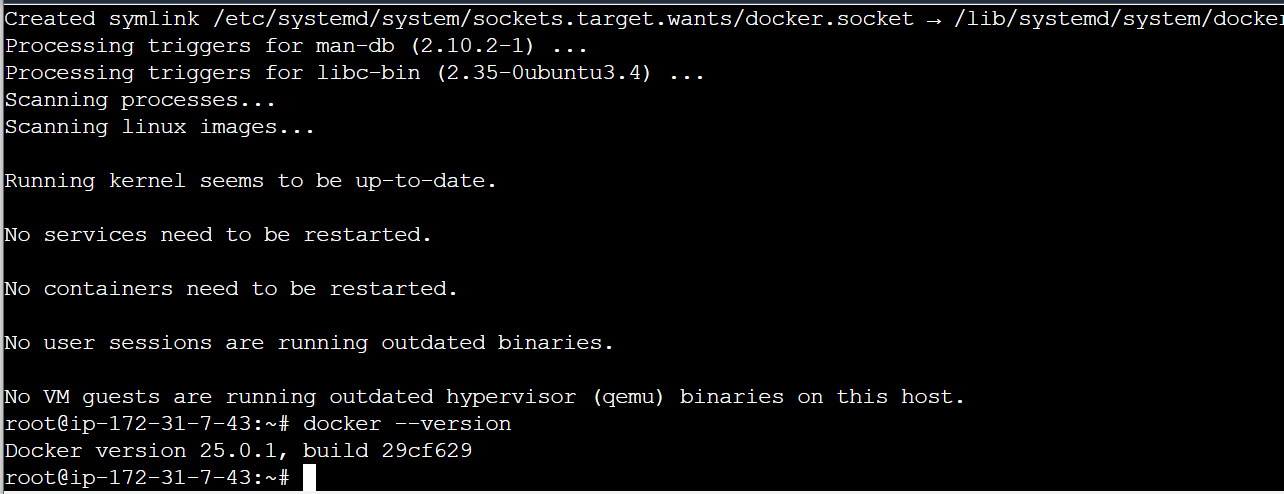
sudo apt update

# Install Docker

sudo apt install -y docker-ce docker-ce-cli containerd.io

#Checking Docker Version

docker --version



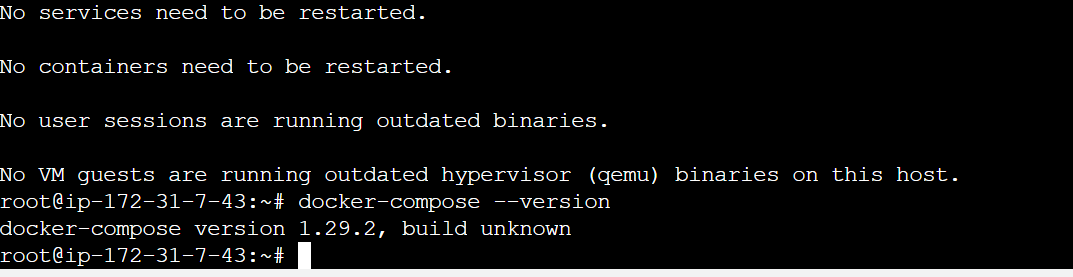
**Installing Docker-Compose**

#To install Docker-Compose

Sudo apt install docker-compose -y

# To check Docker Version

Docker-compose –version



**Cloning the Github repo to the server**

Cloning the repo for the developer code and cloned the repo and written the docker file and docker-compose file for creating an image.

**Docker:**

**-Dockerize the application by creating a Docker. Create a docker-compose file to use the above image**

**# Clone the repo**

**git clone https://github.com/rvsp/reactjs-demo**

**cd reactjs-demo**

**# Create Dockerfile**

**FROM node:14-alpine**

**WORKDIR /app**

**COPY . .**

**RUN npm install**

**RUN npm run build**

**EXPOSE 80**

**CMD ["npm", "start"]**

**# Create docker-compose.yml**

**version: '3'**

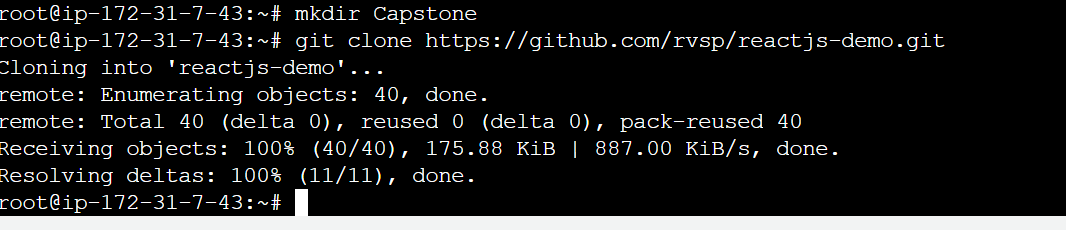
**services:**

**web:**

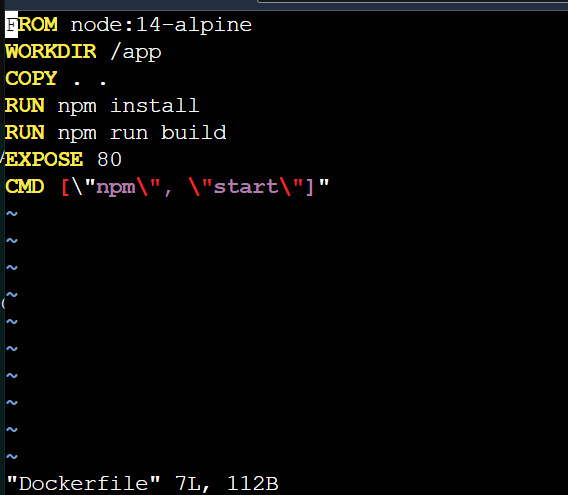
**build: .**

**ports:**

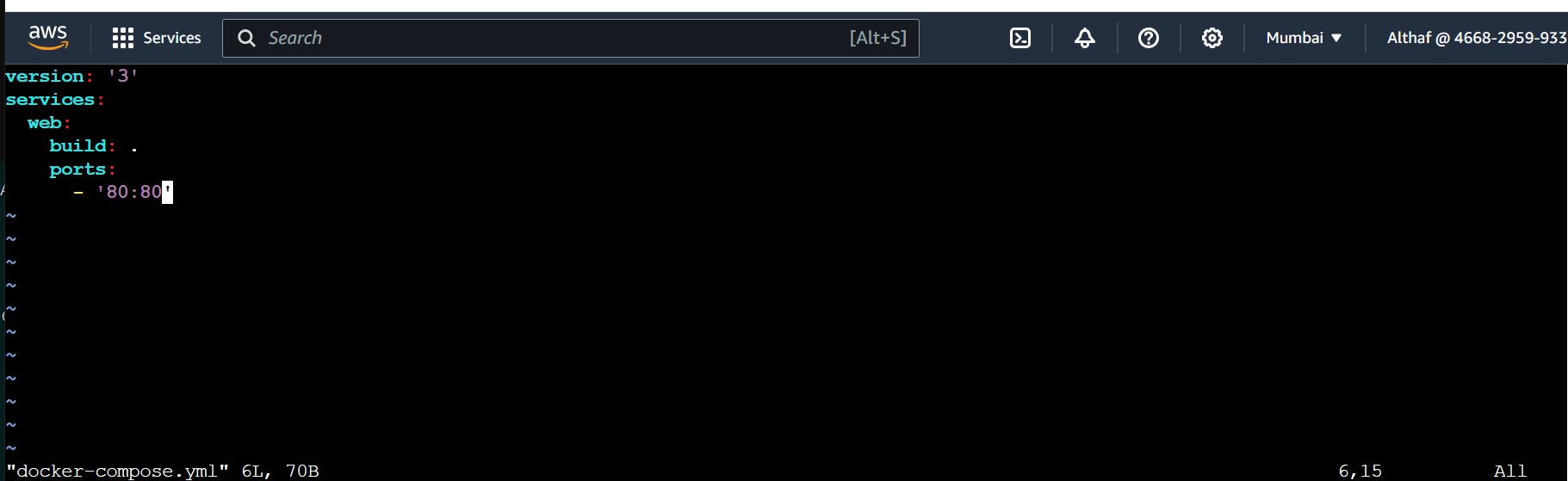
**- '80:80'**



**Docker File**



**Docker-Compose file**

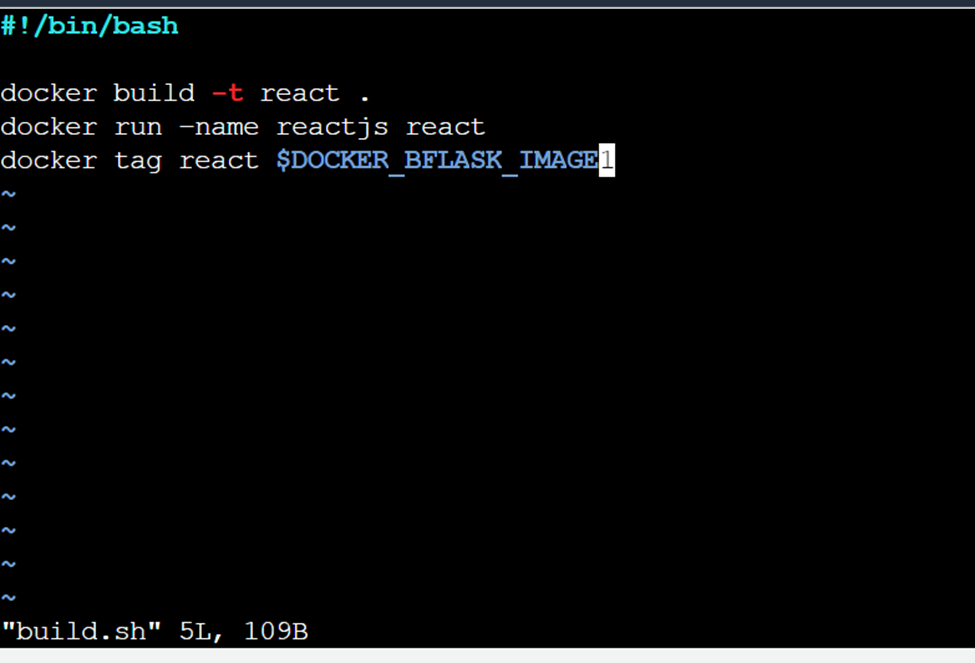


**Bash Scripting:**

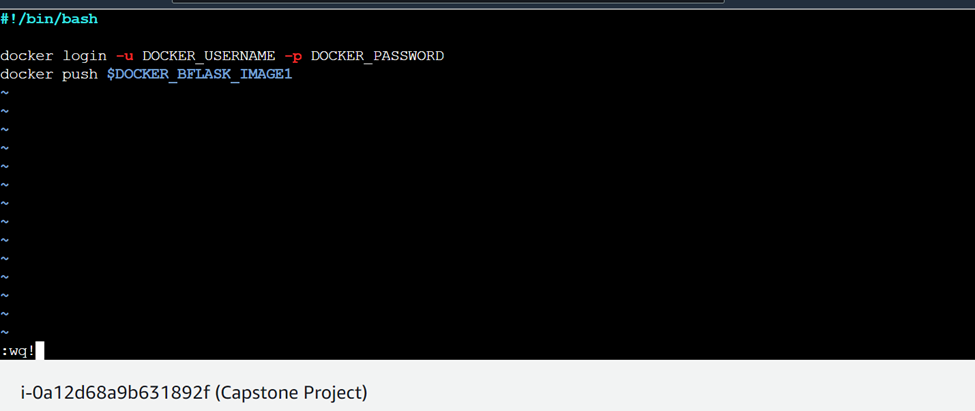
**Write 2 scripts**

**-build.sh —for building docker images Deploy.sh —for deploying the image to server**

**Build.sh**



**Deploy.sh**

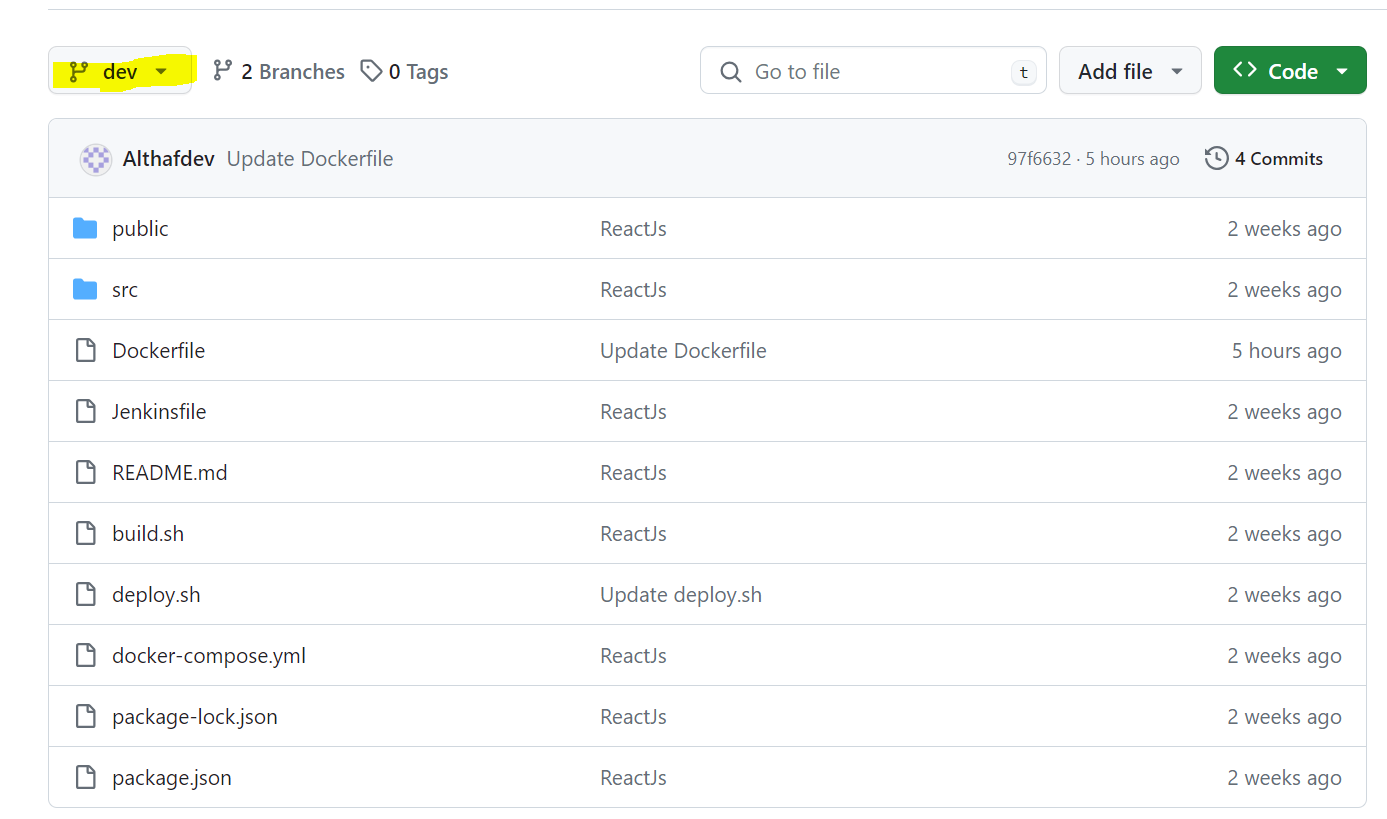


**Docker hub:**

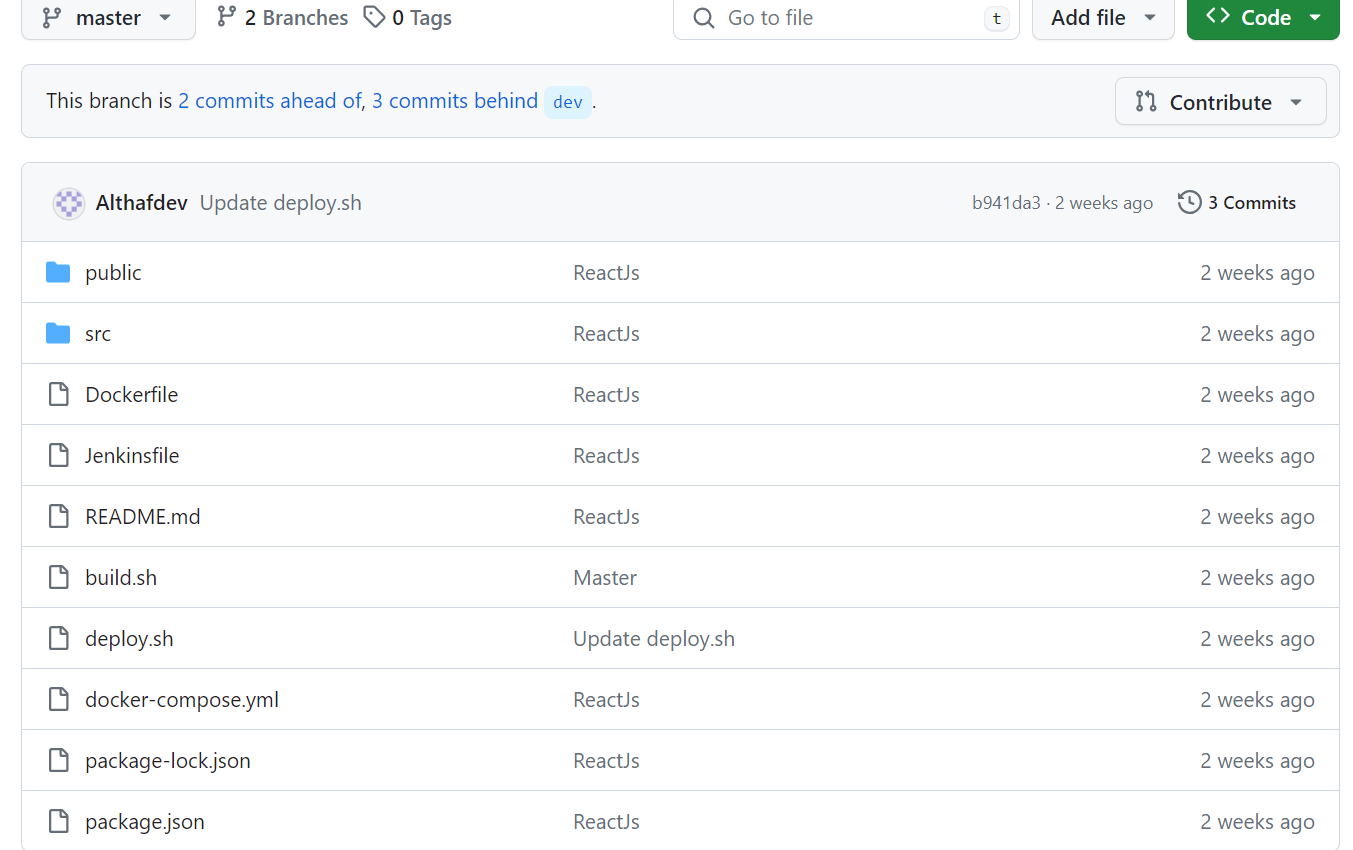
**- Create 2 repos “dev” and “prod” to push images.“Prod” repo must be private and “dev” repo can be public**

I have created 2 repo Dev and prod in Dockerhub, I have made prod as public and dev repo as Public.

**Dev branch: https://github.com/Althafdev/ReactJSA.git**



**Master branch for Private repo Prod https://github.com/Althafdev/ReactJSA.git**

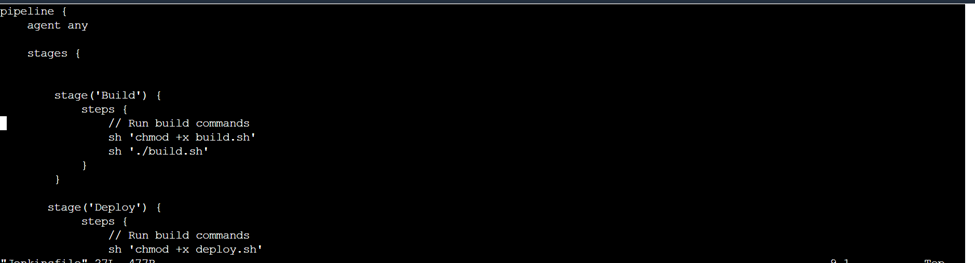


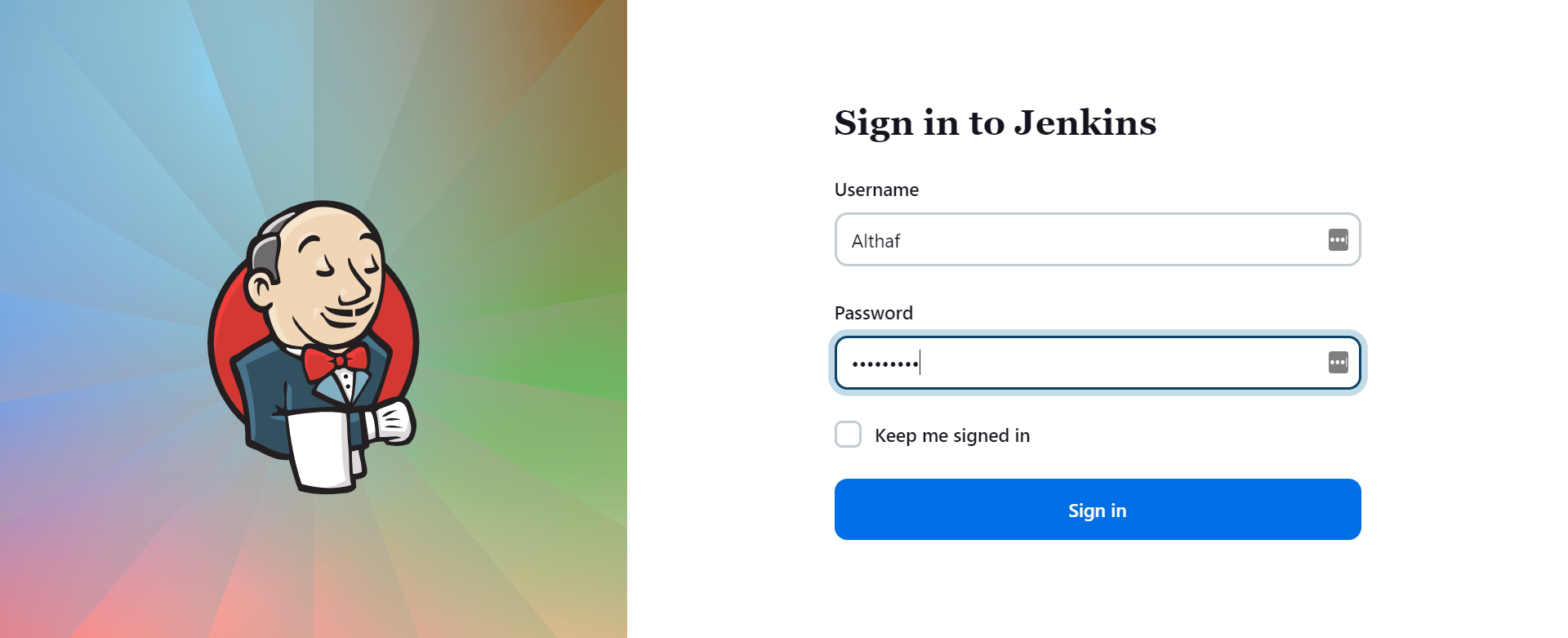
Jenkins:

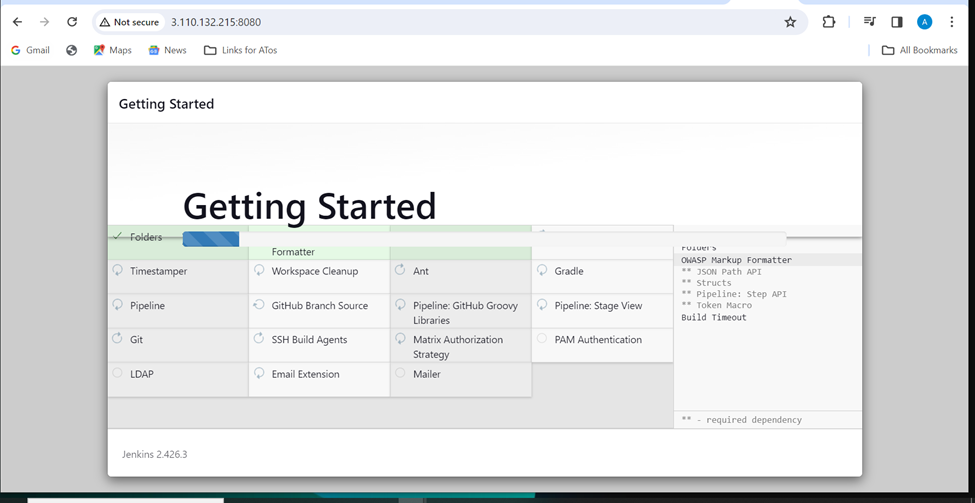
Install and configure jenkins build step as per needs to build, push & deploy the application. Connect jenkins to the github repo with auto build trigger from both dev & master branch If code pushed to dev branch, docker image must build and pushed to dev repo in docker hub. If dev merged to master, then docker image must be pushed to prod repo in docker hub

**Jenkins File:**

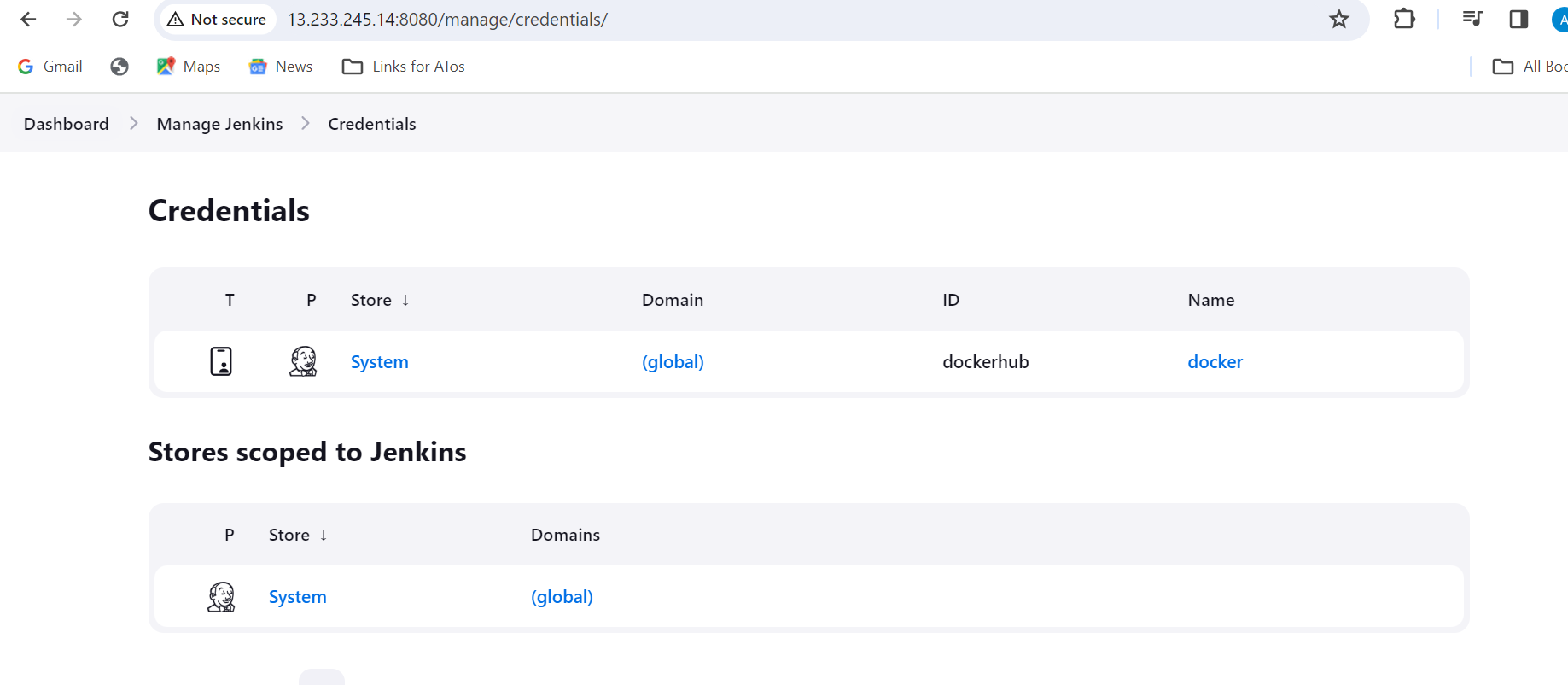
**I have written Jenkins file to deploy the image automatically through CICD to Docker Hub.**

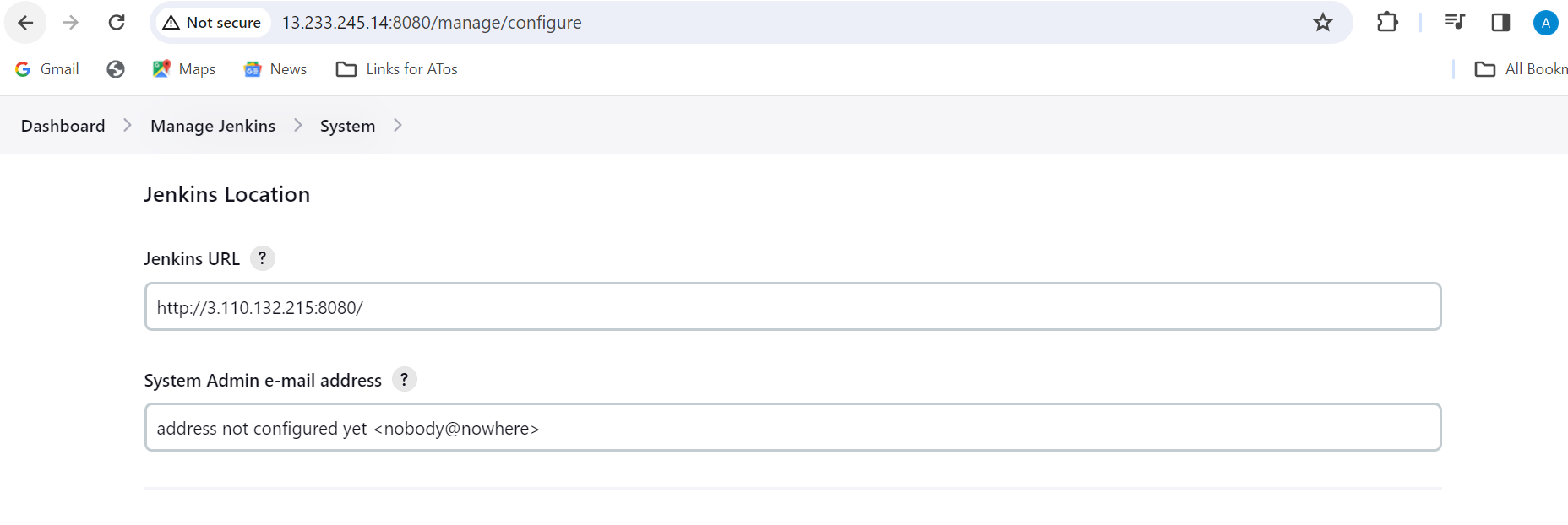


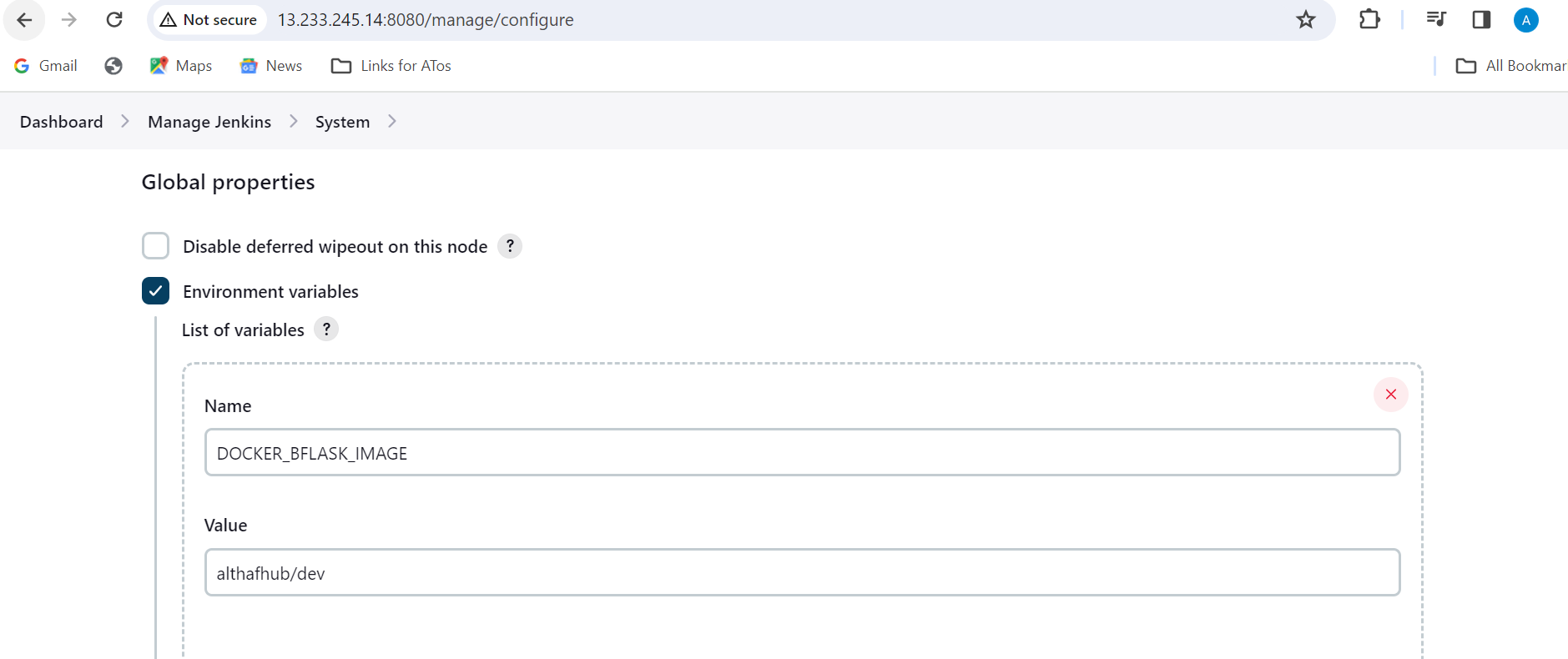


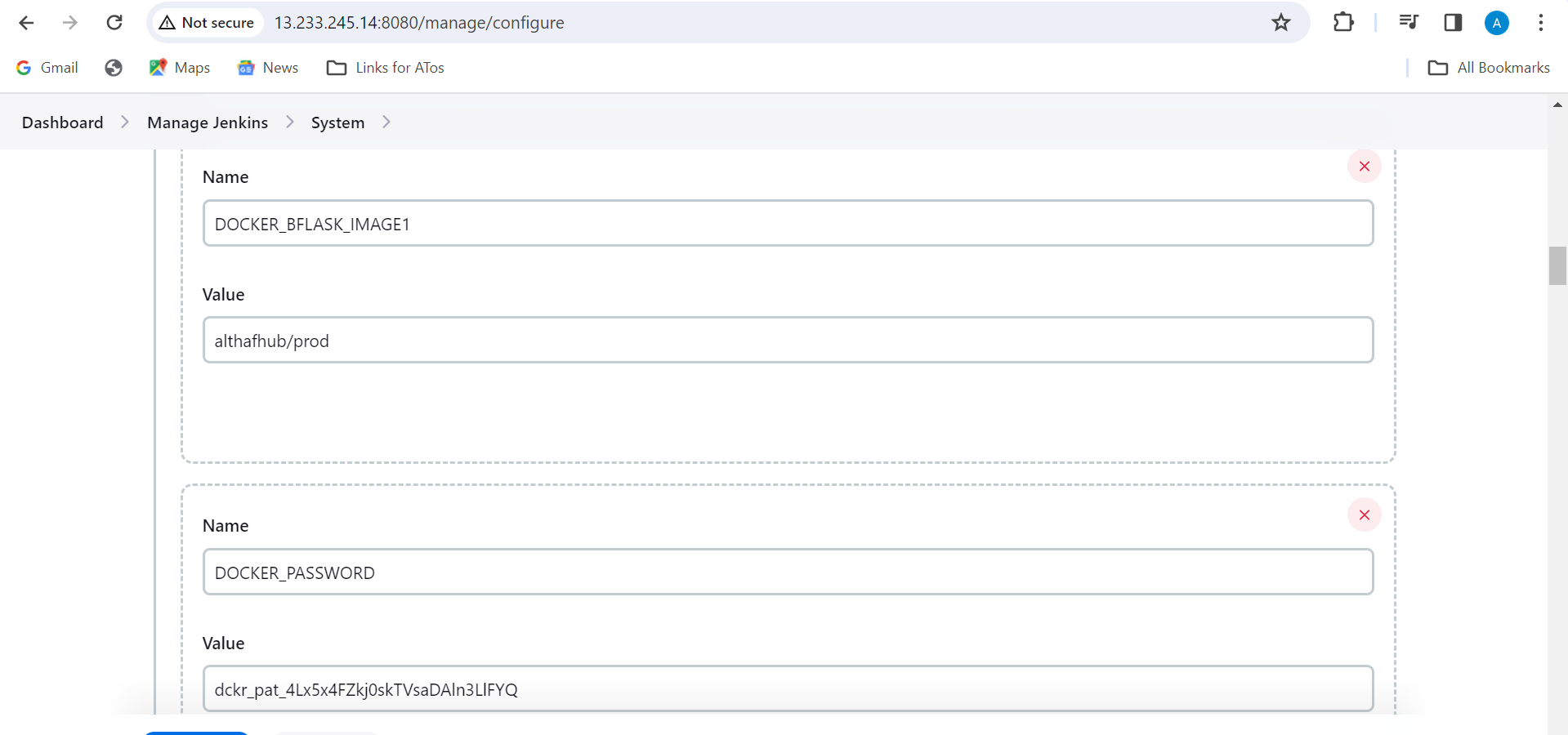


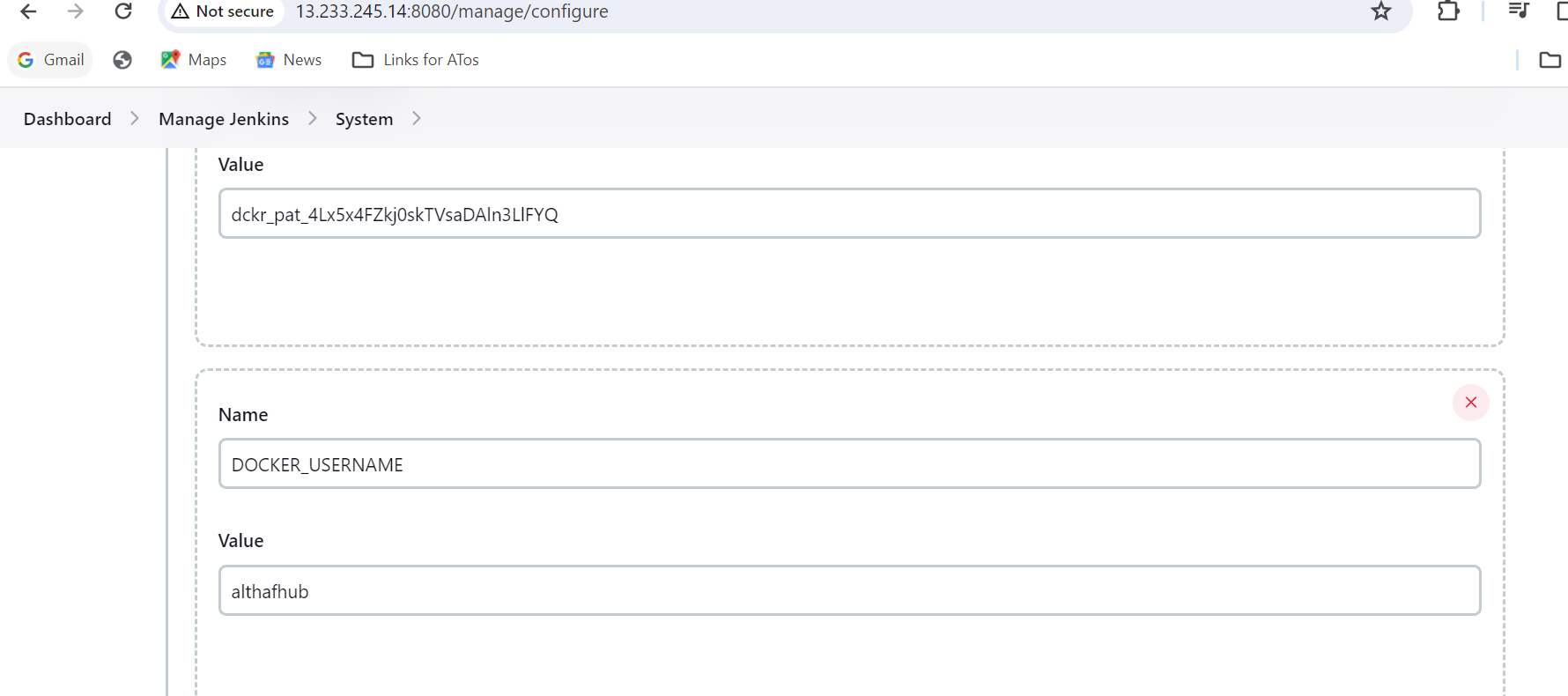
Configured the Environmental variables and ToKens



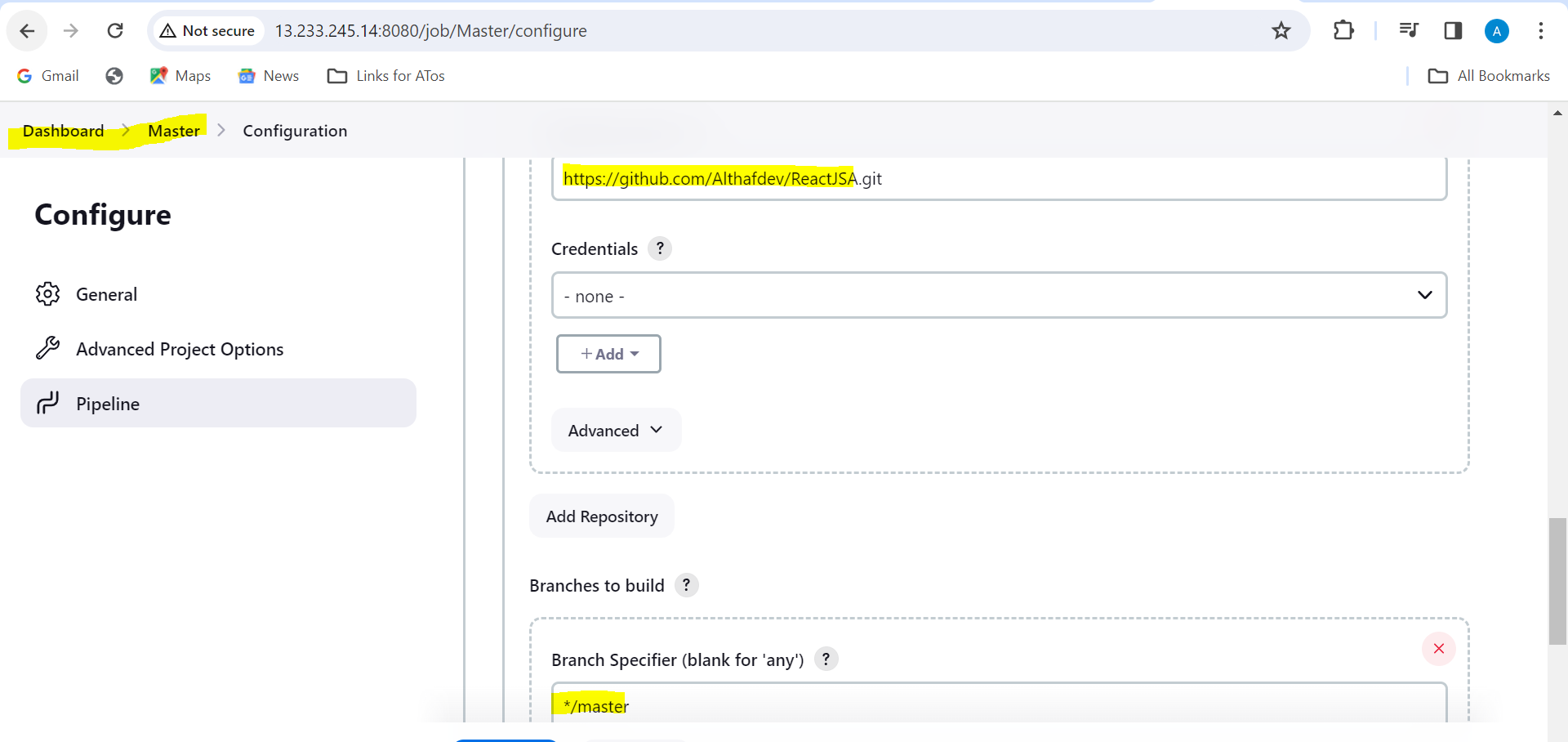


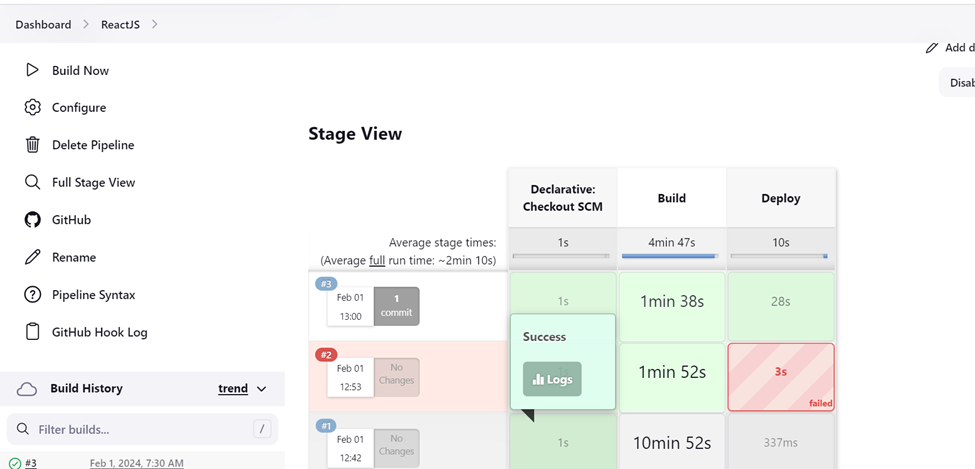


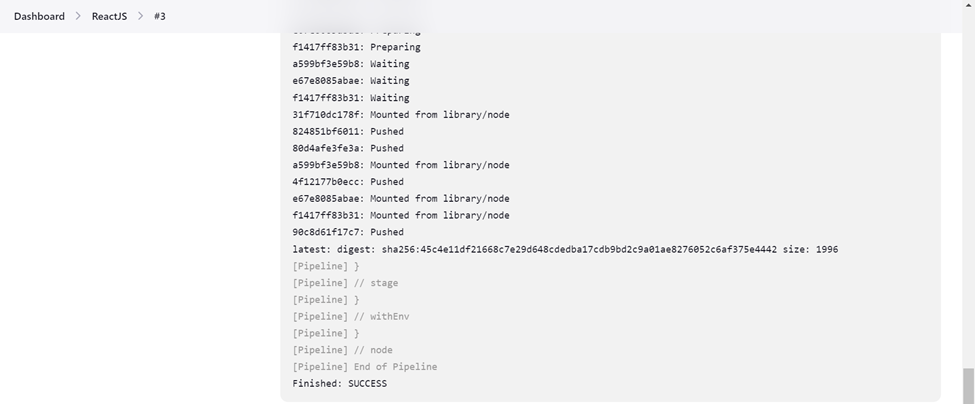




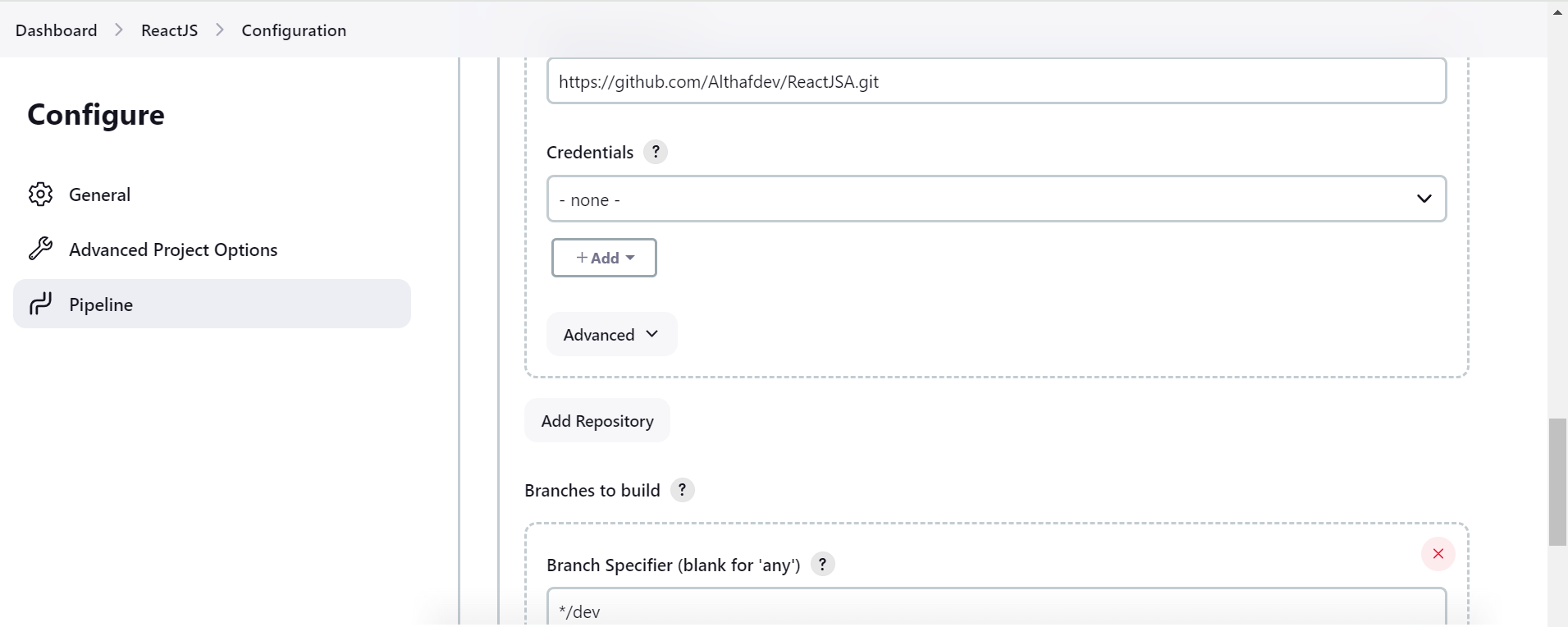
Prod (Master) repo is triggered automatically and push the image to dockerhub

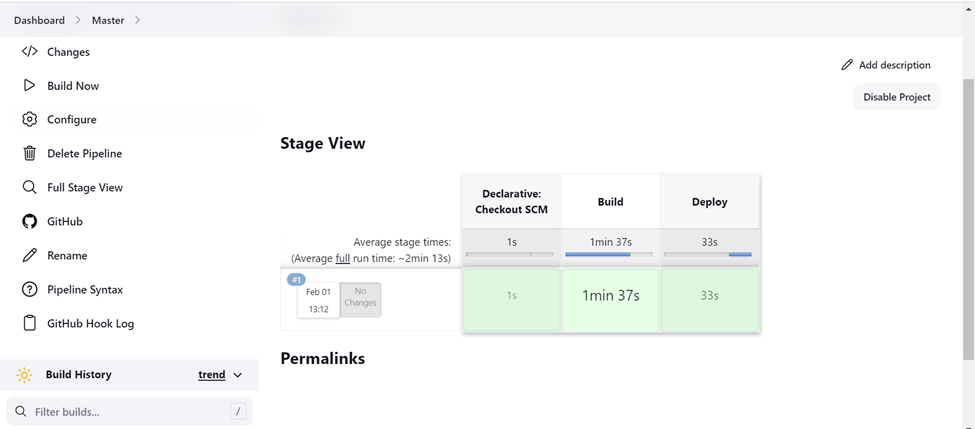


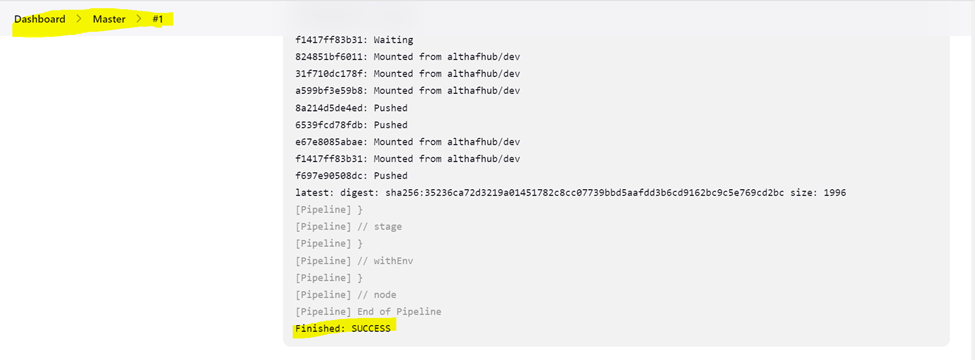


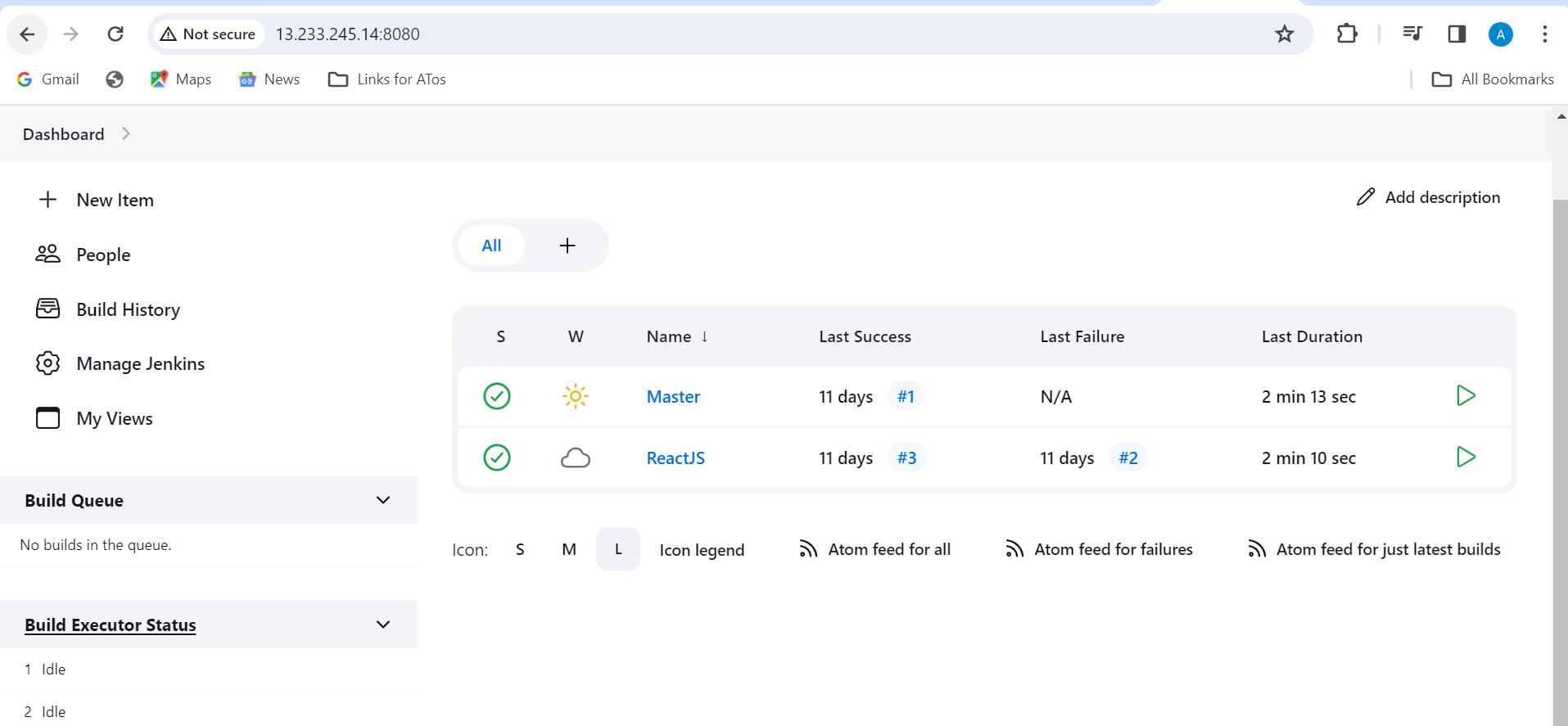


Dev repo is triggered automatically and push the image to dockerhub

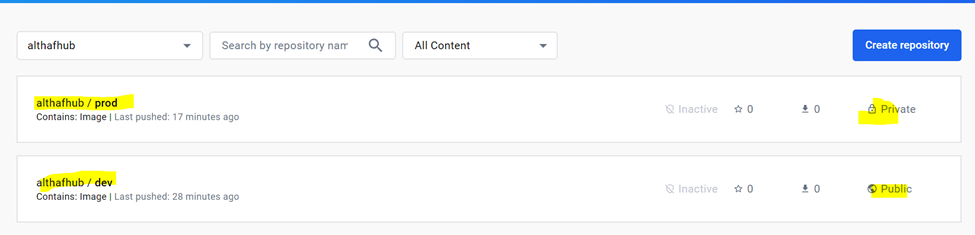




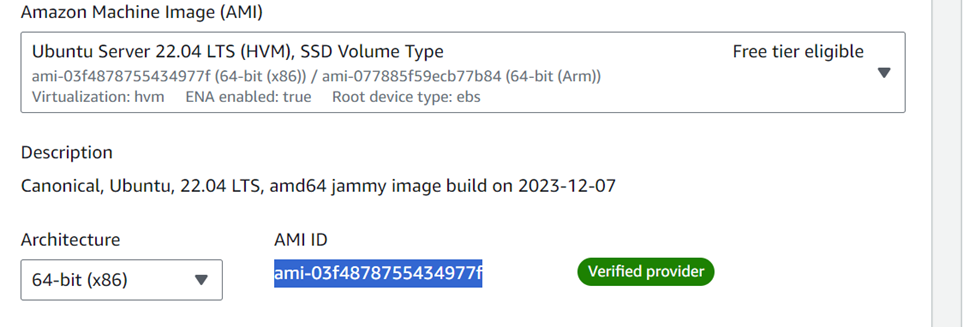


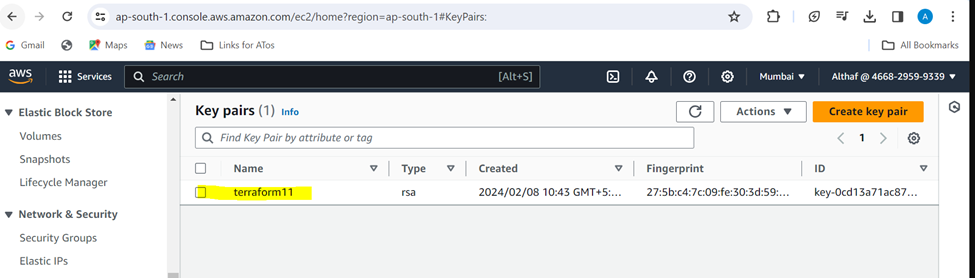


**Image pushed to DockerHub by CI/CD jenkins Automatic**



Launching an Instances to deploy an ReactJs application for that i have written terraform code to launch an instance, selected AMI ubuntu image and created key pair on ap-south-1 location. State file of terraform stored in S3 bucket.



****

**Terraform code :**

**provider "aws" {**

**region = "ap-south-1"**

**}**

**terraform {**

**backend "s3" {**

**bucket = "capstoneprojectbucket1"**

**key = "my-terraform-project"**

**region = "ap-south-1"**

**}**

**}**

**data "aws\_vpcs" "default" {}**

**data "aws\_subnet" "default" {**

**vpc\_id = data.aws\_vpcs.default.ids[0]**

**availability\_zone = "ap-south-1a"**

**}**

**resource "aws\_instance" "capstone" {**

**ami = " ami-03f4878755434977f"**

**instance\_type = "t2.micro"**

**key\_name = "terraform11"**

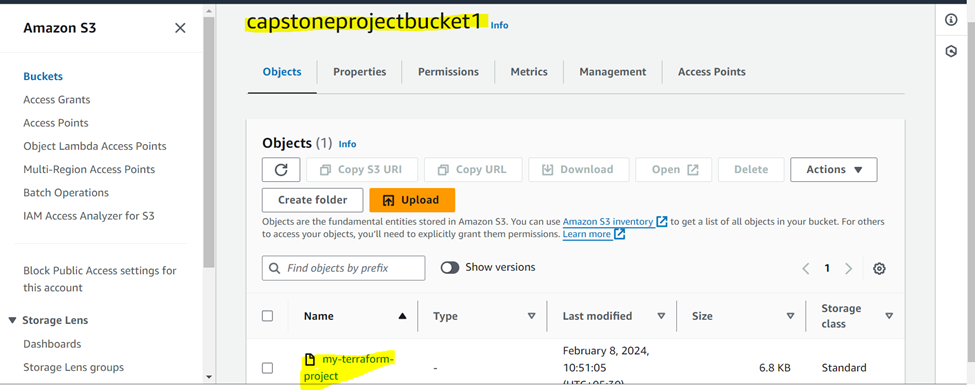
**subnet\_id = data.aws\_subnet.default.id**

**tags = {**

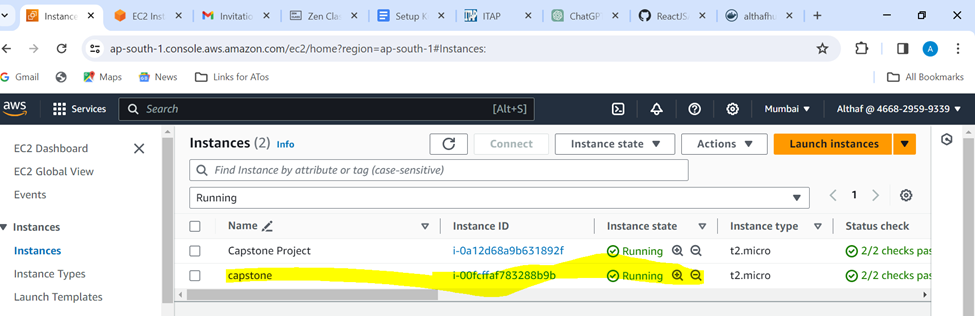
**Name = "capstone"**

**}**

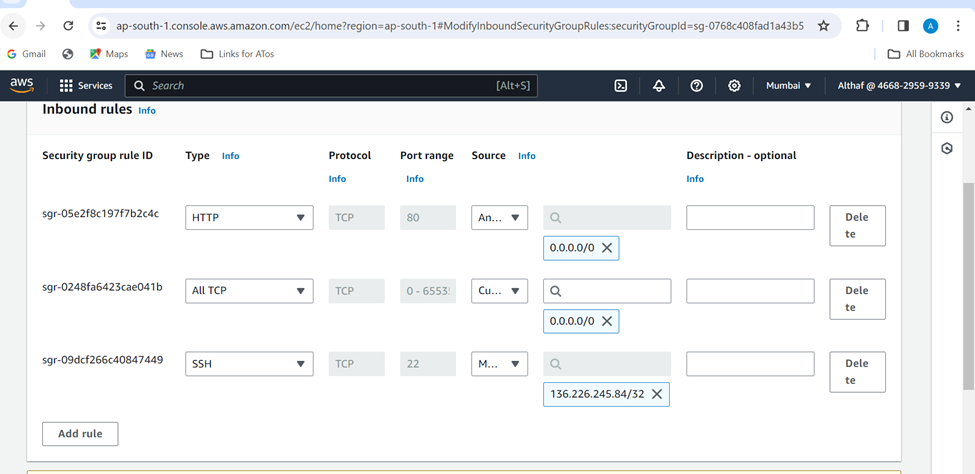
**}**

****

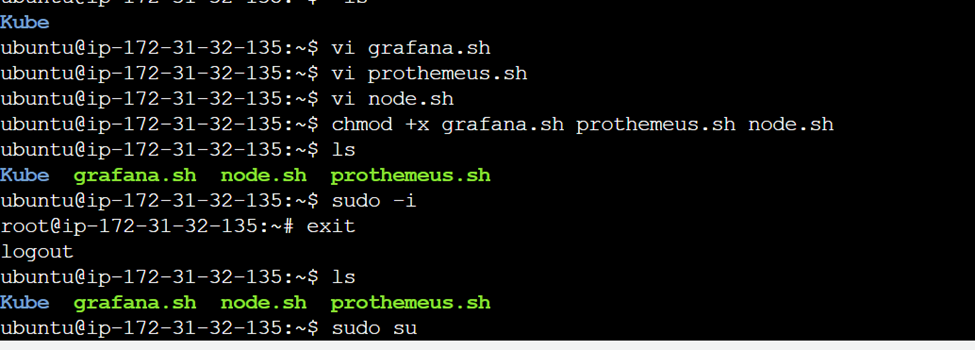
**Using terraform i have created an Infrastructure to deploy an application using image which we pushed to dockerhub**

****

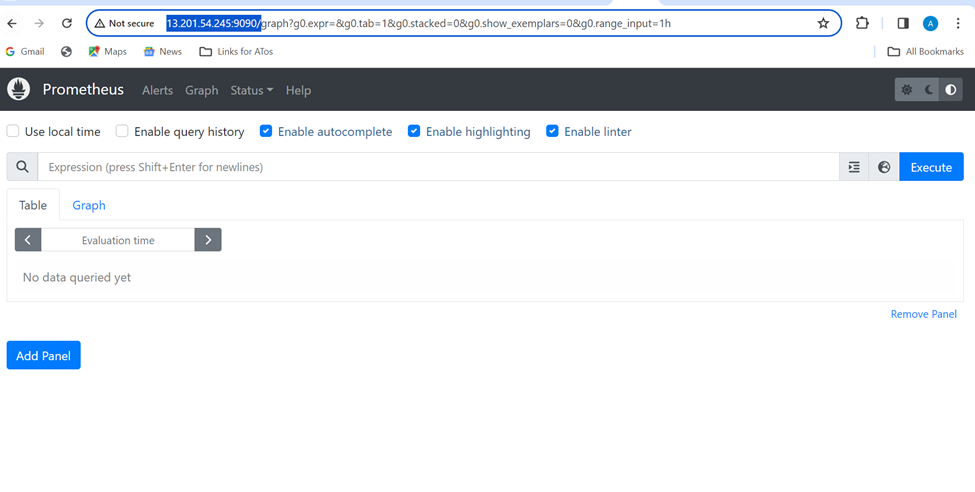
**Security Group is configured**

****

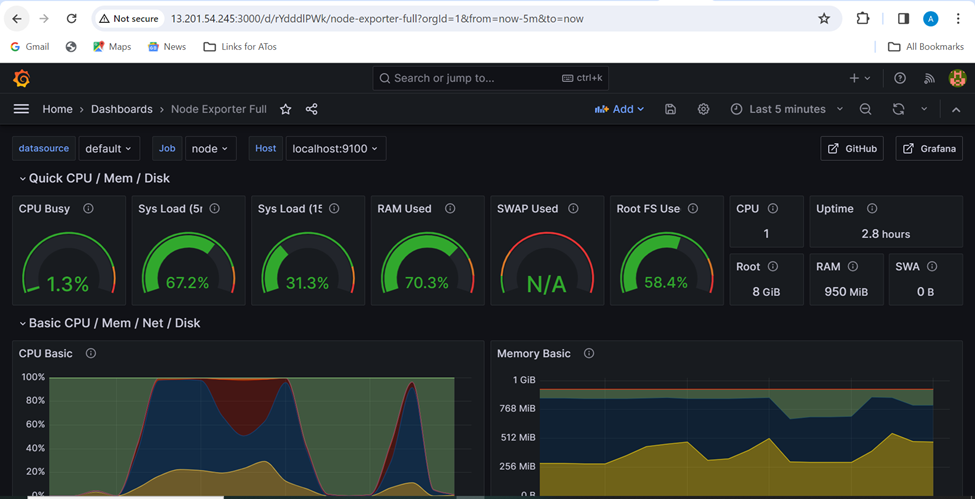
**For Monitoring of the server I have used Prometheus and Grafana**

****

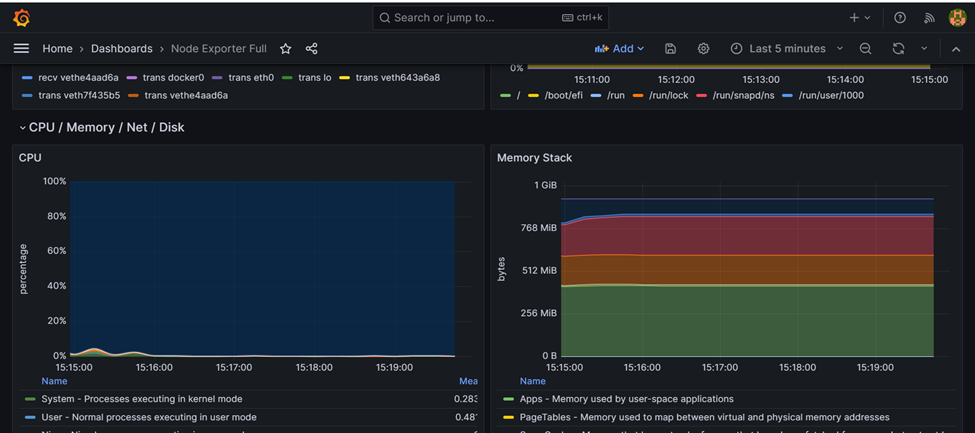
****

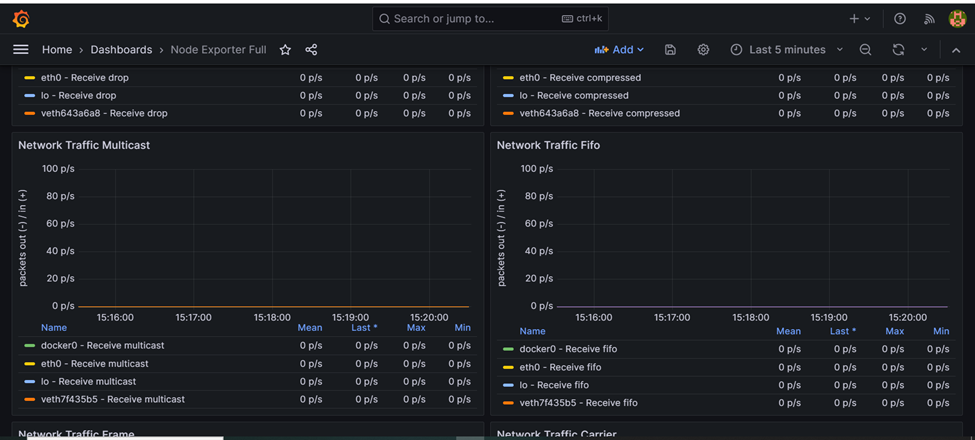
****

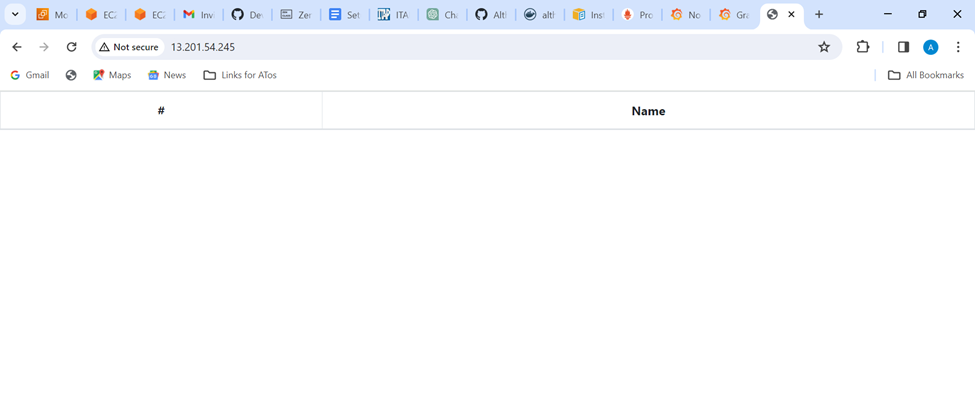
****

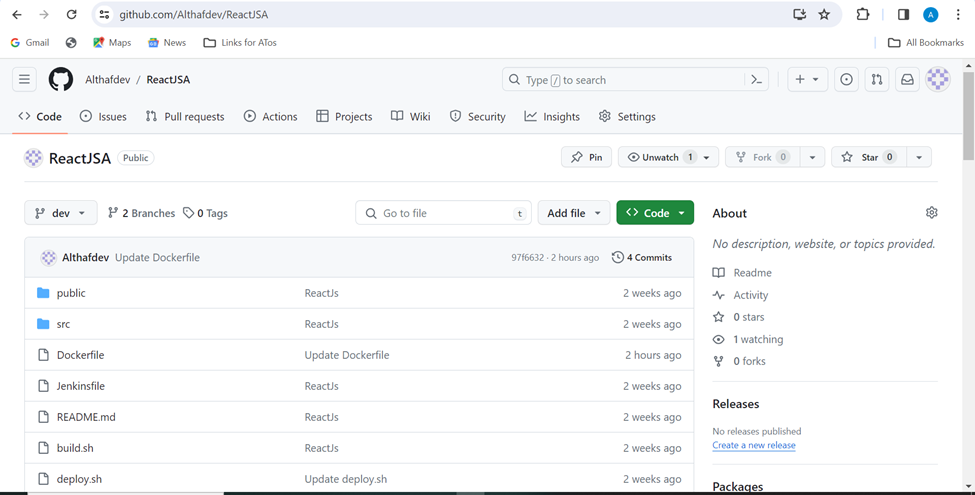
****

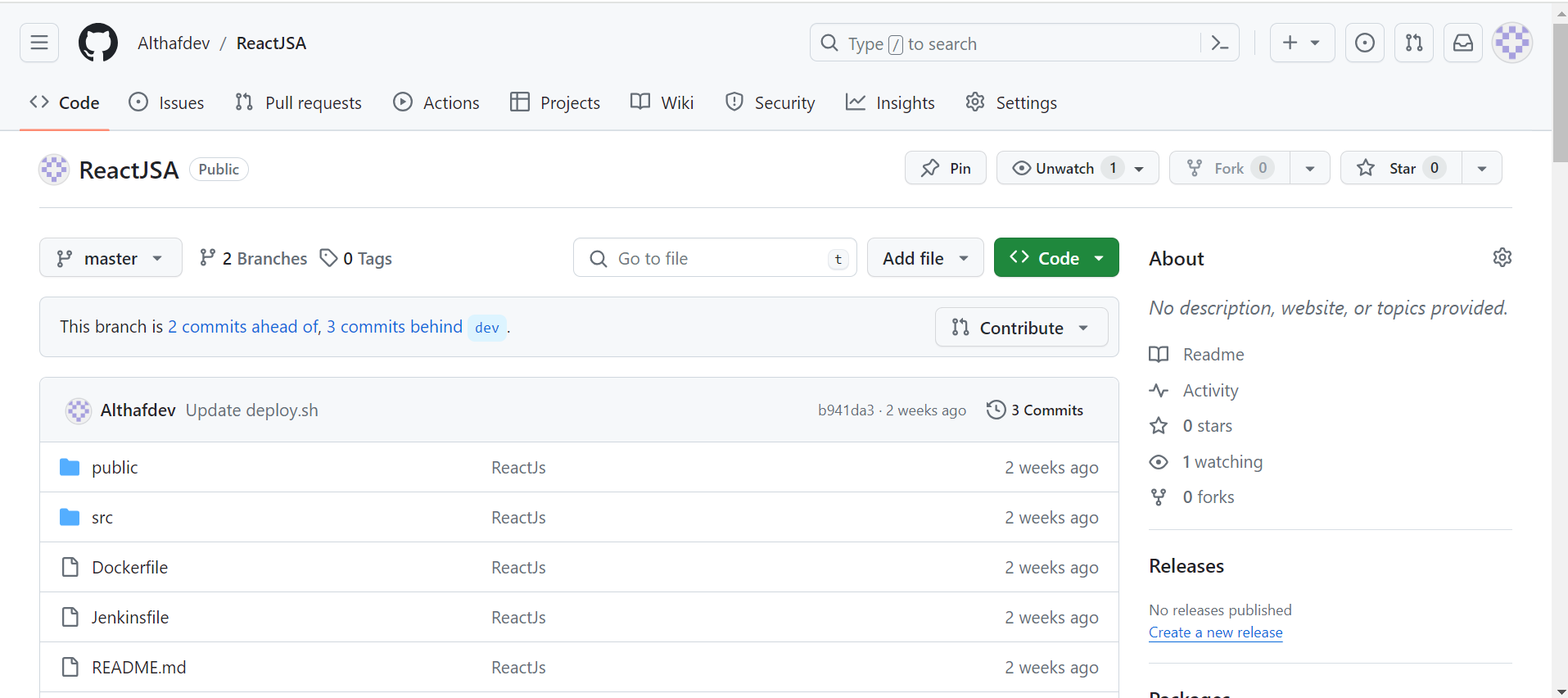
****

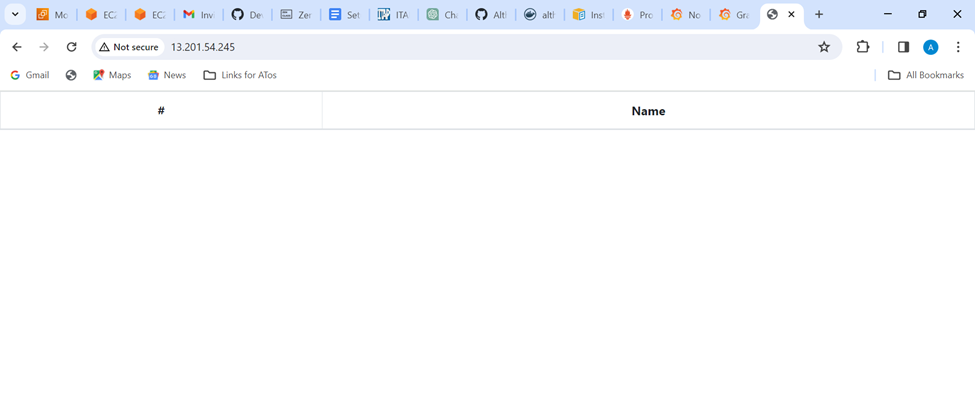
****

****

****

****

****

****