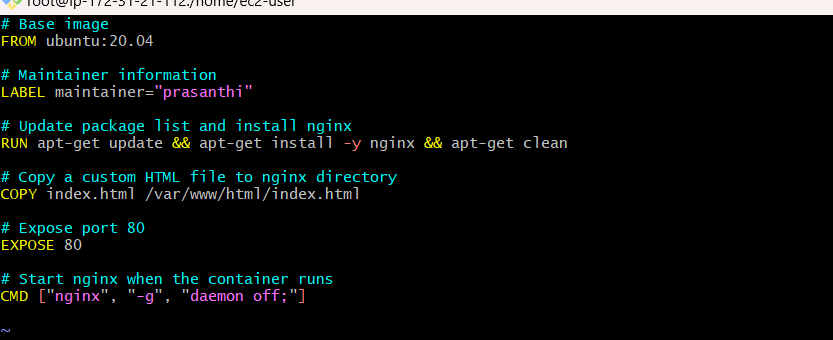
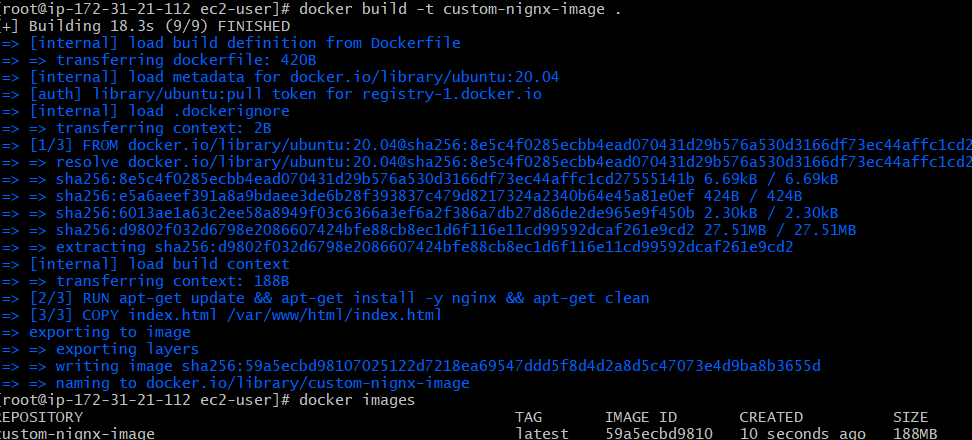
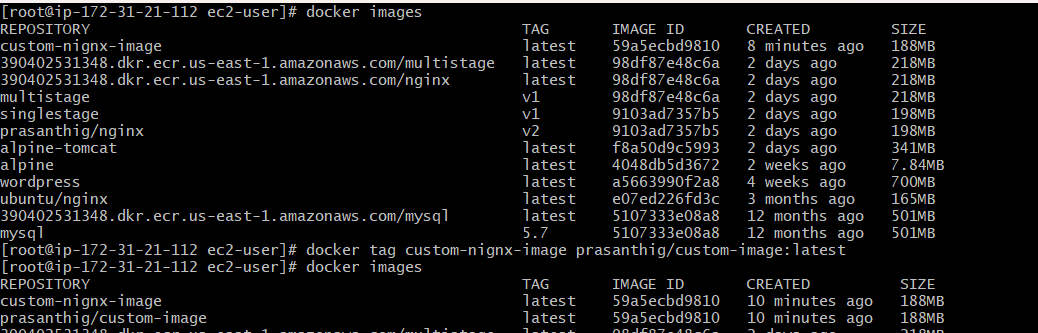
1. Create a customized docker image by using Docker file.



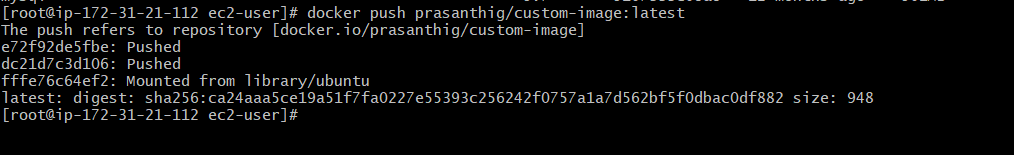


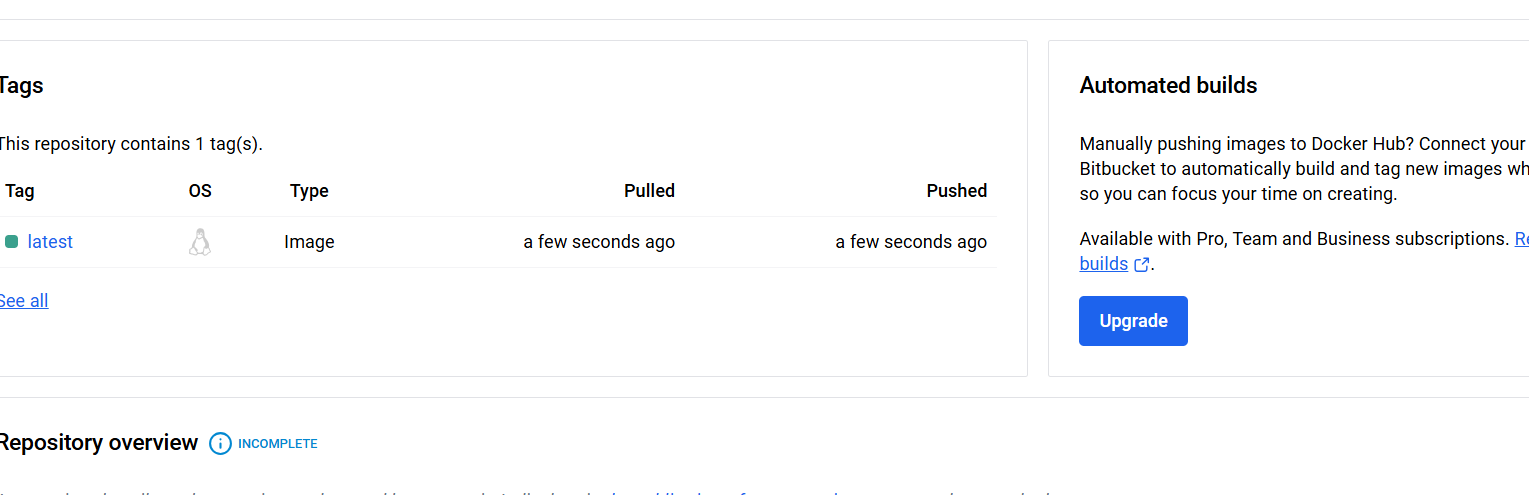
1. Push the image to docker hub

Tag the image

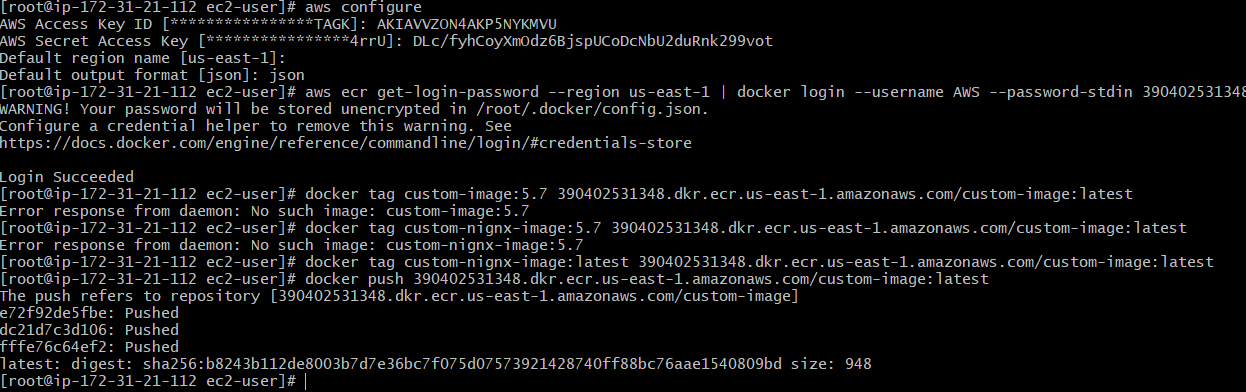


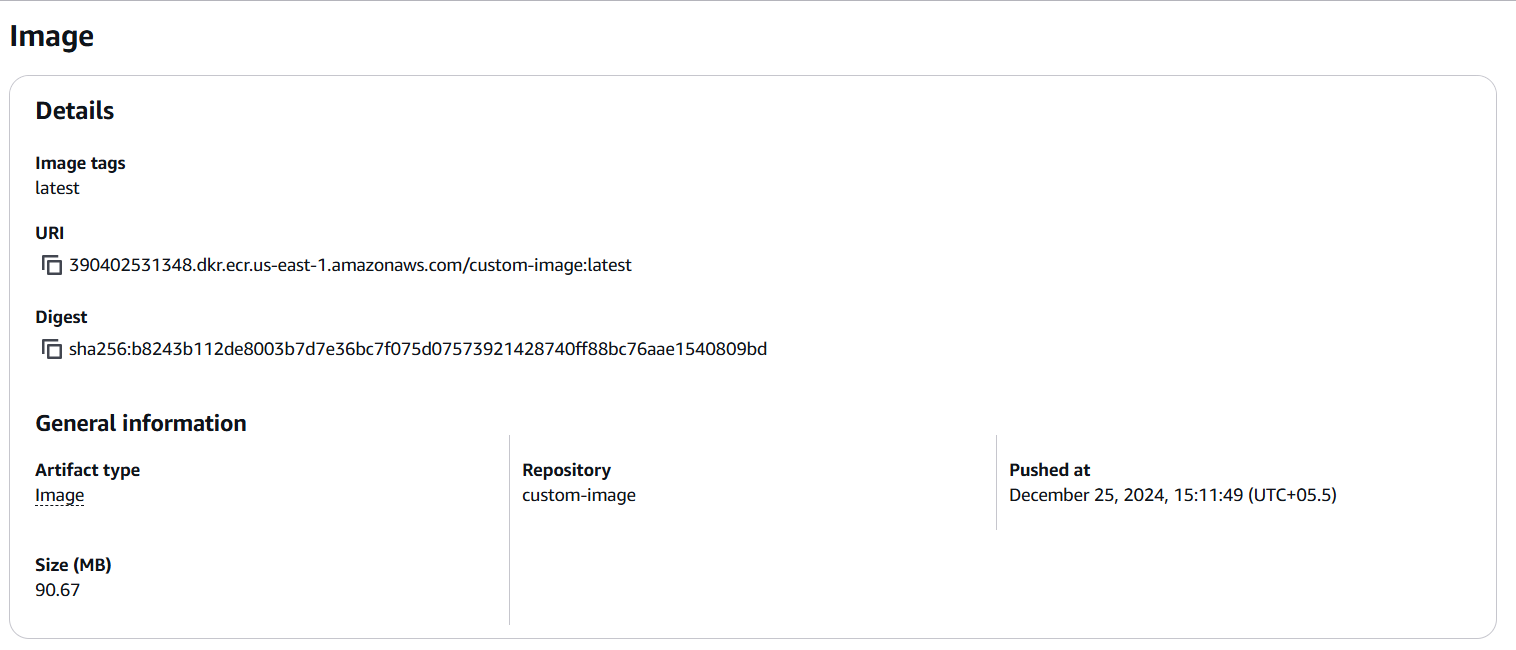
Push the image to docker hub



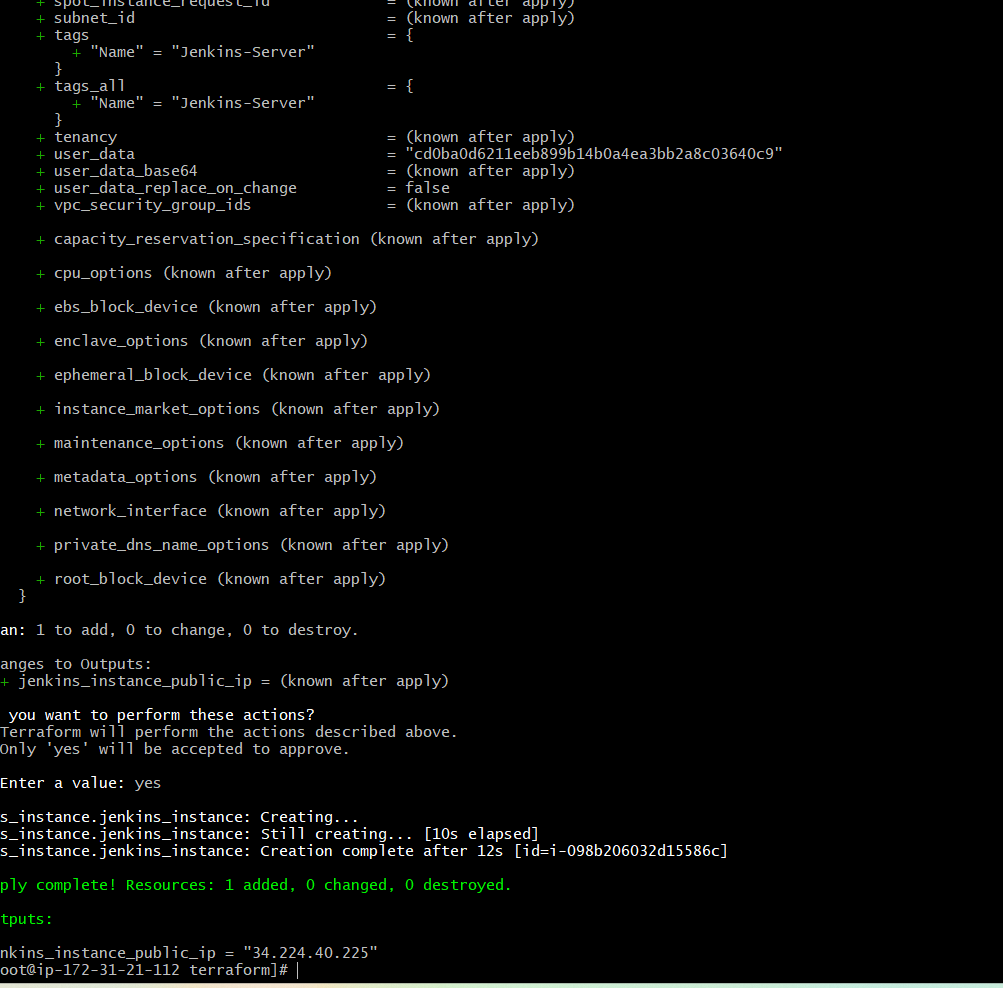


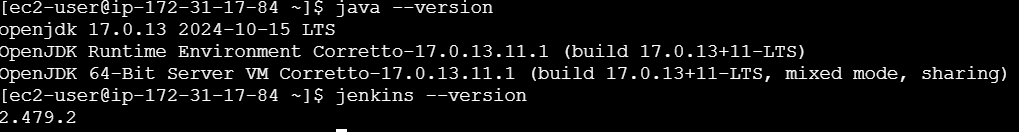
1. Push the same Image to Amazon ECR

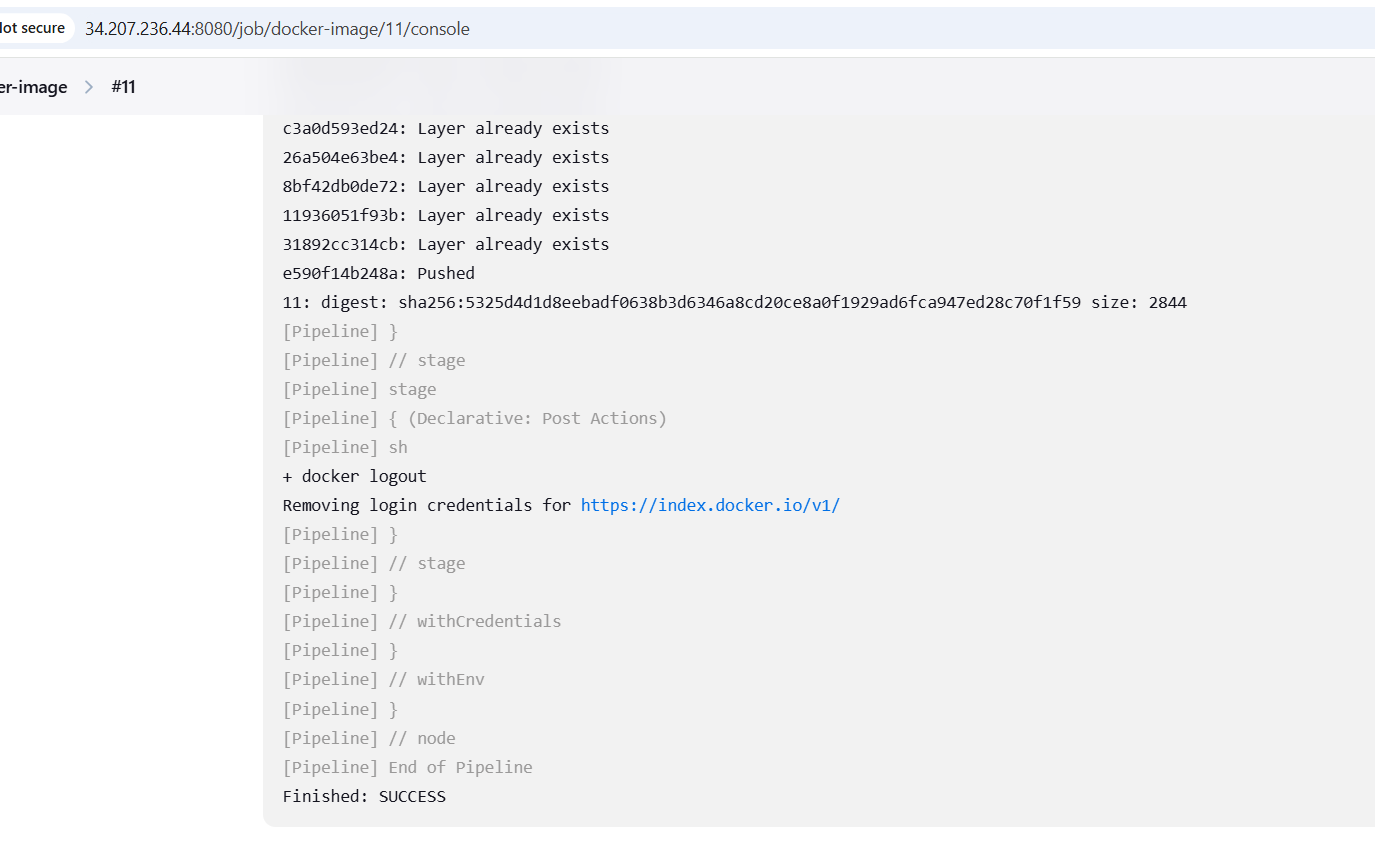




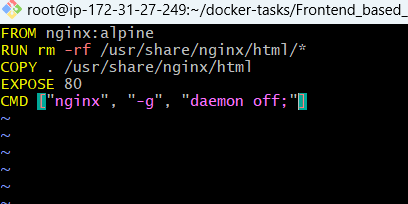
1. Provision one ec2 using terraform and install Jenkins.
2. # Provider Configuration
3. provider "aws" {
4. region = "us-east-1"
5. }
6. # Key Pair
7. variable "key\_name" {
8. default = "nvirg"
9. }
10. # Security Group for Jenkins
11. resource "aws\_security\_group" "jenkins\_sg" {
12. name        = "jenkins-sg"
13. description = "Allow SSH and HTTP access for Jenkins"
14. ingress {
15. from\_port   = 22
16. to\_port     = 22
17. protocol    = "tcp"
18. cidr\_blocks = ["0.0.0.0/0"] # SSH access
19. }
20. ingress {
21. from\_port   = 8080
22. to\_port     = 8080
23. protocol    = "tcp"
24. cidr\_blocks = ["0.0.0.0/0"] # Jenkins UI access
25. }
26. egress {
27. from\_port   = 0
28. to\_port     = 0
29. protocol    = "-1"
30. cidr\_blocks = ["0.0.0.0/0"]
31. }
32. }
33. # EC2 Instance
34. resource "aws\_instance" "jenkins\_instance" {
35. ami           = "ami-0ca9fb66e076a6e32"
36. instance\_type = "t2.micro"
37. key\_name      = var.key\_name
38. security\_groups = [aws\_security\_group.jenkins\_sg.name]
39. tags = {
40. Name = "Jenkins-Server"
41. }
42. # Install Jenkins during EC2 provisioning
43. user\_data = <<-EOF
44. #!/bin/bash
45. yum update -y
46. yum install java-17-amazon-corretto-devel
47. yum install -y wget
48. wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
49. rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
50. yum install -y jenkins
51. systemctl start jenkins
52. systemctl enable jenkins
53. EOF
54. }
55. # Output Jenkins Public IP
56. output "jenkins\_instance\_public\_ip" {
57. value = aws\_instance.jenkins\_instance.public\_ip
58. }

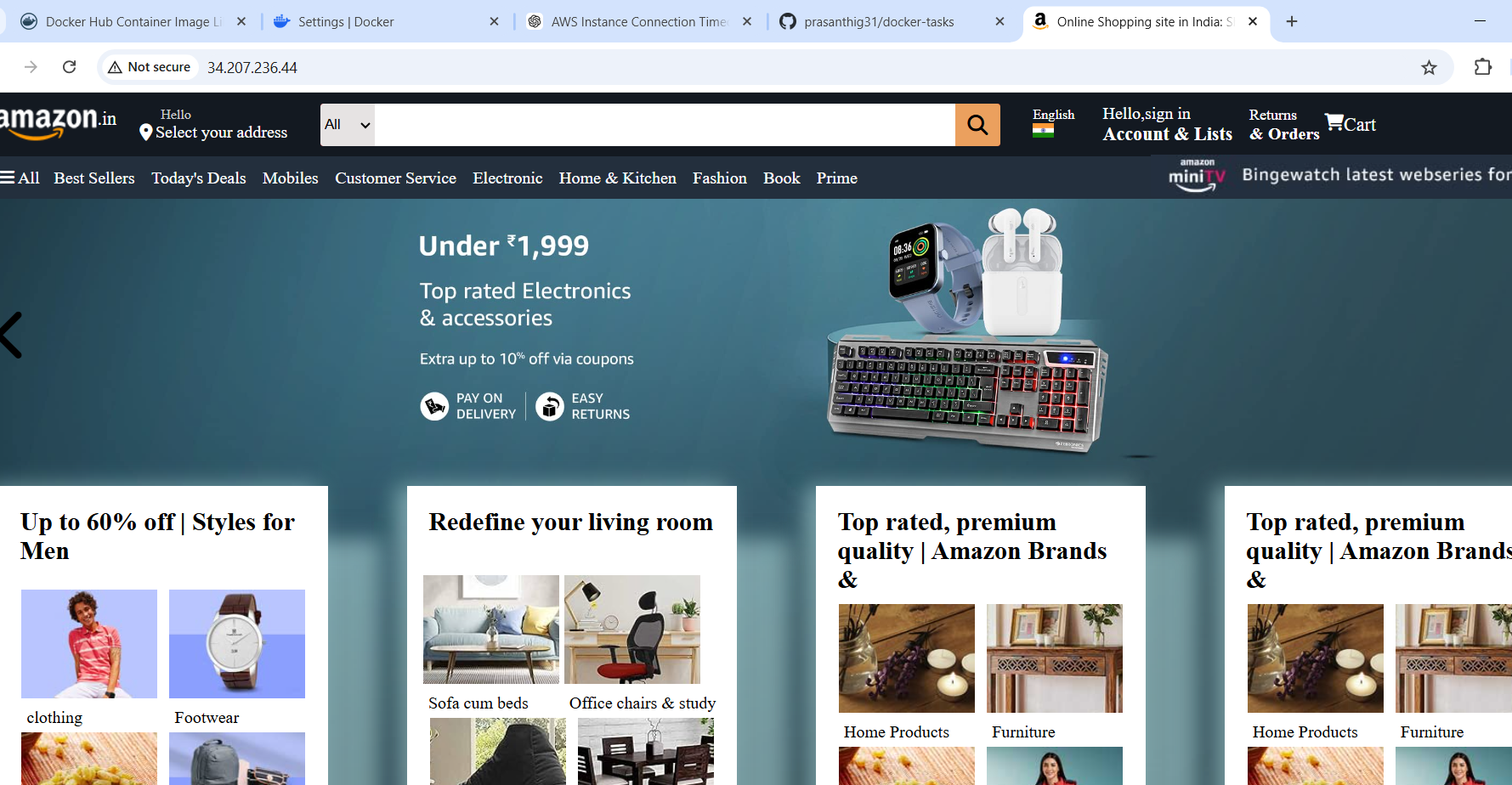


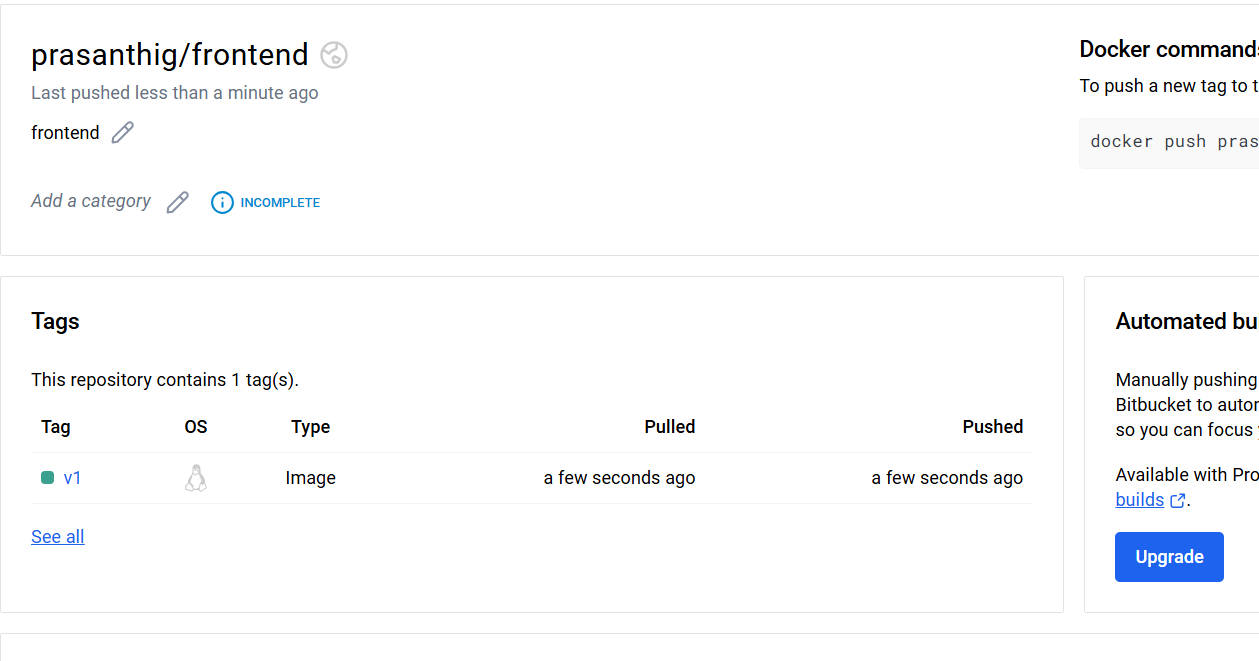




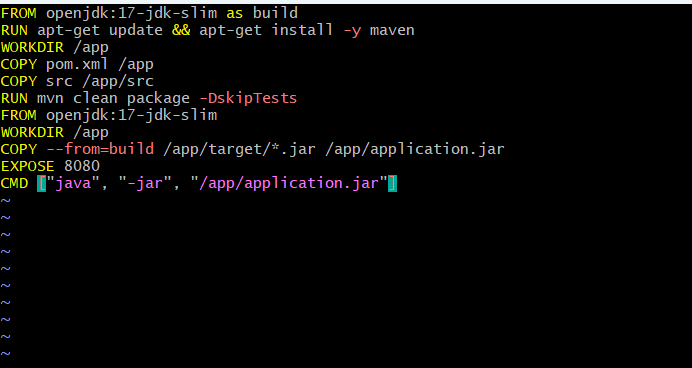
1. From the source code of the frontend, Using that write a Docker file & build a docker image, run & push that image to your docker registry



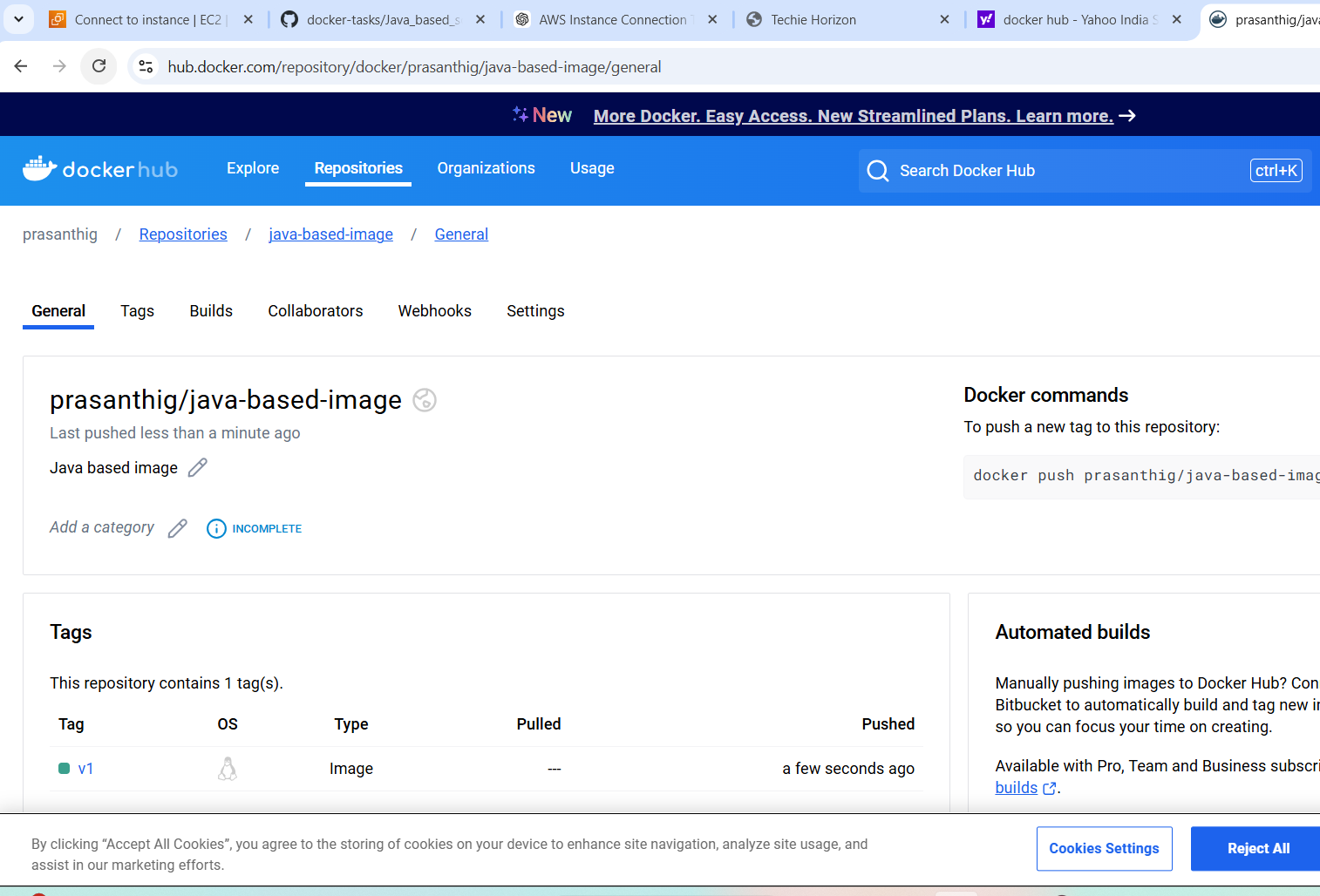




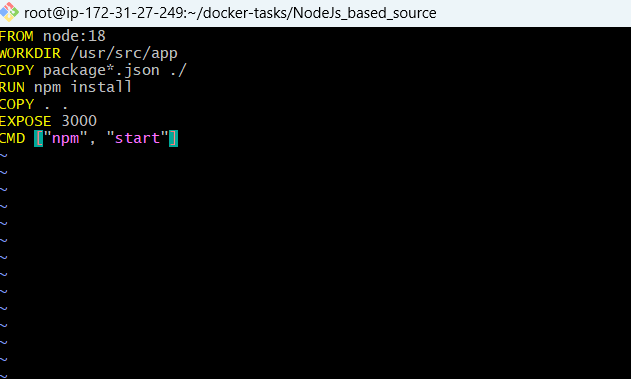
2. From the Java Based Source Code, Write a Docker file, build, run & push to docker registry.

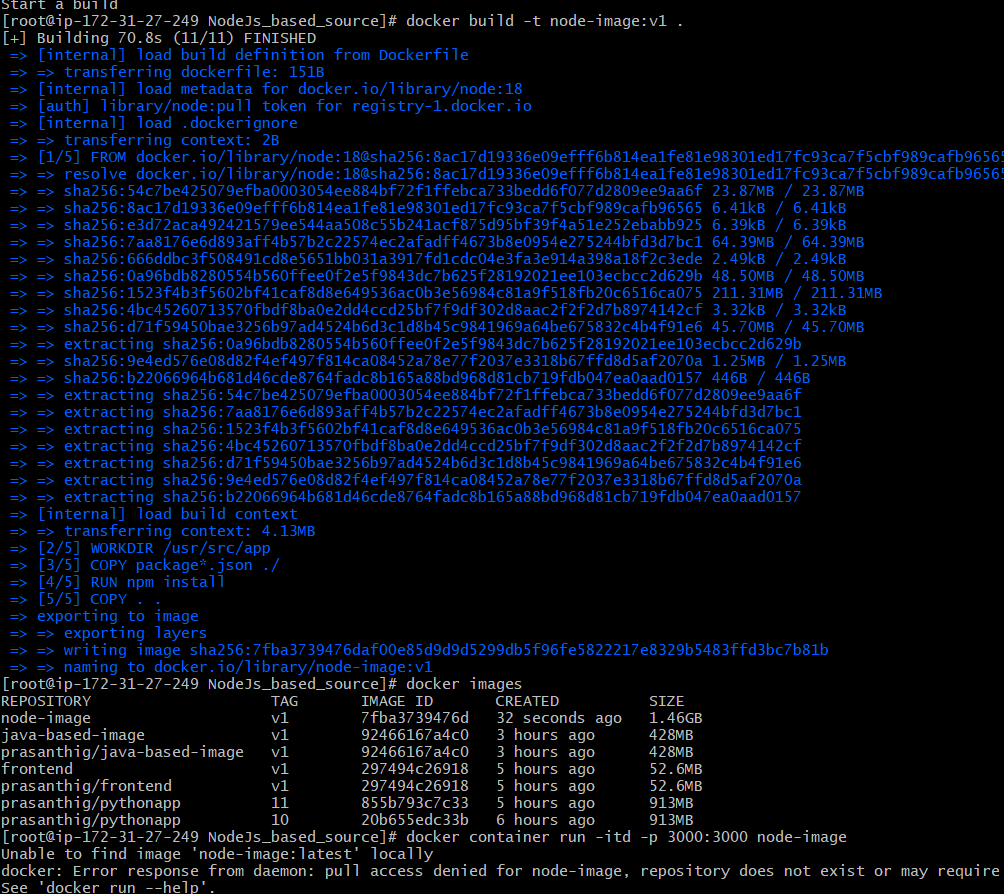


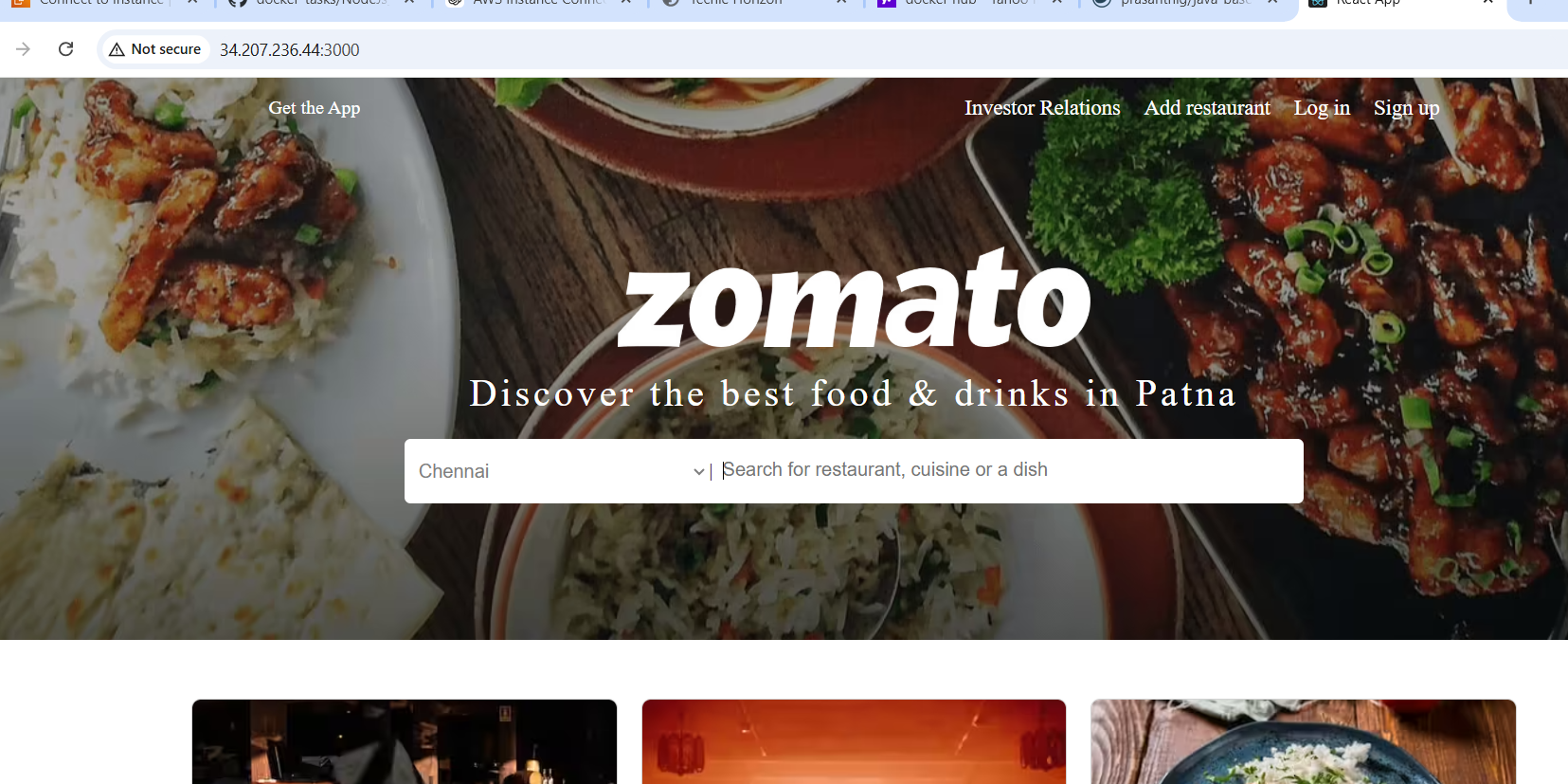




3. From the NodeJS Based Source Code, Write a docker file, build with tag v1, run & push to docker registry.







4. Write a docker-compose docker file to setup WordPress with MySQL database

Install docker compose

sudo curl -L "https://github.com/docker/compose/releases/download/v2.25.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

giving permission create symbolic link

chmod +x /usr/local/bin/docker-compose

ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose

docker-compose.yml

version: '3'

services:

db:

image: mysql:5.7

volumes:

- db\_data:/var/lib/mysql

restart: always

environment:

- MYSQL\_ROOT\_PASSWORD=somewordpress

- MYSQL\_DATABASE=wordpress

- MYSQL\_USER=wordpress

- MYSQL\_PASSWORD=wordpress

wordpress:

depends\_on:

- db

image: wordpress:latest

ports:

- "8000:80"

restart: always

environment:

- WORDPRESS\_DB\_HOST=db:3306

- WORDPRESS\_DB\_USER=wordpress

- WORDPRESS\_DB\_PASSWORD=wordpress

- WORDPRESS\_DB\_NAME=wordpress

volumes:

db\_data: { }

run docker-compose up command

