Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Automated Deployment with Monitoring

Introduction

The following case study will explain in detail how to establish a CI/CD pipeline using Jenkins for a simple web application, which would be deployed on an Amazon EC2 instance. We will also extend this setup by implementing Nagios for monitoring the application availability, enabling the detection of any possible downtime or issues as soon as they arise.

Problem Statement

Automation of deploying a web application will help in improving efficiency and reducing chances of human error throughout deployment. Secondly, it is important to put in place the monitoring that will update on application health and performance.

Technologies Used

- **Jenkins**: An open-source automation server that facilitates CI/CD.
- **Amazon EC2**: A cloud computing service that provides scalable computing capacity.
- **Nagios**: A powerful monitoring system that enables organizations to identify and resolve IT infrastructure issues.
- **Vercel**: A platform for frontend developers, used here for analytics and monitoring the application's performance and usage.
- **Docker:** Containerization for consistent deployment for our web app.

Implementation Steps

1. **Setting Up the Environment**:
    - Provision an EC2 instance where the web application will be deployed.
    - Install necessary software and dependencies on the EC2 instance.

2. **Jenkins Configuration**:
    - Install Jenkins on a dedicated server or the EC2 instance itself.
    - Create a new pipeline job in Jenkins that will:
        - Clone the web application repository.
        - Build the application.
        - Deploy the built application to the EC2 instance using SSH.

3. **Nagios Installation and Configuration**:
    - Install Nagios on a separate server or the same EC2 instance.
    - Configure Nagios to monitor the HTTP status of the deployed web application.

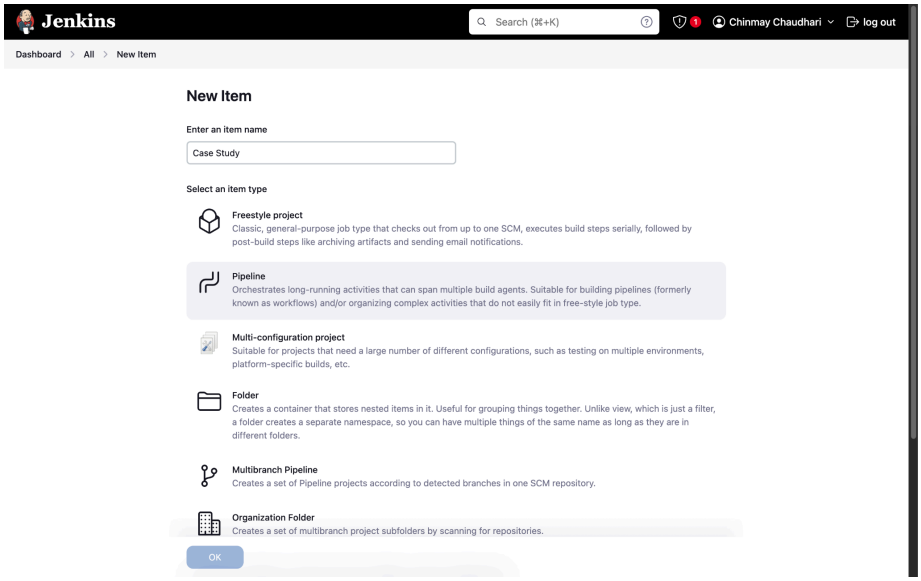
4. **Vercel Analytics Integration**:
    - Integrate Vercel for tracking user interactions and performance metrics.
    - Use the data gathered from Vercel to inform future deployments and enhancements.

5. **Testing the Pipeline**:
    - Trigger a build in Jenkins to deploy the application.
    - Verify the deployment by accessing the web application through a browser.

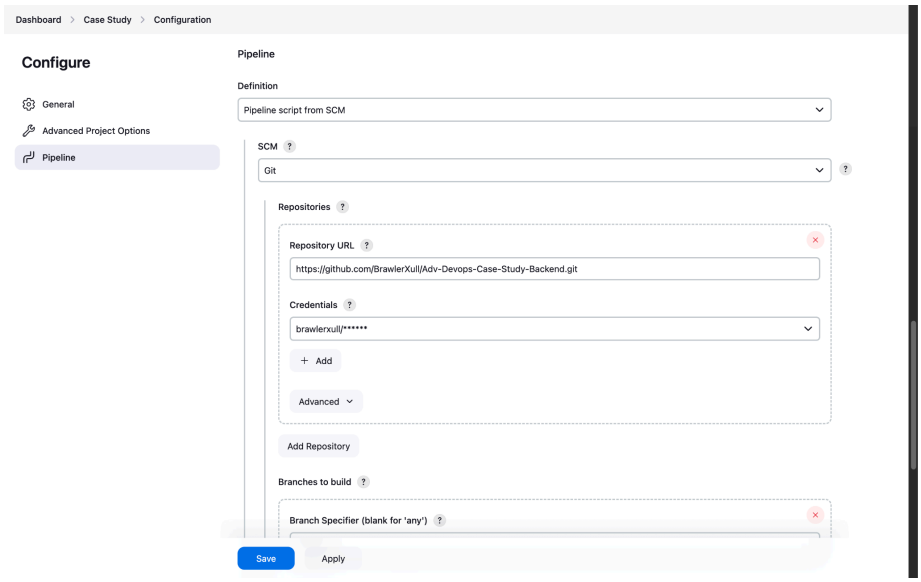
Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Part I - Backend

Step 1 - Go To Jenkins And Start new Pipeline project



Step 2 - Select ‘Pipeline Script from SCM option’ and insert the url of git repo

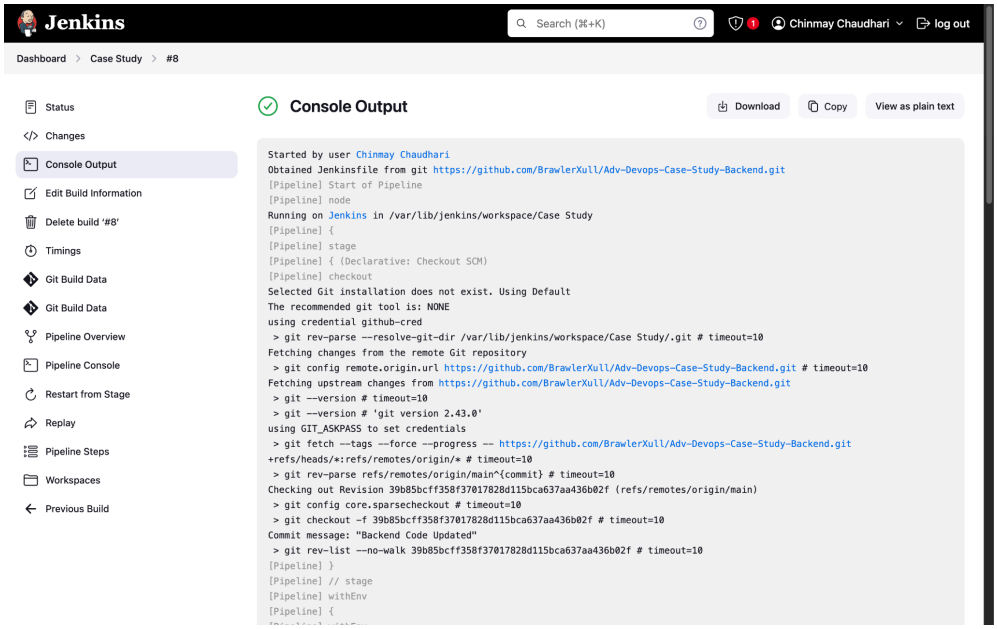


Step 3 - Add the below Jenkins file to the repo

```
pipeline {
   environment {
   imagename = "backend"
   jenkinsProject = 'calculator-webapp-backend'
 }

   agent any
   stages {

       stage('Git Staging'){

           steps{

               checkout([$class: 'GitSCM', branches: [[name: '*/main']], extensions:
[], userRemoteConfigs: [[credentialsId: 'github-cred', url:
'https://github.com/yashrpandit/calculator-webapp-backend.git']]])

           }
       }


       stage('Build image and Run image ') {

           steps{
               sh 'sudo su - jenkins -s/bin/bash'
               //sh 'sudo docker stop $imagename'
               //sh 'sudo docker rm $imagename'
               //sh 'sudo docker rmi $imagename'
               sh 'sudo docker image build -t  $imagename .'

           }
       }

       stage('Run image ') {
           steps{
               sh 'sudo docker run -p5000:5000 --restart=always --name $imagename
-itd $imagename'
           }
       }
   }
}
```

Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

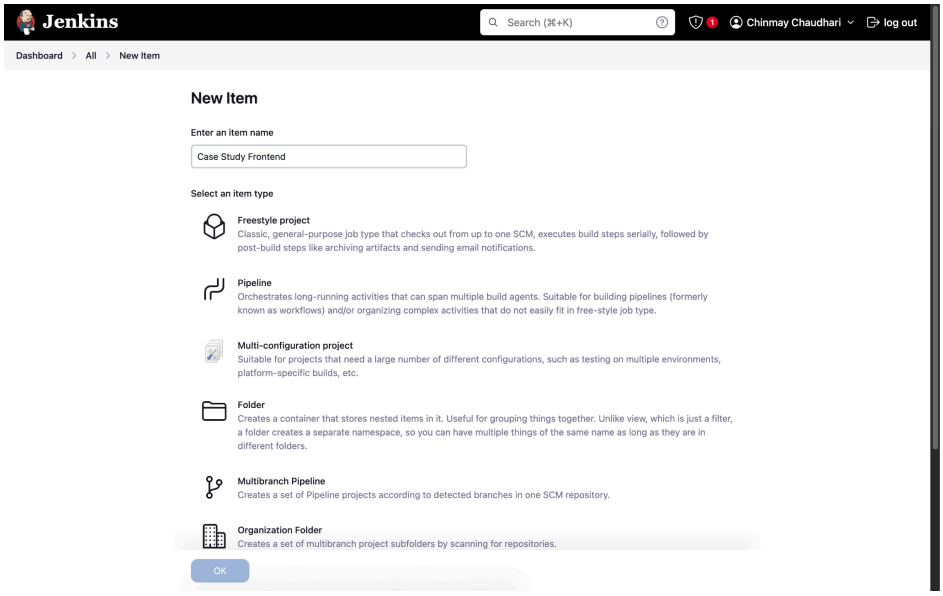
Step 4 - Click on build now and wait for successful build



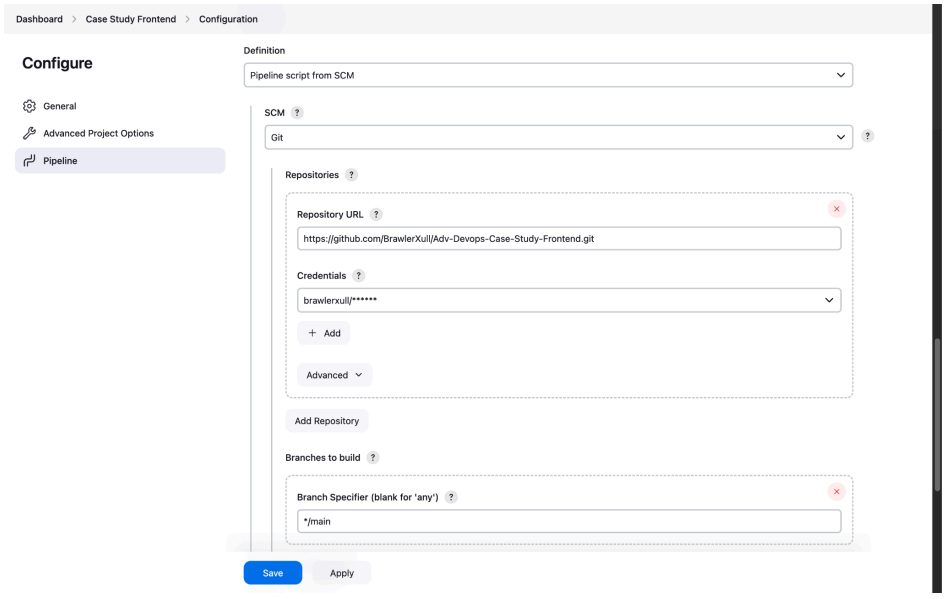
Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Part II - Frontend

Step 1 - Go To Jenkins And Start new Pipeline project



Step 2 - Select ‘Pipeline Script from SCM option’ and insert the url of git repo

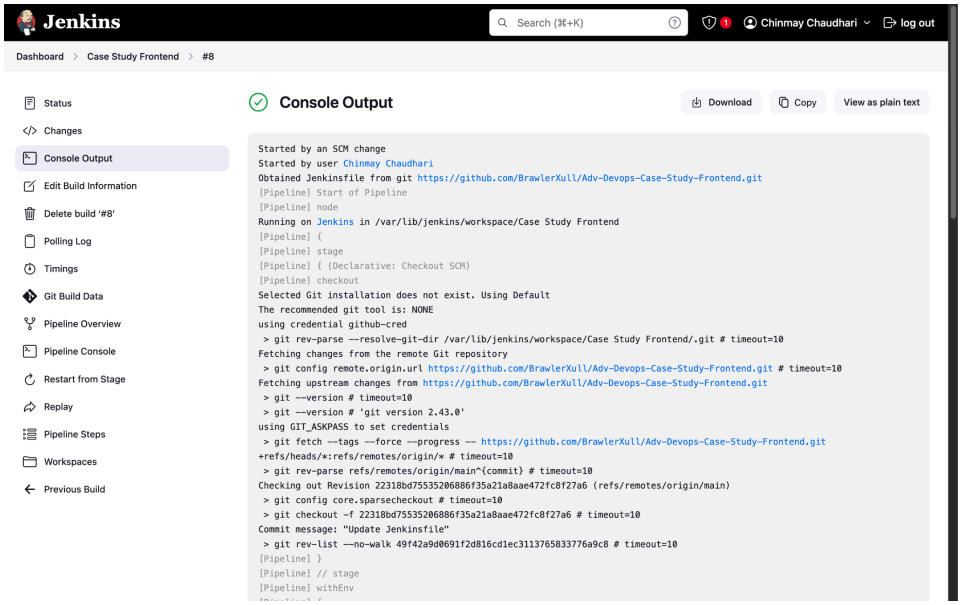


Step 3 - Add the below Jenkins file to the repo

```
pipeline {
   environment {
   imagename = "web"
   jenkinsProject = 'calculator-webapp-frontend'
 }

   agent any
   stages {

       stage('Git Staging'){

           steps{

               checkout([$class: 'GitSCM', branches: [[name: '*/main']], extensions:
[], userRemoteConfigs: [[credentialsId: 'github-cred', url:
'https://github.com/BrawlerXull/Adv-Devops-Case-Study-Frontend.git']]])

           }
       }


       stage('Build image and Run image ') {

           steps{
               sh 'sudo su - jenkins -s/bin/bash'
               //sh 'sudo docker stop $imagename'
               //sh 'sudo docker rm $imagename'
               //sh 'sudo docker rmi $imagename'
               sh 'sudo docker image build -t  $imagename .'

           }
       }

       stage('Run image ') {
           steps{
               sh 'sudo docker run -p81:80 --restart=always --name $imagename  -itd
$imagename'
           }
       }
   }
}
```

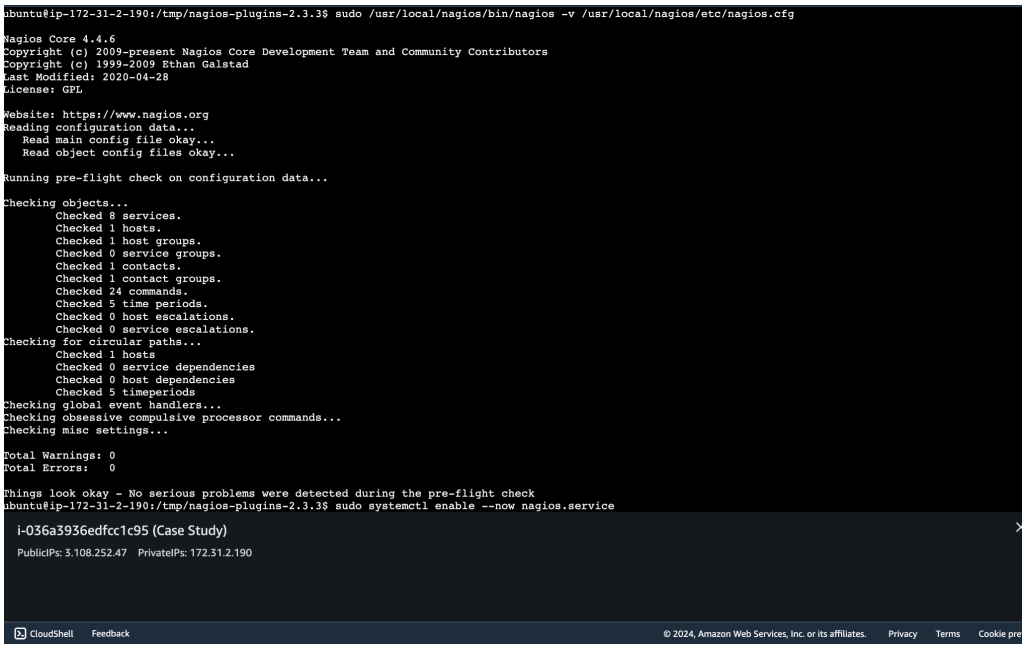
Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Step 4 - Click on build now and wait for successful build

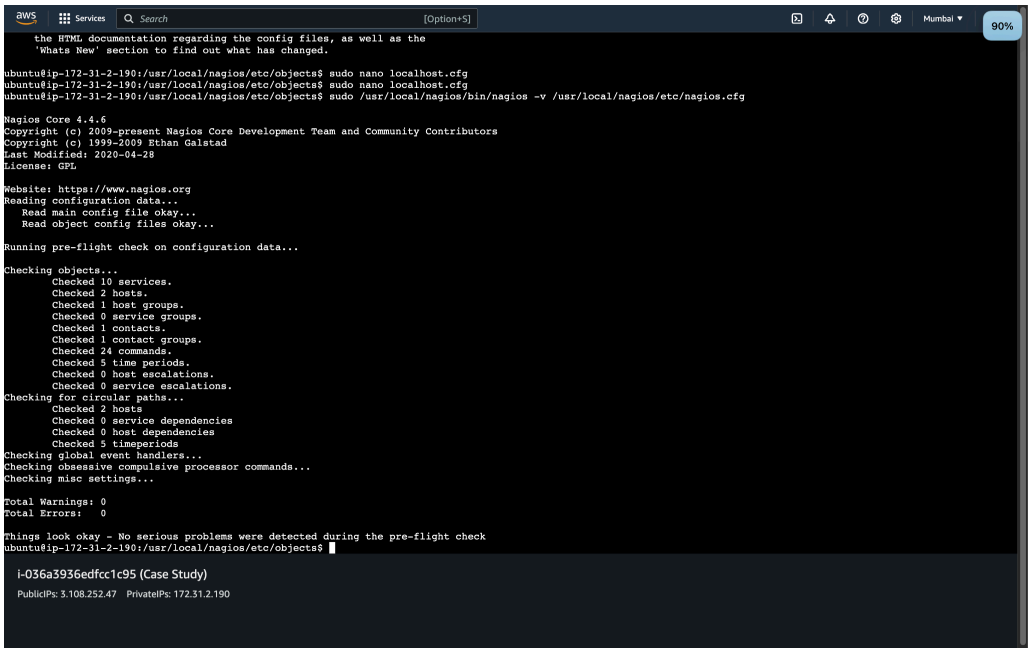


Part III - Nagios

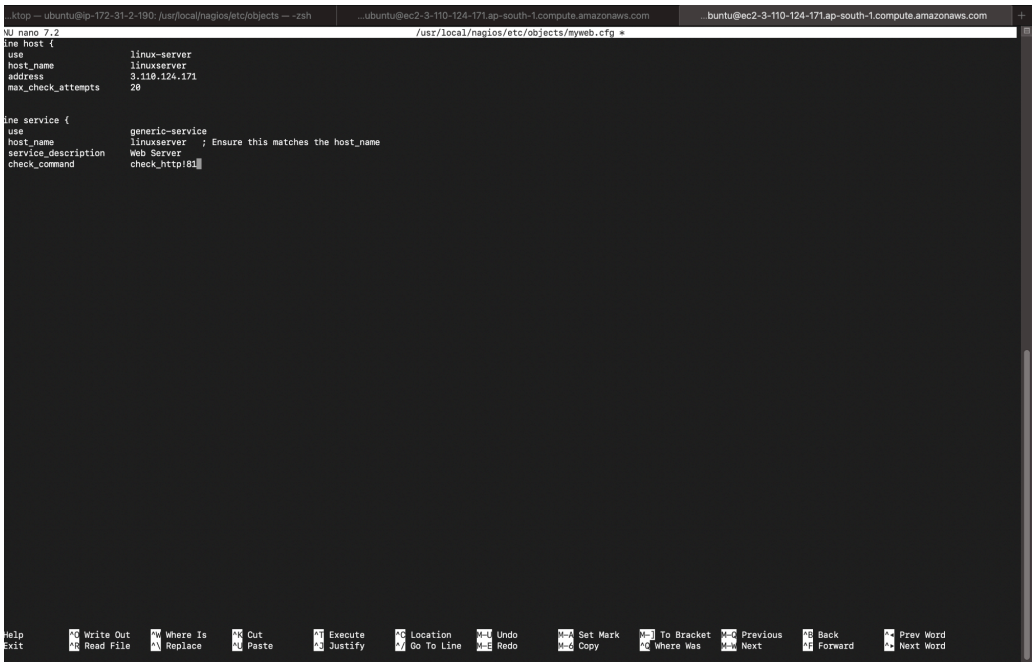
Step 1 - Install Nagios using the official documentation



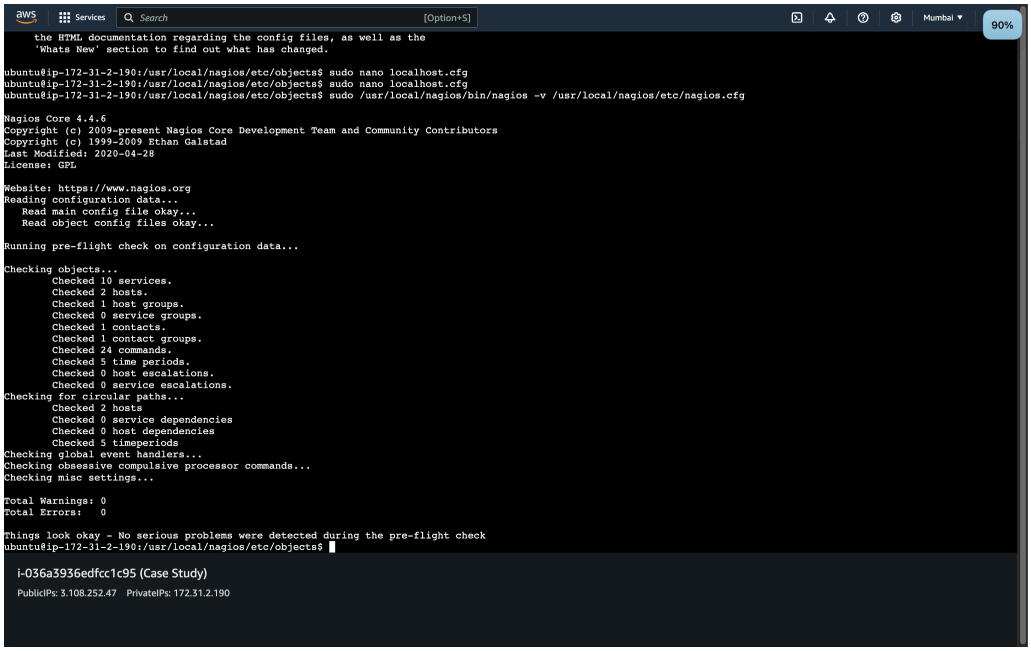
Step 2 - Run *sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg*



Step 3 - Setup the my myweb.cfg file by *sudo nano /usr/local/nagios/etc/objects/myweb.cfg*

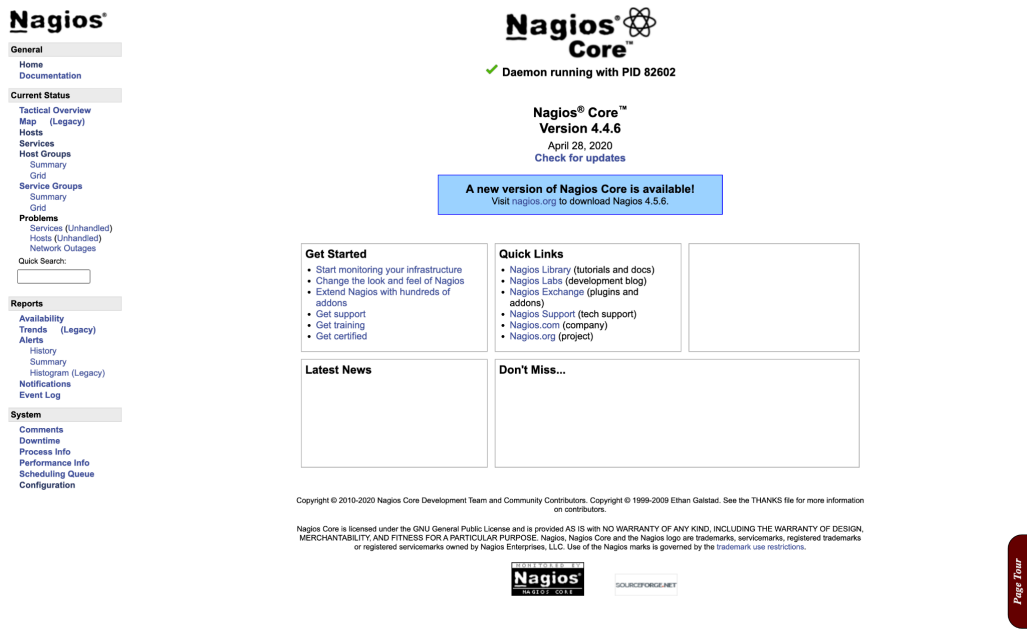


Step 4 - Again Run *sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg*

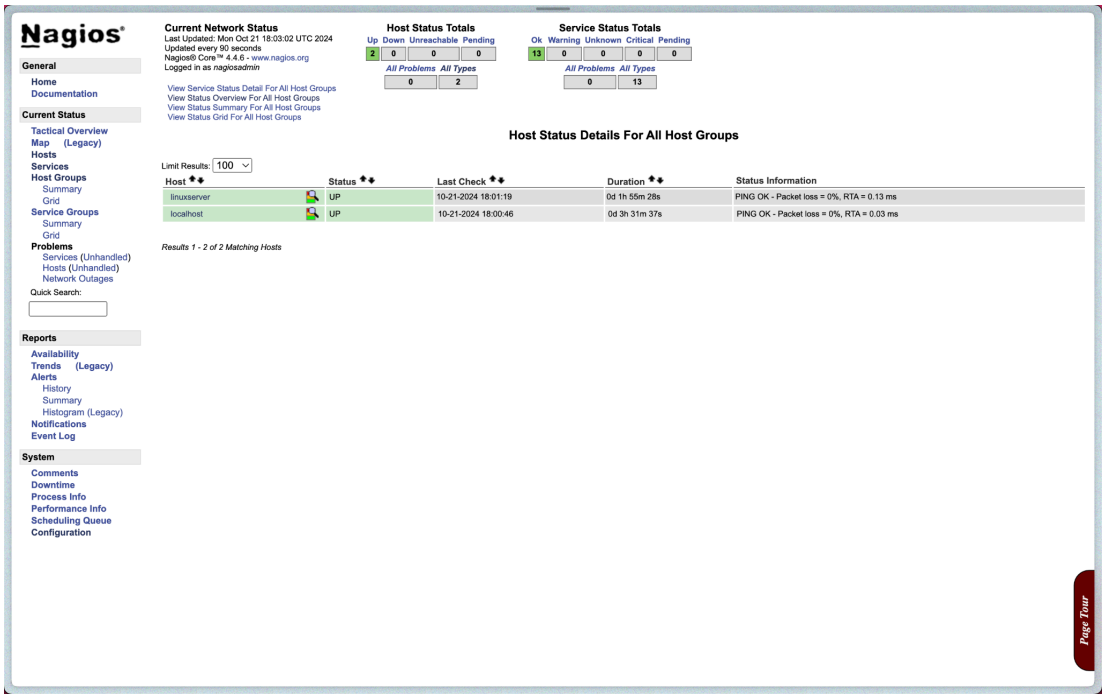


Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Step 5 - Run *sudo systemctl start nagios* **to start Nagios and go to** *https://<YourIP/nagios*

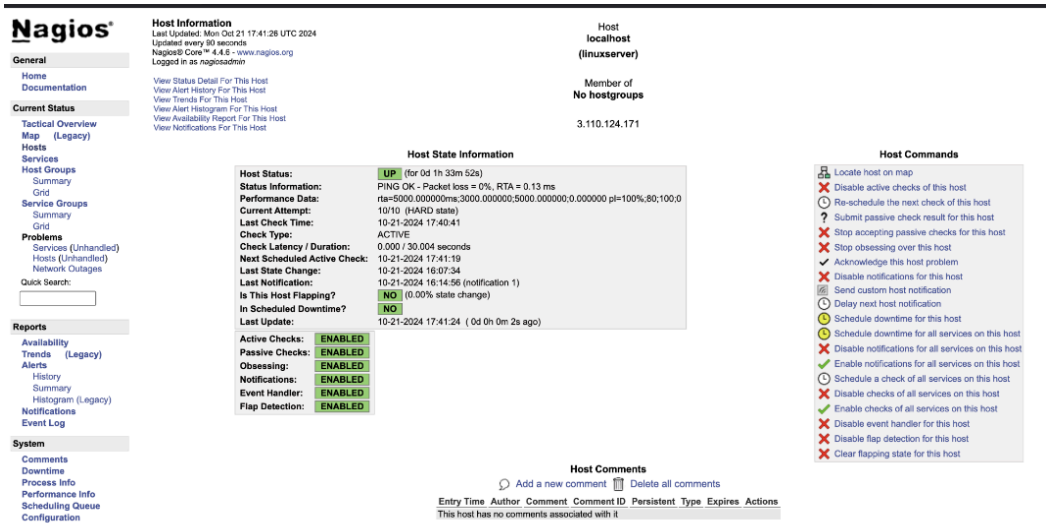


Step 6 - Go to host section from left side bar



Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Step 7 - Click on one of the service to see the details

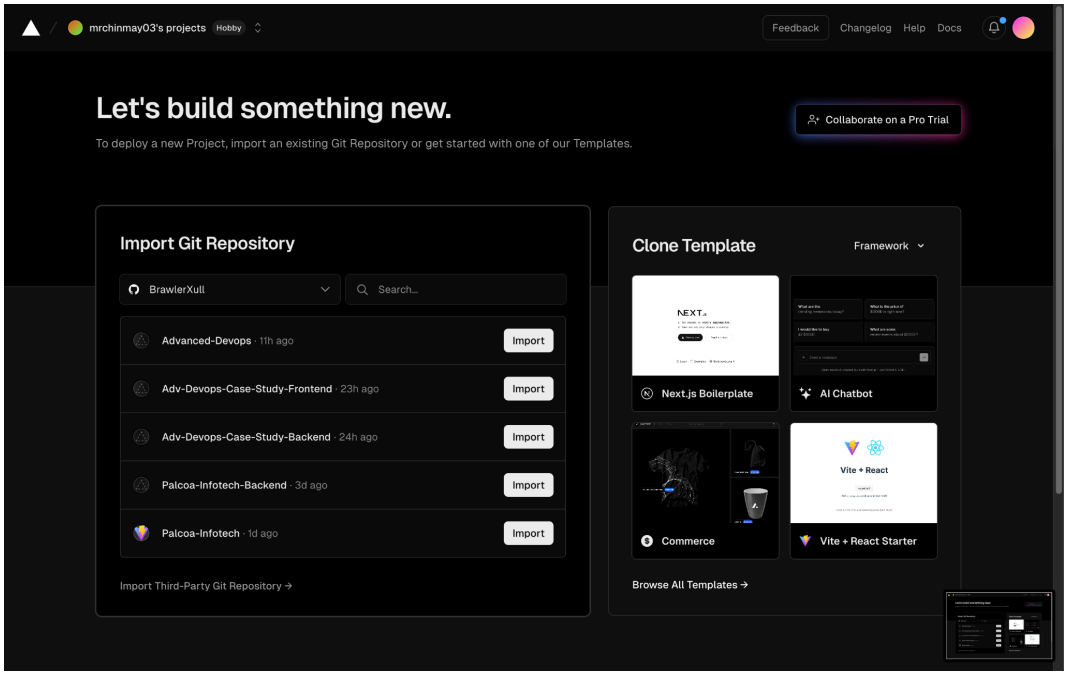


Step 8 - Go to services section to see the status of the services

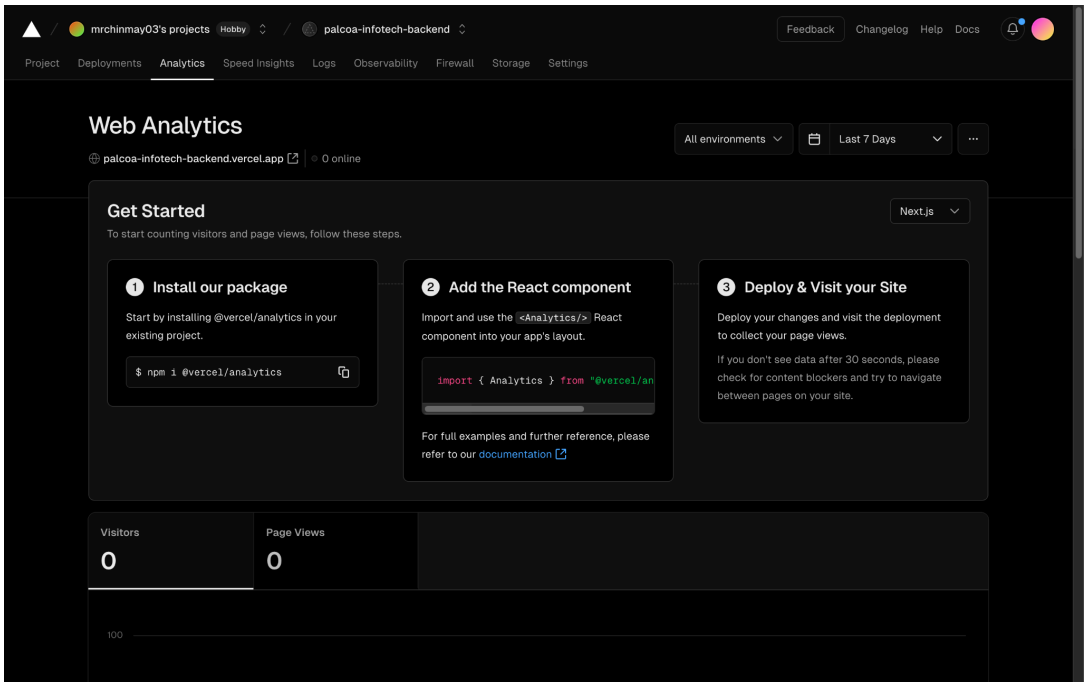
Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Part IV - Additional Integration (Mini Project)

Step 1 - Go to vercel.com login with github account and click on import project



Step 2 - Go to analytics section and follow the below steps

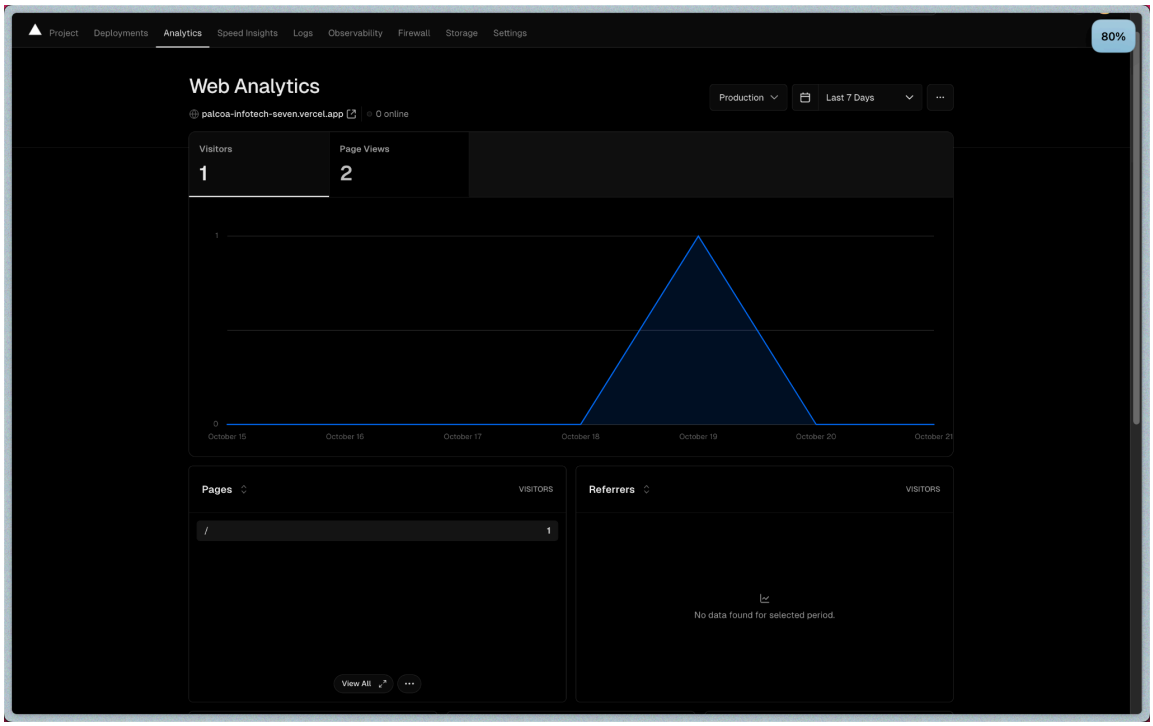


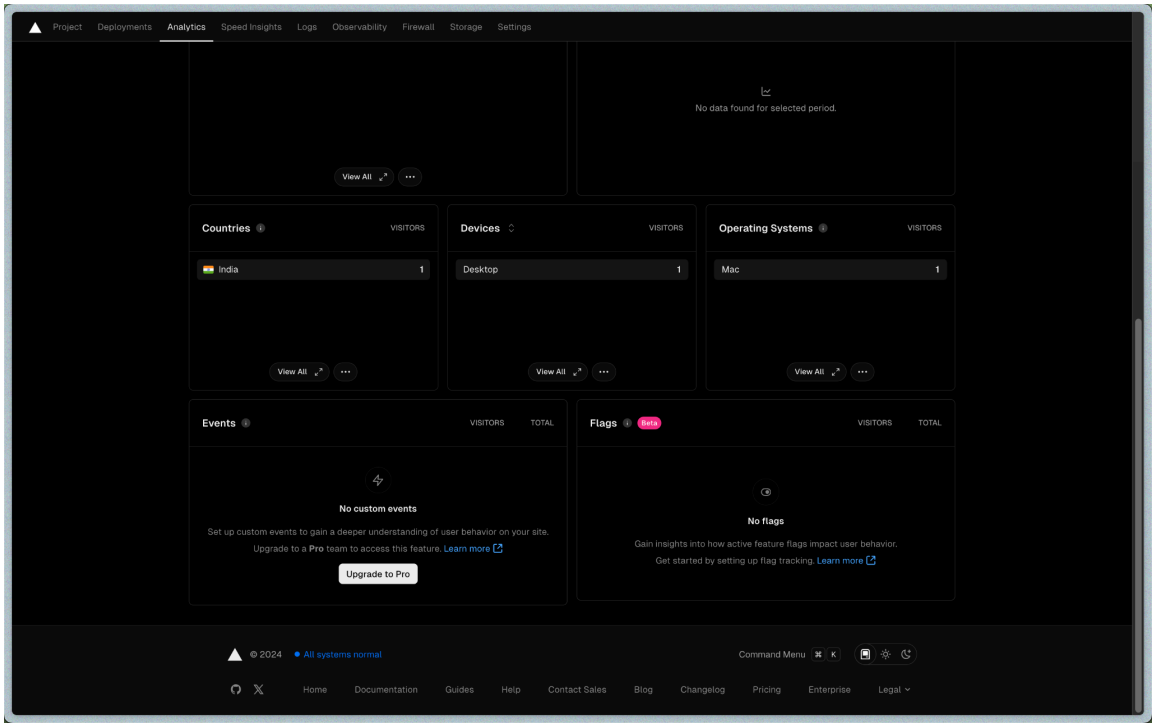
Step 3 - Install the Npm package by *npm i @vercel/analytics* **and insert below code in App.jsx**

```
import { BrowserRouter, Routes, Route } from "react-router-dom";
import { Toaster } from "sonner";
import SignIn from "./pages/SignIn";
import PrivateRoute from "./components/PrivateRoute";
import Dashboard from "./pages/Dashboard";
import About from "./pages/About";
import AddTask from "./pages/AddTask";
import swDev from "./swdev";
import { Analytics } from "@vercel/analytics/react"
export default function App() {
 return (
   <BrowserRouter>
     <Toaster position="top-right" richColors />
     <Routes>
     <Route path="/" element={<SignIn />} />
     <Route path="/about" element={<About />} />
       <Route element={<PrivateRoute />}>
         <Route path="/dashboard" element={<Dashboard />} />
         <Route path="/addtask" element={<AddTask />} />
       </Route>
     </Routes>
     <Analytics />
    </BrowserRouter>
 );
}
```

Name : Chinmay Chaudhari **Div : D15C** **Roll No : 6**

Step 4 - You will be able to see the Analytics of the project in the Analytics section





Outcome

Once this CI/CD pipeline is successfully implemented, the web application deployment will be smooth and agile since the time taken from development to production would be minimized. Moreover, Nagios monitoring provides instant visibility of an application's status in order for proactive issue management to take place.

Conclusion

In this following Experiment we learnt the below things

1. Setting up the Jenkins pipeline for deployment of web applications.
2. Using SCM Pipeline script while setting up the jenkins pipeline.
3. Integrating Nagios for Real Time monitoring of our web server.
4. Setting up custom config files for our Nagios project.
5. Integrating Vercel to view analytics of our frontend project.