

## Automated Deployment with Monitoring

### Introduction

This case study outlines the implementation of a Continuous Integration/Continuous Deployment (CI/CD) pipeline using Jenkins for a simple web application deployed on an Amazon EC2 instance. Additionally, we will set up Nagios for monitoring the application's availability, ensuring that any downtime or issues are promptly identified.

### Problem Statement

The objective is to automate the deployment process of a web application, enhancing efficiency and reducing the potential for human error during deployment. Furthermore, it's essential to implement a monitoring solution to keep track of the application's 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.

### 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:

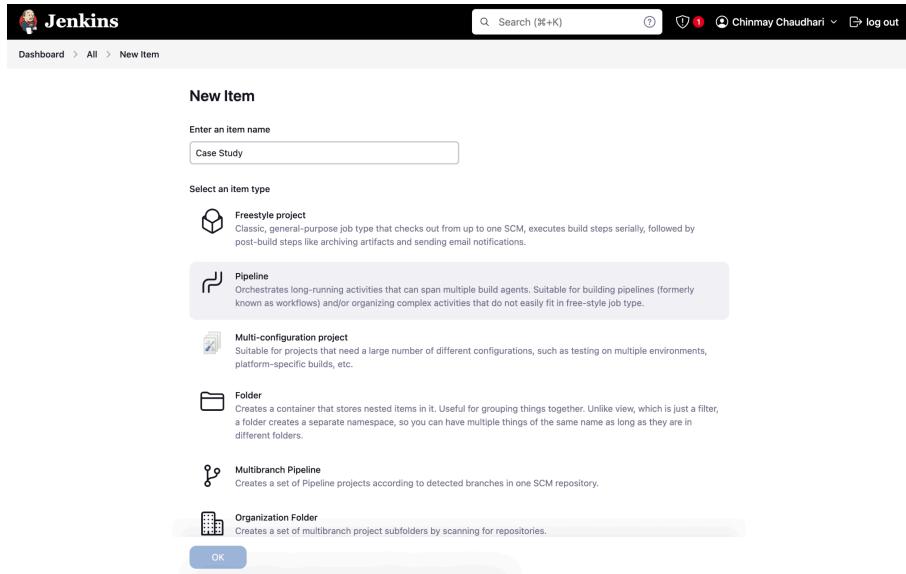
- Integrate Vercel for tracking user interactions and performance metrics.
- Use the data gathered from Vercel to inform future deployments and enhancements.

#### 5. Testing the Pipeline:

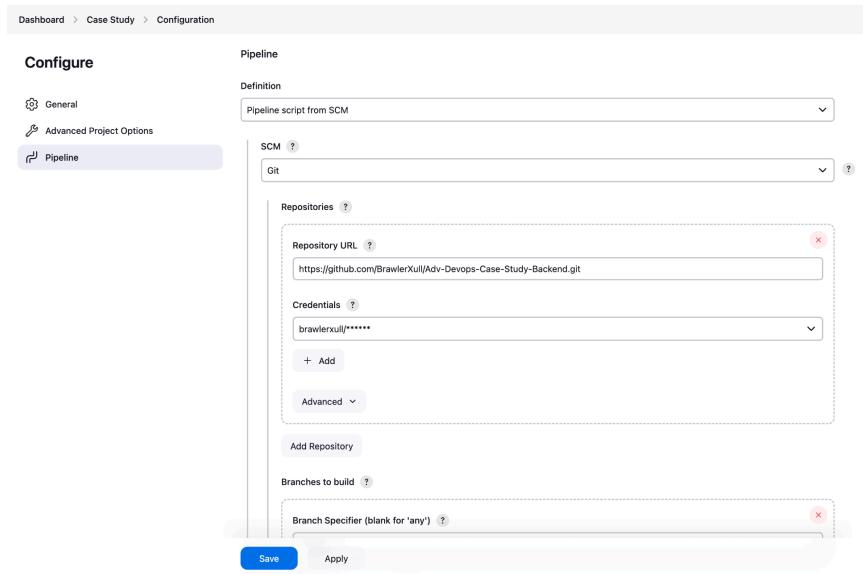
- Trigger a build in Jenkins to deploy the application.
- Verify the deployment by accessing the web application through a browser.

## 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']]]

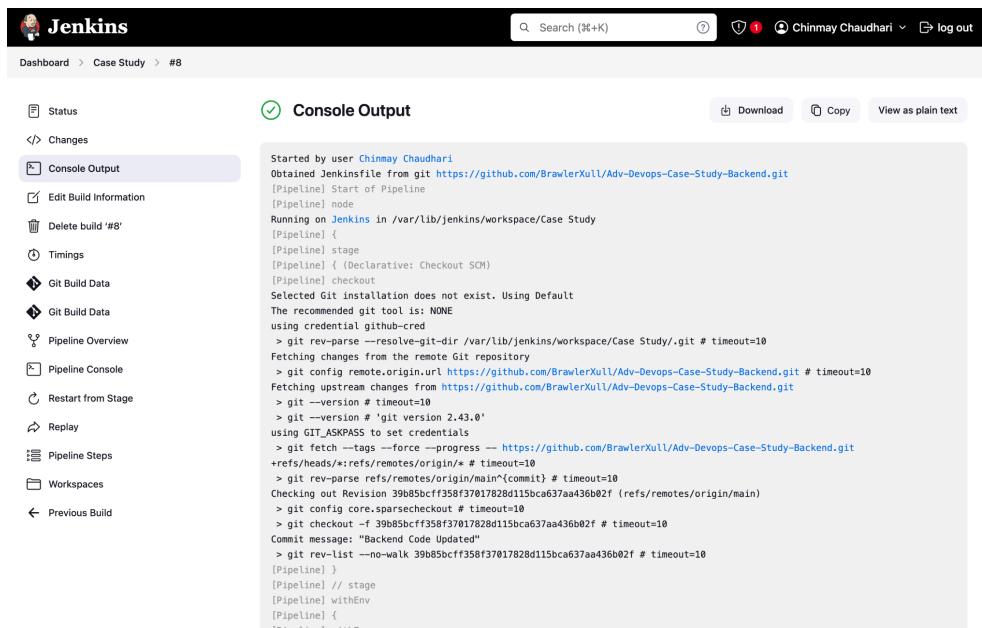
            }
        }

        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'
            }
        }
    }
}

```

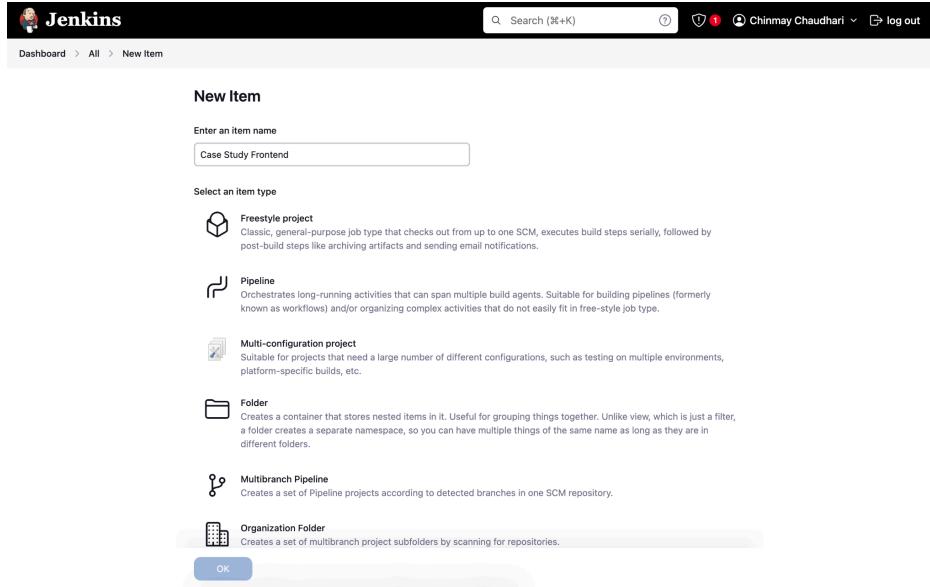
**Step 4 - Click on build now and wait for successful build**

The screenshot shows the Jenkins interface with the 'Console Output' tab selected. The log output is as follows:

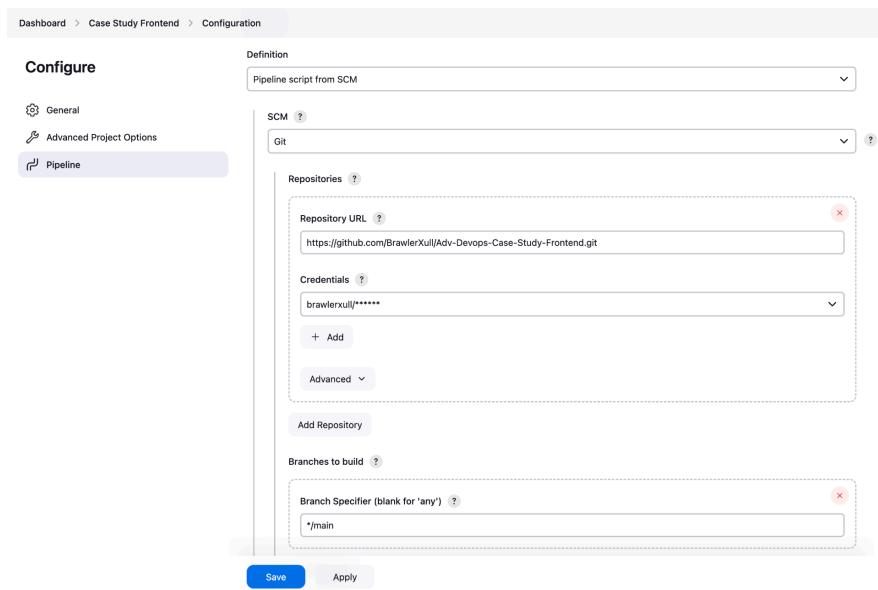
```
Started by user Chinmay Chaudhari
Obtained Jenkinsfile from git https://github.com/BrawlerXull/Adv-Devops-Case-Study-Backend.git
[Pipeline] Start of Pipeline
[Pipeline] node
[Pipeline] {
    Running on Jenkins in /var/lib/jenkins/workspace/Case Study
[Pipeline] stage
[Pipeline] {
    [Pipeline] {
        [Pipeline] {
            [Pipeline] {
                [Pipeline] {
                    Selected Git installation does not exist. Using Default
                    The recommended git tool is: NONE
                    using credential github-cred
                    > git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Case Study/.git # timeout=10
                    Fetching changes from the remote Git repository
                    > git config remote.origin.url https://github.com/BrawlerXull/Adv-Devops-Case-Study-Backend.git # timeout=10
                    Fetching upstream changes from https://github.com/BrawlerXull/Adv-Devops-Case-Study-Backend.git
                    > git --version # timeout=10
                    > git --version # 'git version 2.43.0'
                    using GIT_ASKPASS to set credentials
                    > git fetch --tags --force --progress -- https://github.com/BrawlerXull/Adv-Devops-Case-Study-Backend.git
                    +refs/heads/* refs/remotes/origin/* # timeout=10
                    > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
                    Checking out Revision 39b85bcff358f37017828d115bca637aa436b02f (refs/remotes/origin/main)
                    > git config core.sparsecheckout # timeout=10
                    > git checkout -f 39b85bcff358f37017828d115bca637aa436b02f # timeout=10
                    Commit message: "Backend Code Updated"
                    > git rev-list --no-walk 39b85bcff358f37017828d115bca637aa436b02f # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
```

## 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**

```

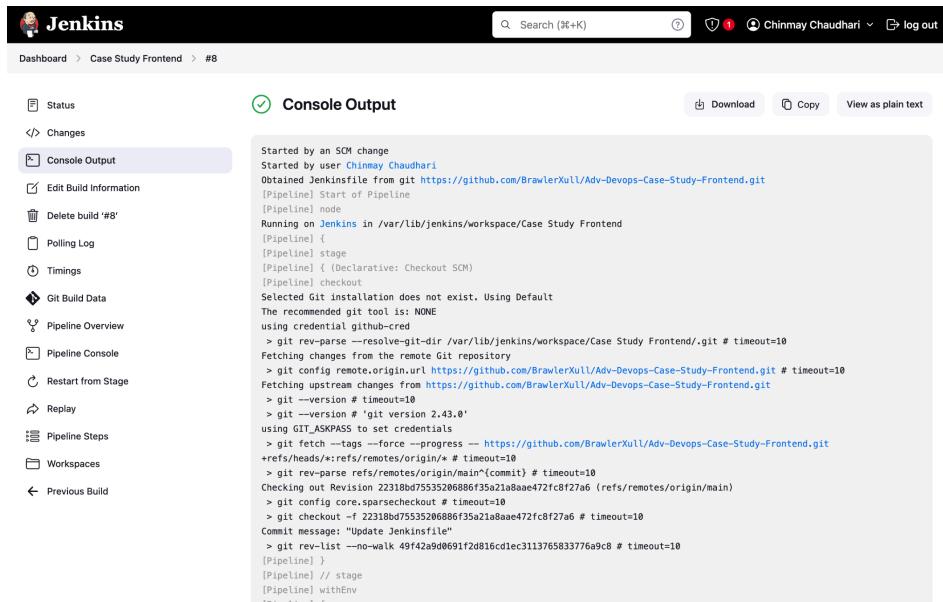
pipeline {
    environment {
        imagename = "web"
        jenkinsProject = 'calculator-webapp-frontend'
    }

    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'
            }
        }
    }
}

```

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

The screenshot shows the Jenkins interface with the 'Console Output' tab selected. The output window displays the log of a pipeline run. The log starts with the pipeline being triggered by an SCM change and user input. It details the cloning of the repository from GitHub, the execution of pipeline stages, and the final successful build message.

```
Started by an SCM change
Started by user Chinmay Chaudhari
Obtained Jenkinsfile from git https://github.com/BrawlerXull/Adv-Devops-Case-Study-Frontend.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/Case Study Frontend
[Pipeline] {
[Pipeline] stage
[Pipeline] {
  [Declarative: Checkout SCM]
  [Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential github-cred
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Case Study Frontend/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/BrawlerXull/Adv-Devops-Case-Study-Frontend.git # timeout=10
Fetching upstream changes from https://github.com/BrawlerXull/Adv-Devops-Case-Study-Frontend.git
> git --version # timeout=10
> git --version # 'git' version 2.43.0'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/BrawlerXull/Adv-Devops-Case-Study-Frontend.git
+refs/heads/* refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^(commit) # timeout=10
Checking out Revision 2231bd75535206886f35a21a8aae472fcf27ab (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 2231bd75535206886f35a21a8aae472fcf27ab # timeout=10
Commit message: "Update Jenkinsfile"
> git rev-list --no-walk 49f42a9d0691fd8316cd1ec3113765833776a9c8 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
```

## Part III - Nagios

### Step 1 - Install Nagios using the official documentation

```
ubuntu@ip-172-31-2-190:/tmp/nagios-plugins-2.3.3$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
License: GPL

Website: https://www.nagios.org
Reading configuration data...
  Read main config file okay...
  Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
  Checked 8 services.
  Checked 1 hosts.
  Checked 1 host groups.
  Checked 0 service groups.
  Checked 1 contacts.
  Checked 1 contact groups.
  Checked 24 commands.
  Checked 5 time periods.
  Checked 0 host escalations.
  Checked 0 service escalations.

Checking for circular paths...
  Checked 1 hosts
  Checked 0 service dependencies
  Checked 0 host dependencies
  Checked 5 timperiods

Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
ubuntu@ip-172-31-2-190:/tmp/nagios-plugins-2.3.3$ sudo systemctl enable --now nagios.service
i-036a3936edfcc1c95 (Case Study)
PublicIPs: 3.108.252.47 PrivateIPs: 172.31.2.190

CloudShell Feedback
```

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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

```
aws Services Q Search [Option+S] 90%
the HTML documentation regarding the config files, as well as the
'Whats New' section to find out what has changed.

ubuntu@ip-172-31-2-190:/usr/local/nagios/etc/objects$ sudo nano localhost.cfg
ubuntu@ip-172-31-2-190:/usr/local/nagios/etc/objects$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
License: GPL

Website: https://www.nagios.org
Reading configuration data...
  Read main config file okay...
  Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
  Checked 10 services.
  Checked 2 hosts.
  Checked 1 host groups.
  Checked 0 service groups.
  Checked 1 contacts.
  Checked 1 contact groups.
  Checked 24 commands.
  Checked 5 time periods.
  Checked 0 host escalations.
  Checked 0 service escalations.

Checking for circular paths...
  Checked 2 hosts
  Checked 0 service dependencies
  Checked 0 host dependencies
  Checked 5 timperiods

Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
ubuntu@ip-172-31-2-190:/usr/local/nagios/etc/objects$ i-036a3936edfcc1c95 (Case Study)
PublicIPs: 3.108.252.47 PrivateIPs: 172.31.2.190
```

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



```

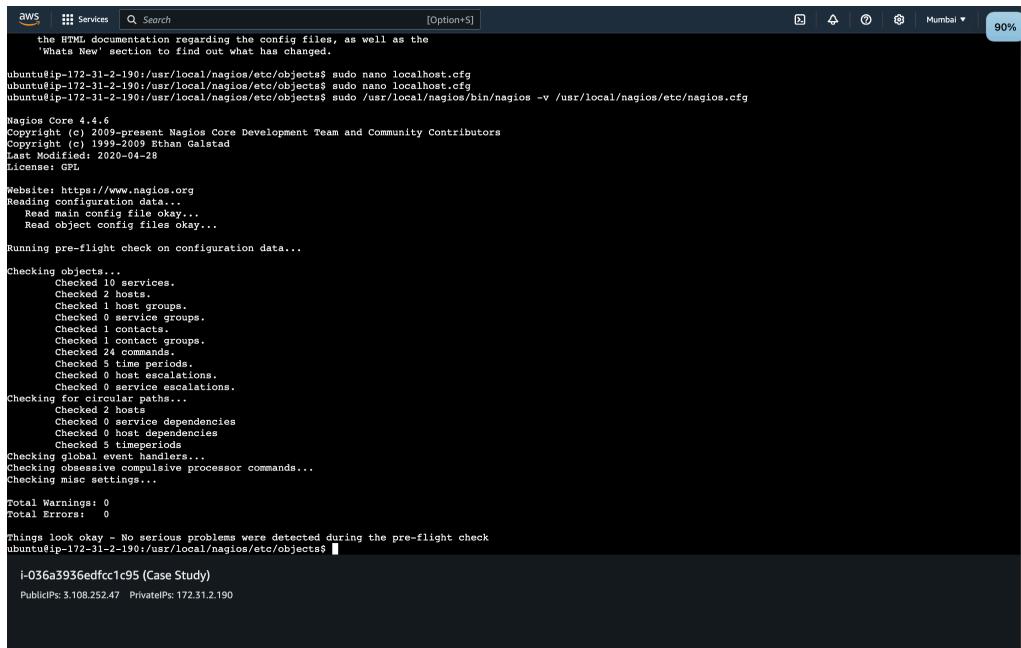
_ktop -- ubuntu@ip-172-31-2-190: /usr/local/nagios/etc/objects -- zsh  ...ubuntu@ec2-3-110-124-171.ap-south-1.compute.amazonaws.com  ...buntu@ec2-3-110-124-171.ap-south-1.compute.amazonaws.com +
Ubuntu 7.2
use host {
    use generic-service
    host_name linuxserver
    address 172.16.124.171
    max_check_attempts 20
}

define service {
    use generic-service
    host_name linuxserver ; Ensure this matches the host_name
    service_description Web Server
    check_command check_http[$]
}

Help   Write Out  Where Is  Cut  Execute  Location  Undo  Set Mark  To Bracket  Previous  Back  Forward  Prev Word
Exit   Read File  Replace  Paste  Justify  Go To Line  Redo  Where Was  Next  M-Back  M-Forward  Next Word

```

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



```

AWS Services Search [Option+S] Mumbai 90%
the HTML documentation regarding the config files, as well as the
'Whats New' section to find out what has changed.

ubuntu@ip-172-31-2-190:/usr/local/nagios/etc/objects$ sudo nano localhost.cfg
ubuntu@ip-172-31-2-190:/usr/local/nagios/etc/objects$ sudo nano localhost.cfg
ubuntu@ip-172-31-2-190:/usr/local/nagios/etc/objects$ sudo /usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg

Nagios Core 4.4.6
Copyright (c) 2009-present Nagios Core Development Team and Community Contributors
Copyright (c) 1999-2009 Ethan Galstad
Last Modified: 2020-04-28
License: GPL

Website: https://www.nagios.org
Reading configuration data...
  Read main config file okay...
  Read object config files okay...

Running pre-flight check on configuration data...

Checking objects...
  Checked 10 services.
  Checked 2 hosts.
  Checked 1 host groups.
  Checked 0 service groups.
  Checked 1 contacts.
  Checked 1 contact groups.
  Checked 1 event handlers.
  Checked 5 time periods.
  Checked 0 host escalations.
  Checked 0 service escalations.
Checking for circular paths...
  Checked 2 hosts
  Checked 0 host dependencies
  Checked 0 host dependencies
  Checked 5 timperiods
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors: 0

Things look okay - No serious problems were detected during the pre-flight check
ubuntu@ip-172-31-2-190:/usr/local/nagios/etc/objects$ i-036a3936edffcc1c95 (Case Study)
PublicIPs: 5.108.252.47 PrivateIPs: 172.31.2.190

```

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

The screenshot shows the Nagios Core 4.4.6 dashboard. At the top right, it says "Daemon running with PID 82602". Below that, the version information is displayed: "Nagios® Core™ Version 4.4.6 April 28, 2020 Check for updates". A blue banner at the top center reads "A new version of Nagios Core is available! Visit [nagios.org](http://nagios.org) to download Nagios 4.5.6." On the left sidebar, there are sections for General, Current Status, Reports, and System. The Current Status section includes a "Get Started" box with links to monitoring infrastructure, look-and-feel, and add-ons. The Reports section includes Availability, Trends (Legacy), Alerts, History, Summary, Histogram (Legacy), Notifications, and Event Log. The System section includes Comments, Downtime, Process Info, Performance Info, Scheduling Queue, and Configuration.

**Step 6 - Go to host section from left side bar**

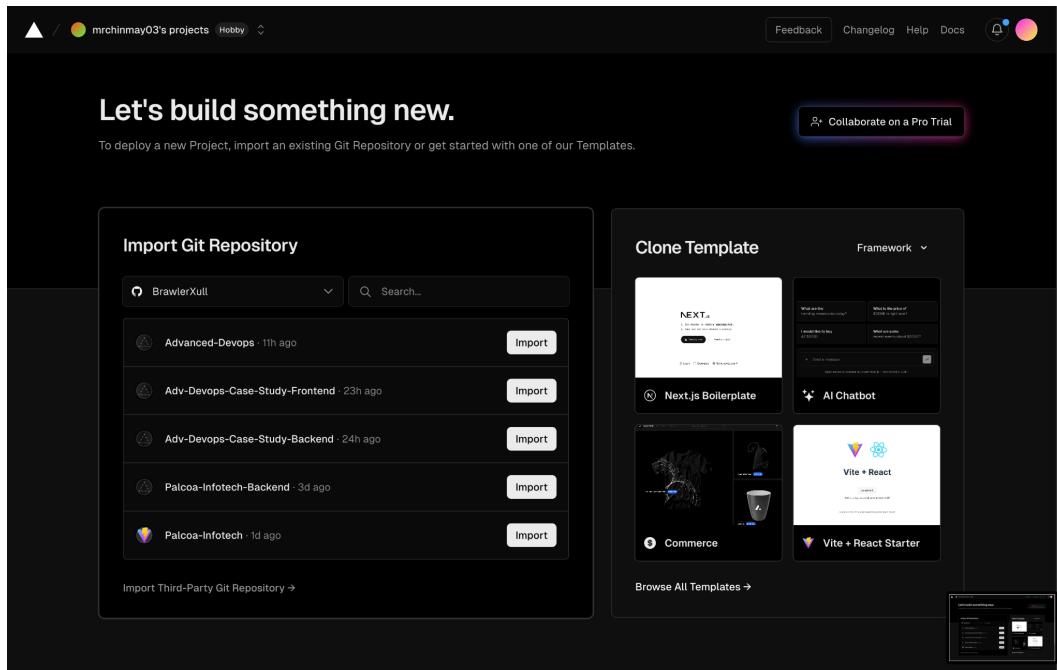
The screenshot shows the "Host Status Details For All Host Groups" page. It displays two hosts: "lunixserver" and "localhost". Both hosts are marked as "UP". The "Status Information" column indicates "PING OK - Packet loss = 0%, RTA = 0.13 ms" for "lunixserver" and "PING OK - Packet loss = 0%, RTA = 0.03 ms" for "localhost". The page also shows "Host Status Totals" and "Service Status Totals" tables. The "Host Status Totals" table shows 2 hosts up, 0 down, 0 unreachable, and 0 pending. The "Service Status Totals" table shows 13 services ok, 0 warning, 0 unknown, 0 critical, and 0 pending. The left sidebar is identical to the one in the previous screenshot, showing General, Current Status, Reports, and System sections.

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

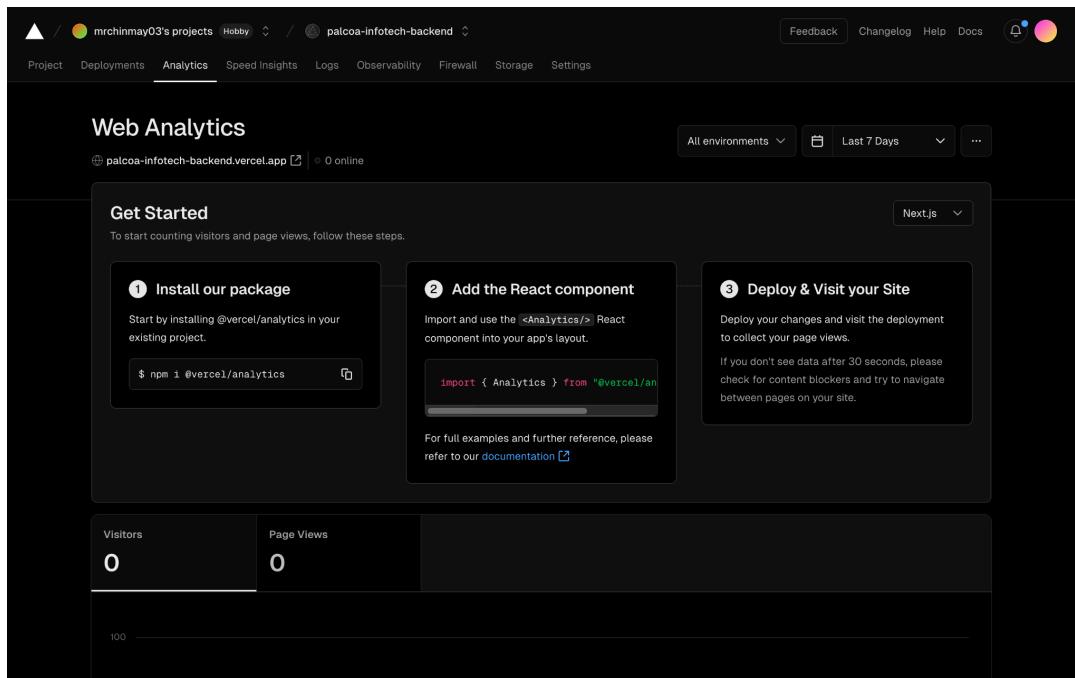
## Step 8 - Go to services section to see the status of the services

## Part IV - Additional Integration (Mini Project)

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



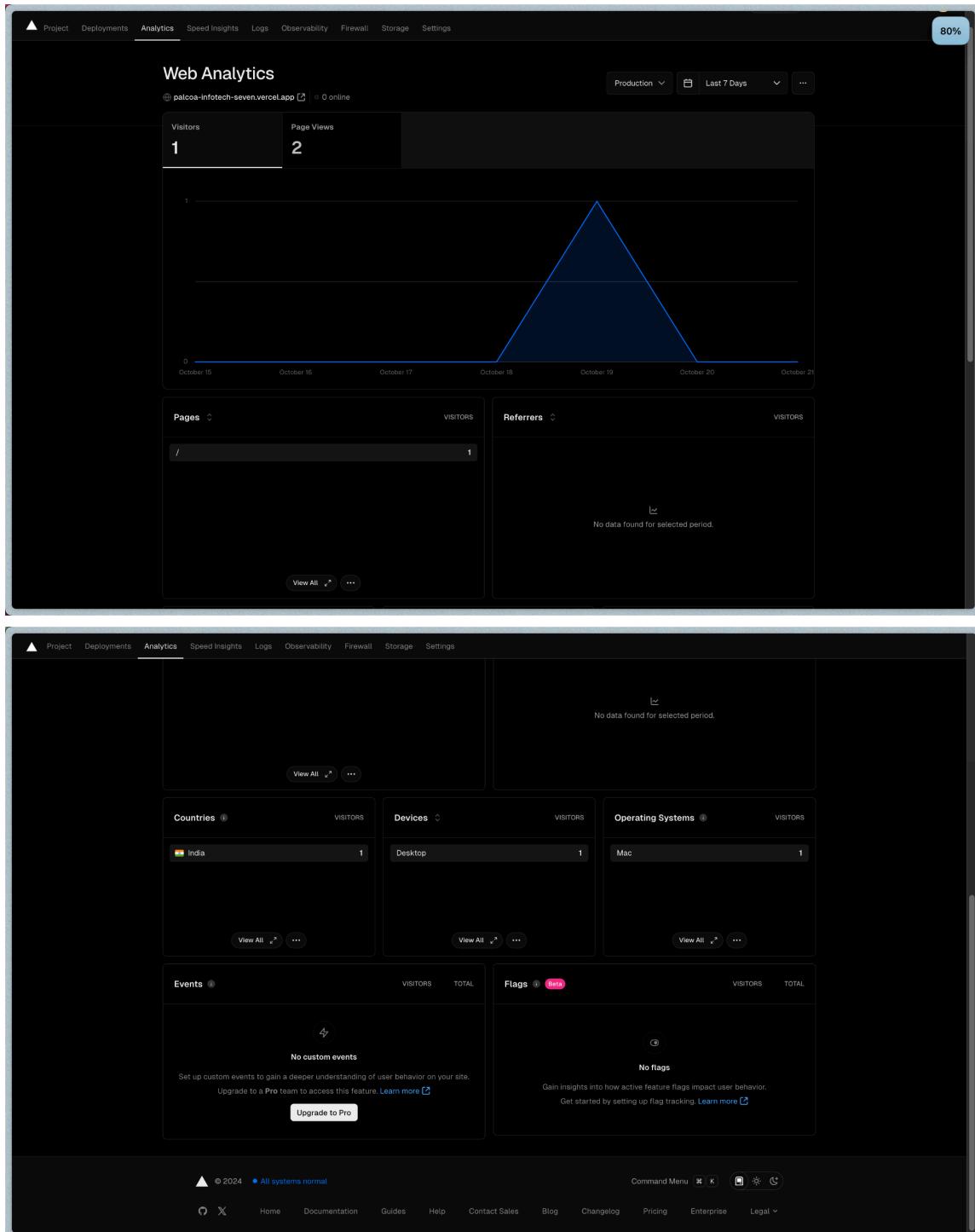
**Step 2 - Go to analytics section and follow the below steps**



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>
  );
}
```

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



## **Outcome**

The successful implementation of this CI/CD pipeline results in a streamlined process for deploying web applications, reducing the time from development to production. The addition of Nagios monitoring provides immediate visibility into the application's status, allowing for proactive management of any issues.

## **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.