

# 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:**

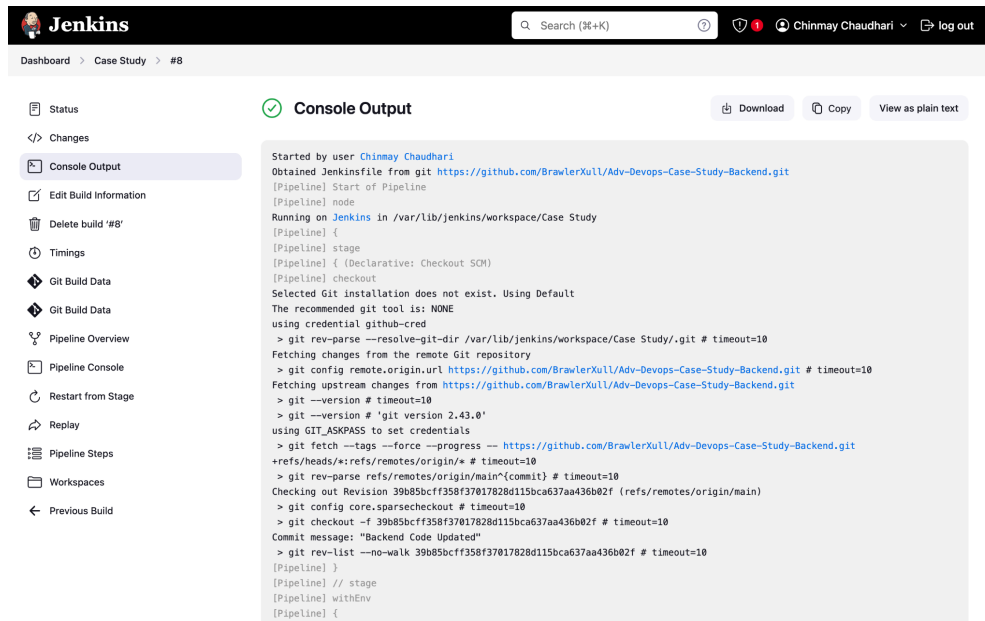
- Integrate Vercel for tracking user interactions and performance metrics.
- Use the data gathered from Vercel to inform future deployments and enhancements.

### **5. Testing the Pipeline:**

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

## Part I - Backend

Setup a Jenkins Pipeline to automate the process of deployment of our web app's backend

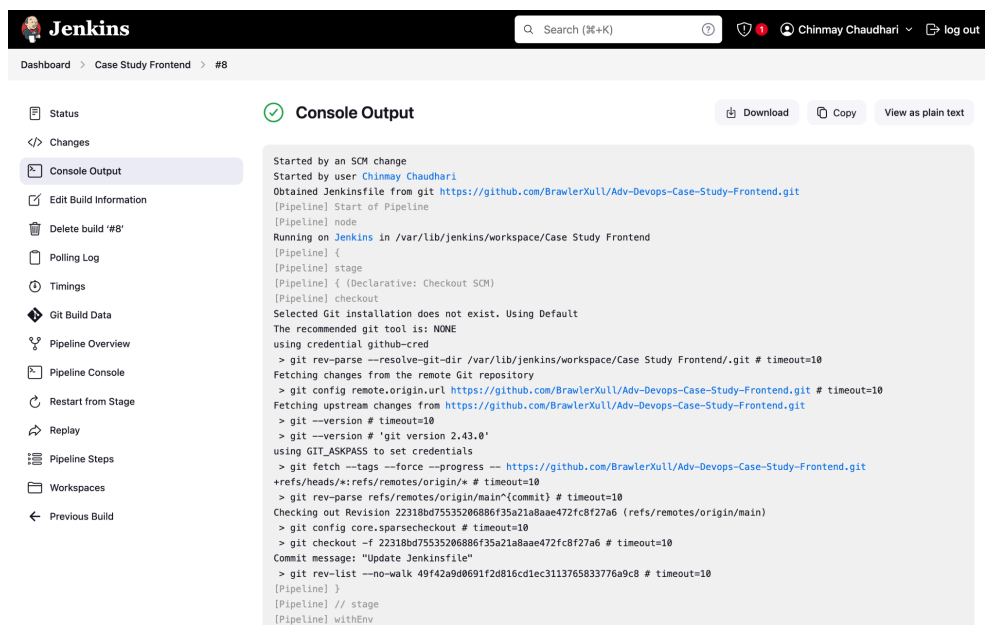


The screenshot shows the Jenkins web interface for a pipeline named 'Case Study'. The 'Console Output' tab is selected, displaying the execution log. The log starts with 'Started by user Chinmay Chaudhari' and 'Obtained Jenkinsfile from git https://github.com/BrawlerXull/Adv-Devops-Case-Study-Backend.git'. It then shows the pipeline starting, running on Jenkins in the workspace, and checking out the code from the remote repository. The log includes commands for git rev-parse, git config, git fetch, git rev-parse, git checkout, and git rev-list, along with their respective outputs and timestamps.

```
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
Running on Jenkins in /var/lib/jenkins/workspace/Case Study
[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/.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

Setup a Jenkins Pipeline to automate the process of deployment of our web app's frontend.



The screenshot shows the Jenkins web interface for a pipeline named 'Case Study Frontend'. The 'Console Output' tab is selected, displaying the execution log. The log starts with 'Started by an SCM change' and 'Started by user Chinmay Chaudhari'. It then shows the pipeline starting, running on Jenkins in the workspace, and checking out the code from the remote repository. The log includes commands for git rev-parse, git config, git fetch, git rev-parse, git checkout, and git rev-list, along with their respective outputs and timestamps.

```
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 22318bd75535206886f35a21a8aae472fc8f27a6 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 22318bd75535206886f35a21a8aae472fc8f27a6 # timeout=10
Commit message: "Update Jenkinsfile"
> git rev-list --no-walk 49f42a9d0691f2d816cd1ec3113765833776a9c8 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
```

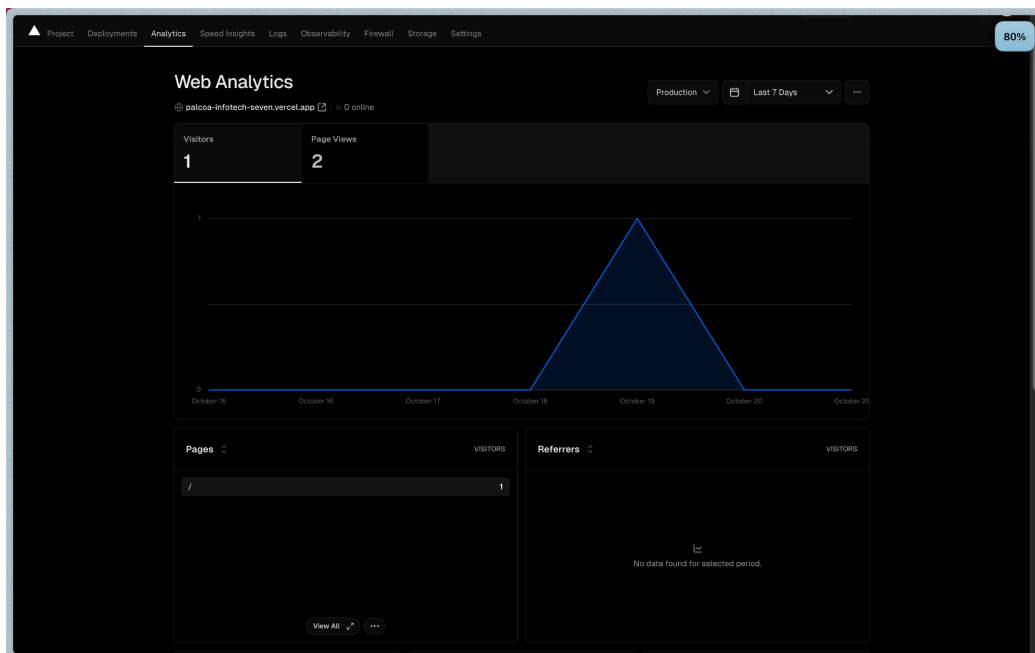
## Part III - Nagios

Install Nagios and go to hosts section so see the running webapps

The screenshot displays the Nagios web interface. On the left is a sidebar with navigation links: General, Home, Documentation, Current Status, Tactical Overview, Map (Legacy), Hosts, Services, Host Groups, Summary, Grid, Service Groups, Summary, Grid, Problems, Services (Unhandled), Hosts (Unhandled), Network Outages, Quick Search, Reports, Availability, Trends (Legacy), Alerts, History, Summary, Histogram (Legacy), Notifications, Event Log, System, Comments, Downtime, Process Info, Performance Info, Scheduling Queue, and Configuration. The main content area shows the 'Hosts' section. At the top, it displays 'Current Network Status' (Last Updated: Mon Oct 21 18:03:02 UTC 2024), 'Host Status Totals' (Up: 2, Down: 0, Unreachable: 0, Pending: 0), and 'Service Status Totals' (OK: 15, Warning: 0, Unknown: 0, Critical: 0, Pending: 0). Below this is a table titled 'Host Status Details For All Host Groups' showing two hosts: 'linuxserver' and 'localhost', both with a status of 'UP'. The table includes columns for Host, Status, Last Check, Duration, and Status Information. A 'Page Tour' button is visible in the bottom right corner.

## Part IV - Additional Integration (Mini Project)

Login to vercel using your github account and setup the analytics for our webapp.



## **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.