CASE STUDY

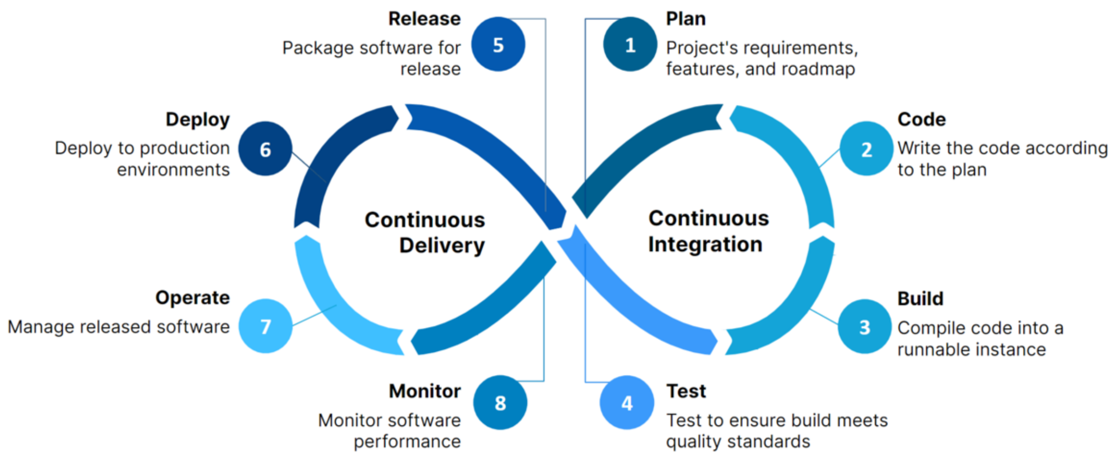
Aim: Automated Deployment with Monitoring

- Concepts Used: Jenkins, EC2, Nagios.
- Problem Statement: "Set up a Jenkins CI/CD pipeline to deploy a simple web application on an EC2 instance. Configure Nagios to monitor the deployed application's availability."
- Tasks:
  - Create a Jenkins pipeline that builds and deploys a sample web app to an EC2 instance.
  - Install and configure Nagios to monitor the HTTP status of the deployed application.
  - Verify the pipeline by triggering a build and checking the monitoring status in Nagios.

Theory:

Introduction

Automated deployment is a critical aspect of modern software development practices that emphasizes agility, reliability, and efficiency. In today's fast-paced tech environment, where frequent updates and quick releases are expected, organizations must streamline their development processes to maintain a competitive edge. Automated deployment allows teams to manage their software lifecycle effectively, enabling them to deliver updates and new features quickly and reliably.


Release
5
Package software for release
Plan
1
Project's requirements, features, and roadmap
Deploy
6
Deploy to production environments
Code
2
Write the code according to the plan
Continuous Delivery
Continuous Integration
Operate
7
Manage released software
Build
3
Compile code into a runnable instance
Monitor
8
Monitor software performance
Test
4
Test to ensure build meets quality standards

Continuous Integration and Continuous Deployment (CI/CD) :are foundational practices in achieving automated deployment. CI involves automatically testing and integrating code changes from multiple contributors into a shared repository, while CD focuses on automatically deploying these integrated changes to production environments. By leveraging CI/CD pipelines, development teams can automate the entire process of building, testing, and deploying applications. This automation minimizes manual errors, reduces the time required for deployments, and enhances overall software quality.

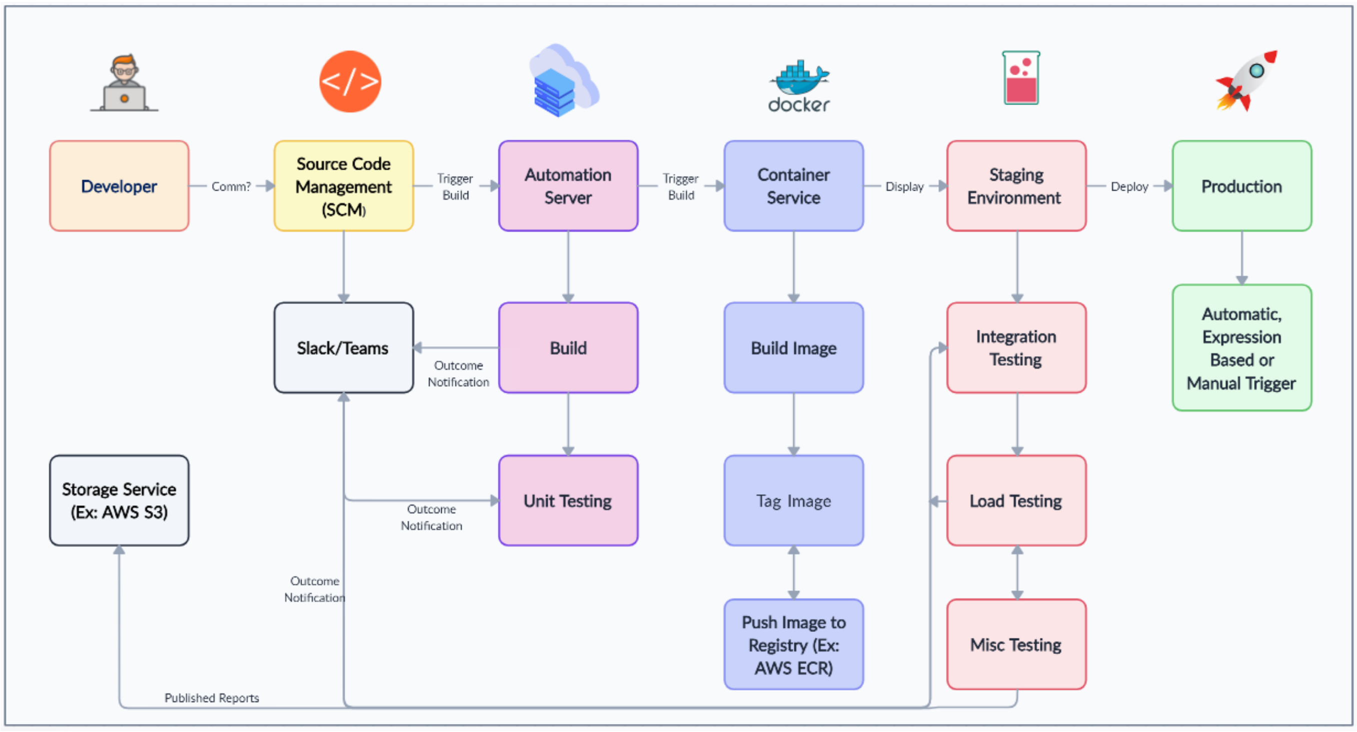
Concepts Used

1. **Jenkins**:
   - Jenkins is an open-source automation server that facilitates continuous integration and continuous delivery of software projects. It enables developers to automate various stages of development, such as building, testing, and deploying applications.
   - Jenkins supports a rich ecosystem of plugins that extend its capabilities, allowing seamless integration with various tools, including version control systems, build tools, and deployment platforms.

2. **EC2 (Elastic Compute Cloud)**:
   - Amazon EC2 is a web service that provides resizable compute capacity in the cloud. It allows users to launch virtual servers (instances) on demand, offering flexibility in scaling resources based on application needs.
   - EC2 instances can run various operating systems and are commonly used for hosting applications, websites, and services.
3. **Nagios**:
   - Nagios is an open-source monitoring system that provides comprehensive monitoring of applications, servers, and network infrastructure. It helps ensure system reliability by tracking the status of services and notifying administrators of any issues.
   - Nagios supports plugins that allow users to monitor various types of resources, including HTTP status, system metrics, and performance data.

Importance of CI/CD in Automated Deployment

1. **Speed and Efficiency**:
   Automated pipelines reduce the time to deploy code changes, allowing teams to iterate rapidly and respond to feedback effectively.
2. **Consistency and Reliability**:
   Automation ensures each release is uniform and dependable, minimizing errors associated with manual deployments.
3. **Rapid Recovery from Failures**:
   CI/CD pipelines can quickly revert to stable versions after a failure, reducing downtime and enhancing user experience.
4. **Improved Collaboration**:
   CI/CD fosters teamwork between development and operations, promoting shared responsibility and better communication across teams.



Integrating Monitoring into Automated Deployment

While automated deployment enhances the speed and reliability of software delivery, continuous monitoring is equally vital to ensure that applications perform as expected once deployed. **Monitoring** provides insights into the application's health, performance, and availability, allowing teams to proactively address issues before they impact users.

Nagios as a Monitoring Solution

1. **Real-Time Monitoring**:
   Nagios delivers real-time insights into application and infrastructure status, enabling teams to swiftly detect and address potential issues.
2. **Alerting and Notifications**:
   It sends alerts for problems like server downtime and performance errors, allowing teams to respond promptly and minimize user impact.
3. **Historical Data Analysis**:
   Nagios collects and analyses performance data over time, helping teams identify trends and optimize application scaling and efficiency.

<u>Real-World Use Cases of Automated Deployment with Monitoring:</u>

1. Netflix: Continuous Delivery and Monitoring

Scenario:
Netflix operates one of the largest streaming platforms globally, serving millions of users simultaneously. To handle the enormous scale and provide uninterrupted service, Netflix employs a highly automated deployment pipeline.

Implementation:

- Netflix utilizes Spinnaker, an open-source multi-cloud continuous delivery platform, to automate the deployment of its microservices architecture.
- The company combines this with robust monitoring tools like Atlas for real-time metrics and Hystrix for circuit-breaking to manage service failures.

Outcomes:

- Netflix can deploy hundreds of changes daily with minimal downtime, showcasing an impressive 99.99% uptime.
- The automation enables rapid recovery from failures, with incident response times significantly reduced to just minutes.

2. GitHub: Continuous Deployment for Developer Tools

Scenario:
GitHub, a platform for version control and collaboration, continuously enhances its features and performance. The company relies on automated deployment and monitoring to ensure a seamless user experience.

Implementation:

- GitHub employs Circle CI for continuous integration and delivery, automating tests and deployments for new features.
- They utilize Prometheus for monitoring application performance and identifying potential issues in real time.

Outcomes:

- GitHub successfully deploys updates to its platform multiple times a day, enabling them to respond swiftly to user feedback and improve functionalities.
- Their monitoring capabilities have led to a 25% increase in performance metrics, resulting in higher user engagement and satisfaction.

Future Trends:

1.. Self-Healing Applications:

- **Overview**: Future applications will increasingly incorporate self-healing capabilities, automatically recovering from failures. Monitoring tools like Nagios will work alongside these systems to ensure quick remediation.
- **Impact**: Self-healing applications will enhance resilience, allowing for minimal disruption to users during outages. Automated deployments will be coupled with mechanisms for self-repair, leading to higher reliability

2. ***AI and Machine Learning for Predictive Monitoring***:

- **Overview:** AI and machine learning algorithms will be leveraged in monitoring tools like Nagios to analyse historical data and predict potential issues before they occur.
- **Impact:** Predictive analytics will help teams proactively address performance bottlenecks or failures, reducing downtime and improving overall system reliability.

3. ***Multi-Cloud Deployments***:

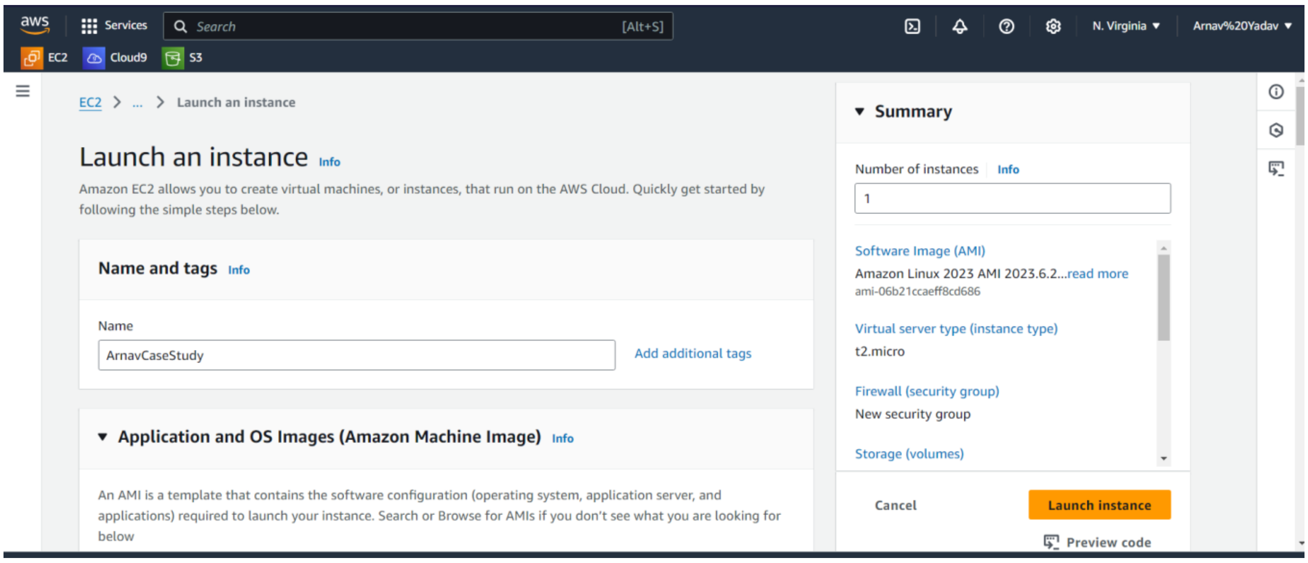
- **Overview**: Organizations will adopt multi-cloud strategies to leverage the best features of different cloud providers. Jenkins pipelines will be configured to deploy applications across various clouds, including AWS, Azure, and Google Cloud.
- **Impact**: This flexibility will enhance disaster recovery options and allow businesses to avoid vendor lock-in, providing more resilient and scalable deployment solutions

Implementation Steps:

Prerequisites:

1. **AWS Account:** Ensure you have an AWS account.
2. **Basic Knowledge of Git:** Understand how to use Git for version control.
3. **Familiarity with Linux:** You'll be working on an EC2 instance, so basic Linux commands are necessary.
4. **Jenkins Installed:** Install Jenkins on a server (or locally) to get started

Step 1: Set Up Your EC2 Instance


EC2 > ... > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name: ArnavCaseStudy

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Summary

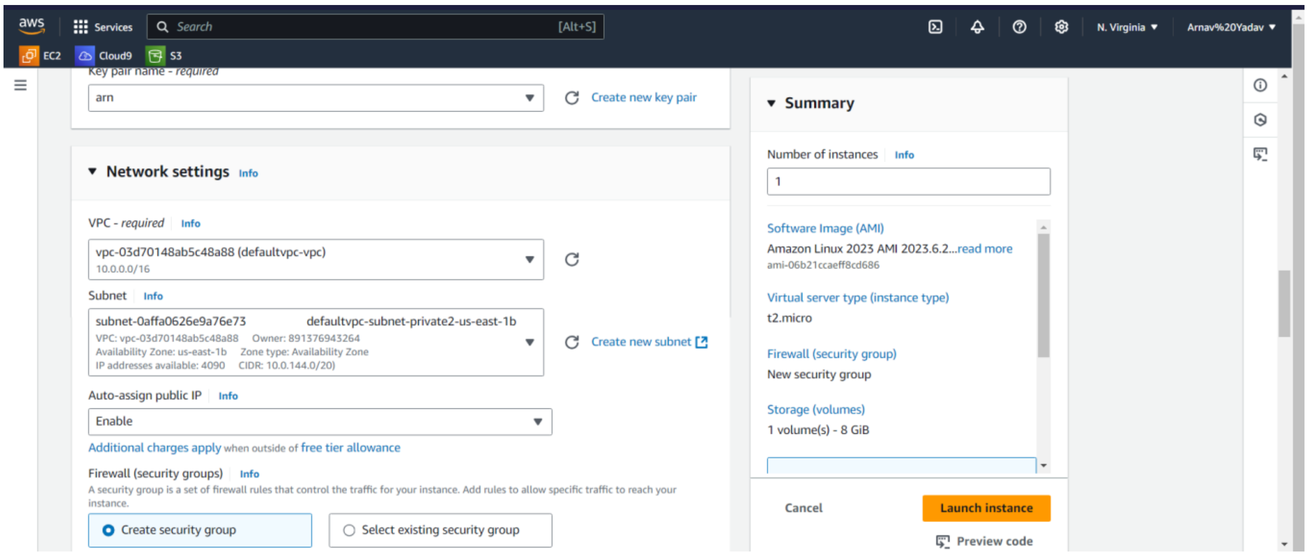
Number of instances Info: 1

Software Image (AMI): Amazon Linux 2023 AMI 2023.6.2...read more ami-06b21ccaeff8cd686

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

Storage (volumes)


Key pair name - required: arn

Network settings Info

VPC - required Info: vpc-03d70148ab5c48a88 (defaultvpc-vpc) 10.0.0.0/16

Subnet Info: subnet-0affa0626e9a76e73 defaultvpc-subnet-private2-us-east-1b
VPC: vpc-03d70148ab5c48a88 Owner: 891376943264
Availability Zone: us-east-1b Zone type: Availability Zone
IP addresses available: 4090 CIDR: 10.0.144.0/20

Auto-assign public IP Info: Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

[✓ Create security group] Select existing security group

Summary

Number of instances Info: 1

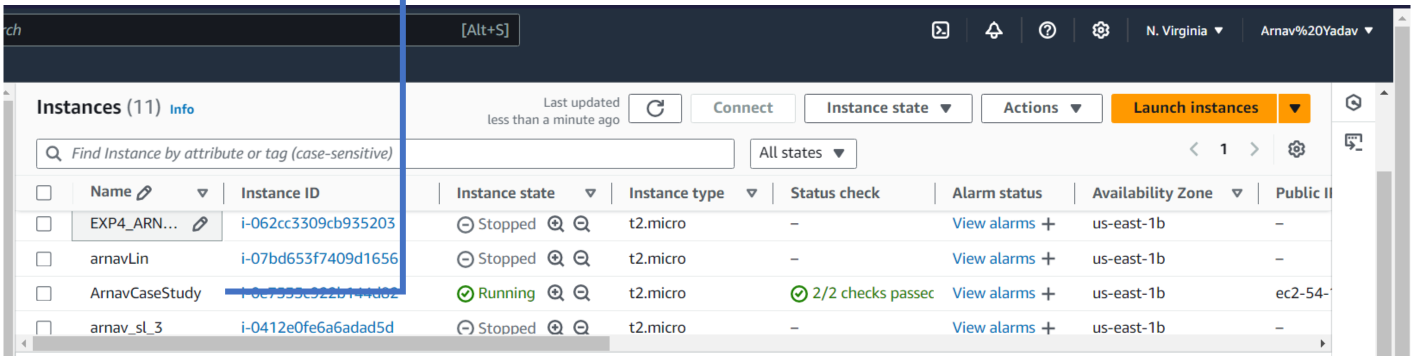
Software Image (AMI): Amazon Linux 2023 AMI 2023.6.2...read more ami-06b21ccaeff8cd686

Virtual server type (instance type): t2.micro

Firewall (security group): New security group

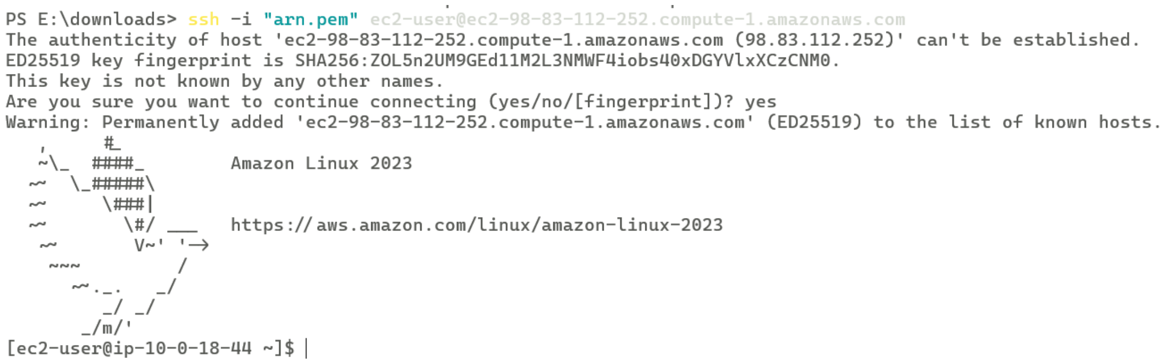
Storage (volumes): 1 volume(s) - 8 GiB

Connect to Your EC2 Instance:



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public II
☐	EXP4_ARN...	i-062cc3309cb935203	Stopped	t2.micro	–	View alarms +	us-east-1b	–
☐	arnavLin	i-07bd653f7409d1656	Stopped	t2.micro	–	View alarms +	us-east-1b	–
☐	ArnavCaseStudy	i-0e7333c322b144d02	Running	t2.micro	2/2 checks passec	View alarms +	us-east-1b	ec2-54-
☐	arnav_sl_3	i-0412e0fe6a6adad5d	Stopped	t2.micro	–	View alarms +	us-east-1b	–

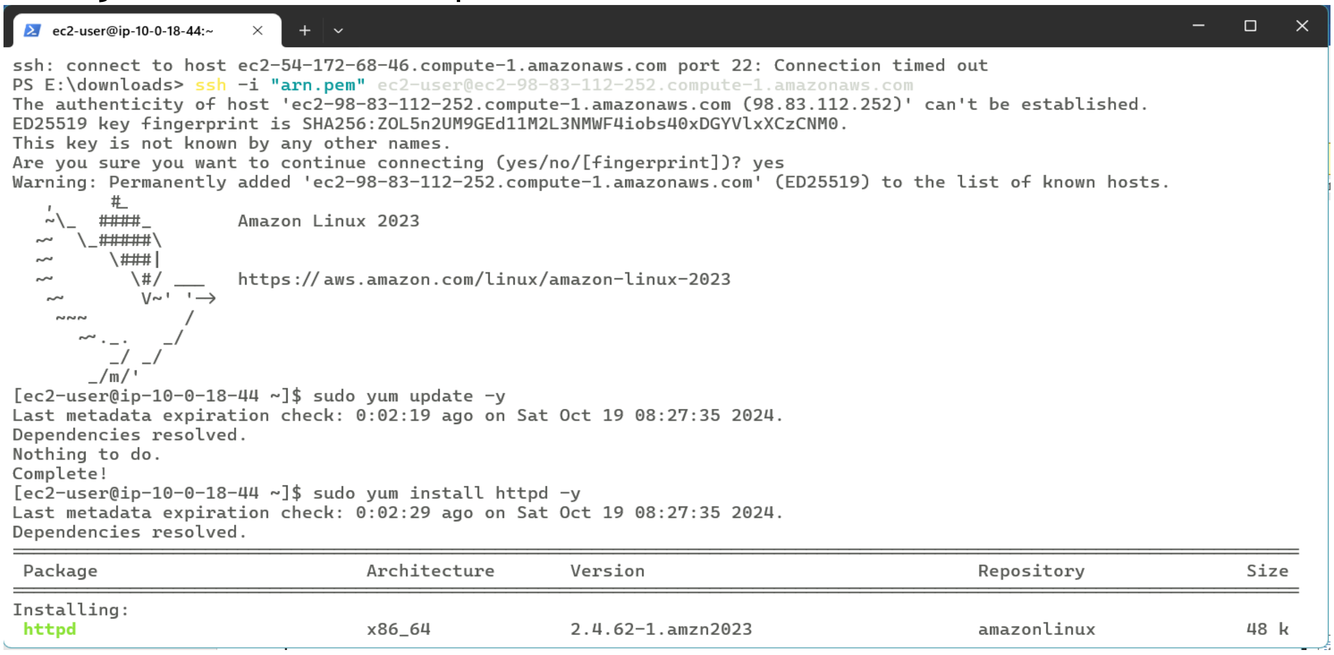
```
ssh -i "arn.pem" ec2-user@ec2-98-83-112-252.compute-1.amazonaws.com
```



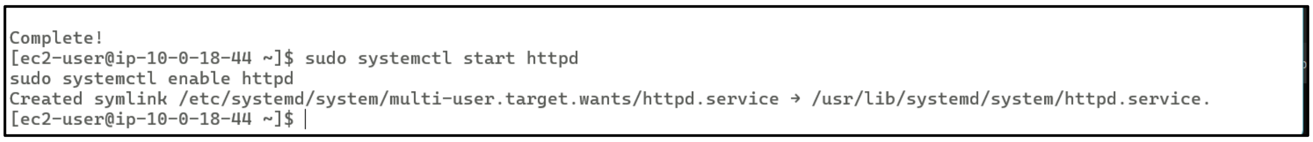
```
PS E:\downloads> ssh -i "arn.pem" ec2-user@ec2-98-83-112-252.compute-1.amazonaws.com
The authenticity of host 'ec2-98-83-112-252.compute-1.amazonaws.com (98.83.112.252)' can't be established.
ED25519 key fingerprint is SHA256:ZOL5n2UM9GEd11M2L3NMWF4iobs40xDGYVlxXCzCNM0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-98-83-112-252.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-18-44 ~]$ |
```

Step 2: Install Required Software on EC2

```
sudo yum update -y
sudo yum install httpd -y
sudo systemctl start httpd
sudo systemctl enable httpd
```



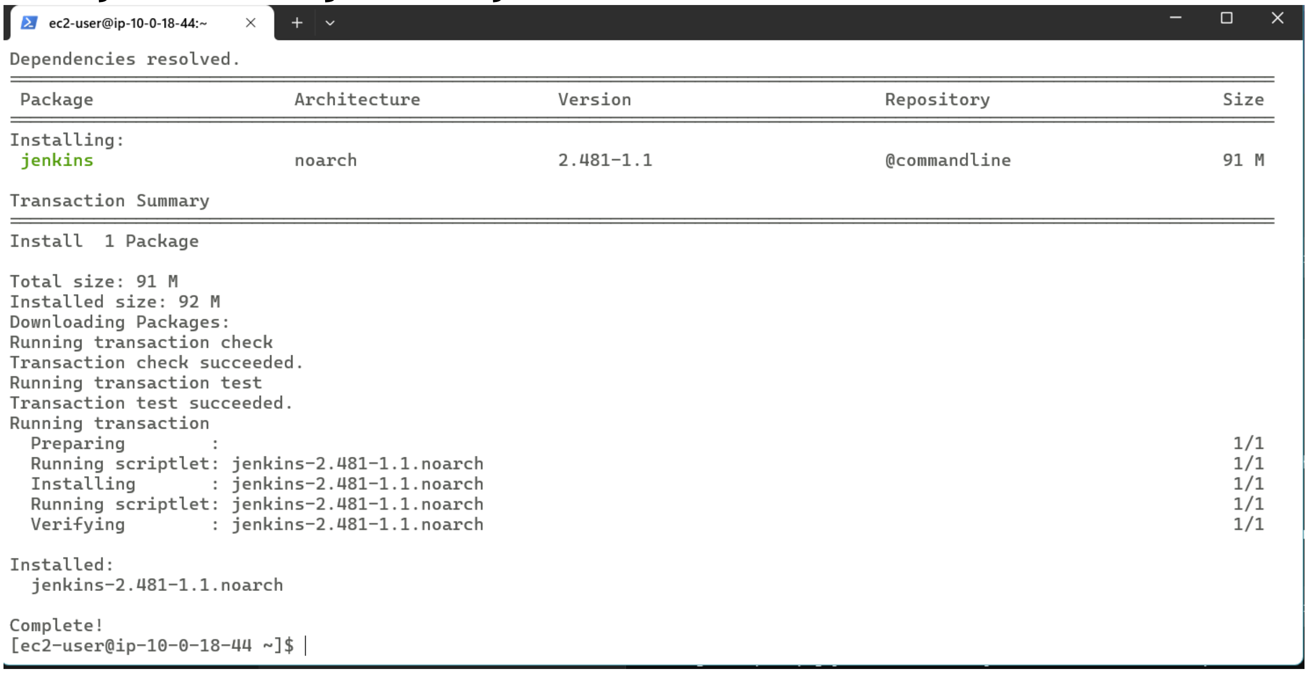
```
ssh: connect to host ec2-54-172-68-46.compute-1.amazonaws.com port 22: Connection timed out
PS E:\downloads> ssh -i "arn.pem" ec2-user@ec2-98-83-112-252.compute-1.amazonaws.com
The authenticity of host 'ec2-98-83-112-252.compute-1.amazonaws.com (98.83.112.252)' can't be established.
ED25519 key fingerprint is SHA256:ZOL5n2UM9GEd11M2L3NMWF4iobs40xDGYVlxXCzCNM0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-98-83-112-252.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-10-0-18-44 ~]$ sudo yum update -y
Last metadata expiration check: 0:02:19 ago on Sat Oct 19 08:27:35 2024.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-10-0-18-44 ~]$ sudo yum install httpd -y
Last metadata expiration check: 0:02:29 ago on Sat Oct 19 08:27:35 2024.
Dependencies resolved.
=========================================================================================================
 Package                     Architecture     Version                               Repository      Size
=========================================================================================================
Installing:
 httpd                       x86_64           2.4.62-1.amzn2023                     amazonlinux     48 k
```



```
Complete!
[ec2-user@ip-10-0-18-44 ~]$ sudo systemctl start httpd
sudo systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-10-0-18-44 ~]$ |
```

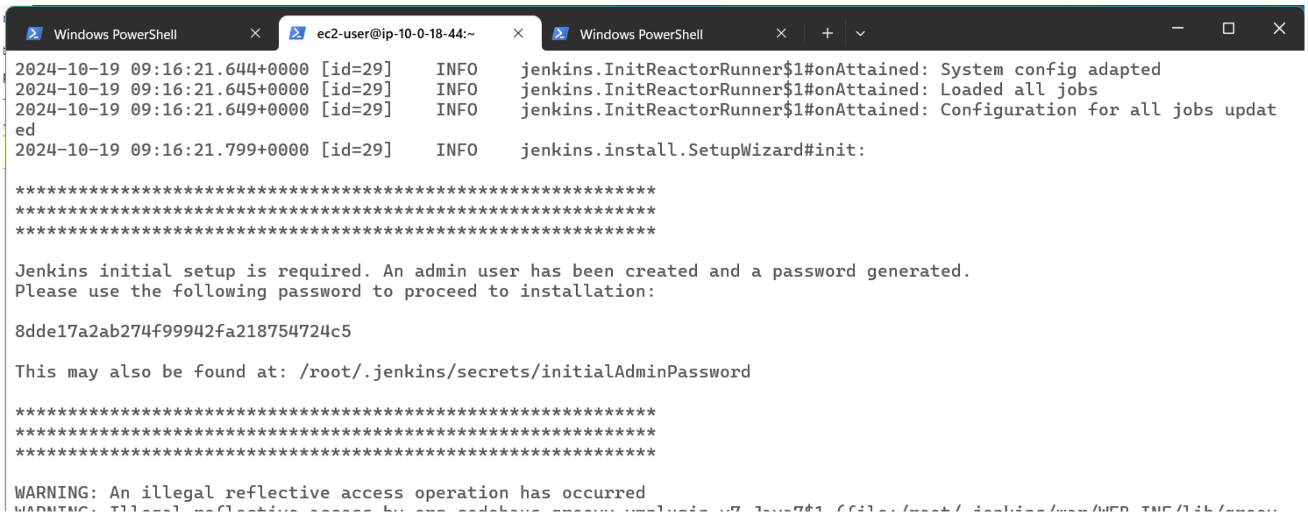
Step 3: Set Up Jenkins

```
sudo yum install java-1.8.0-openjdk-devel -y
sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat/jenkins.io.key
sudo yum install jenkins -y
```



```
Dependencies resolved.
================================================================================================================
 Package                   Architecture         Version                      Repository                  Size
================================================================================================================
Installing:
 jenkins                   noarch               2.481-1.1                    @commandline                91 M

Transaction Summary
================================================================================================================
Install  1 Package

Total size: 91 M
Installed size: 92 M
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                                      1/1
  Running scriptlet: jenkins-2.481-1.1.noarch                                                              1/1
  Installing       : jenkins-2.481-1.1.noarch                                                              1/1
  Running scriptlet: jenkins-2.481-1.1.noarch                                                              1/1
  Verifying        : jenkins-2.481-1.1.noarch                                                              1/1

Installed:
  jenkins-2.481-1.1.noarch

Complete!
[ec2-user@ip-10-0-18-44 ~]$ |
```

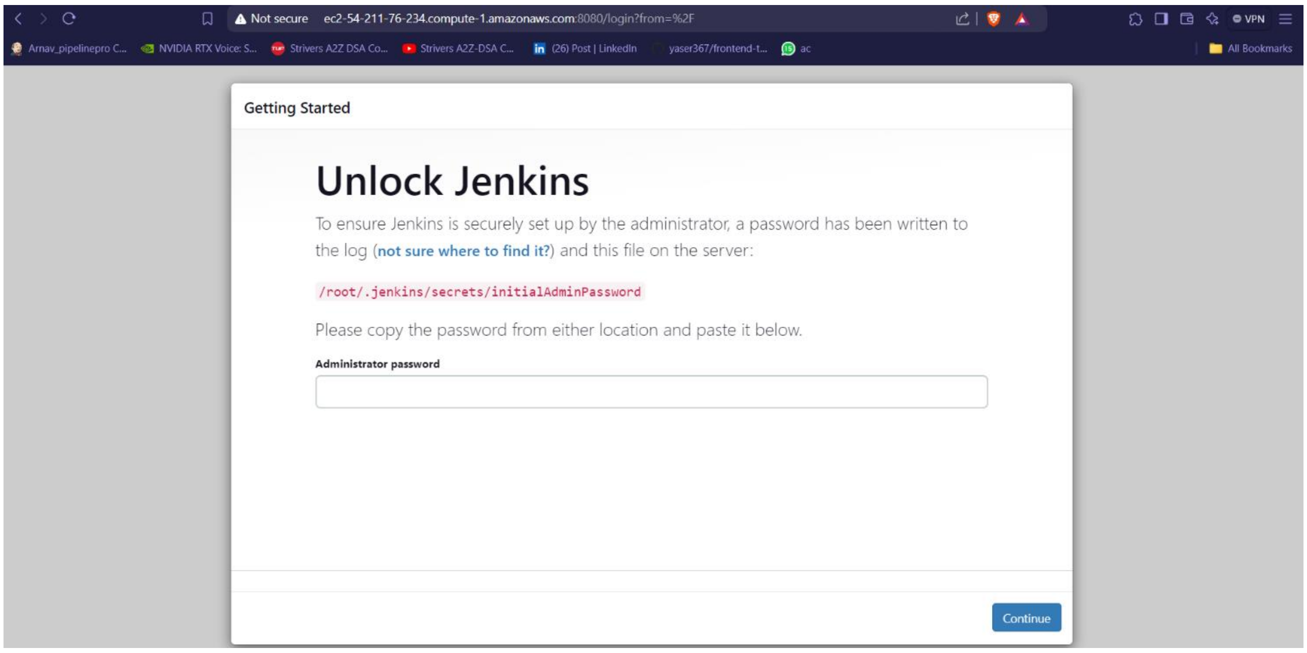
Start Jenkins sudo systemctl start jenkins
sudo systemctl enable Jenkins



```
2024-10-19 09:16:21.644+0000 [id=29]    INFO    jenkins.InitReactorRunner$1#onAttained: System config adapted
2024-10-19 09:16:21.645+0000 [id=29]    INFO    jenkins.InitReactorRunner$1#onAttained: Loaded all jobs
2024-10-19 09:16:21.649+0000 [id=29]    INFO    jenkins.InitReactorRunner$1#onAttained: Configuration for all jobs updat
ed
2024-10-19 09:16:21.799+0000 [id=29]    INFO    jenkins.install.SetupWizard#init:

*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

8dde17a2ab274f99942fa218754724c5

This may also be found at: /root/.jenkins/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************

WARNING: An illegal reflective access operation has occurred
```

Access Jenkinks
http://<your-ec2-public-dns>:8080

Complete the setup wizard:



Not secure ec2-54-211-76-234.compute-1.amazonaws.com:8080/login?from=%2F

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

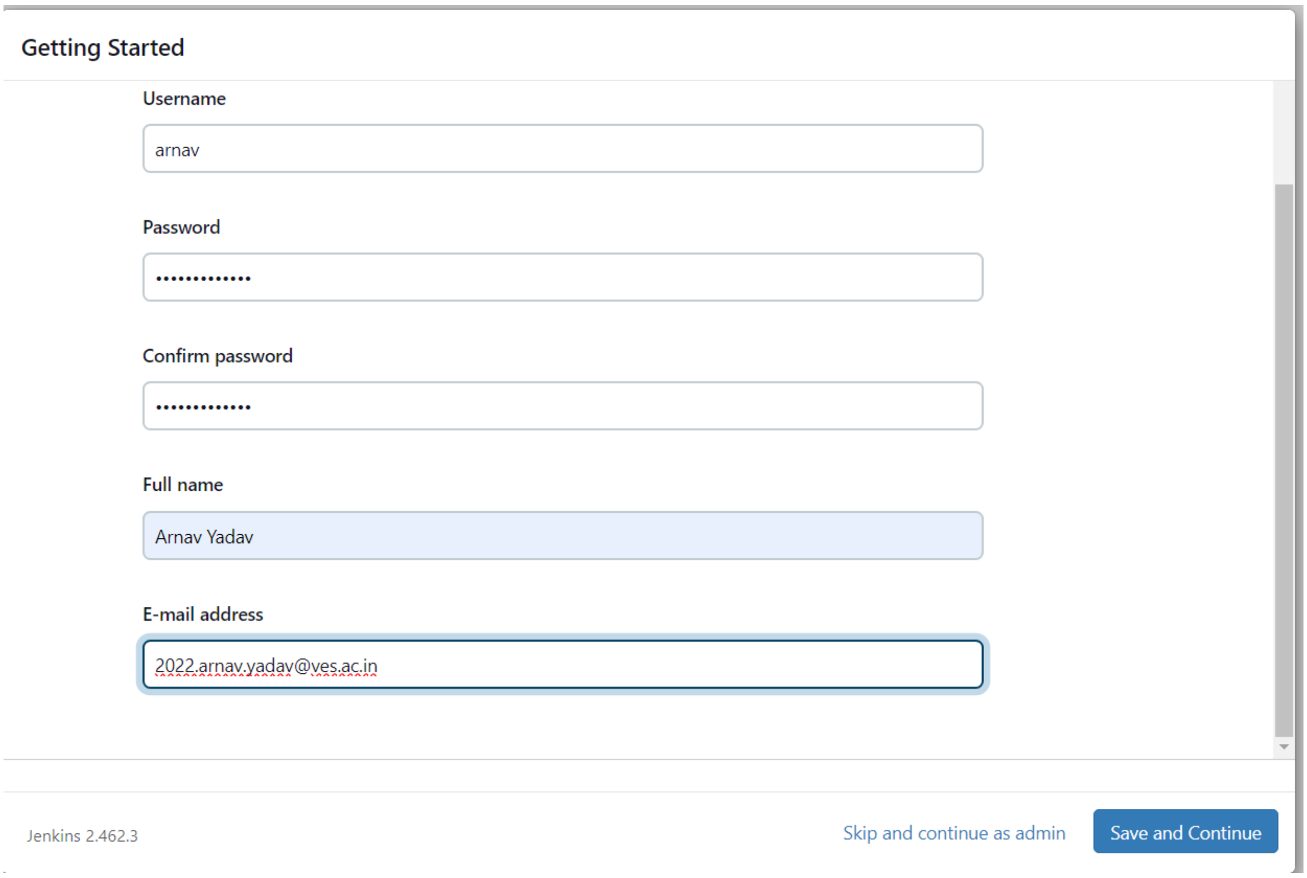
`/root/.jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

Create an admin user.



Getting Started

Username: arnav

Password: •••••••••••••

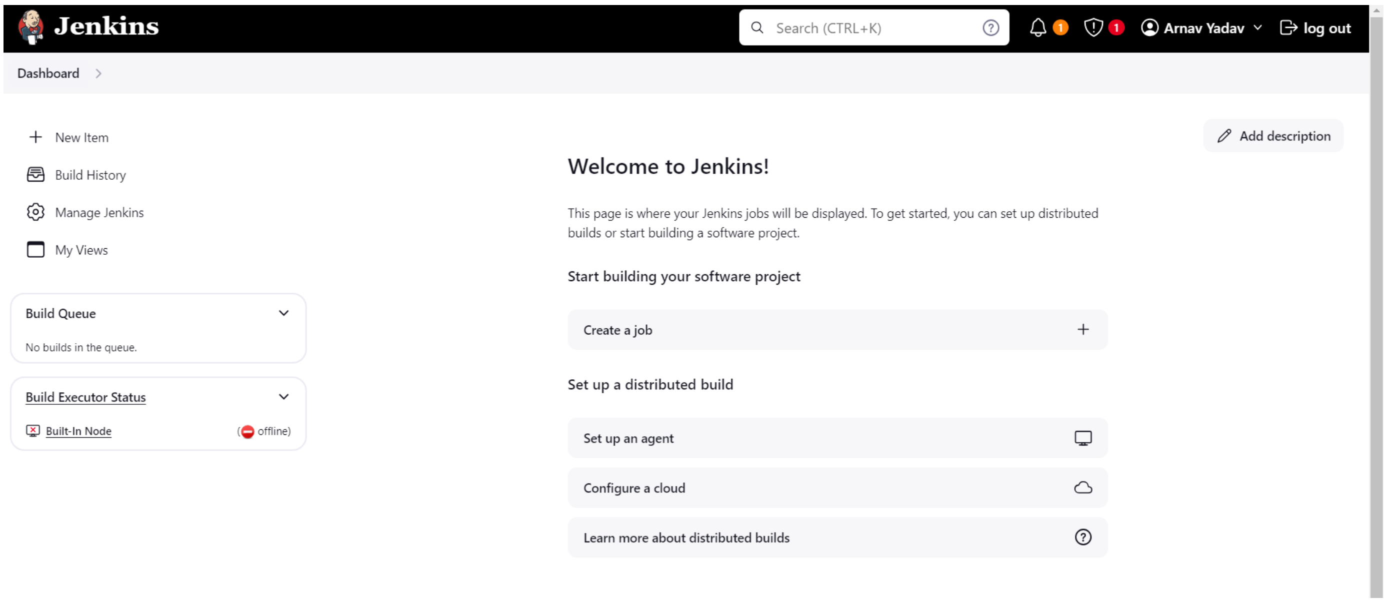
Confirm password: •••••••••••••

Full name: Arnav Yadav

E-mail address: 2022.arnav.yadav@ves.ac.in

Jenkins 2.462.3

Skip and continue as admin | Save and Continue



No builds in the queue.

Welcome to Jenkins!

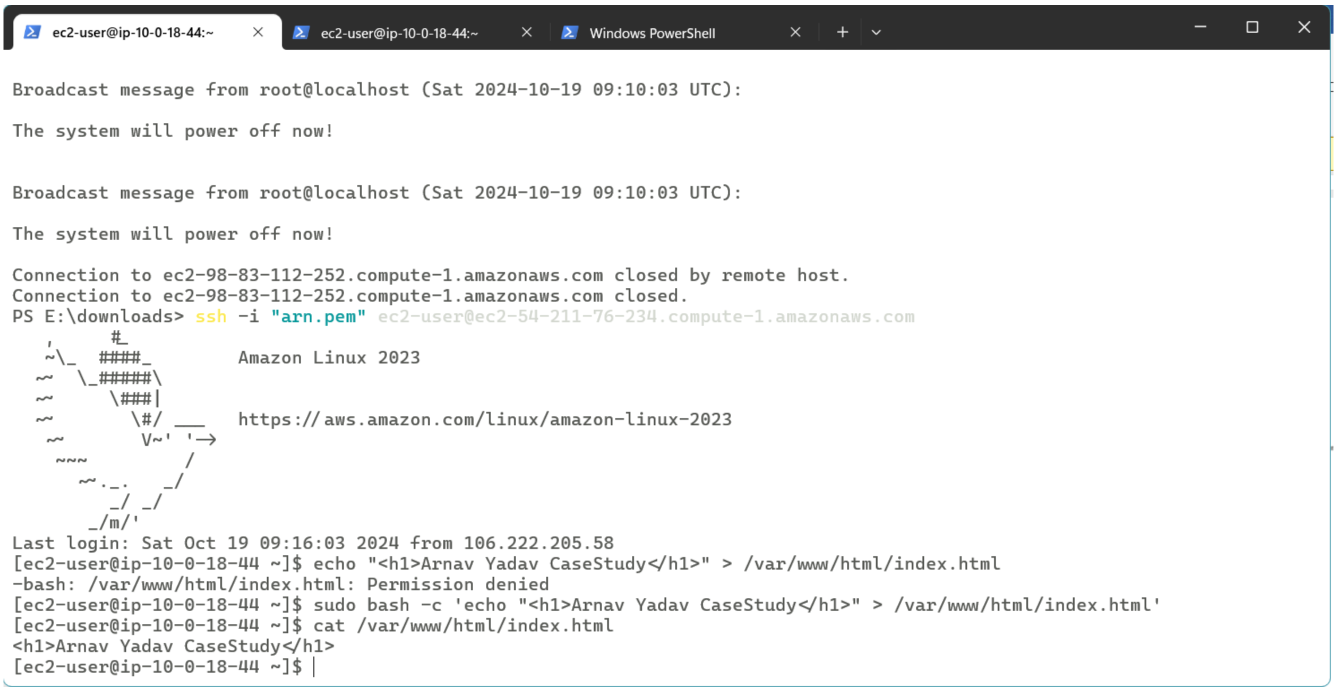
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

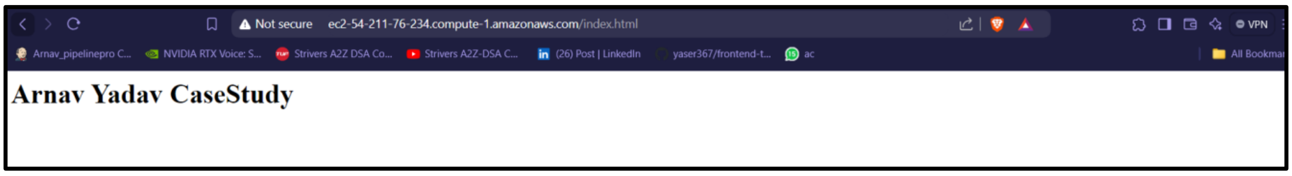
Set up a distributed build

Step 4: Create a Sample Web Application

```
echo "<h1>Arnav Yadav CaseStudy!</h1>" > /var/www/html/index.html
```

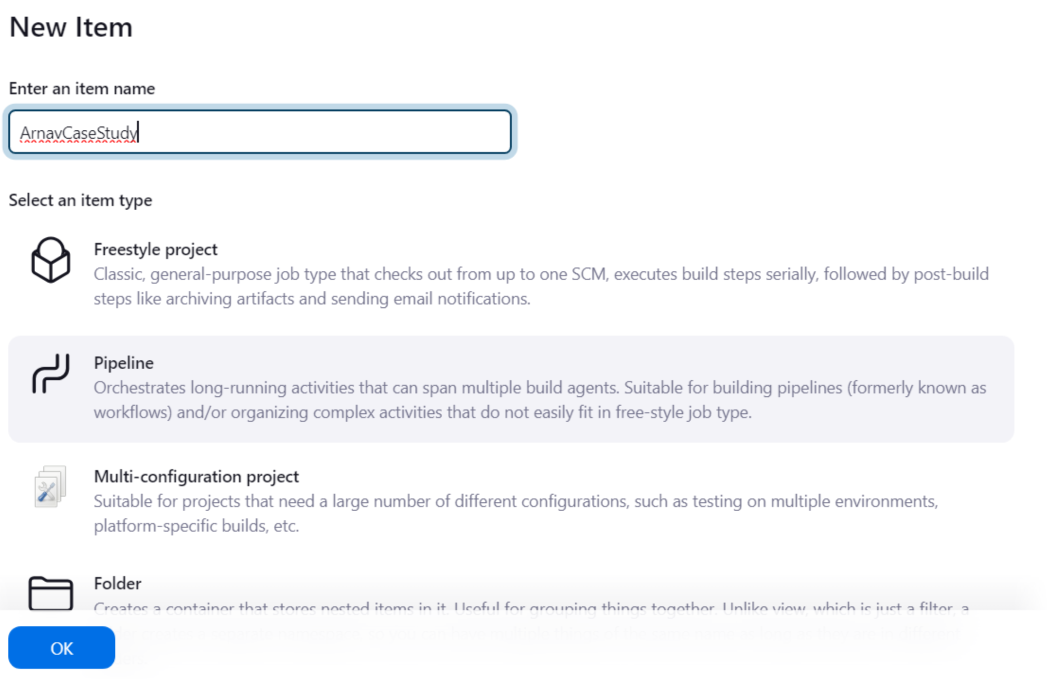


```
Broadcast message from root@localhost (Sat 2024-10-19 09:10:03 UTC):

The system will power off now!


Broadcast message from root@localhost (Sat 2024-10-19 09:10:03 UTC):

The system will power off now!

Connection to ec2-98-83-112-252.compute-1.amazonaws.com closed by remote host.
Connection to ec2-98-83-112-252.compute-1.amazonaws.com closed.
PS E:\downloads> ssh -i "arn.pem" ec2-user@ec2-54-211-76-234.compute-1.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Sat Oct 19 09:16:03 2024 from 106.222.205.58
[ec2-user@ip-10-0-18-44 ~]$ echo "<h1>Arnav Yadav CaseStudy</h1>" > /var/www/html/index.html
-bash: /var/www/html/index.html: Permission denied
[ec2-user@ip-10-0-18-44 ~]$ sudo bash -c 'echo "<h1>Arnav Yadav CaseStudy</h1>" > /var/www/html/index.html'
[ec2-user@ip-10-0-18-44 ~]$ cat /var/www/html/index.html
<h1>Arnav Yadav CaseStudy</h1>
[ec2-user@ip-10-0-18-44 ~]$ |
```



Arnav Yadav CaseStudy

Step 5: Configure Jenkins Pipeline



New Item

Enter an item name

ArnavCaseStudy

Select an item type

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

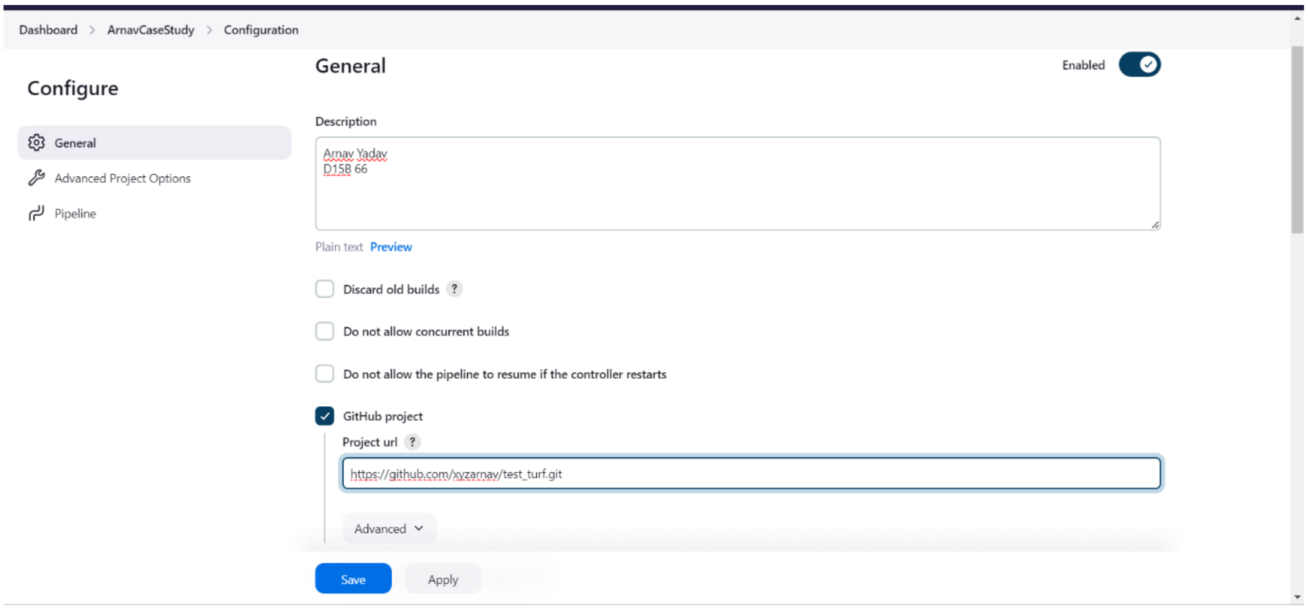
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a

OK

Set Up Your Pipeline Script:



Dashboard > ArnavCaseStudy > Configuration

Configure

- General
- Advanced Project Options
- Pipeline

General — Enabled

Description

Arnav Yadav
D15B 66

Plain text Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project

Project url: https://github.com/xyzarnav/test_turf.git

Advanced

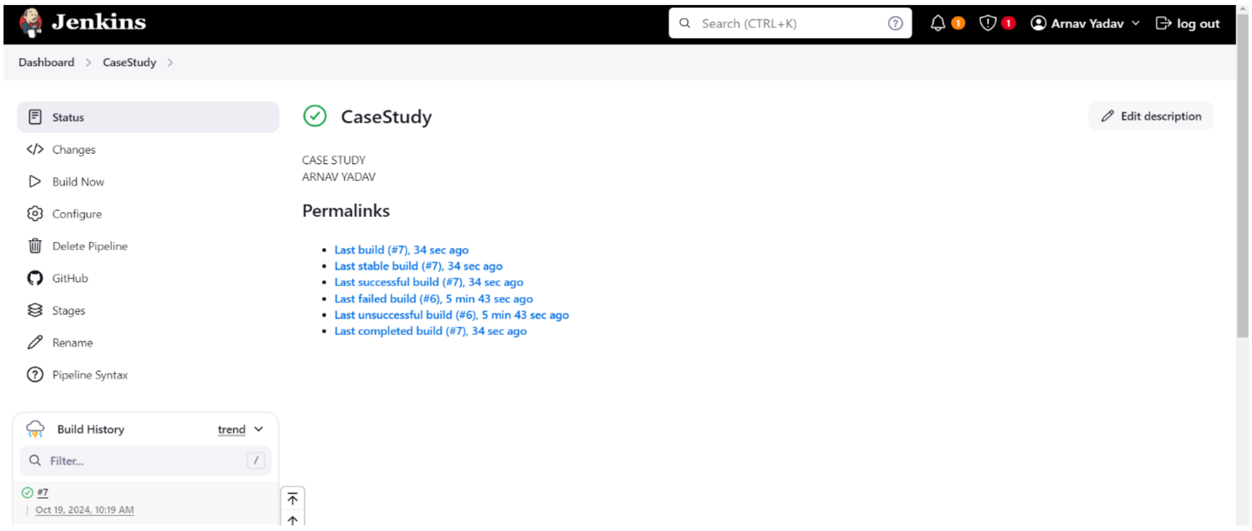
Save Apply

Updated Jenkins Pipeline Script

```
        pipeline {
            agent any
            stages {
                stage('Clone Repository') {
                    steps {
                        // Specify the branch name as 'main'
                        git branch: 'main', url:
  'https://github.com/xyzarnav/casestudy.git'
                    }
                }
                stage('Deploy') {
                    steps {
                        script {
                            // Your deployment commands here
                            sh 'cp -r * /var/www/html/'
                        }
                    }
                }
            }
            post {
                failure {
                    echo 'Deployment failed!'
                }
            }
        }
```

Apply and Save the Pipeline.

Step 6: Build and Run



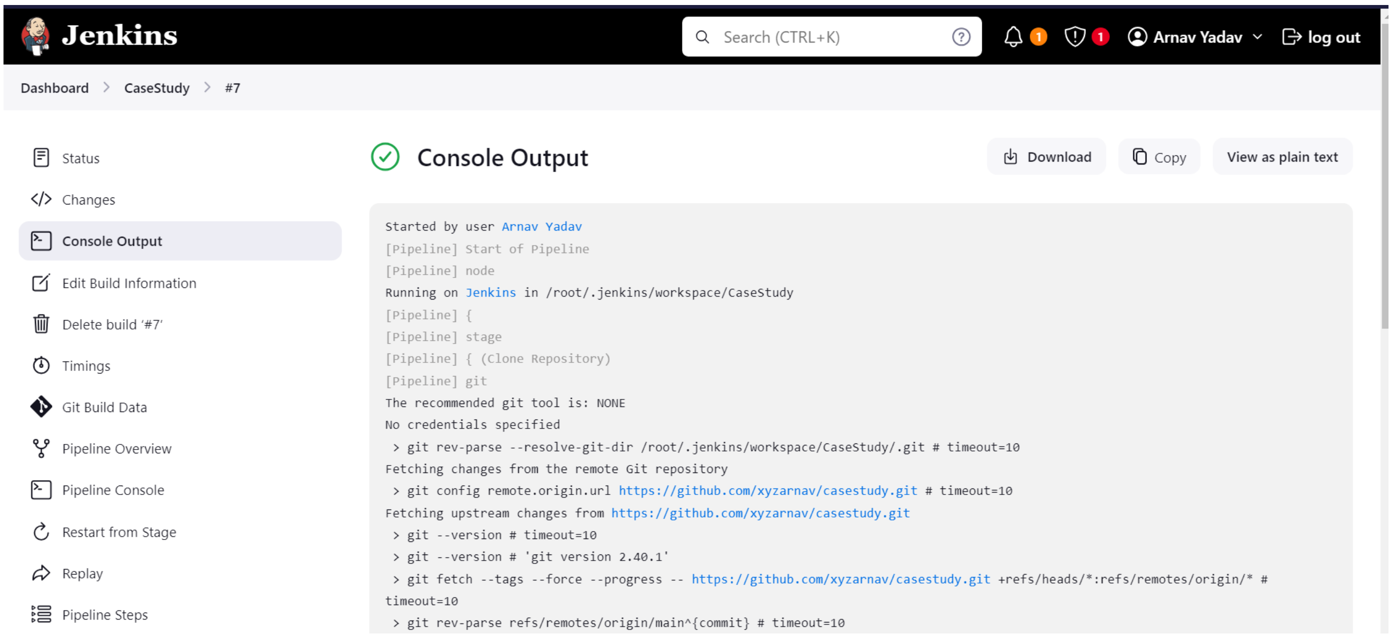
CaseStudy

CASE STUDY
ARNAV YADAV

Permalinks

- Last build (#7), 34 sec ago
- Last stable build (#7), 34 sec ago
- Last successful build (#7), 34 sec ago
- Last failed build (#6), 5 min 43 sec ago
- Last unsuccessful build (#6), 5 min 43 sec ago
- Last completed build (#7), 34 sec ago

#7
Oct 19, 2024, 10:19 AM



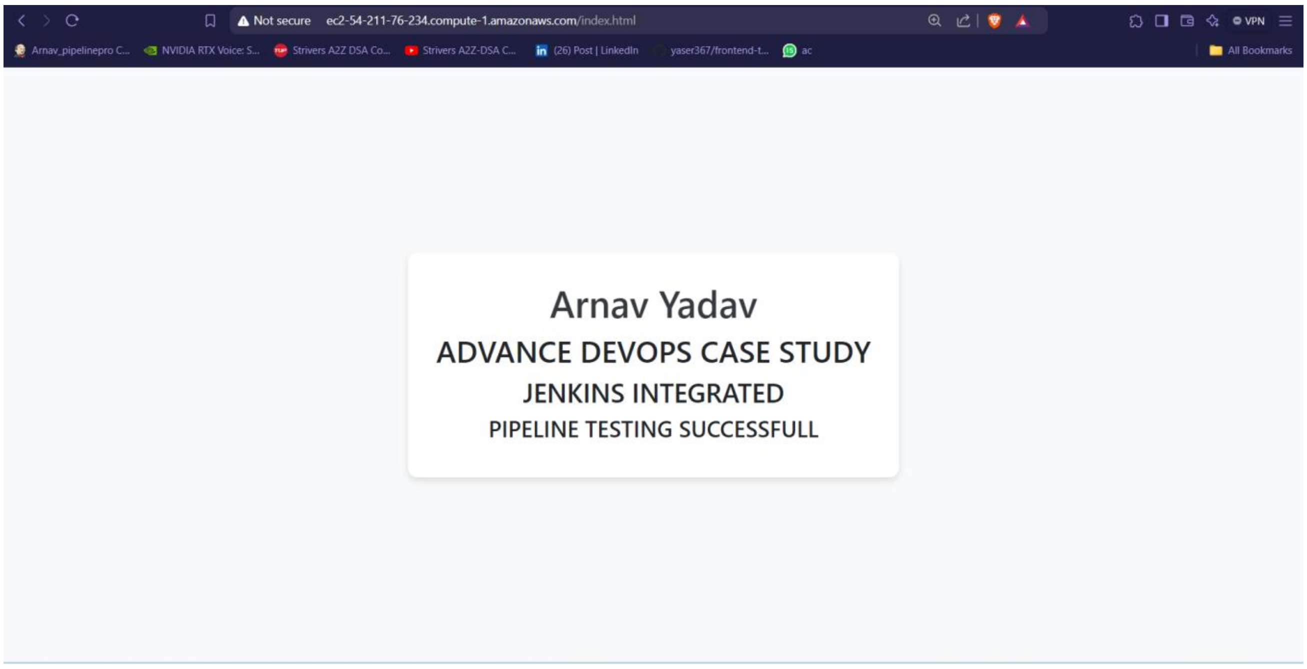
Console Output

```
Started by user Arnav Yadav
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /root/.jenkins/workspace/CaseStudy
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone Repository)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --resolve-git-dir /root/.jenkins/workspace/CaseStudy/.git # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/xyzarnav/casestudy.git # timeout=10
Fetching upstream changes from https://github.com/xyzarnav/casestudy.git
 > git --version # timeout=10
 > git --version # 'git version 2.40.1'
 > git fetch --tags --force --progress -- https://github.com/xyzarnav/casestudy.git +refs/heads/*:refs/remotes/origin/* #
timeout=10
 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10
```

Now, you should be able to access this file from your browser by going to:

http://<your-server-ip>/index.html

http://ec2-98-83-112-252.compute-1.amazonaws.com/index.html


Not secure ec2-54-211-76-234.compute-1.amazonaws.com/index.html

Arnav Yadav

ADVANCE DEVOPS CASE STUDY

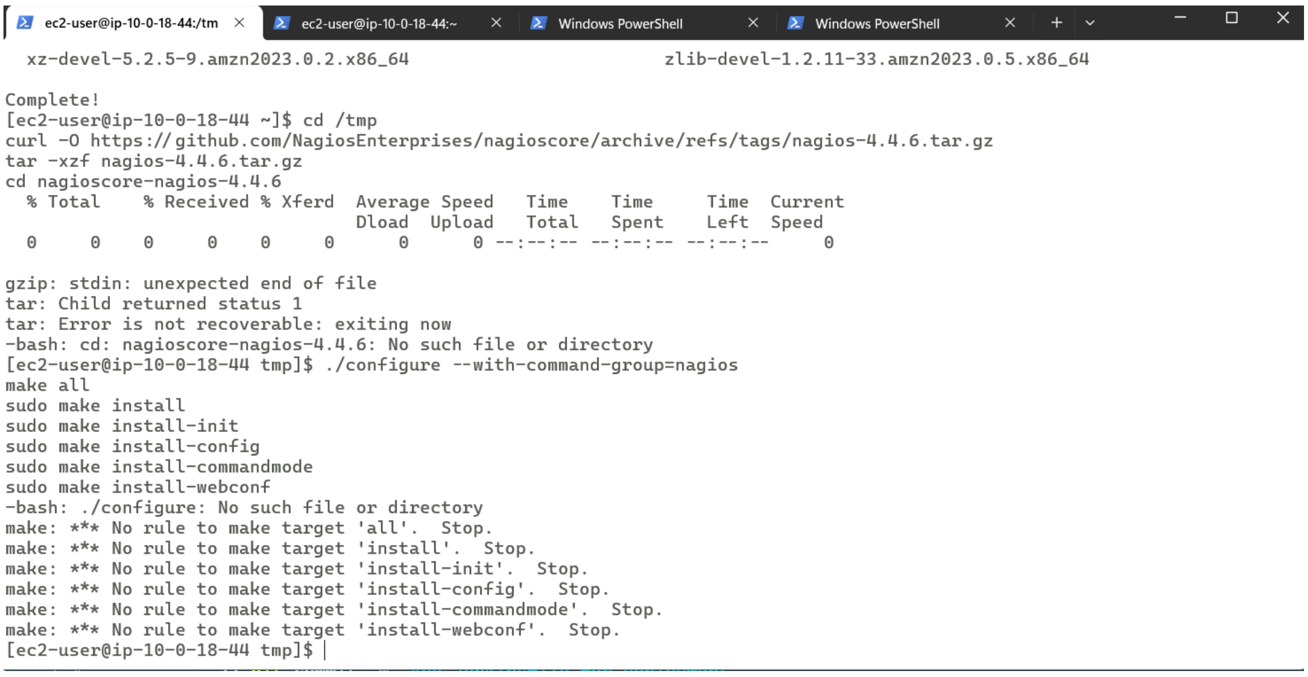
JENKINS INTEGRATED

PIPELINE TESTING SUCCESSFULL

!!!!Task 1 Completed

Step 7: Install and Configure Nagios

```
sudo yum install -y gcc glibc glibc-common perl httpd php
sudo yum install -y httpd-devel php-devel php-gd php-mbstring
```

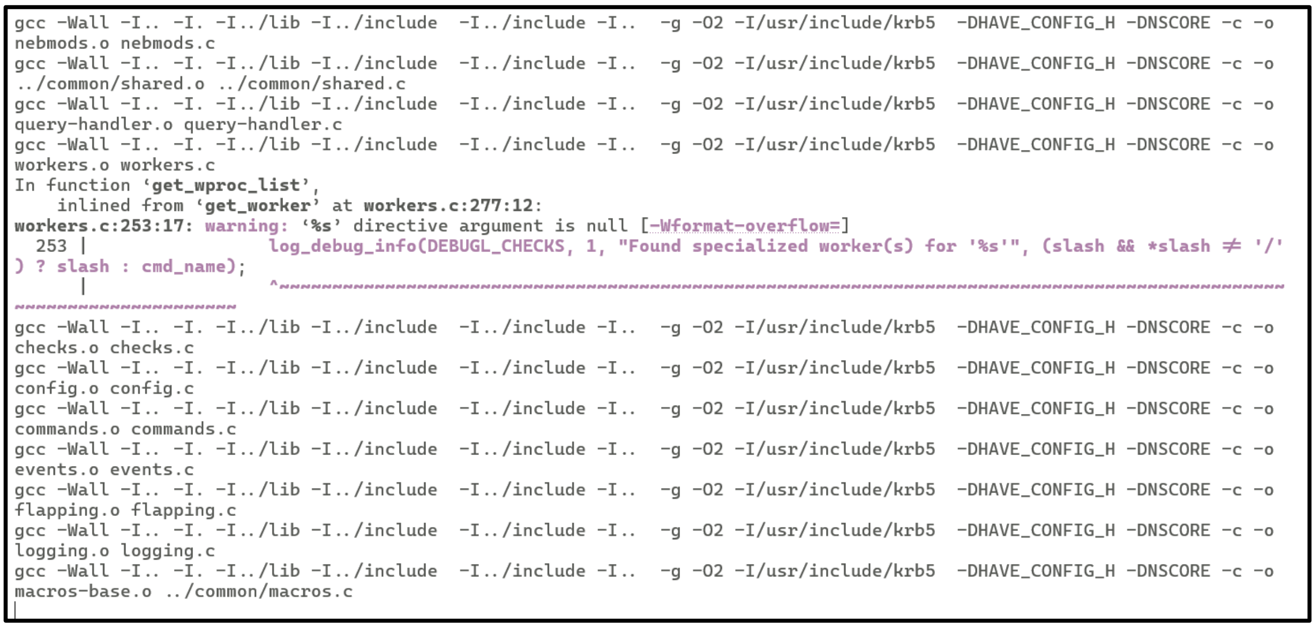


```
  xz-devel-5.2.5-9.amzn2023.0.2.x86_64                         zlib-devel-1.2.11-33.amzn2023.0.5.x86_64

Complete!
[ec2-user@ip-10-0-18-44 ~]$ cd /tmp
curl -O https://github.com/NagiosEnterprises/nagioscore/archive/refs/tags/nagios-4.4.6.tar.gz
tar -xzf nagios-4.4.6.tar.gz
cd nagioscore-nagios-4.4.6
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0

gzip: stdin: unexpected end of file
tar: Child returned status 1
tar: Error is not recoverable: exiting now
-bash: cd: nagioscore-nagios-4.4.6: No such file or directory
[ec2-user@ip-10-0-18-44 tmp]$ ./configure --with-command-group=nagios
make all
sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
sudo make install-webconf
-bash: ./configure: No such file or directory
make: *** No rule to make target 'all'.  Stop.
make: *** No rule to make target 'install'.  Stop.
make: *** No rule to make target 'install-init'.  Stop.
make: *** No rule to make target 'install-config'.  Stop.
make: *** No rule to make target 'install-commandmode'.  Stop.
make: *** No rule to make target 'install-webconf'.  Stop.
[ec2-user@ip-10-0-18-44 tmp]$ |
```

Download Nagios:

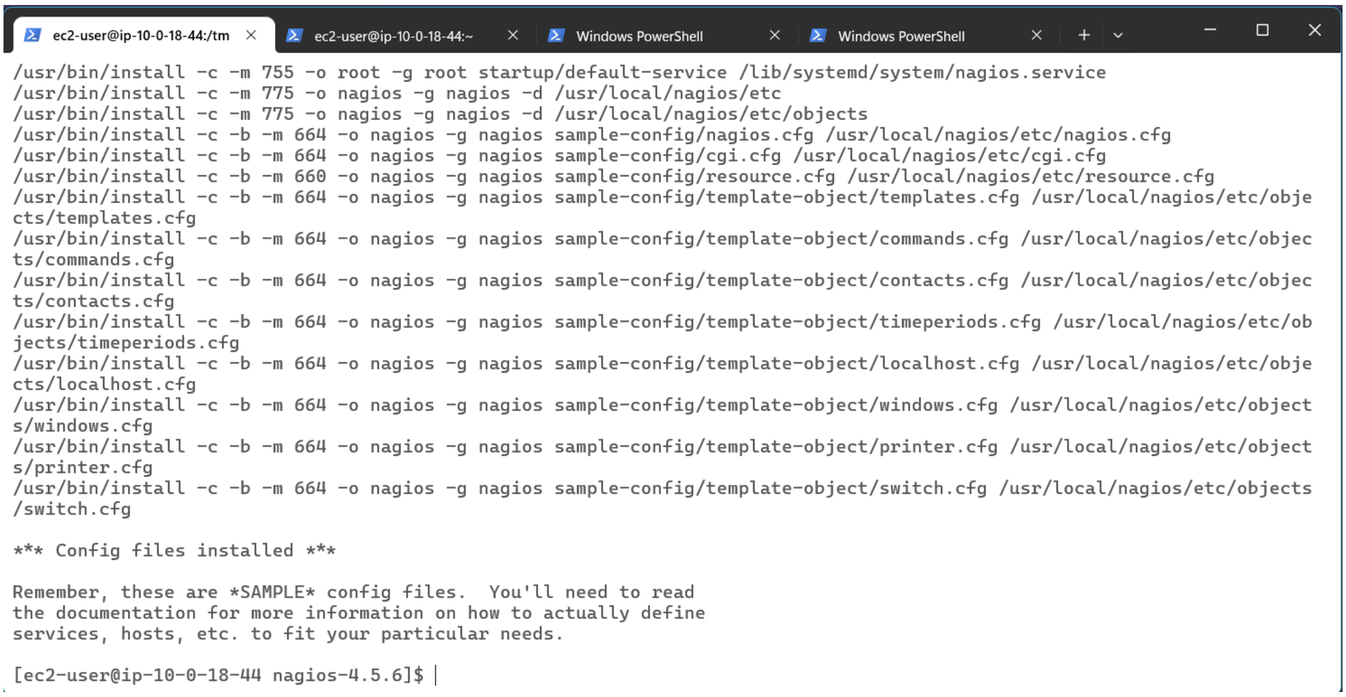
```
cd /tmp
curl -O
https://github.com/NagiosEnterprises/nagioscore/archive/refs/tags/nag
ios-4.4.6.tar.gz
tar -xzf nagios-4.4.6.tar.gz
```

```
cd nagioscore-nagios-4.4.6
```



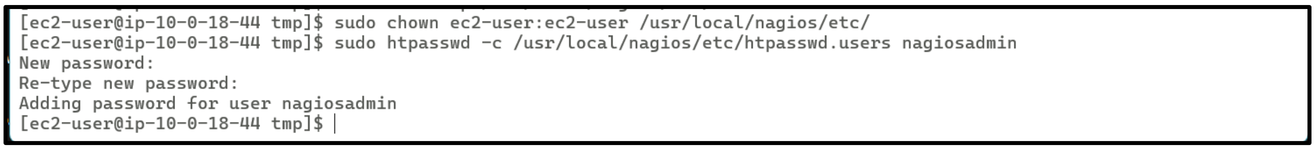
```
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
nebmods.o nebmods.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
../common/shared.o ../common/shared.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
query-handler.o query-handler.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
workers.o workers.c
In function ‘get_wproc_list’,
    inlined from ‘get_worker’ at workers.c:277:12:
workers.c:253:17: warning: ‘%s’ directive argument is null [-Wformat-overflow=]
  253 |                 log_debug_info(DEBUGL_CHECKS, 1, "Found specialized worker(s) for '%s'", (slash && *slash != '/'
) ? slash : cmd_name);
      |                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
checks.o checks.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
config.o config.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
commands.o commands.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
events.o events.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
flapping.o flapping.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
logging.o logging.c
gcc -Wall -I.. -I. -I../lib -I../include  -I../include -I..  -g -O2 -I/usr/include/krb5  -DHAVE_CONFIG_H -DNSCORE -c -o
macros-base.o ../common/macros.c
```

```
./configure --with-command-group=nagios
make all
sudo make install
sudo make install-init
sudo make install-config
sudo make install-commandmode
sudo make install-webconf
```



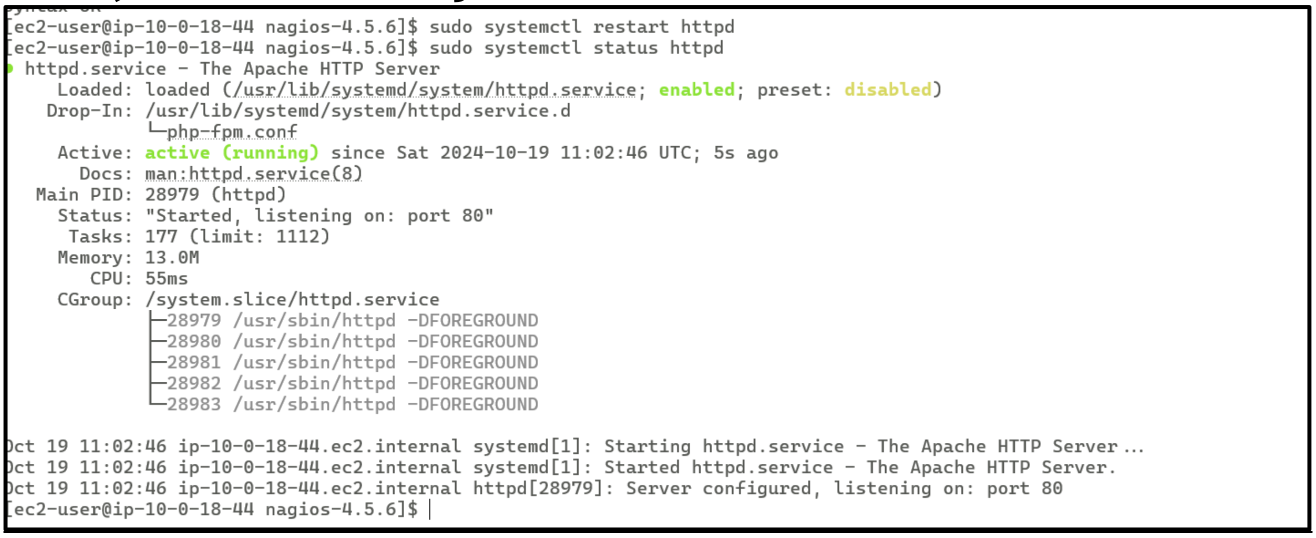
```
/usr/bin/install -c -m 755 -o root -g root startup/default-service /lib/systemd/system/nagios.service
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc
/usr/bin/install -c -m 775 -o nagios -g nagios -d /usr/local/nagios/etc/objects
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/nagios.cfg /usr/local/nagios/etc/nagios.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/cgi.cfg /usr/local/nagios/etc/cgi.cfg
/usr/bin/install -c -b -m 660 -o nagios -g nagios sample-config/resource.cfg /usr/local/nagios/etc/resource.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/templates.cfg /usr/local/nagios/etc/obje
cts/templates.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/commands.cfg /usr/local/nagios/etc/objec
ts/commands.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/contacts.cfg /usr/local/nagios/etc/objec
ts/contacts.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/timeperiods.cfg /usr/local/nagios/etc/ob
jects/timeperiods.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/localhost.cfg /usr/local/nagios/etc/obje
cts/localhost.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/windows.cfg /usr/local/nagios/etc/object
s/windows.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/printer.cfg /usr/local/nagios/etc/object
s/printer.cfg
/usr/bin/install -c -b -m 664 -o nagios -g nagios sample-config/template-object/switch.cfg /usr/local/nagios/etc/objects
/switch.cfg

*** Config files installed ***

Remember, these are *SAMPLE* config files.  You'll need to read
the documentation for more information on how to actually define
services, hosts, etc. to fit your particular needs.

[ec2-user@ip-10-0-18-44 nagios-4.5.6]$ |
```

```
sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
sudo vi /usr/local/nagios/etc/nagios.cfg
```



```
[ec2-user@ip-10-0-18-44 tmp]$ sudo chown ec2-user:ec2-user /usr/local/nagios/etc/
[ec2-user@ip-10-0-18-44 tmp]$ sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
[ec2-user@ip-10-0-18-44 tmp]$ |
```

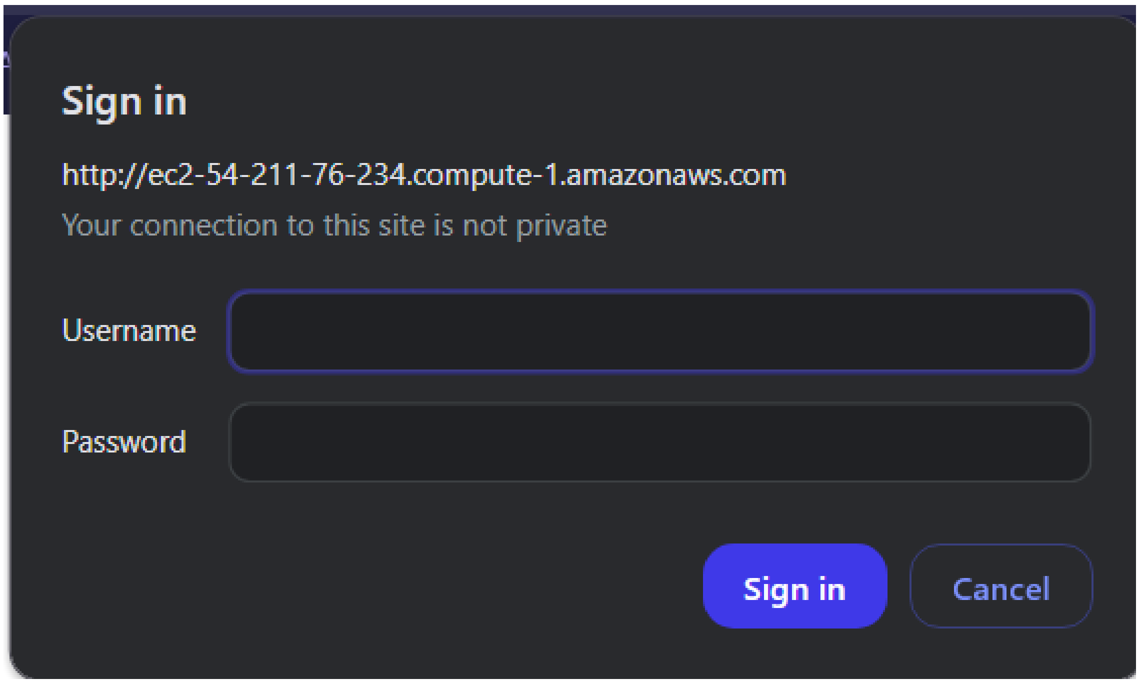
sudo systemctl start nagios
sudo systemctl enable nagios



```
[ec2-user@ip-10-0-18-44 nagios-4.5.6]$ sudo systemctl restart httpd
[ec2-user@ip-10-0-18-44 nagios-4.5.6]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Sat 2024-10-19 11:02:46 UTC; 5s ago
       Docs: man:httpd.service(8)
   Main PID: 28979 (httpd)
     Status: "Started, listening on: port 80"
      Tasks: 177 (limit: 1112)
     Memory: 13.0M
        CPU: 55ms
     CGroup: /system.slice/httpd.service
             ├─28979 /usr/sbin/httpd -DFOREGROUND
             ├─28980 /usr/sbin/httpd -DFOREGROUND
             ├─28981 /usr/sbin/httpd -DFOREGROUND
             ├─28982 /usr/sbin/httpd -DFOREGROUND
             └─28983 /usr/sbin/httpd -DFOREGROUND

Oct 19 11:02:46 ip-10-0-18-44.ec2.internal systemd[1]: Starting httpd.service - The Apache HTTP Server...
Oct 19 11:02:46 ip-10-0-18-44.ec2.internal systemd[1]: Started httpd.service - The Apache HTTP Server.
Oct 19 11:02:46 ip-10-0-18-44.ec2.internal httpd[28979]: Server configured, listening on: port 80
[ec2-user@ip-10-0-18-44 nagios-4.5.6]$ |
```

Step 8: Access Nagios:

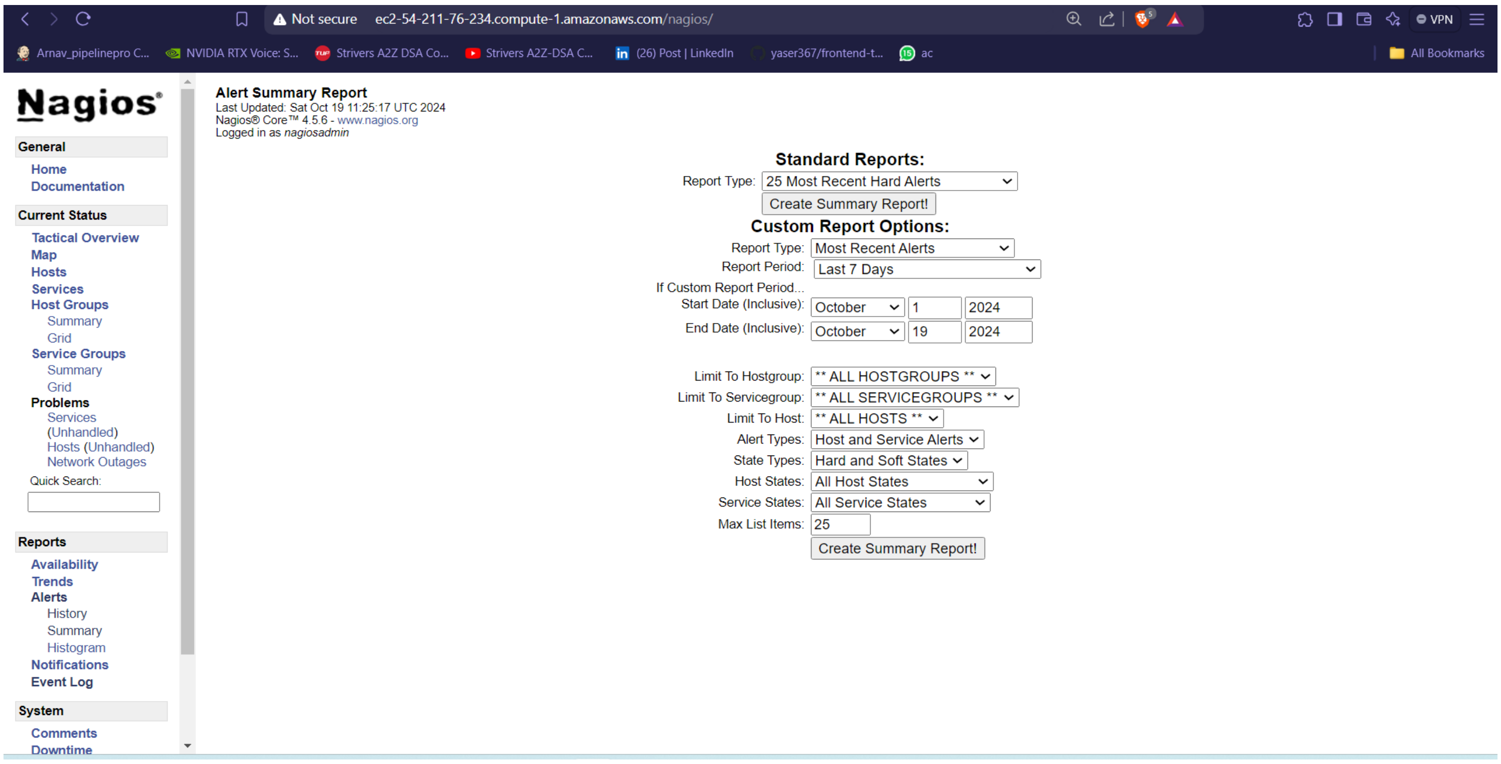
- *Go to http://your-ec2-public-dns/nagios and log in with the user you created.*



Sign in

http://ec2-54-211-76-234.compute-1.amazonaws.com

Your connection to this site is not private


Alert Summary Report
Last Updated: Sat Oct 19 11:25:17 UTC 2024
Nagios® Core™ 4.5.6 - www.nagios.org
Logged in as *nagiosadmin*

Standard Reports:
Report Type: 25 Most Recent Hard Alerts
Create Summary Report!

Custom Report Options:
Report Type: Most Recent Alerts
Report Period: Last 7 Days
If Custom Report Period...
Start Date (Inclusive): October 1 2024
End Date (Inclusive): October 19 2024

Limit To Hostgroup: ** ALL HOSTGROUPS **
Limit To Servicegroup: ** ALL SERVICEGROUPS **
Limit To Host: ** ALL HOSTS **
Alert Types: Host and Service Alerts
State Types: Hard and Soft States
Host States: All Host States
Service States: All Service States
Max List Items: 25
Create Summary Report!

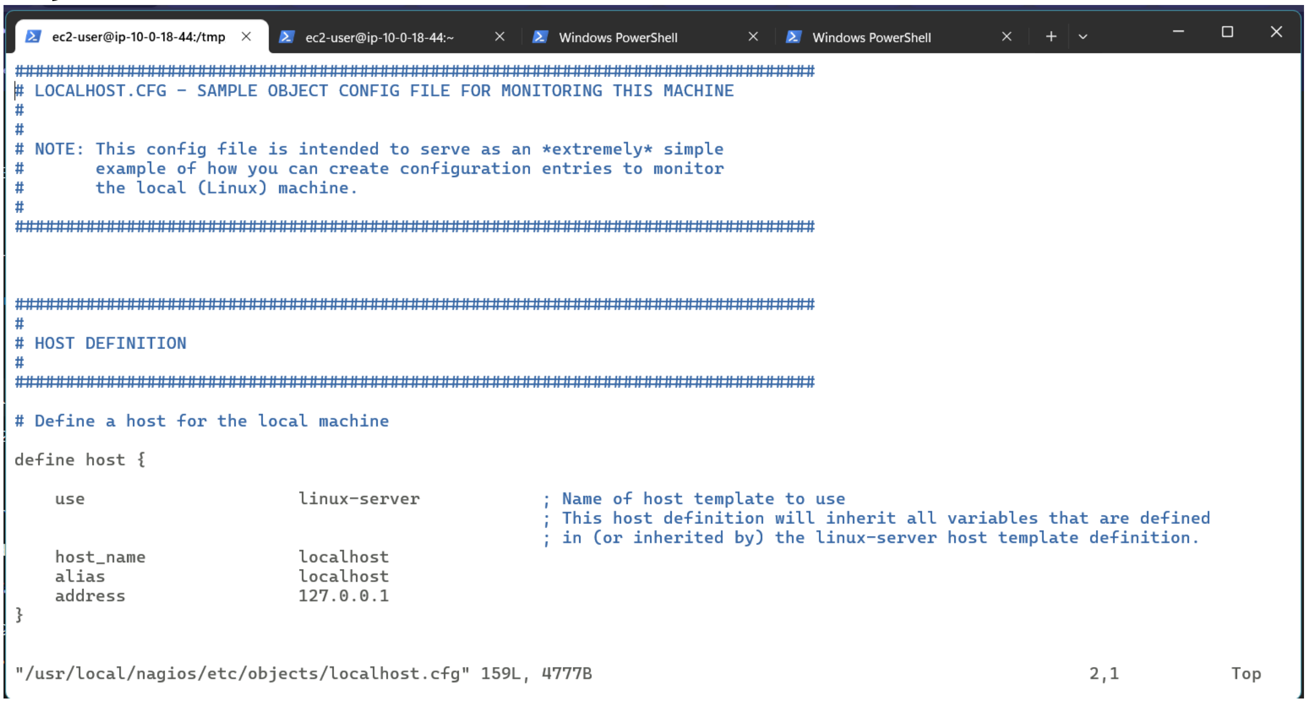
!!!!Task 2 Completed

Step 9: Monitor the Application with Nagios

Configure Nagios to Monitor Your Web Application:
Add a new command in /usr/local/nagios/etc/objects/localhost.cfg:

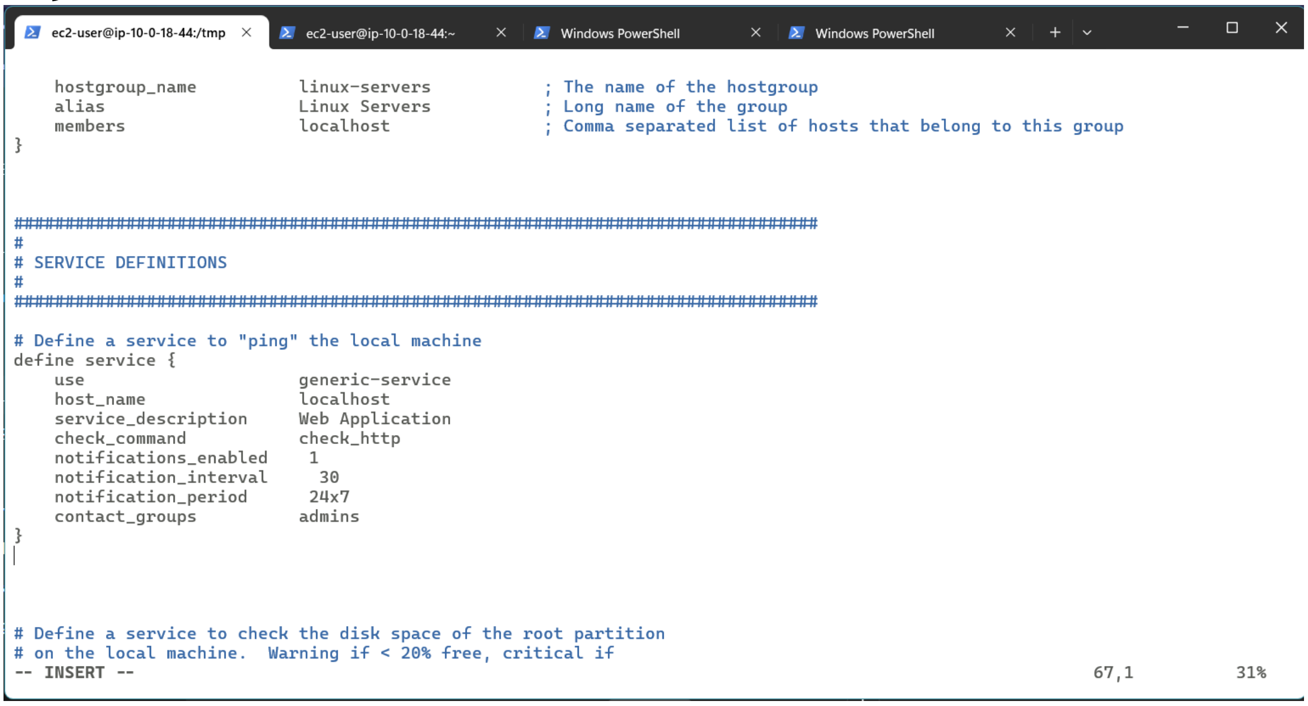
```
define service {
    use                     generic-service
    host_name               localhost
    service_description     Web Application
    check_command           check_http
}

define host {
    use                     linux-server
    host_name               EC2-Server
    alias                   EC2 Server
    address                 ec2-54-211-76-234.compute-1.amazonaws.com
    max_check_attempts      3
    check_period            24x7
    notification_interval    30
    notification_period     24x7
    contact_groups          admins
    }
```

Define Host



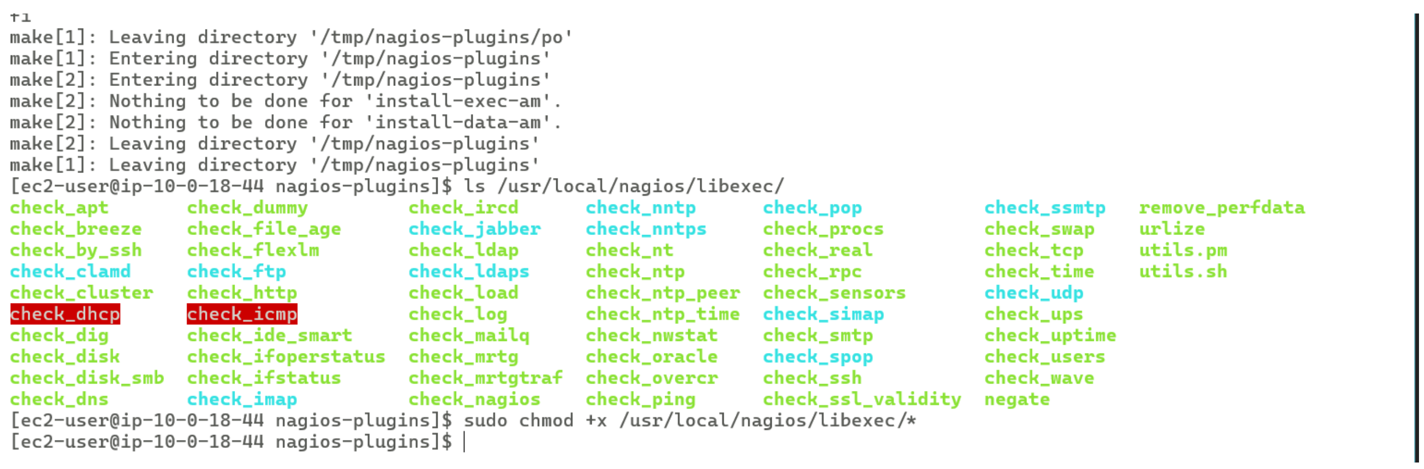
```
###############################################################################
# LOCALHOST.CFG - SAMPLE OBJECT CONFIG FILE FOR MONITORING THIS MACHINE
#
#
# NOTE: This config file is intended to serve as an *extremely* simple
#       example of how you can create configuration entries to monitor
#       the local (Linux) machine.
#
###############################################################################



###############################################################################
#
# HOST DEFINITION
#
###############################################################################

# Define a host for the local machine

define host {

    use                     linux-server            ; Name of host template to use
                                                    ; This host definition will inherit all variables that are defined
                                                    ; in (or inherited by) the linux-server host template definition.

    host_name               localhost
    alias                   localhost
    address                 127.0.0.1
}

"/usr/local/nagios/etc/objects/localhost.cfg" 159L, 4777B                                   2,1           Top
```

Define Service



```
    hostgroup_name          linux-servers           ; The name of the hostgroup
    alias                   Linux Servers           ; Long name of the group
    members                 localhost               ; Comma separated list of hosts that belong to this group
}


###############################################################################
#
# SERVICE DEFINITIONS
#
###############################################################################

# Define a service to "ping" the local machine
define service {
    use                     generic-service
    host_name               localhost
    service_description     Web Application
    check_command           check_http
    notifications_enabled     1
    notification_interval      30
    notification_period       24x7
    contact_groups          admins
}



# Define a service to check the disk space of the root partition
# on the local machine.  Warning if < 20% free, critical if
-- INSERT --                                                                                 67,1          31%
```

Install Nagios Necessary Plugins



```
+1
make[1]: Leaving directory '/tmp/nagios-plugins/po'
make[1]: Entering directory '/tmp/nagios-plugins'
make[2]: Entering directory '/tmp/nagios-plugins'
make[2]: Nothing to be done for 'install-exec-am'.
make[2]: Nothing to be done for 'install-data-am'.
make[2]: Leaving directory '/tmp/nagios-plugins'
make[1]: Leaving directory '/tmp/nagios-plugins'
[ec2-user@ip-10-0-18-44 nagios-plugins]$ ls /usr/local/nagios/libexec/
check_apt        check_dummy         check_ircd       check_nntp        check_pop            check_ssmtp    remove_perfdata
check_breeze     check_file_age      check_jabber     check_nntps       check_procs          check_swap     urlize
check_by_ssh     check_flexlm        check_ldap       check_nt          check_real           check_tcp      utils.pm
check_clamd      check_ftp           check_ldaps      check_ntp         check_rpc            check_time     utils.sh
check_cluster    check_http          check_load       check_ntp_peer    check_sensors        check_udp
check_dhcp       check_icmp          check_log        check_ntp_time    check_simap          check_ups
check_dig        check_ide_smart     check_mailq      check_nwstat      check_smtp           check_uptime
check_disk       check_ifoperstatus  check_mrtg       check_oracle      check_spop           check_users
check_disk_smb   check_ifstatus      check_mrtgtraf   check_overcr      check_ssh            check_wave
check_dns        check_imap          check_nagios     check_ping        check_ssl_validity   negate
[ec2-user@ip-10-0-18-44 nagios-plugins]$ sudo chmod +x /usr/local/nagios/libexec/*
[ec2-user@ip-10-0-18-44 nagios-plugins]$ |
```

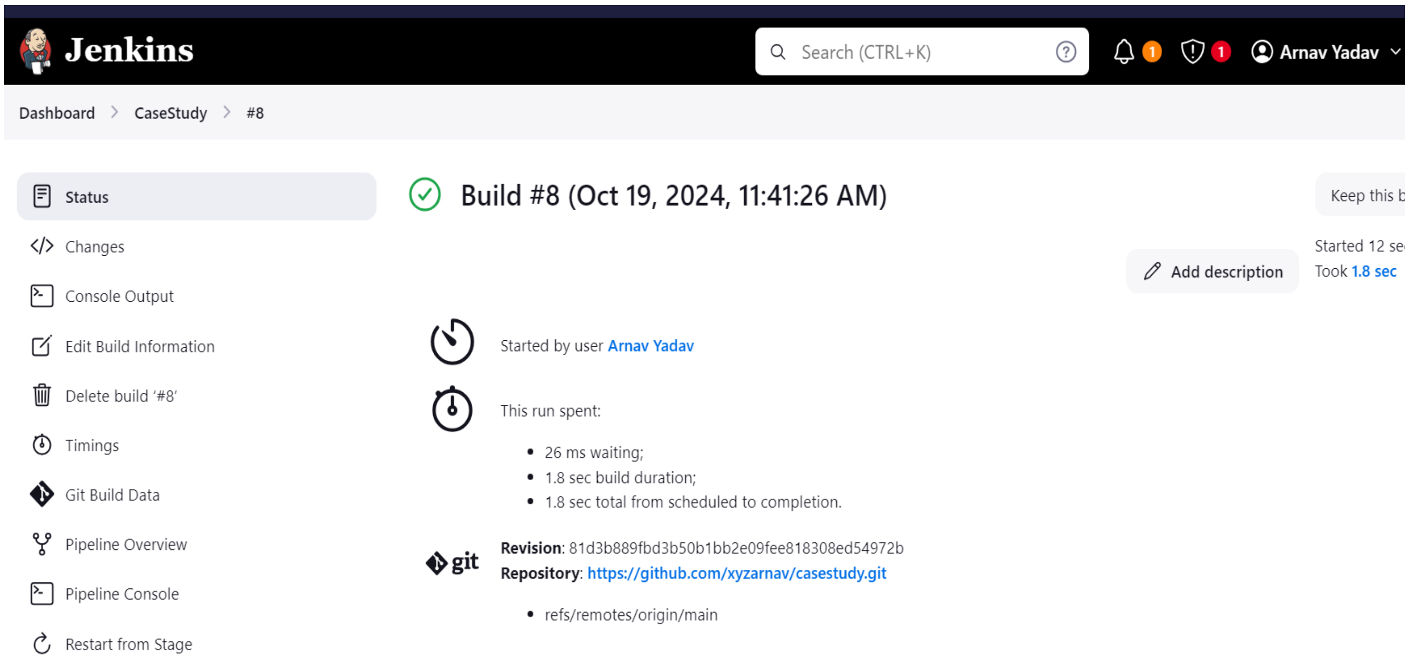
Restart Nagios
sudo systemctl restart nagios

Verify Configuration
Finally, verify your configuration:
bash
Copy code
sudo nagios -v /usr/local/nagios/etc/nagios.cfg

Step 10 : Test the Pipeline

Trigger a Build:

- *Go to your Jenkins job and click "Build Now."*



Jenkins | Search (CTRL+K) | Arnav Yadav

Dashboard > CaseStudy > #8

- Status
- Changes
- Console Output
- Edit Build Information
- Delete build '#8'
- Timings
- Git Build Data
- Pipeline Overview
- Pipeline Console
- Restart from Stage

Build #8 (Oct 19, 2024, 11:41:26 AM)

Keep this b

Started 12 se
Took 1.8 sec

Add description

Started by user Arnav Yadav

This run spent:

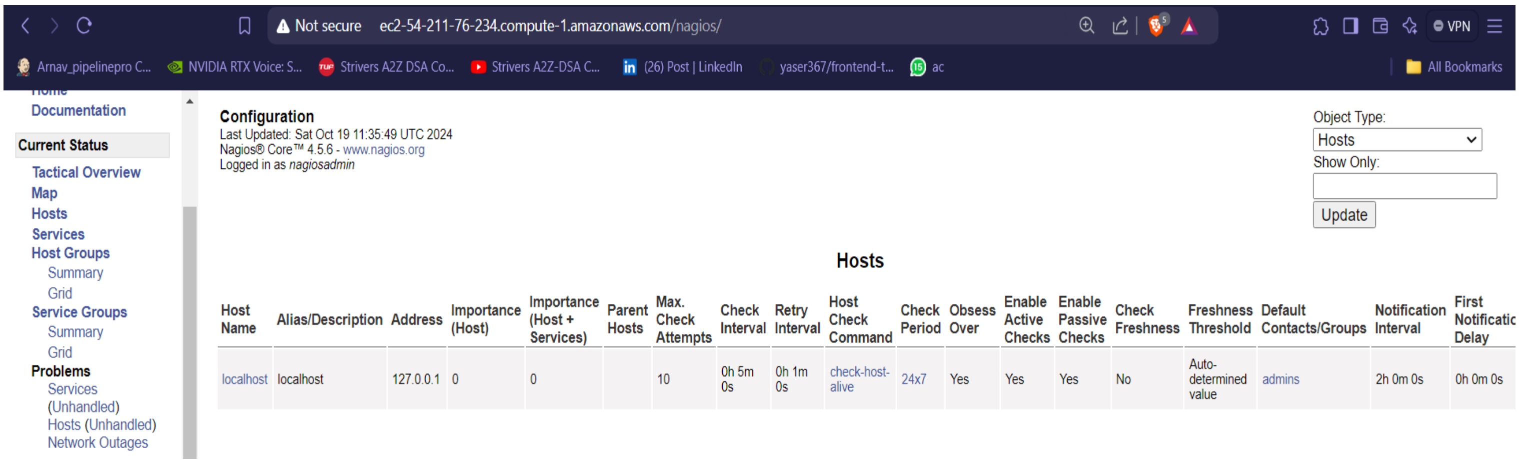
- 26 ms waiting;
- 1.8 sec build duration;
- 1.8 sec total from scheduled to completion.

Revision: 81d3b889fbd3b50b1bb2e09fee818308ed54972b
Repository: https://github.com/xyzarnav/casestudy.git

- refs/remotes/origin/main

ARNAV YADAV (D15B - 66)

Check Nagios Configuration:

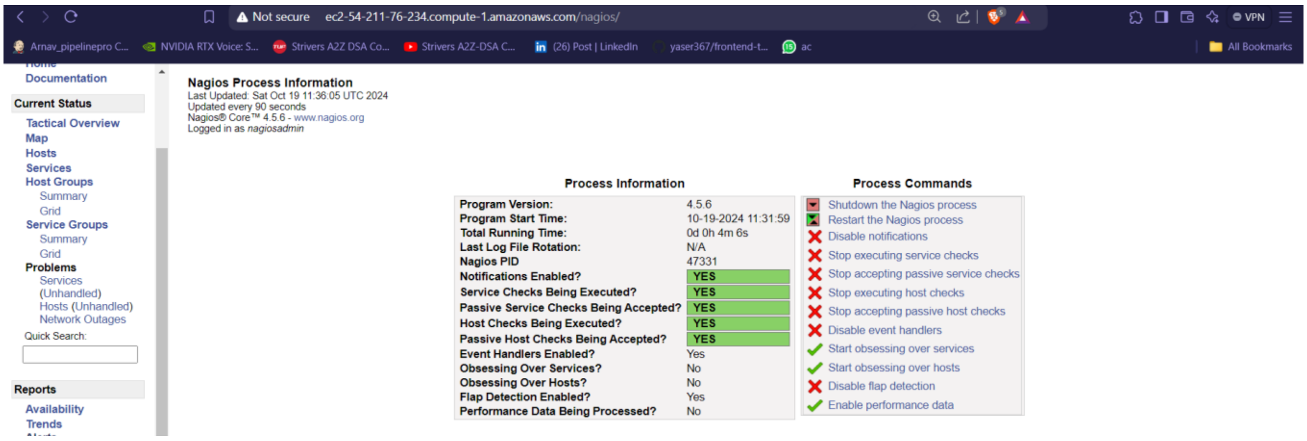


Configuration
Last Updated: Sat Oct 19 11:35:49 UTC 2024
Nagios® Core™ 4.5.6 - www.nagios.org
Logged in as *nagiosadmin*

Hosts

Host Name	Alias/Description	Address	Importance (Host)	Importance (Host + Services)	Parent Hosts	Max. Check Attempts	Check Interval	Retry Interval	Host Check Command	Check Period	Obsess Over	Enable Active Checks	Enable Passive Checks	Check Freshness	Freshness Threshold	Default Contacts/Groups	Notification Interval	First Notificatic Delay
localhost	localhost	127.0.0.1	0	0		10	0h 5m 0s	0h 1m 0s	check-host-alive	24x7	Yes	Yes	Yes	No	Auto-determined value	admins	2h 0m 0s	0h 0m 0s

ProcessInfo



Nagios Process Information
Last Updated: Sat Oct 19 11:36:05 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.6 - www.nagios.org
Logged in as *nagiosadmin*

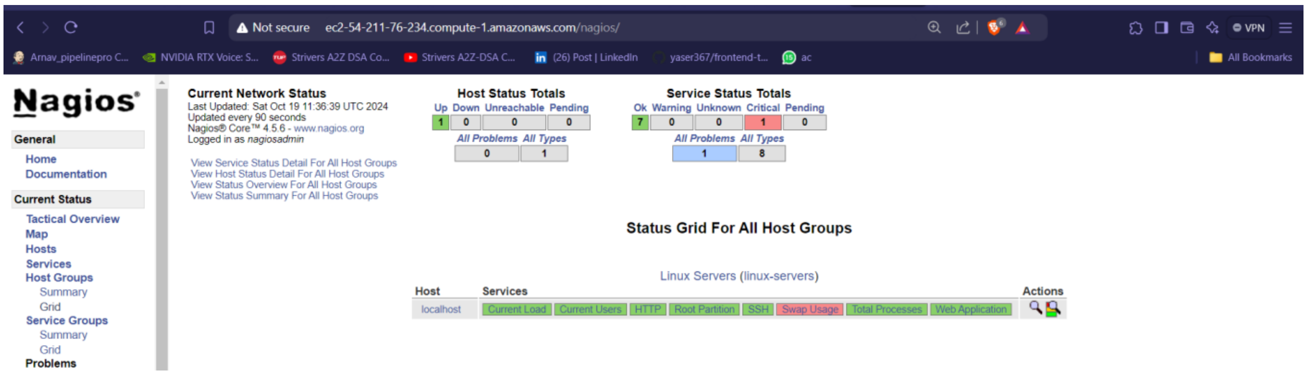
Process Information

Program Version:	4.5.6
Program Start Time:	10-19-2024 11:31:59
Total Running Time:	0d 0h 4m 6s
Last Log File Rotation:	N/A
Nagios PID	47331
Notifications Enabled?	YES
Service Checks Being Executed?	YES
Passive Service Checks Being Accepted?	YES
Host Checks Being Executed?	YES
Passive Host Checks Being Accepted?	YES
Event Handlers Enabled?	Yes
Obsessing Over Services?	No
Obsessing Over Hosts?	No
Flap Detection Enabled?	Yes
Performance Data Being Processed?	No

Process Commands

- Shutdown the Nagios process
- Restart the Nagios process
- ✗ Disable notifications
- ✗ Stop executing service checks
- ✗ Stop accepting passive service checks
- ✗ Stop executing host checks
- ✗ Stop accepting passive host checks
- ✗ Disable event handlers
- ✓ Start obsessing over services
- ✓ Start obsessing over hosts
- ✗ Disable flap detection
- ✓ Enable performance data

Network Status-Grid



Current Network Status
Last Updated: Sat Oct 19 11:36:39 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.6 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Host Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
1	0	0	0

All Problems	*All Types*
0	1

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
7	0	0	1	0

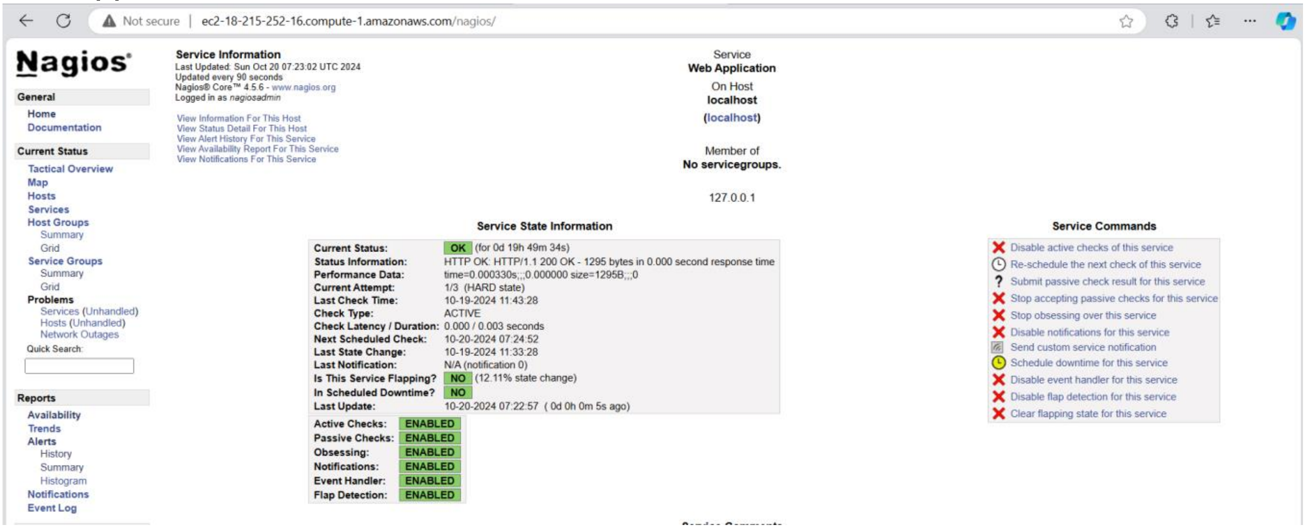
All Problems	*All Types*
1	8

Status Grid For All Host Groups

Linux Servers (linux-servers)

Host	Services	Actions
localhost	Current Load, Current Users, HTTP, Root Partition, SSH, Swap Usage, Total Processes, Web Application	

Web-Application Status



Service Information
Last Updated: Sun Oct 20 07:23:02 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.6 - www.nagios.org
Logged in as *nagiosadmin*

View Information For This Host
View Status Detail For This Host
View Alert History For This Service
View Availability Report For This Service
View Notifications For This Service

Service
Web Application
On Host
localhost
(localhost)

Member of
No servicegroups.

127.0.0.1

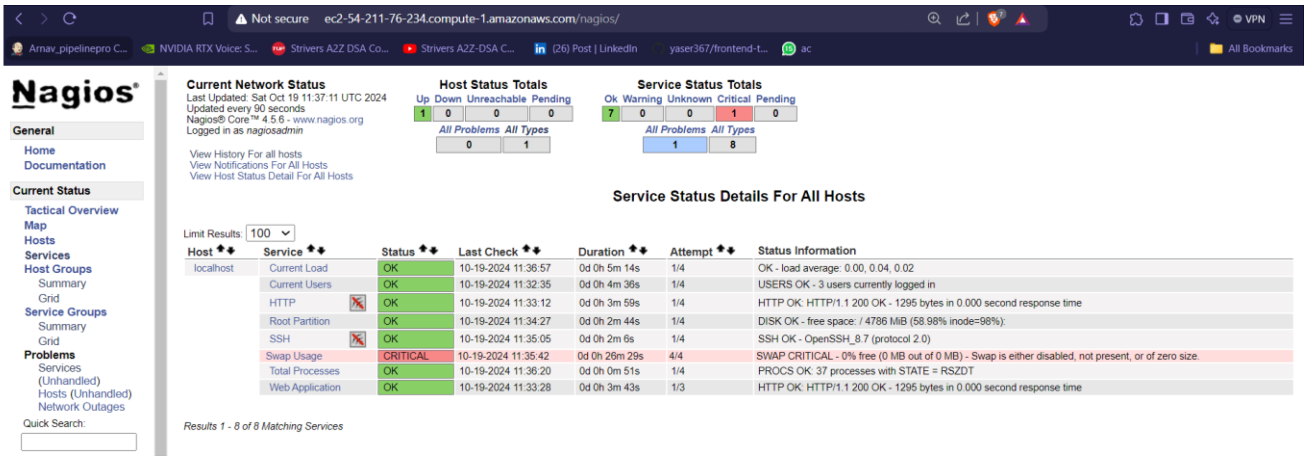
Service State Information

Current Status:	OK (for 0d 19h 49m 34s)
Status Information:	HTTP OK: HTTP/1.1 200 OK - 1295 bytes in 0.000 second response time
Performance Data:	time=0.000330s;;;0.000000 size=1295B;;;0
Current Attempt:	1/3 (HARD state)
Last Check Time:	10-19-2024 11:43:28
Check Type:	ACTIVE
Check Latency / Duration:	0.000 / 0.003 seconds
Next Scheduled Check:	10-20-2024 07:24:52
Last State Change:	10-19-2024 11:33:28
Last Notification:	N/A (notification 0)
Is This Service Flapping?	NO (12.11% state change)
In Scheduled Downtime?	NO
Last Update:	10-20-2024 07:22:57 (0d 0h 0m 5s ago)

Active Checks:	ENABLED
Passive Checks:	ENABLED
Obsessing:	ENABLED
Notifications:	ENABLED
Event Handler:	ENABLED
Flap Detection:	ENABLED

Service Commands
- Disable active checks of this service
- Re-schedule the next check of this service
- Submit passive check result for this service
- Stop accepting passive checks for this service
- Stop obsessing over this service
- Disable notifications for this service
- Send custom service notification
- Schedule downtime for this service
- Disable event handler for this service
- Disable flap detection for this service
- Clear flapping state for this service

HostNetwork Status



Current Network Status
Last Updated: Sat Oct 19 11:37:11 UTC 2024
Updated every 90 seconds
Nagios® Core™ 4.5.6 - www.nagios.org
Logged in as *nagiosadmin*

View History For all hosts
View Notifications For All Hosts
View Host Status Detail For All Hosts

Host Status Totals

Up	Down	Unreachable	Pending
1	0	0	0

All Problems	All Types
0	1

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
7	0	0	1	0

All Problems	All Types
1	8

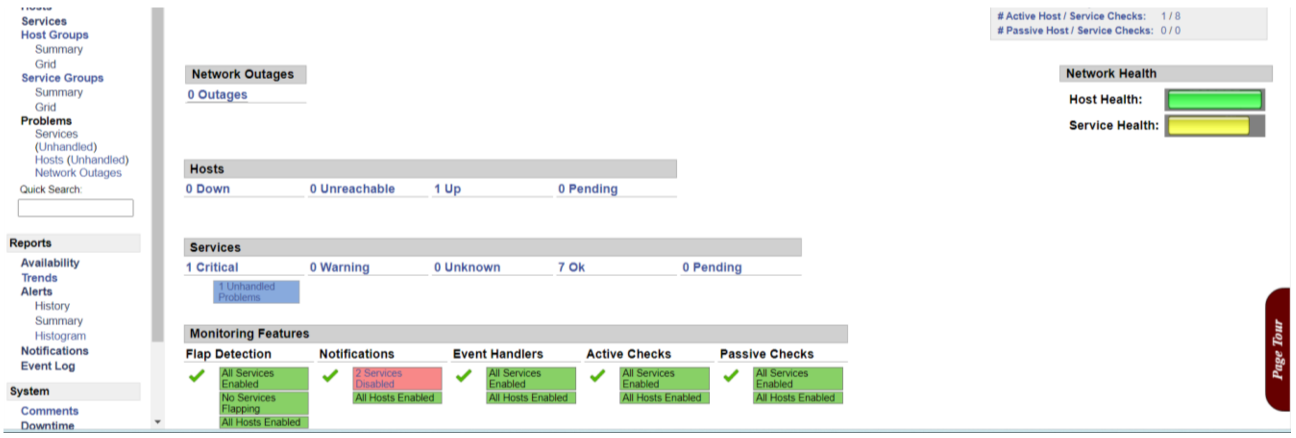
Service Status Details For All Hosts

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
localhost	Current Load	OK	10-19-2024 11:36:57	0d 0h 5m 14s	1/4	OK - load average: 0.00, 0.04, 0.02
	Current Users	OK	10-19-2024 11:32:35	0d 0h 4m 36s	1/4	USERS OK - 3 users currently logged in
	HTTP	OK	10-19-2024 11:33:12	0d 0h 3m 59s	1/4	HTTP OK: HTTP/1.1 200 OK - 1295 bytes in 0.000 second response time
	Root Partition	OK	10-19-2024 11:34:27	0d 0h 2m 44s	1/4	DISK OK - free space: / 4786 MiB (58.98% inode=98%):
	SSH	OK	10-19-2024 11:35:05	0d 0h 2m 6s	1/4	SSH OK - OpenSSH_8.7 (protocol 2.0)
	Swap Usage	CRITICAL	10-19-2024 11:35:42	0d 0h 26m 29s	4/4	SWAP CRITICAL - 0% free (0 MB out of 0 MB) - Swap is either disabled, not present, or of zero size.
	Total Processes	OK	10-19-2024 11:36:20	0d 0h 0m 51s	1/4	PROCS OK: 37 processes with STATE = RSZDT
	Web Application	OK	10-19-2024 11:33:28	0d 0h 3m 43s	1/3	HTTP OK: HTTP/1.1 200 OK - 1295 bytes in 0.000 second response time

Results 1 - 8 of 8 Matching Services

Overview



Active Host / Service Checks: 1 / 8
Passive Host / Service Checks: 0 / 0

Network Outages
0 Outages

Network Health
Host Health:
Service Health:

Hosts

0 Down	0 Unreachable	1 Up	0 Pending

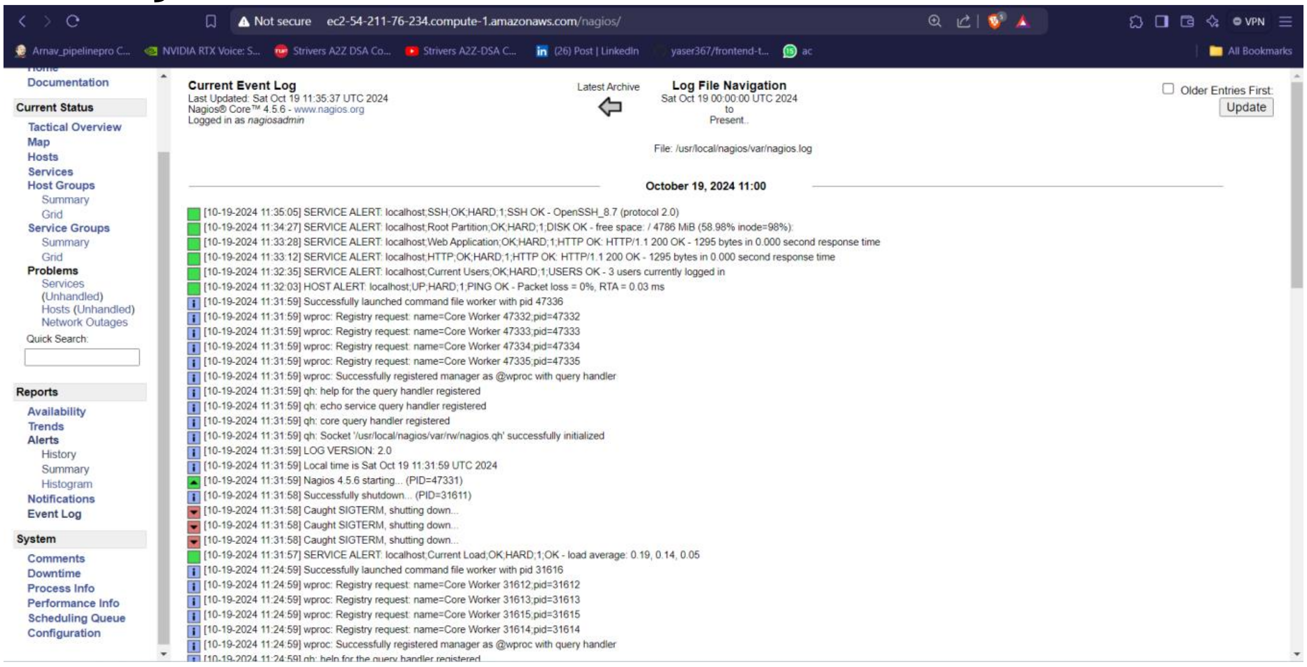
Services

1 Critical	0 Warning	0 Unknown	7 Ok	0 Pending
1 Unhandled Problems				

Monitoring Features

Flap Detection	Notifications	Event Handlers	Active Checks	Passive Checks
✓ All Services Enabled No Services Flapping All Hosts Enabled	✓ 2 Services Disabled All Hosts Enabled	✓ All Services Enabled All Hosts Enabled	✓ All Services Enabled All Hosts Enabled	✓ All Services Enabled All Hosts Enabled

Event Logs:



Current Event Log
Last Updated: Sat Oct 19 11:35:37 UTC 2024
Nagios® Core™ 4.5.6 - www.nagios.org
Logged in as *nagiosadmin*

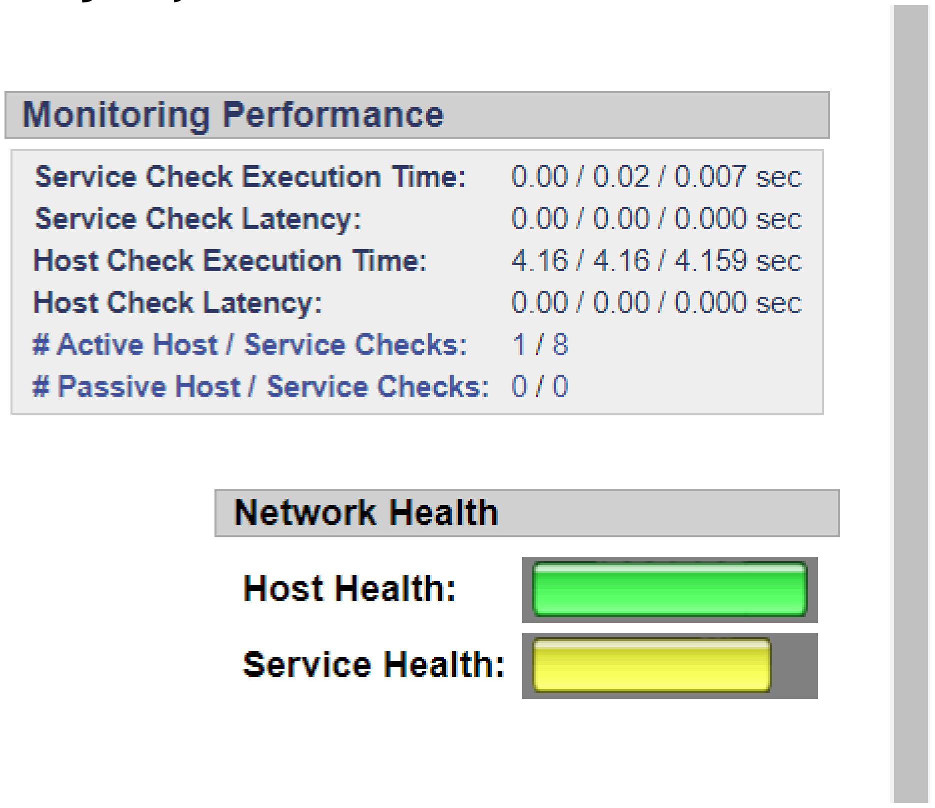
Log File Navigation
Sat Oct 19 00:00:00 UTC 2024
to
Present..

File: /usr/local/nagios/var/nagios.log

October 19, 2024 11:00

```
[10-19-2024 11:35:05] SERVICE ALERT: localhost;SSH;OK;HARD;1;SSH OK - OpenSSH_8.7 (protocol 2.0)
[10-19-2024 11:34:27] SERVICE ALERT: localhost;Root Partition;OK;HARD;1;DISK OK - free space: / 4786 MiB (58.98% inode=98%):
[10-19-2024 11:33:28] SERVICE ALERT: localhost;Web Application;OK;HARD;1;HTTP OK: HTTP/1.1 200 OK - 1295 bytes in 0.000 second response time
[10-19-2024 11:33:12] SERVICE ALERT: localhost;HTTP;OK;HARD;1;HTTP OK: HTTP/1.1 200 OK - 1295 bytes in 0.000 second response time
[10-19-2024 11:32:35] SERVICE ALERT: localhost;Current Users;OK;HARD;1;USERS OK - 3 users currently logged in
[10-19-2024 11:32:03] HOST ALERT: localhost;UP;HARD;1;PING OK - Packet loss = 0%, RTA = 0.03 ms
[10-19-2024 11:31:59] Successfully launched command file worker with pid 47336
[10-19-2024 11:31:59] wproc: Registry request: name=Core Worker 47332;pid=47332
[10-19-2024 11:31:59] wproc: Registry request: name=Core Worker 47333;pid=47333
[10-19-2024 11:31:59] wproc: Registry request: name=Core Worker 47334;pid=47334
[10-19-2024 11:31:59] wproc: Registry request: name=Core Worker 47335;pid=47335
[10-19-2024 11:31:59] wproc: Successfully registered manager as @wproc with query handler
[10-19-2024 11:31:59] qh: help for the query handler registered
[10-19-2024 11:31:59] qh: echo service query handler registered
[10-19-2024 11:31:59] qh: core query handler registered
[10-19-2024 11:31:59] qh: Socket '/usr/local/nagios/var/rw/nagios.qh' successfully initialized
[10-19-2024 11:31:59] LOG VERSION: 2.0
[10-19-2024 11:31:59] Local time is Sat Oct 19 11:31:59 UTC 2024
[10-19-2024 11:31:59] Nagios 4.5.6 starting... (PID=47331)
[10-19-2024 11:31:58] Successfully shutdown... (PID=31611)
[10-19-2024 11:31:58] Caught SIGTERM, shutting down...
[10-19-2024 11:31:58] Caught SIGTERM, shutting down...
[10-19-2024 11:31:58] Caught SIGTERM, shutting down...
[10-19-2024 11:31:57] SERVICE ALERT: localhost;Current Load;OK;HARD;1;OK - load average: 0.19, 0.14, 0.05
[10-19-2024 11:24:59] Successfully launched command file worker with pid 31616
[10-19-2024 11:24:59] wproc: Registry request: name=Core Worker 31612;pid=31612
[10-19-2024 11:24:59] wproc: Registry request: name=Core Worker 31613;pid=31613
[10-19-2024 11:24:59] wproc: Registry request: name=Core Worker 31615;pid=31615
[10-19-2024 11:24:59] wproc: Registry request: name=Core Worker 31614;pid=31614
[10-19-2024 11:24:59] wproc: Successfully registered manager as @wproc with query handler
[10-19-2024 11:24:59] qh: help for the query handler registered
```

Monitoring Performance



Monitoring Performance

Service Check Execution Time:	0.00 / 0.02 / 0.007 sec
Service Check Latency:	0.00 / 0.00 / 0.000 sec
Host Check Execution Time:	4.16 / 4.16 / 4.159 sec
Host Check Latency:	0.00 / 0.00 / 0.000 sec
# Active Host / Service Checks:	1 / 8
# Passive Host / Service Checks:	0 / 0

Network Health

Host Health:

Service Health:

!!!!Task 3 Completed

Conclusion:

In this experiment, we successfully installed and configured Nagios Core and Nagios Plugins on an Amazon EC2 instance running a Linux distribution. The process involved several key steps, including:

1. *Setting Up the Environment: We started by ensuring the necessary dependencies were installed, such as Apache, PHP, and required libraries.*
2. *Downloading and Compiling Nagios Core: We downloaded the Nagios Core source from its GitHub repository, compiled the source code, and installed it along with essential configurations.*
3. *Installing Nagios Plugins: After resolving issues with incorrect download links, we fetched and compiled the Nagios Plugins, which are essential for monitoring services.*
4. *Configuring Services and Permissions: We set up appropriate user and group permissions, created Nagios configuration files, and linked them to the Apache web server for Nagios' web interface.*
5. *Troubleshooting: Throughout the installation, we encountered several issues, such as missing users and permissions, Nagios service failures, and plugin errors. We addressed these by creating the required nagios user, ensuring proper permissions, and verifying plugin installations.*