**Adv DevOps CaseStudy**

**Kubernetes Application with Basic Monitoring Concepts Used: Kubernetes, AWS Cloud9, and Nagios.**

**Problem Statement: "Deploy a simple application (e.g., an Nginx server) on a Kubernetes cluster using AWS Cloud9 and monitor its health using Nagios."**

**Tasks: Use AWS Cloud9 to deploy the Nginx server on a Kubernetes cluster. Install and configure Nagios to monitor the Nginx pod's availability. Verify that Nagios can detect when the Nginx pod is running or not.**

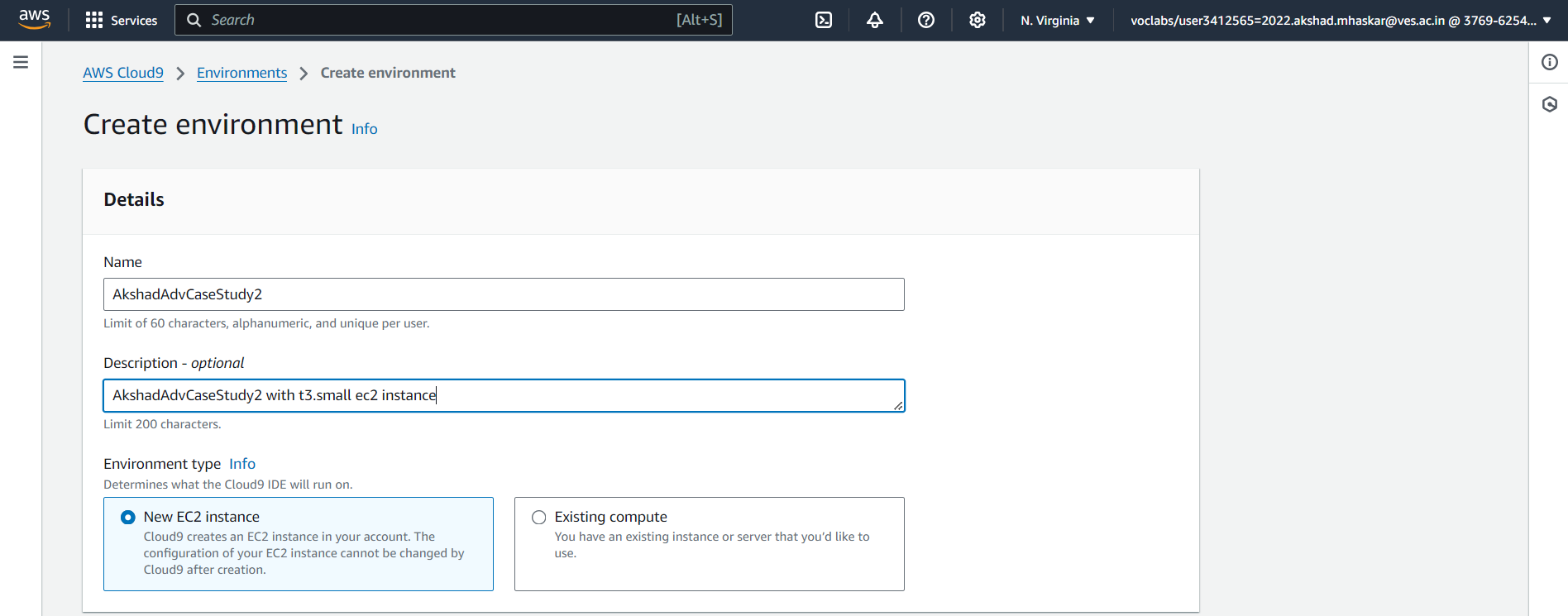
**Theory:**

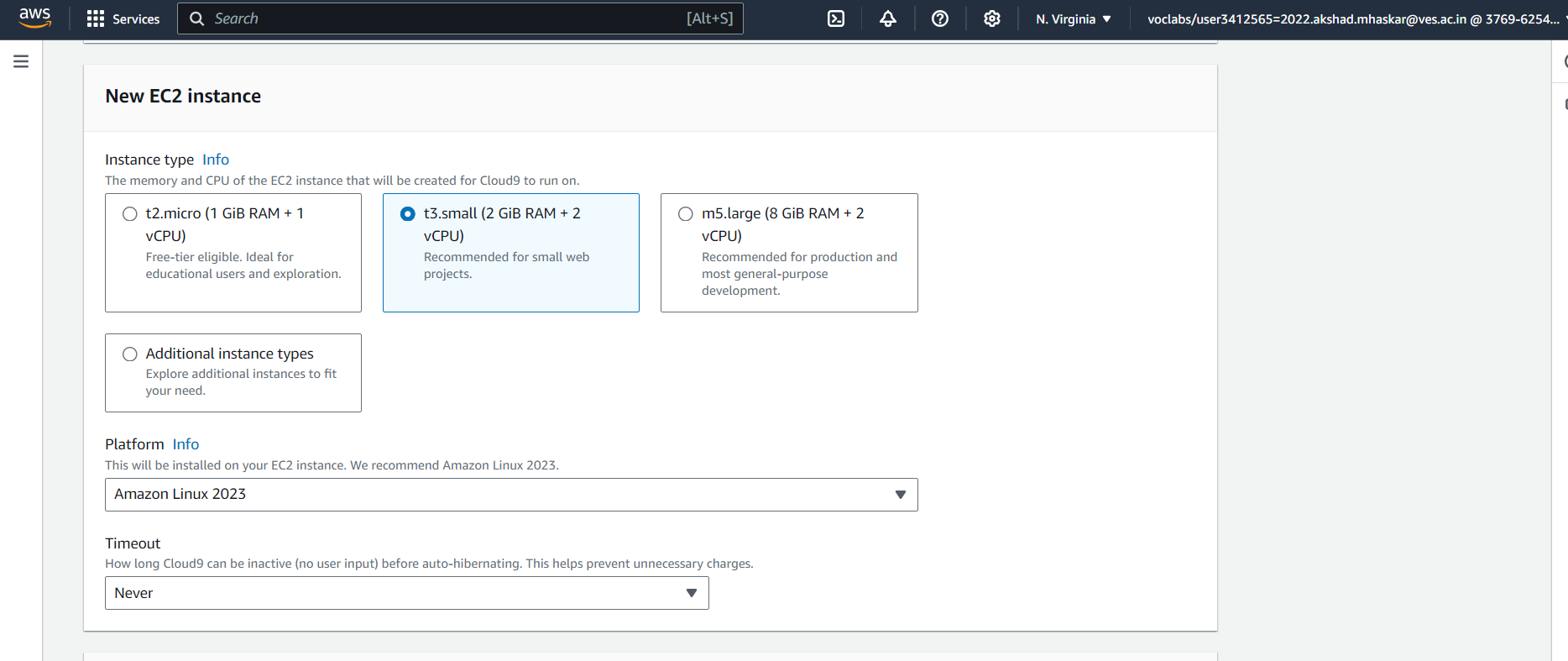
The task involves deploying an **Nginx server** on a **Kubernetes cluster** using **AWS Cloud9** and monitoring its health with **Nagios**. Kubernetes, an orchestration platform, manages containerized applications like Nginx, ensuring scalability and reliability. In this setup, Cloud9 serves as the IDE for managing the Kubernetes cluster using **Minikube**, a tool to run Kubernetes locally. After deploying the Nginx server, exposing it with a NodePort makes the web server accessible externally.

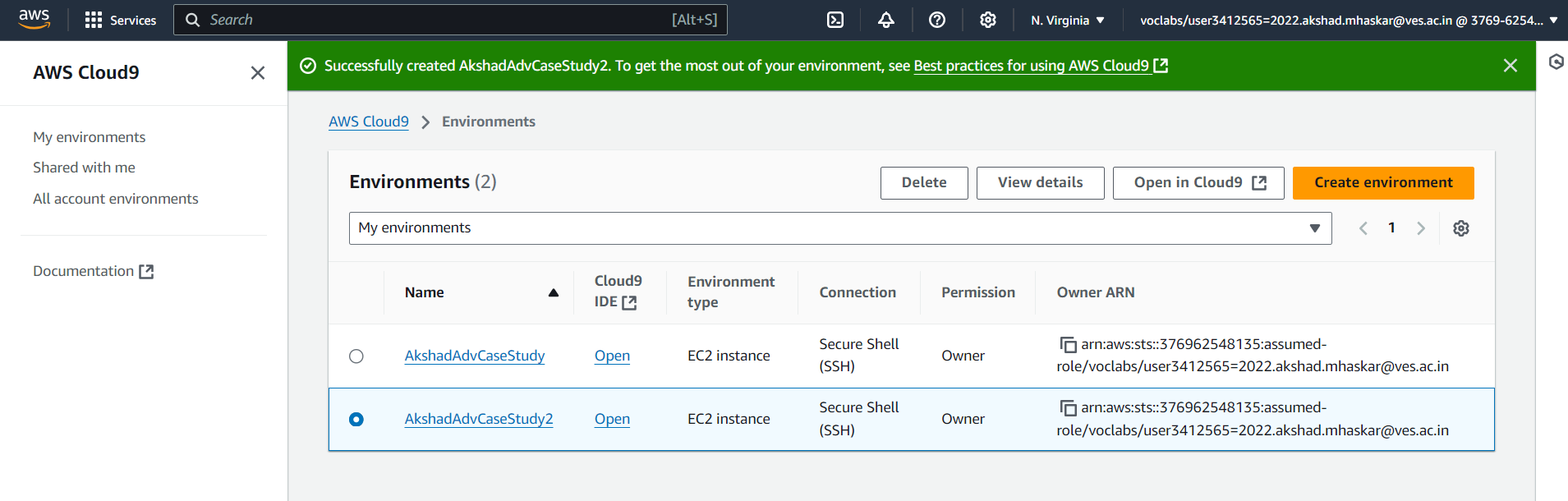
Monitoring is critical in ensuring high availability, and **Nagios**, a widely-used open-source monitoring tool, is configured to observe the status of the Nginx pod. If the Nginx pod stops running, Nagios triggers an alert, allowing timely intervention. This solution demonstrates the integration of Kubernetes for container orchestration and Nagios for active monitoring, ensuring that the application remains operational with minimal downtime.

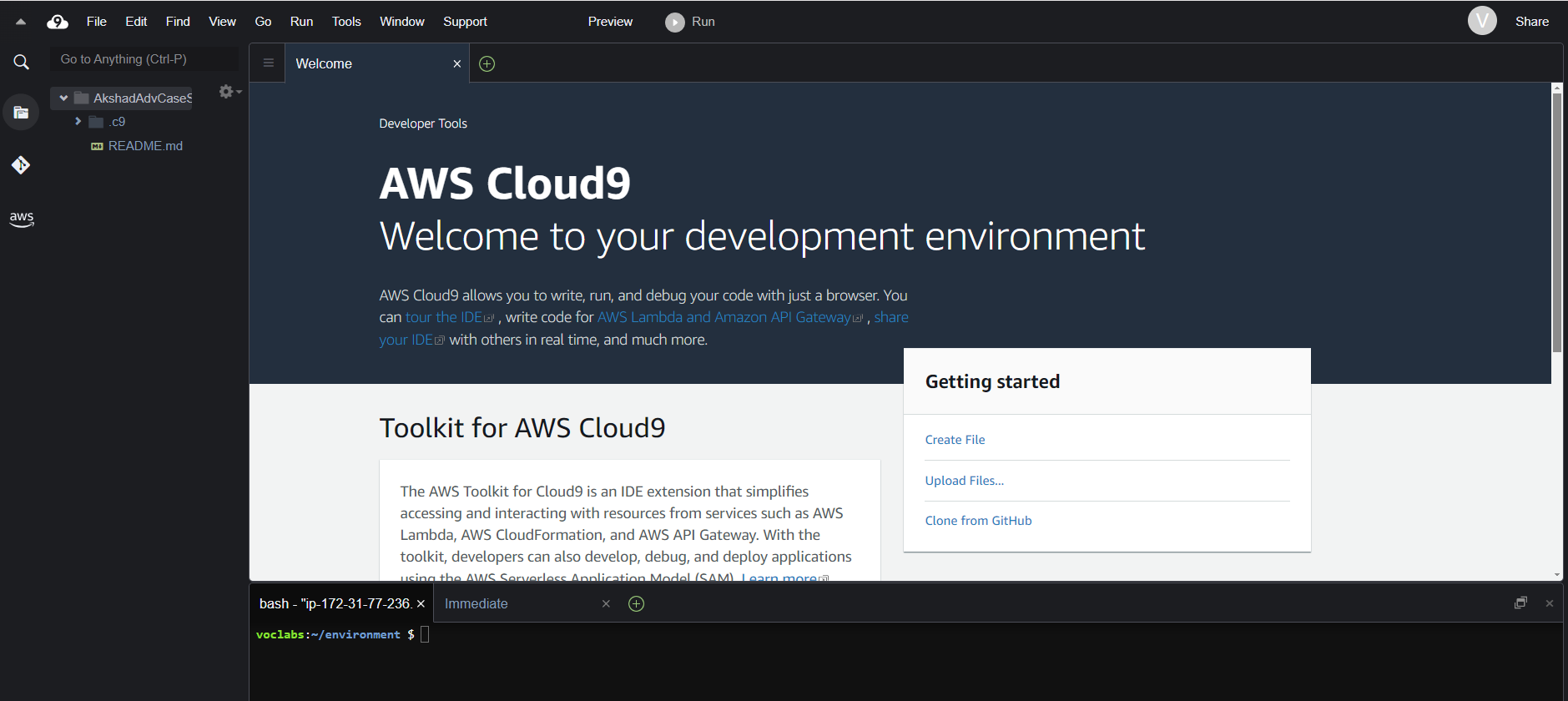
**Steps:**

**1)Create Environment in cloud9**





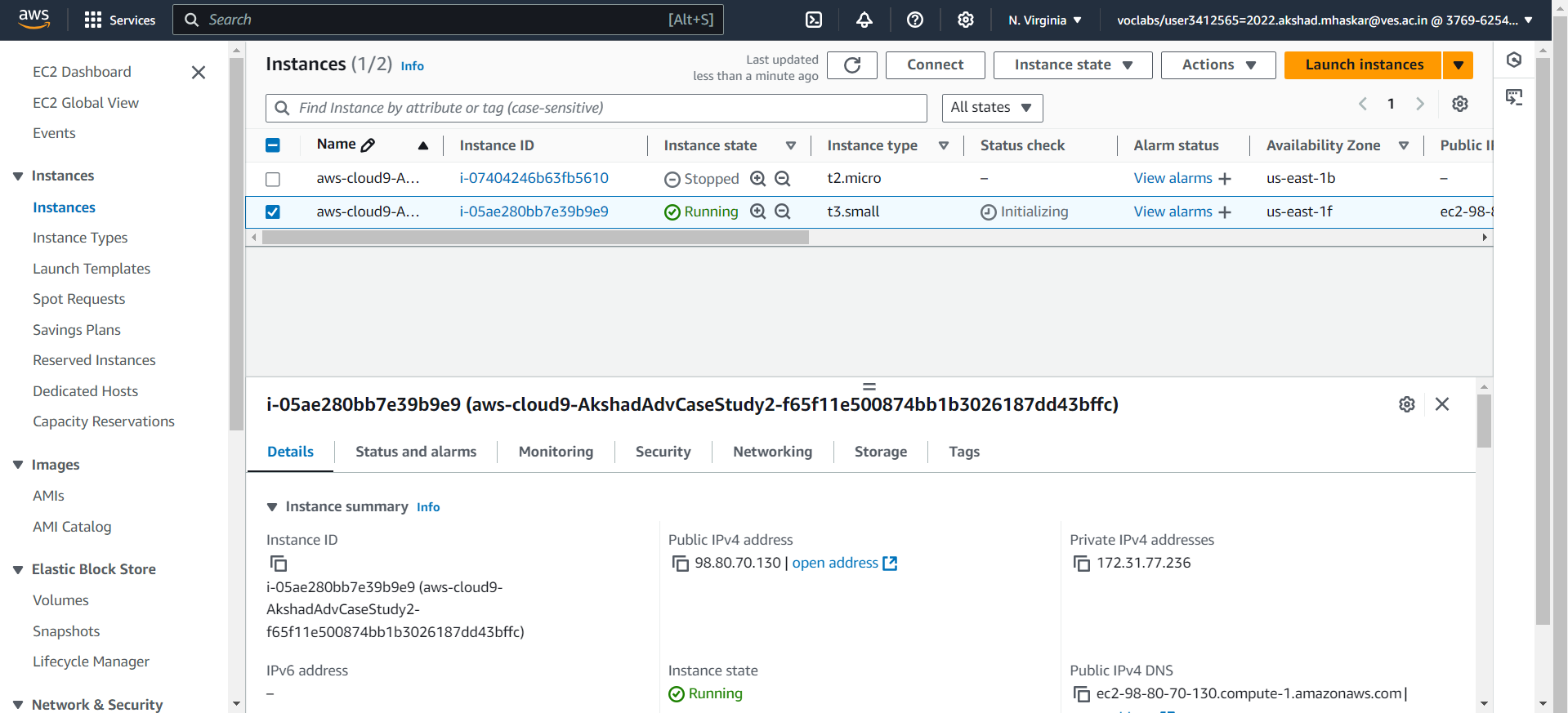


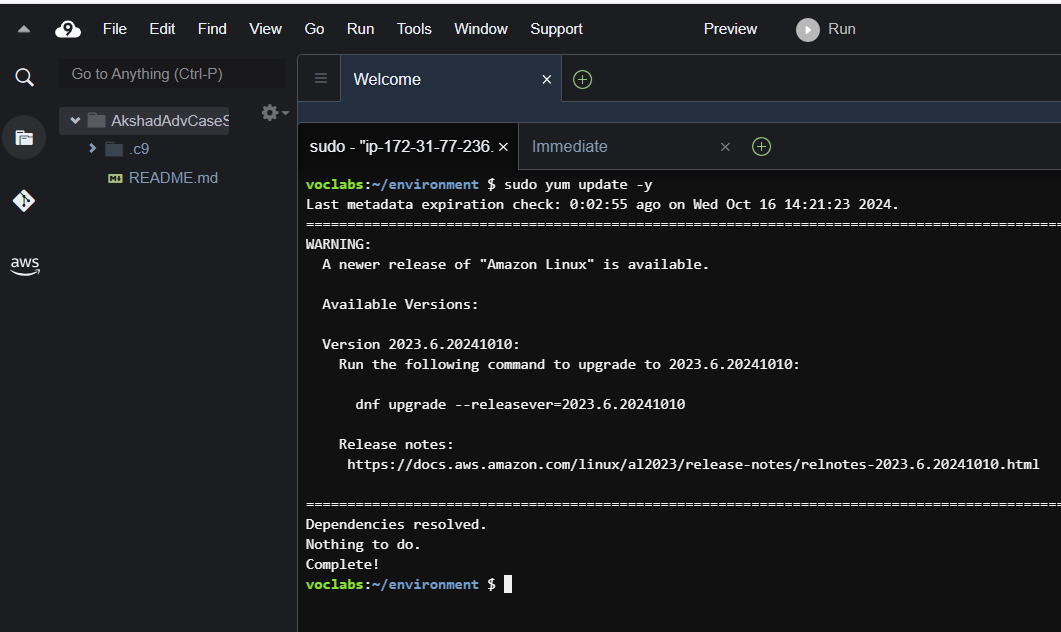
**2)Open Cloud9 Terminal** 

#### **2.1)Install Docker**

Docker is needed to run containers that will host your applications.

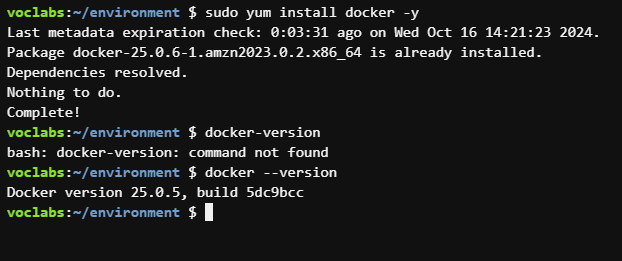
1. **Update the Package List**:
   * In the terminal at the bottom of your Cloud9 IDE, type the following command and press Enter:  
     sudo yum update -y





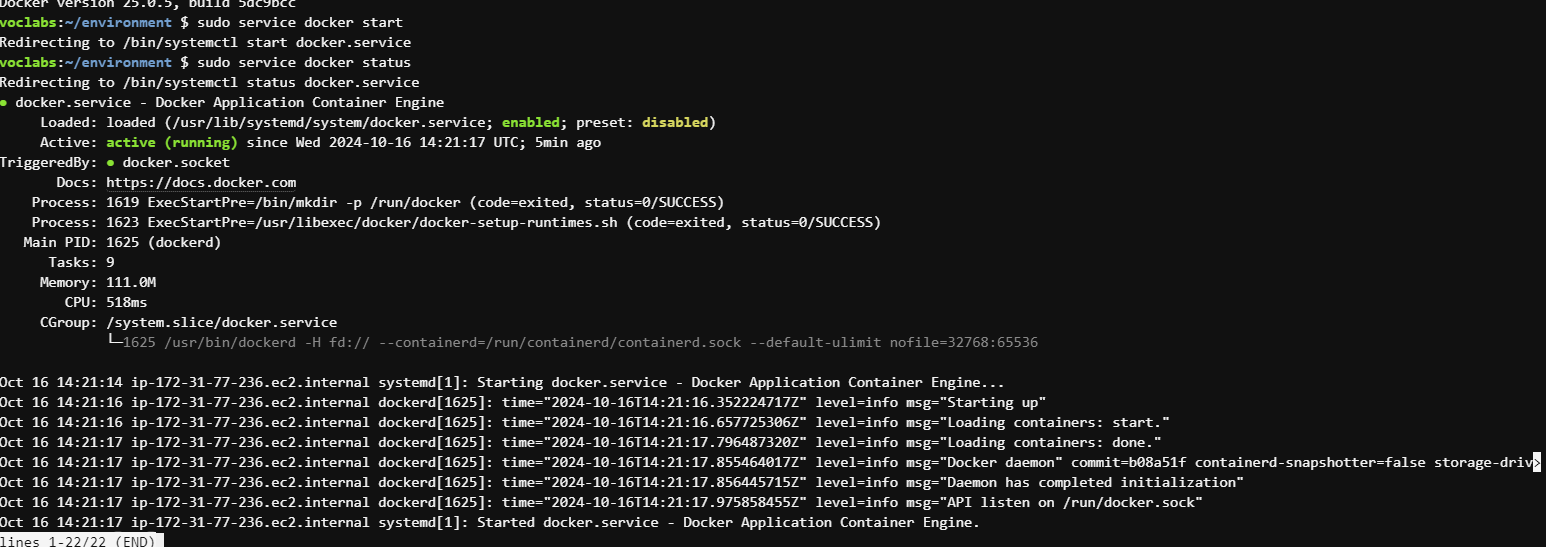
2.Install Docker:

* Type the following command and press Enter:  
  sudo yum install docker -y



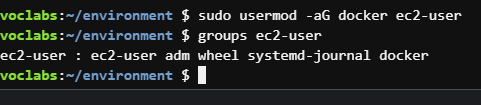
3.Start Docker:

* Run this command to start the Docker service:  
  sudo service docker start



4.Add Your User to the Docker Group:

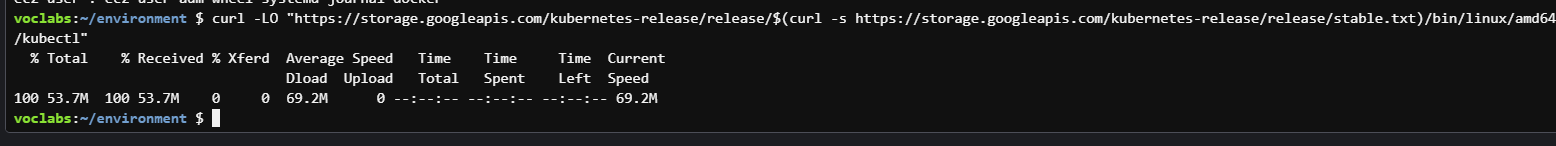
* This allows you to run Docker commands without using sudo. Type:  
  bash  
  Copy code  
  sudo usermod -aG docker ec2-user



#### **2.2 Install kubectl**

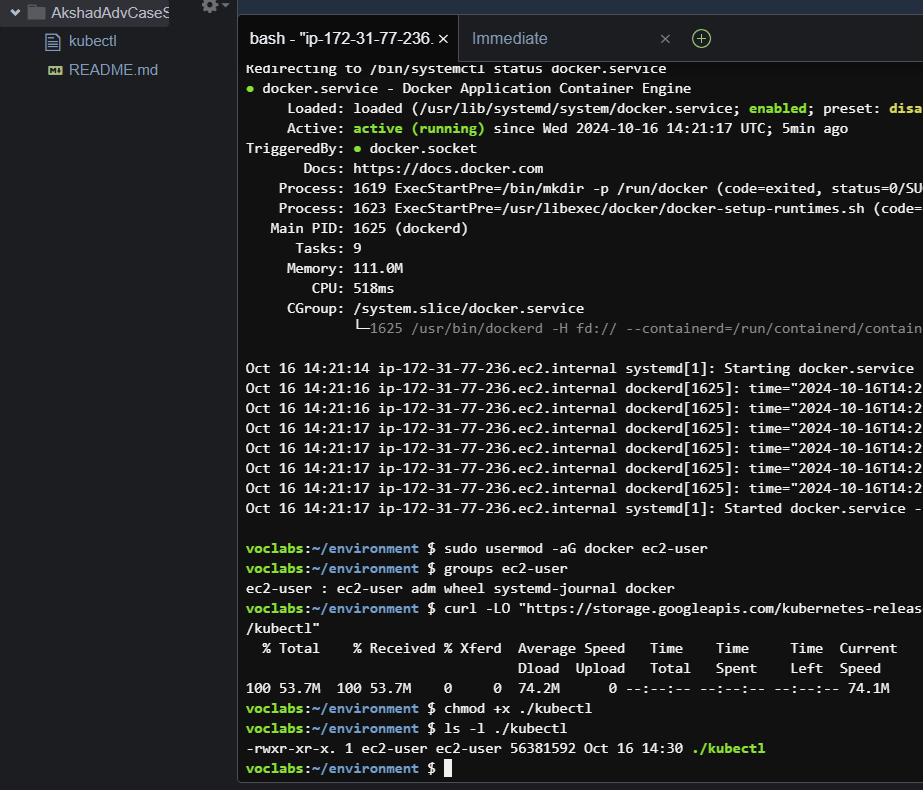
kubectl is a tool to manage Kubernetes from the command line.

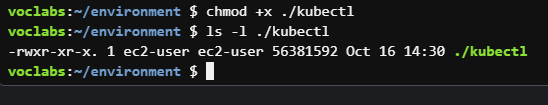
1. Download the kubectl tool:  
   curl -LO "https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl"



2.Make it executable (like turning on a switch):

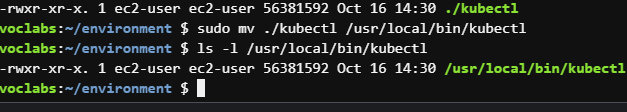
chmod +x ./kubectl





3.Move it to your system path so you can use it:

sudo mv ./kubectl /usr/local/bin/kubectl

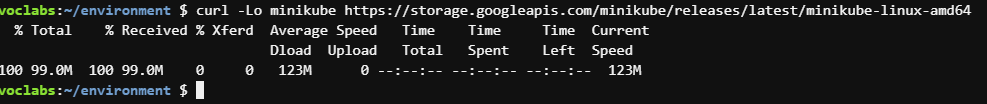


#### **2.3 Install Minikube**

Minikube lets you run Kubernetes on your own computer.

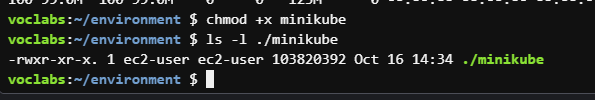
1. Download Minikube:

curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64



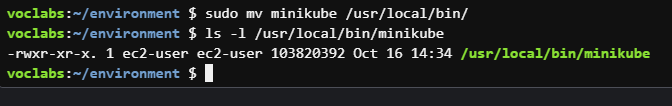
2.Make the Minikube Binary Executable: Run the command below to give permission to the downloaded file to be executed:

chmod +x minikube



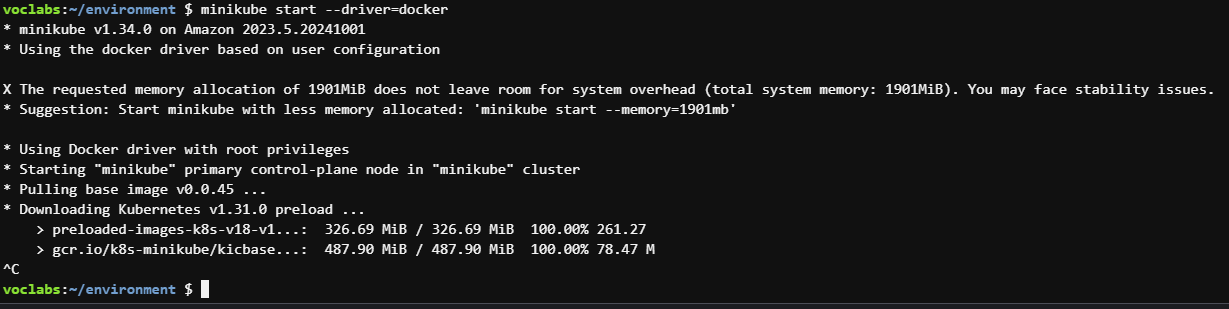
3.Move Minikube to a Directory in Your PATH: Move the Minikube binary to /usr/local/bin, which is a common directory for executable files:

sudo mv minikube /usr/local/bin/

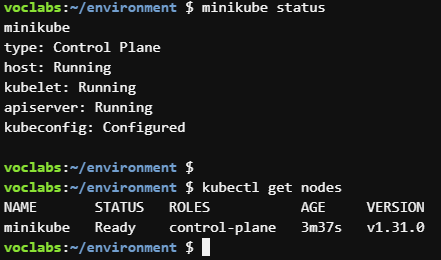


#### **2.4 Start Minikube**

1. Run Minikube (this starts Kubernetes):  
   minikube start --driver=docker



2.Check if Kubernetes is running by typing:  
kubectl get nodes



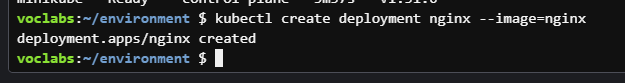
### **3) Deploy Nginx on Kubernetes**

Nginx is a web server. Now, we’ll deploy it on Kubernetes.

#### **3.1 Create Nginx Deployment**

We tell Kubernetes to create a pod (a small unit that runs our app).

1. Run this command to deploy Nginx:  
   kubectl create deployment nginx --image=nginx



#### **3.2 Expose Nginx**

We need to expose the Nginx server so we can access it from a browser.

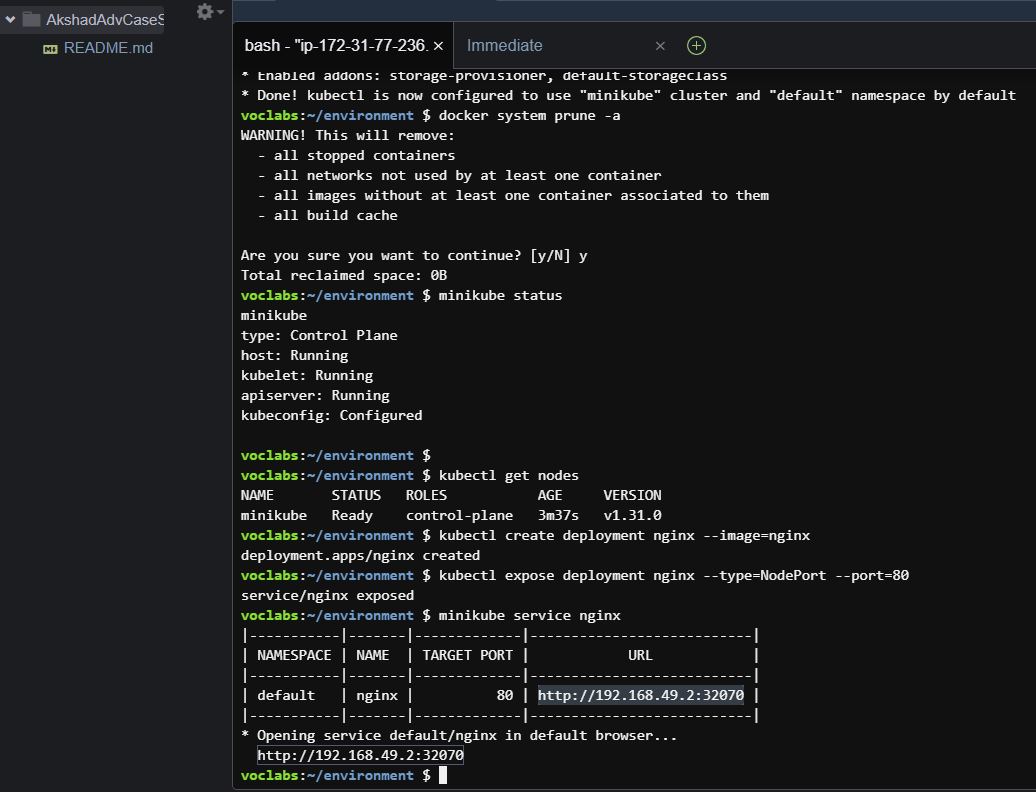
1. Run this command to expose Nginx on port 80:  
   kubectl expose deployment nginx --type=NodePort --port=80

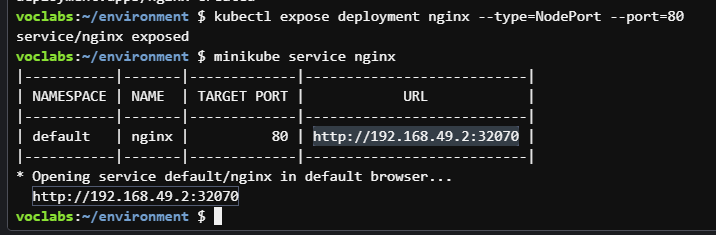


#### **3.3 Get the Service URL**

1. To access the Nginx server, run this command:  
   minikube service nginx

It will show a URL. Copy that URL and open it in a browser. You should see the Nginx welcome page!





### 

### 

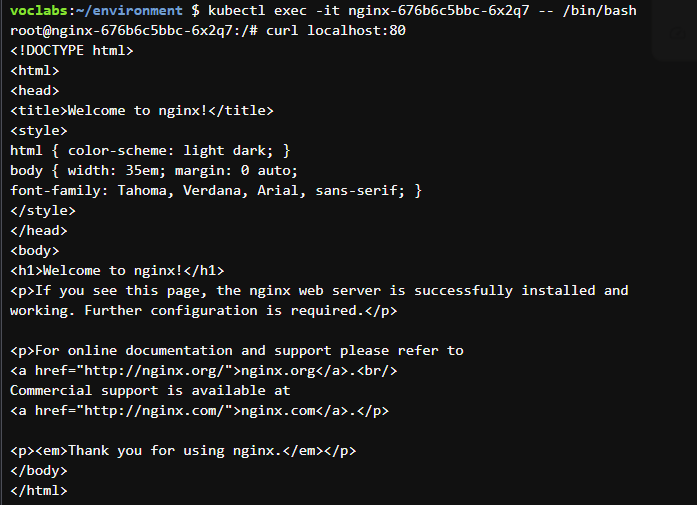
### 

### **As Nginx server is not opening**

### **Step to Exec into the Pod**

**Run the Exec Command**: Use the following command to exec into the pod:  
kubectl exec -it nginx-676b6c5bbc-6x2q7 -- /bin/bash

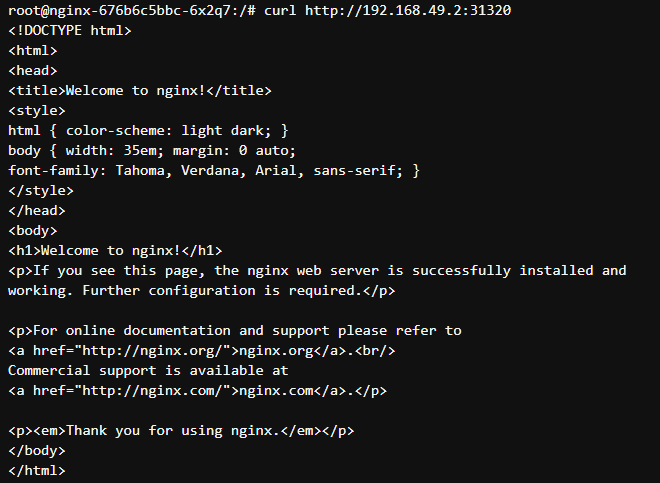
**Test Connectivity**: Once you're inside the pod, run:  
curl localhost:80



### **Using Curl in the Terminal (to Test)**

If you want to see the output directly in the terminal, you can run:

curl <http://192.168.49.2:31320>

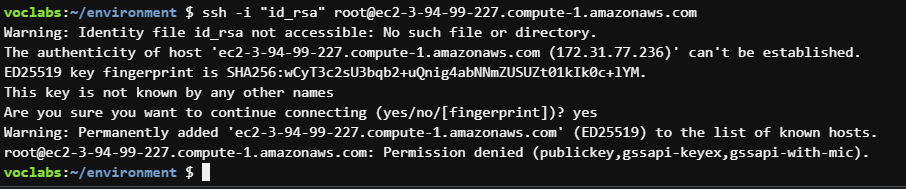


**4) Install and Configure Nagios**

#### **1.Install Nagios**

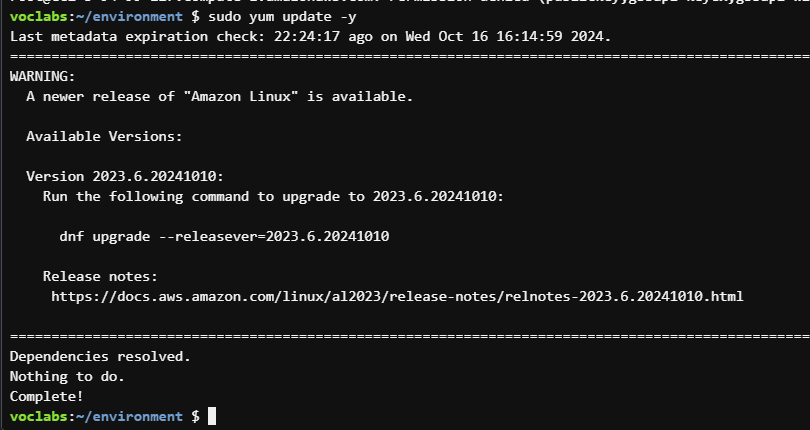
1. **SSH into your Cloud9 environment** (if not already):

ssh -i "id\_rsa" [root@ec2-3-94-99-227.compute-1.amazonaws.com](mailto:root@ec2-3-94-99-227.compute-1.amazonaws.com)



**2.Update the package manager**:

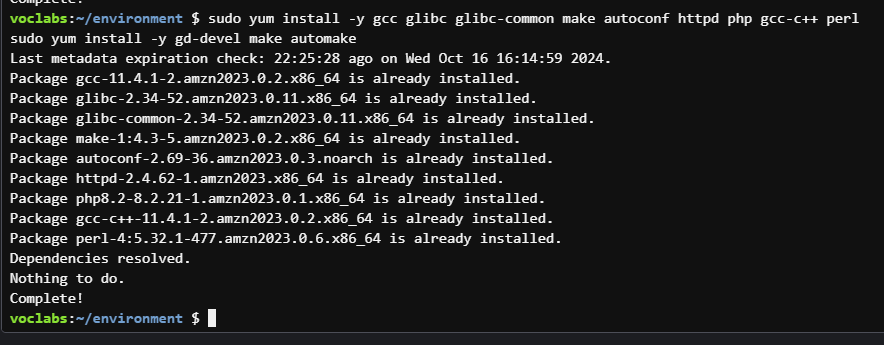
sudo yum update -y # For Amazon Linux



**Install required dependencies**:

sudo yum install -y gcc glibc glibc-common make autoconf httpd php gcc-c++ perl

sudo yum install -y gd-devel make automake



### **Download and Install Nagios 4.4.9**

1. **Change to the /tmp directory**:  
   cd /tmp



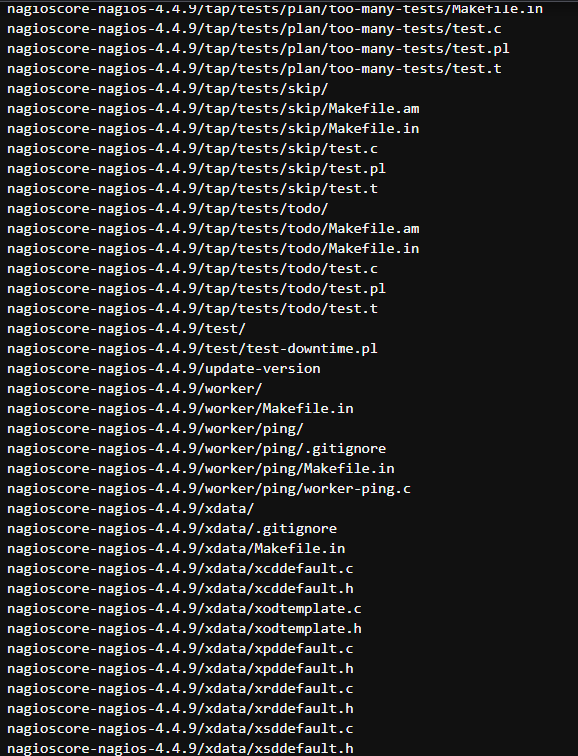
**Download Nagios 4.4.9**:

wget https://github.com/NagiosEnterprises/nagioscore/archive/refs/tags/nagios-4.4.9.tar.gz



**Extract the downloaded file**:

tar -zxvf nagios-4.4.9.tar.gz



**Navigate to the extracted Nagios directory**:

cd /nagioscore-nagios-4.4.9



**Compile and Install Nagios**:

./configure --with-command-group=nagios

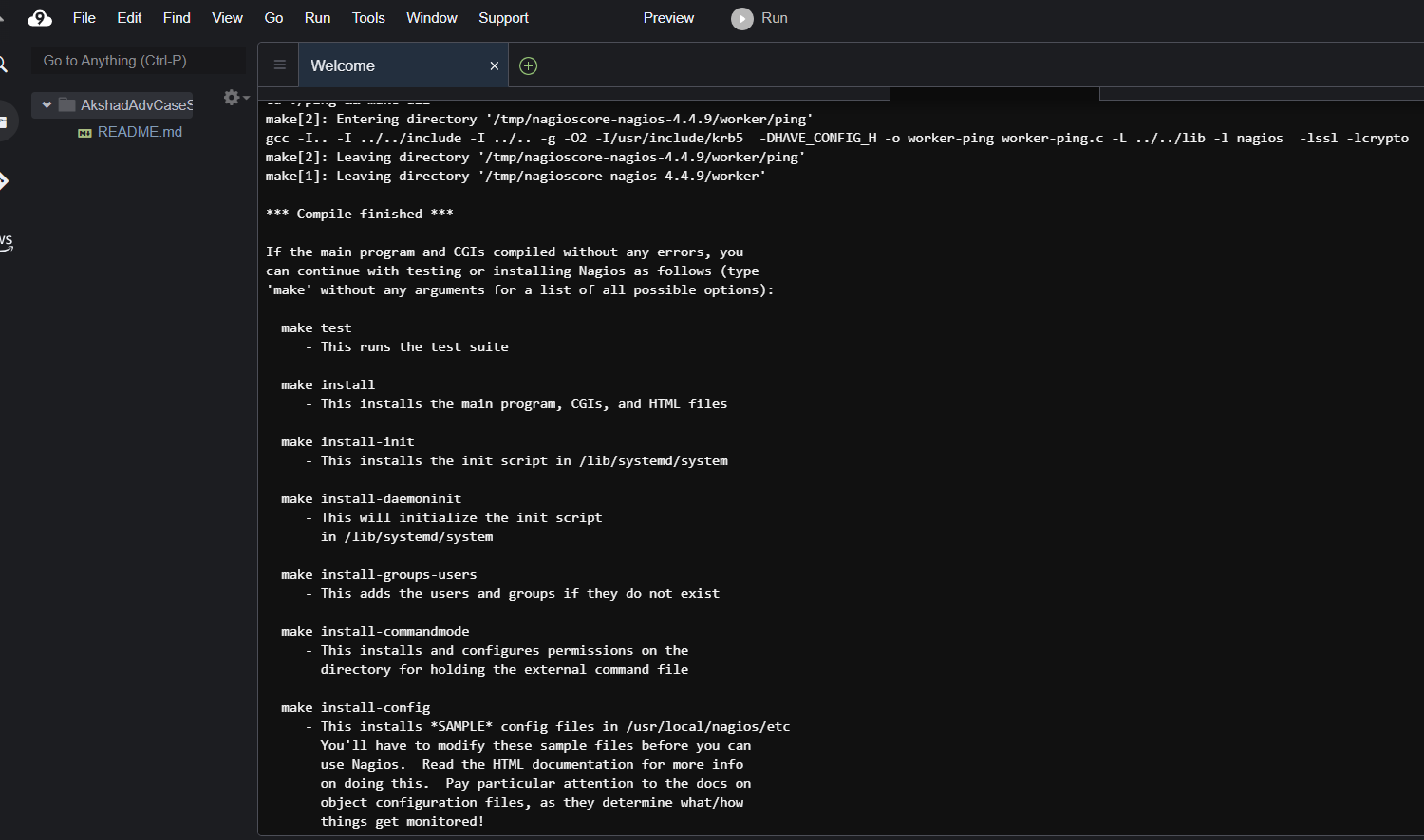
make all

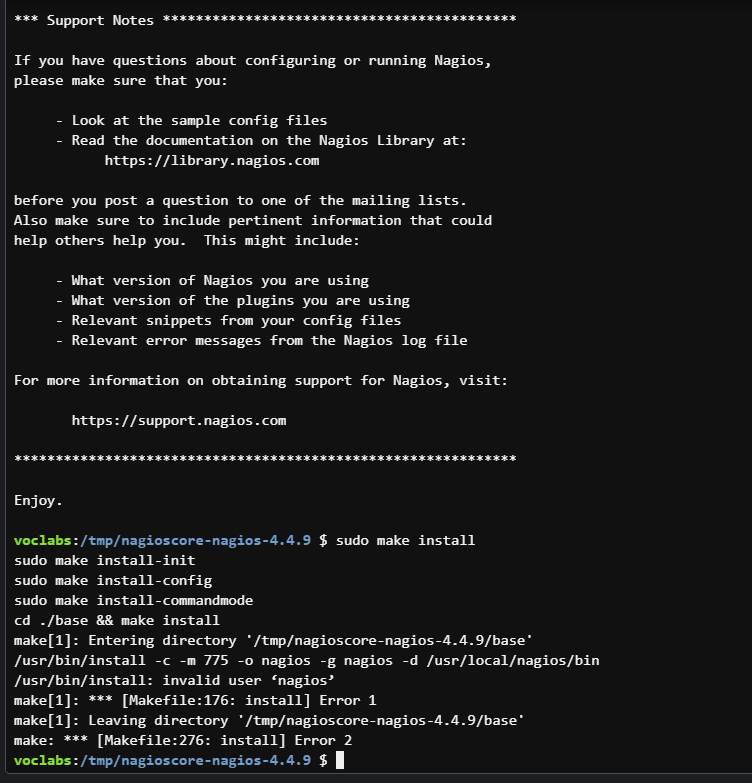
sudo make install

sudo make install-init

sudo make install-config

sudo make install-commandmode

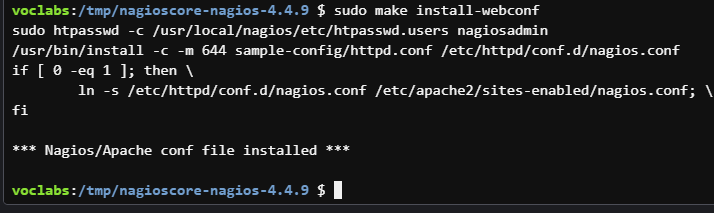




**Set Up Nagios Web Interface**:

sudo make install-webconf

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

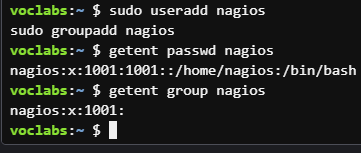


### **Creating Nagios User**

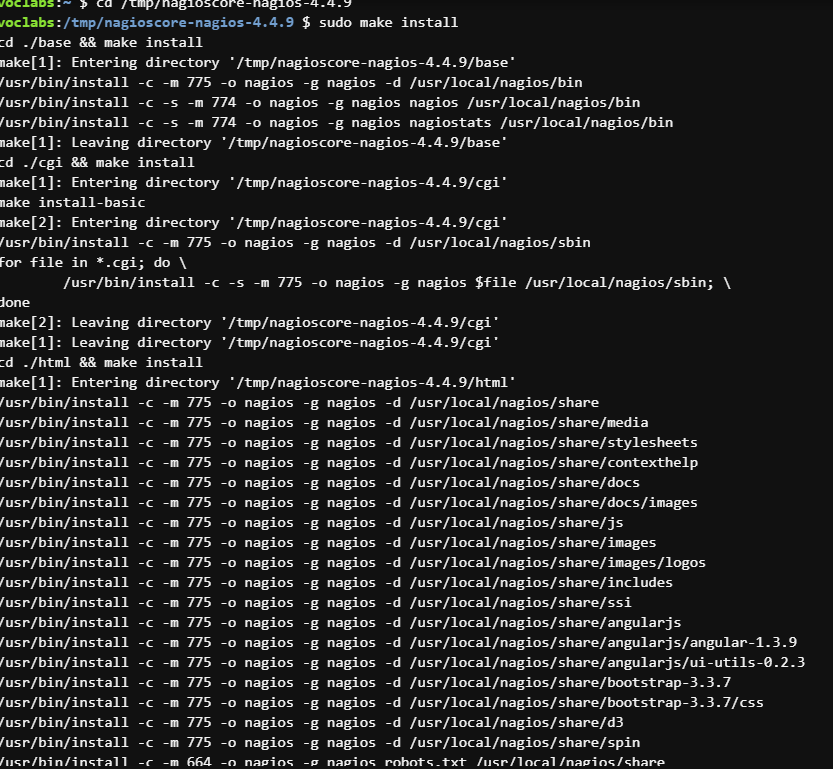
Run the following command to create the nagios user and group:

sudo useradd nagios

sudo groupadd nagios

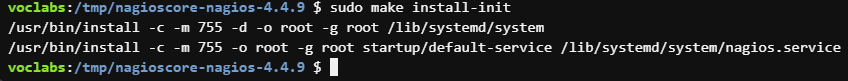


Rerun sudo make install all



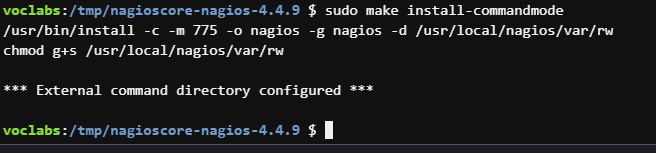
**Install Init Script**:

sudo make install-init



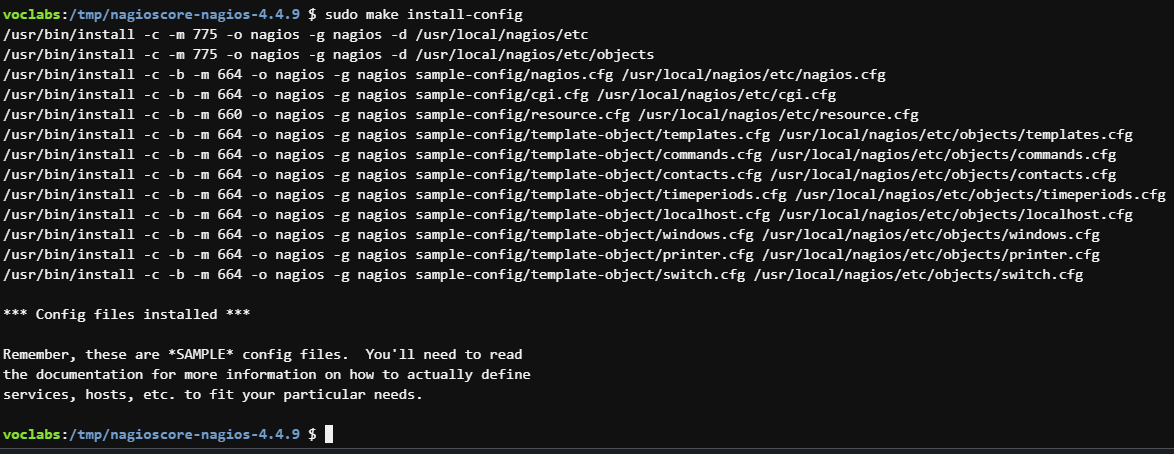
**Set Up Command Mode**:

sudo make install-commandmode



**Install Configuration Files**:

sudo make install-config



**Start Nagios**: After completing the above steps, start Nagios with:

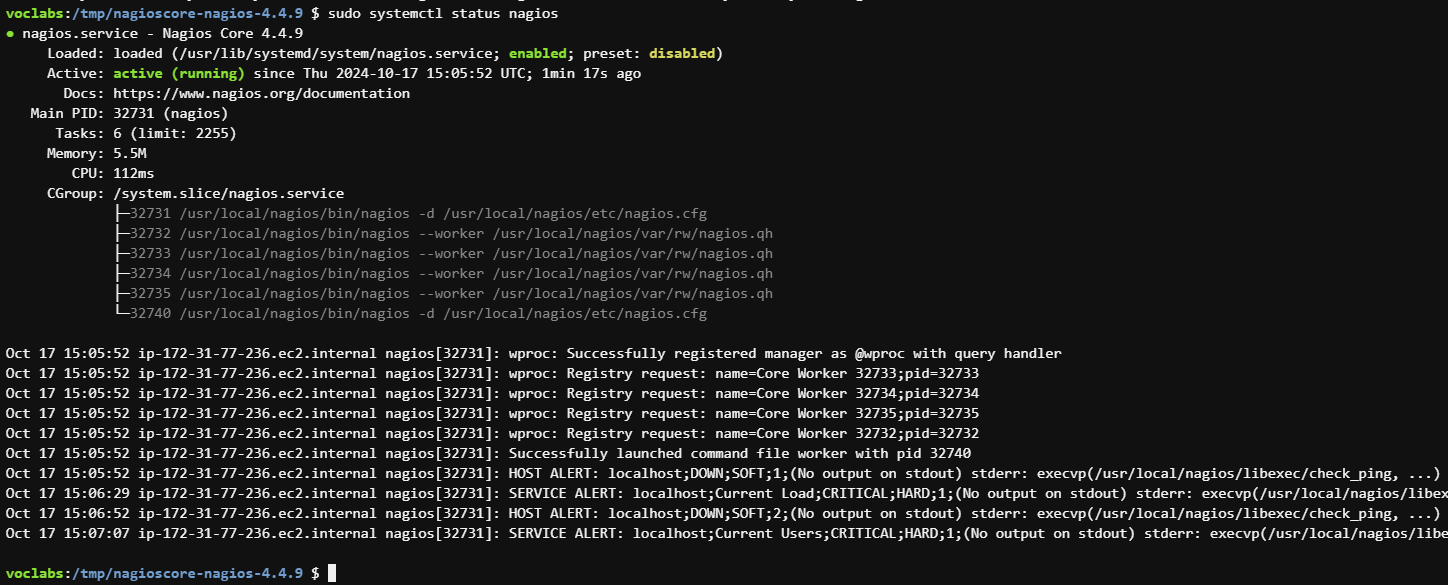
sudo systemctl start nagios

sudo systemctl enable nagios



**Check Status**: To confirm that Nagios is running, you can check its status with:

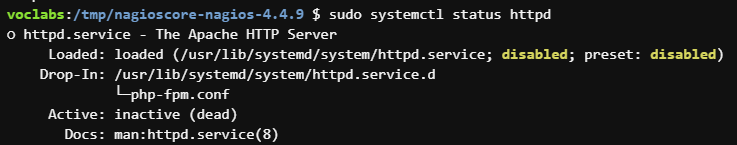
sudo systemctl status nagios



### **Installing Apache**

### **Check for Available Package Managers**

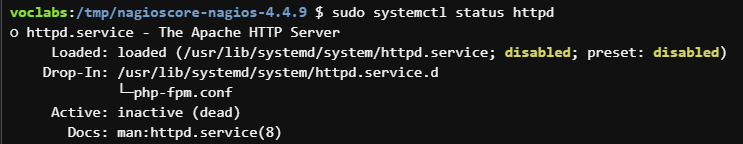
1. **Use yum (if on CentOS or RHEL-based systems)**: If you're on a Red Hat-based system, you might use yum instead:  
   sudo yum install httpd



### **Check Apache Status**

Run the following command to check if Apache is running:

sudo systemctl status httpd



### **Start Apache**

Run the following command to start the Apache server:

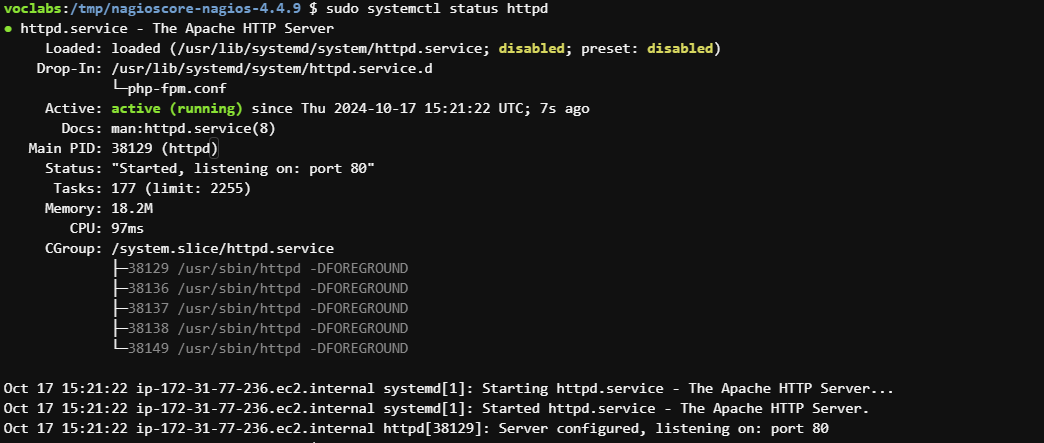
sudo systemctl start httpd



### **Check the Status Again**

After starting it, check the status again to ensure it's running:

sudo systemctl status httpd



### **Enable Apache to Start on Boot**

If it's running, enable it to start on boot with:

sudo systemctl enable httpd



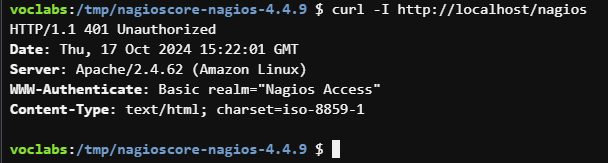
### 

### 

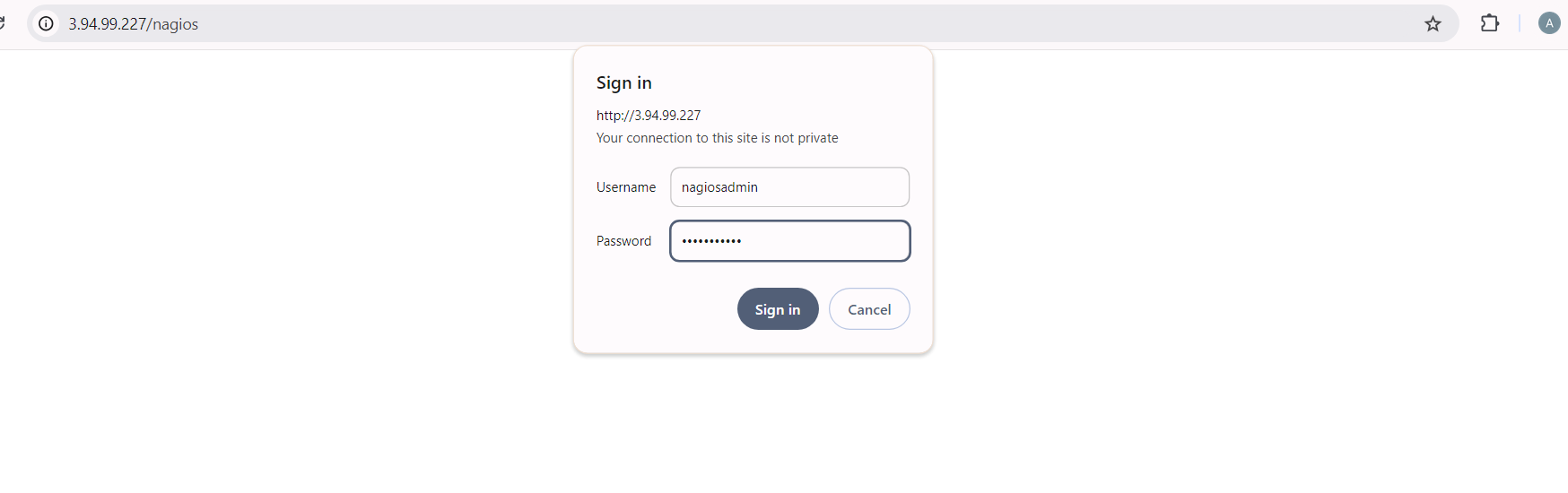
### **Access Nagios**

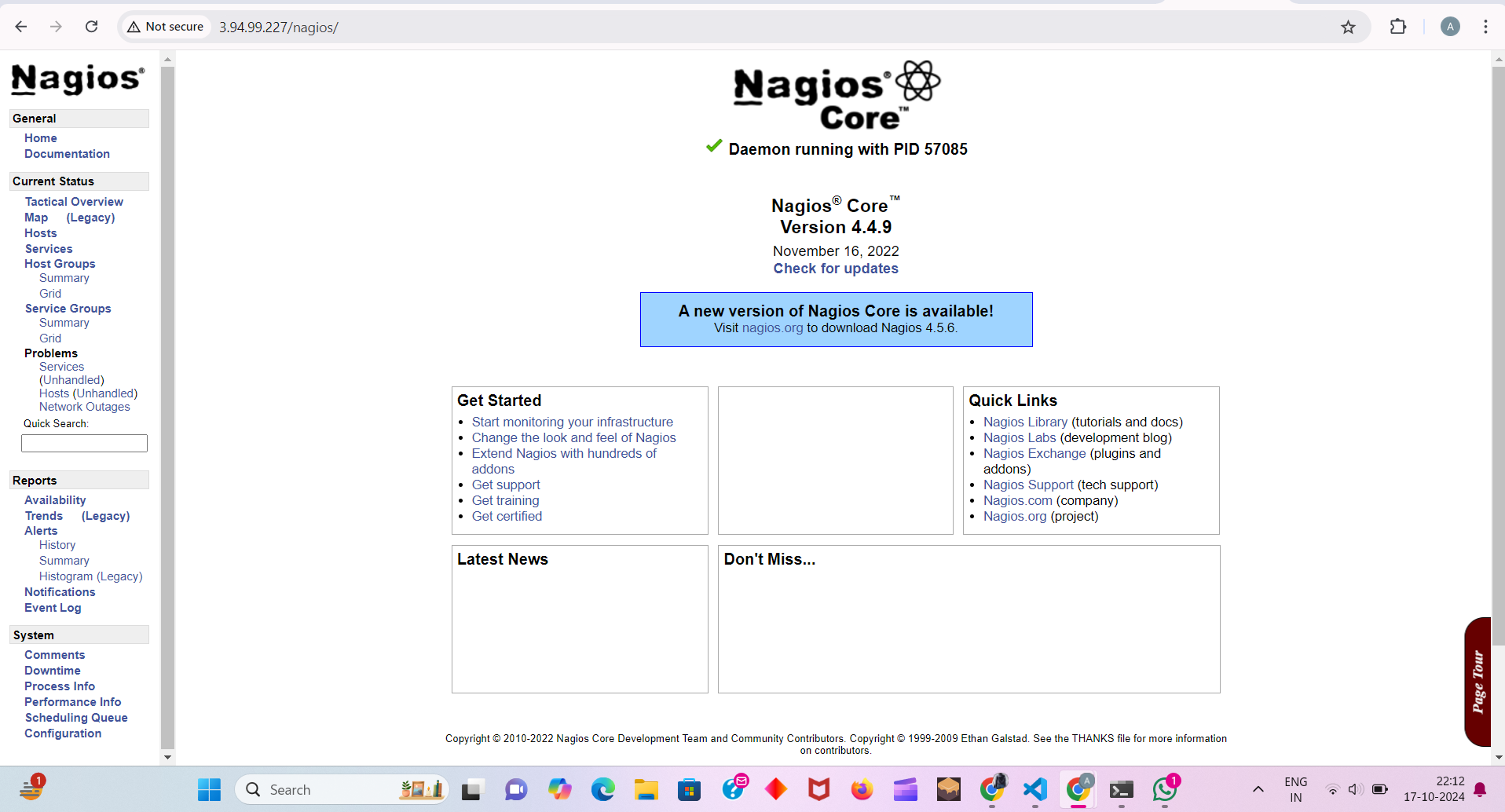
Now try accessing the Nagios web interface again:

curl -I http://localhost/nagios



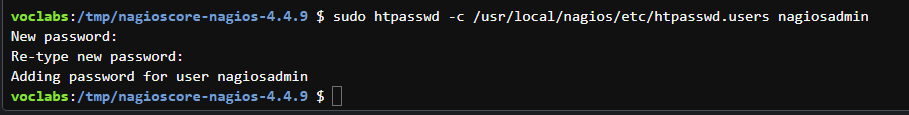
Check on browser http://3.94.99.227/nagios/





### **Login to Nagios**

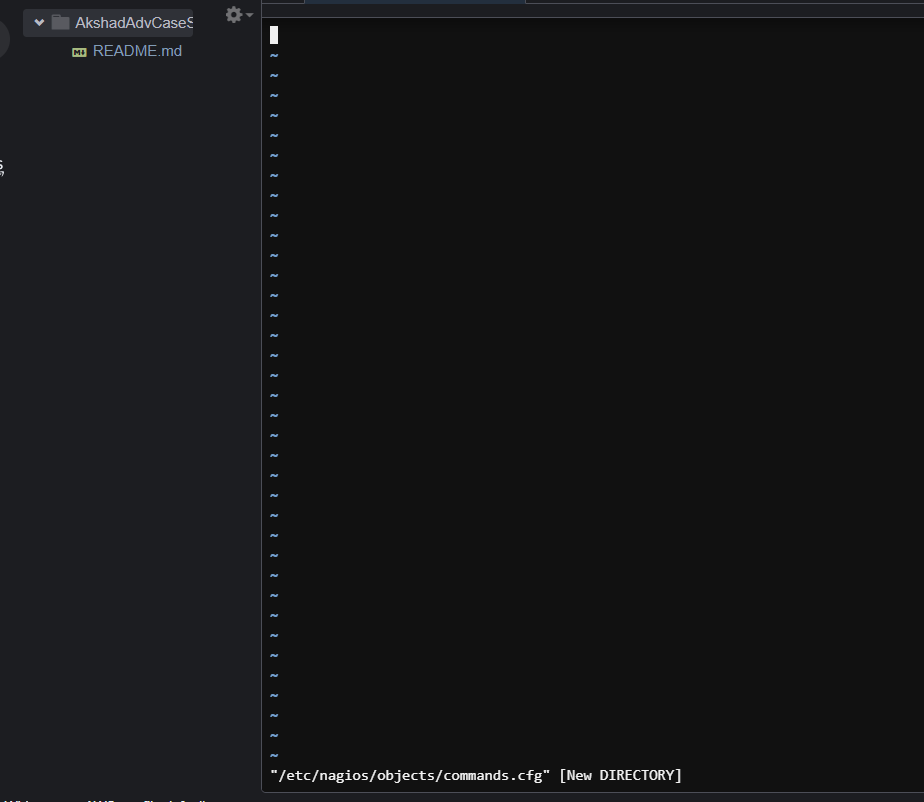
* Use the credentials you set up earlier. If you followed the previous steps, the username is likely nagiosadmin.
* You will be prompted for a password, which you set with:  
  bash  
  Copy code  
  sudo htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin



Password:Akshad.2004

### **Define a Nagios Command**

1. **Open the Commands Configuration File:**  
   sudo vim /etc/nagios/objects/commands.cfg



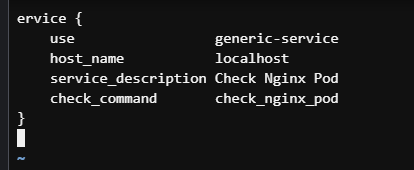
**Add the Command:** Add the following lines to define the command to check the Nginx pod's availability. You may want to ensure the command is added within the existing define command blocks or at the end of the file:

define command {

command\_name check\_nginx\_pod

command\_line /usr/local/bin/kubectl get pods | grep nginx | grep Running

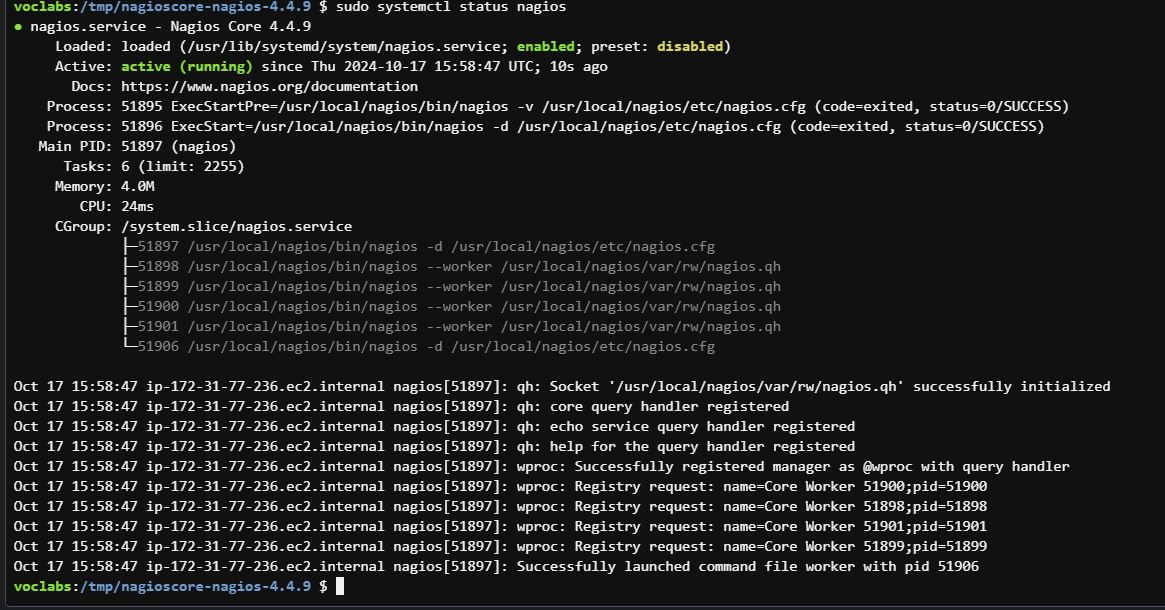
}



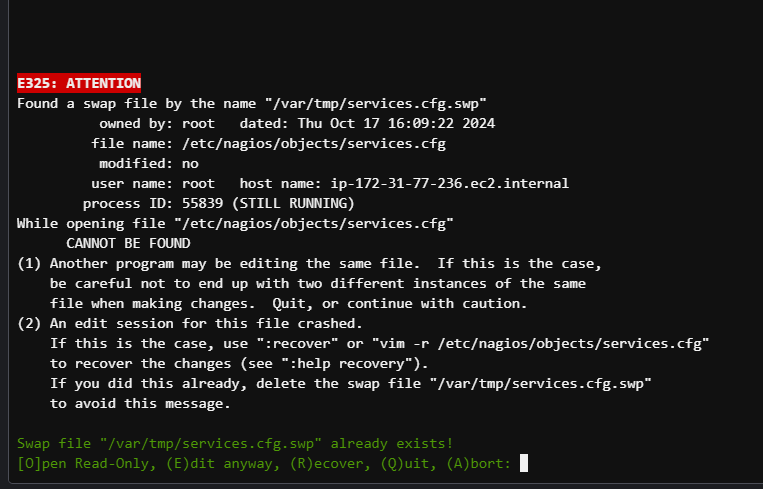
**Restart Nagios** to apply the changes:  
sudo systemctl restart nagios



**Check the status of Nagios** to confirm it's running:  
sudo systemctl status nagios



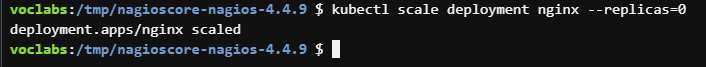
Changes:(not happening)



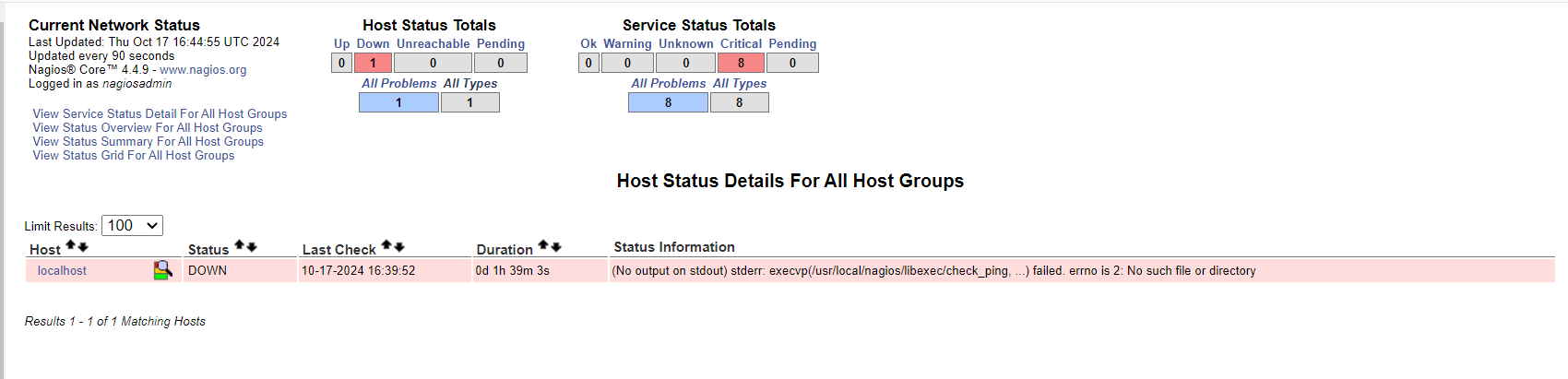
### **Verify That Nagios Can Detect When the Nginx Pod is Running or Not**

#### **1. Simulate Failure (Stop Nginx Pod):**

1. **Scale Down the Nginx Deployment** to simulate a failure:  
   kubectl scale deployment nginx --replicas=0



**Check Nagios**: Open the Nagios web interface. You should see an alert indicating that the Nginx pod is down.



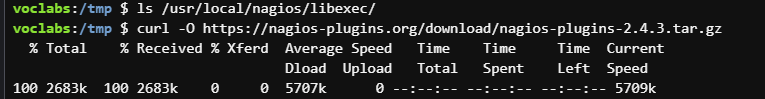
**To Change the status to UP**

#### **Install Nagios Plugins**

If the plugins are missing, follow these steps to install them:

**Download Nagios Plugins**:  
cd /tmp

1. curl -O https://nagios-plugins.org/download/nagios-plugins-2.4.3.tar.gz



**Extract the Plugins**:

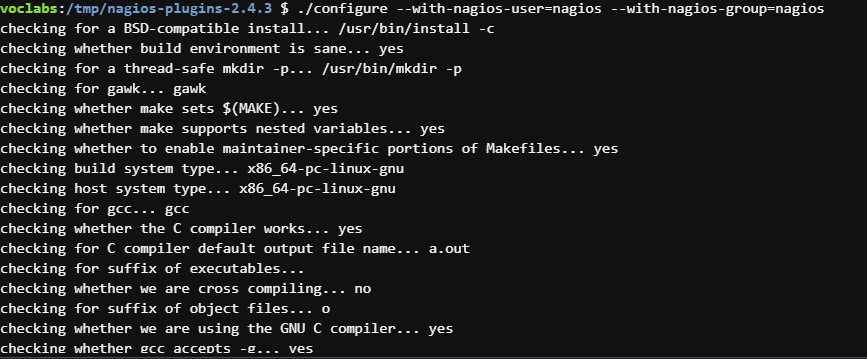
tar -xzf nagios-plugins-2.4.3.tar.gz

cd nagios-plugins-2.4.3

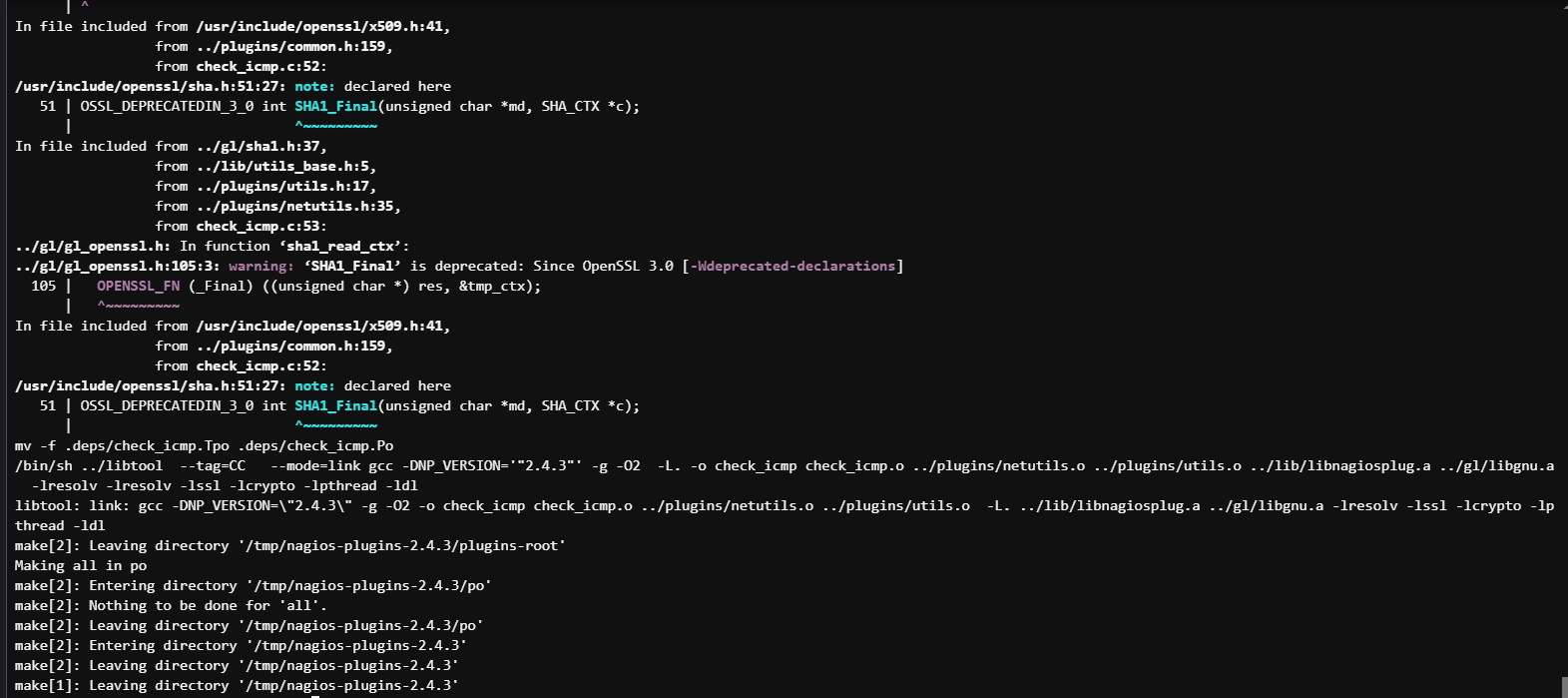


**Compile and Install the Plugins**:

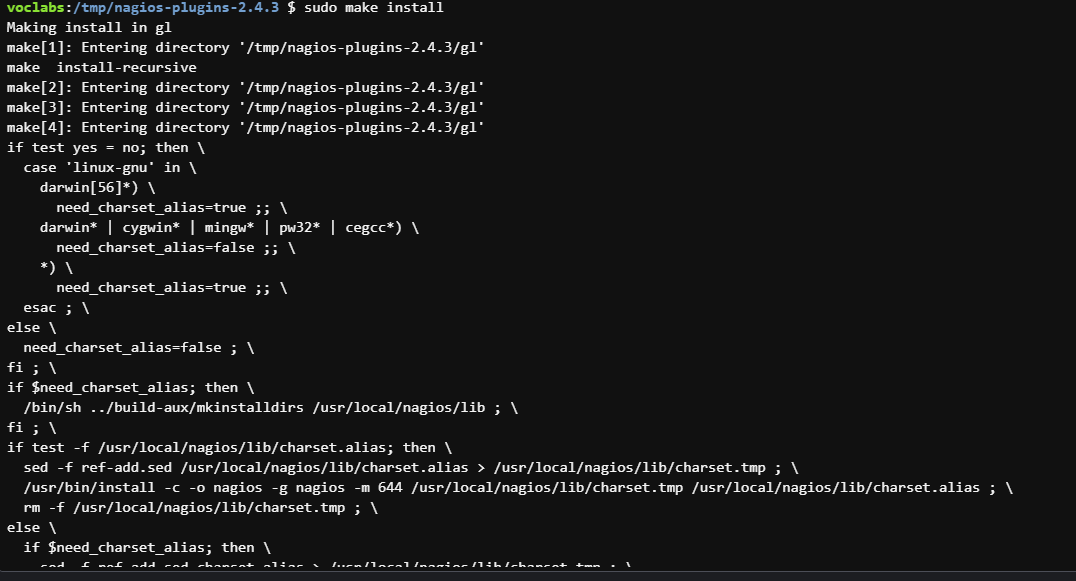
./configure --with-nagios-user=nagios --with-nagios-group=nagios



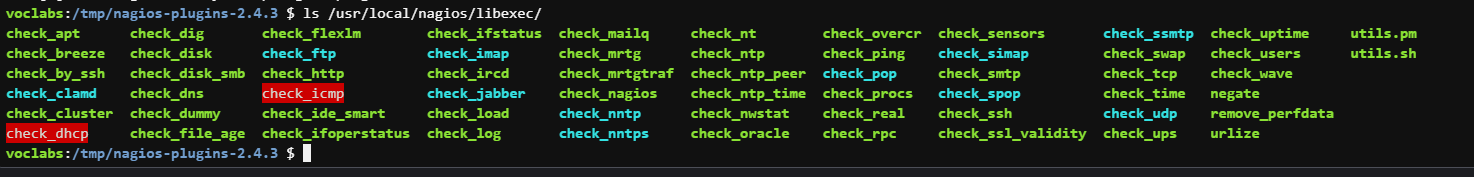
Make



sudo make install



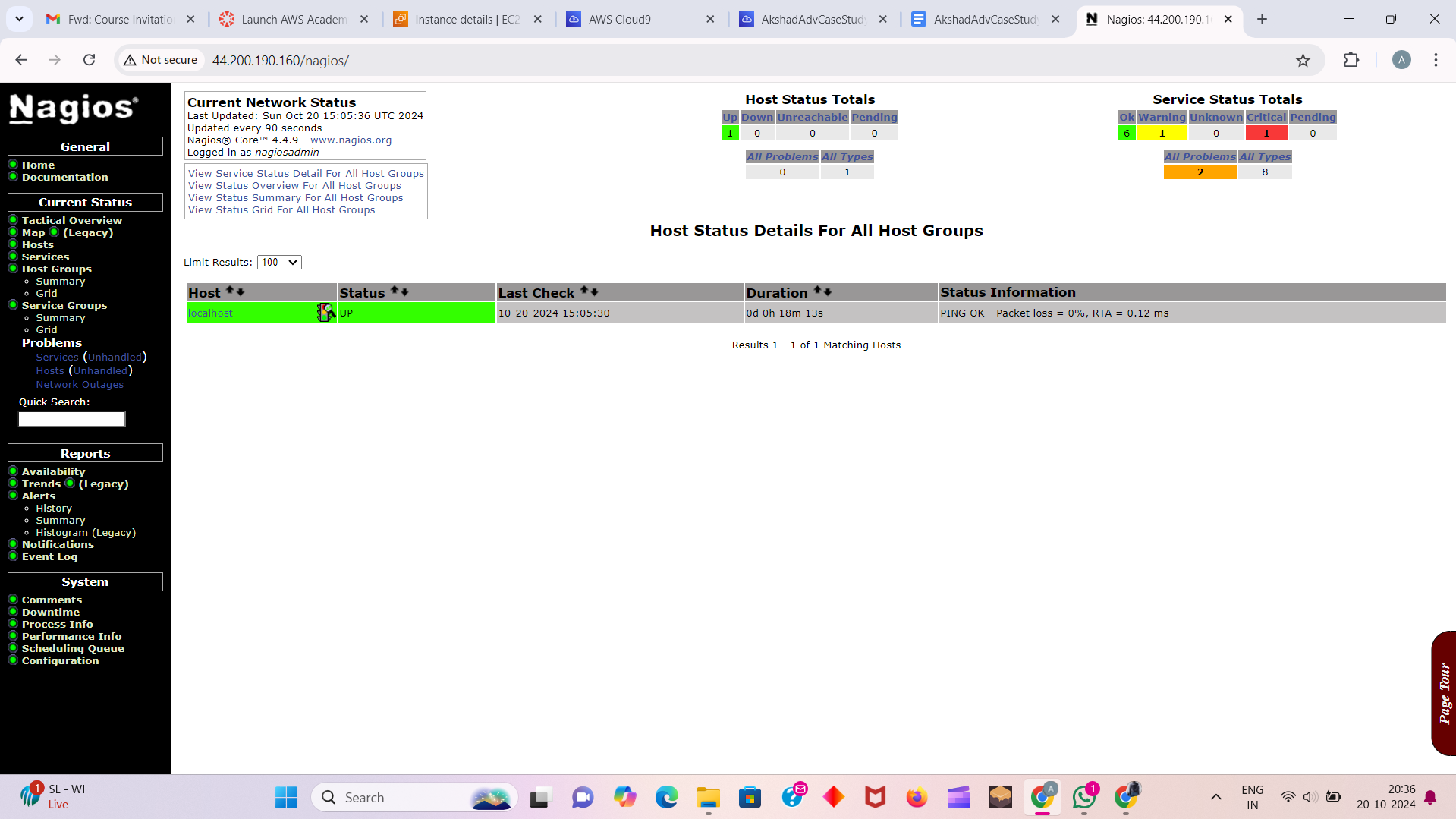
ls /usr/local/nagios/libexec/

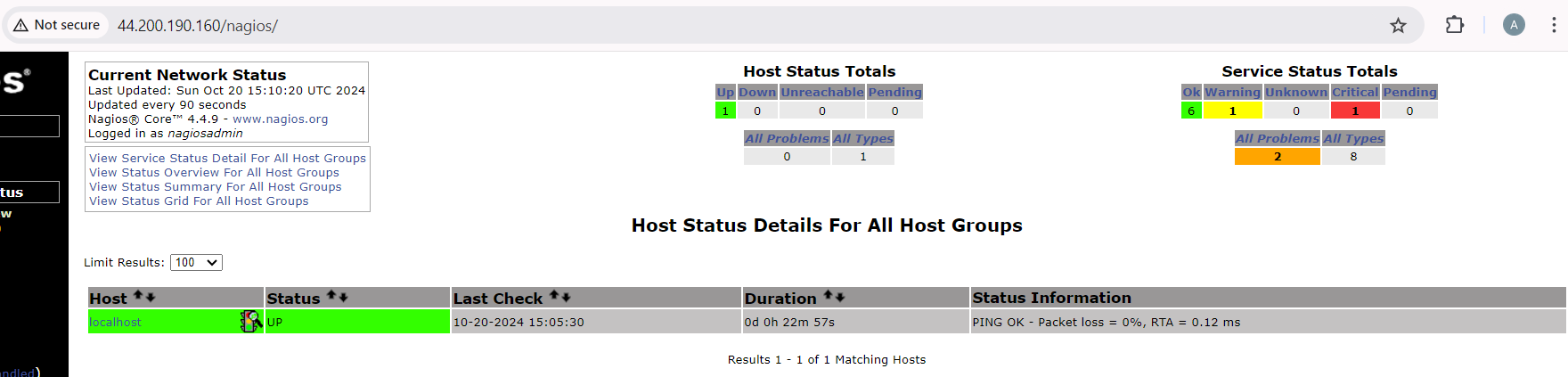


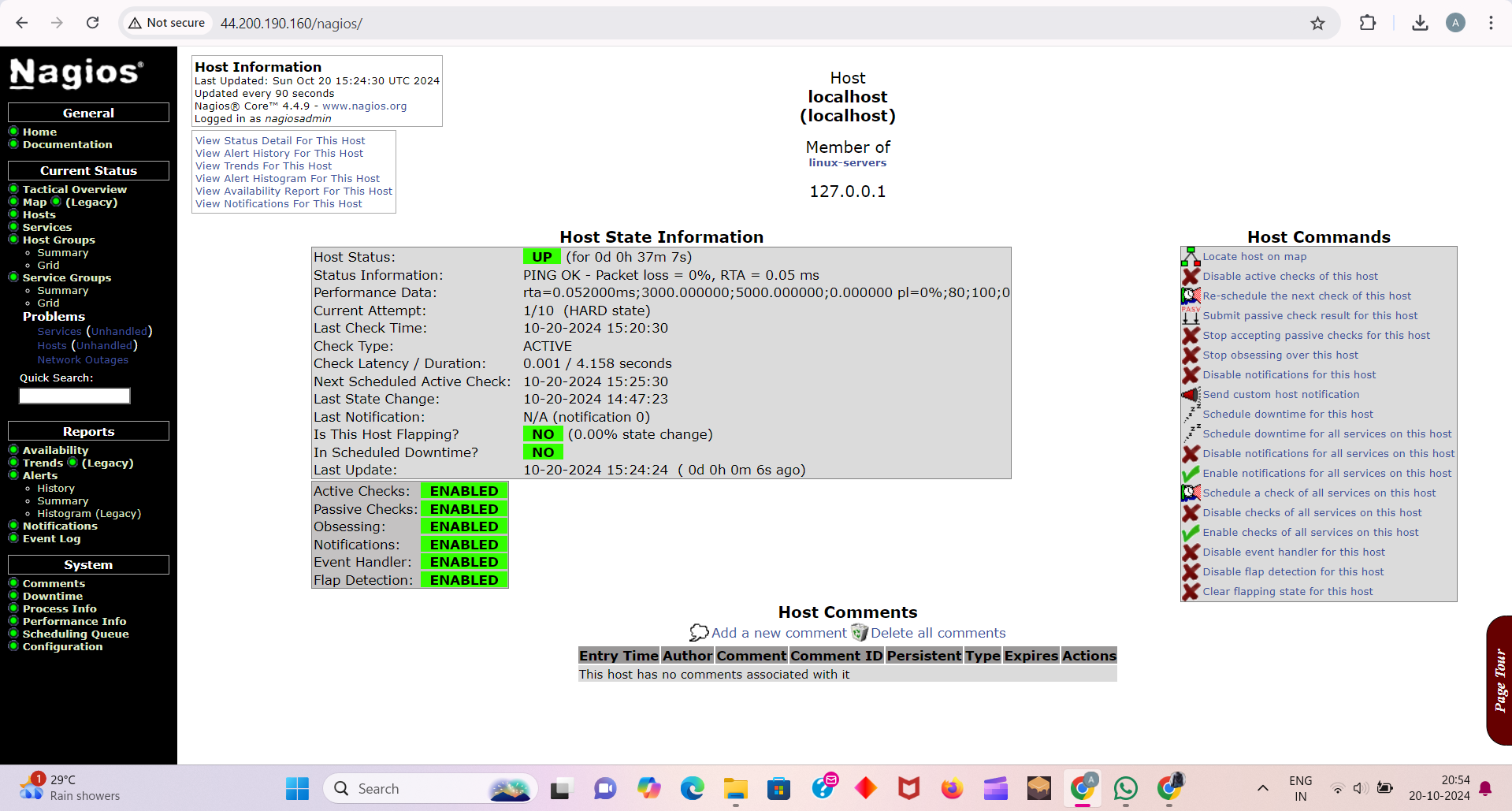
sudo systemctl restart nagios



**Now Status is UP**







**Conclusion:**

In conclusion, deploying an Nginx server on a Kubernetes cluster using AWS Cloud9 allows for a scalable and manageable application environment. Integrating Nagios provides robust monitoring capabilities, enabling real-time tracking of application health and performance. This setup is beneficial for both learning and practical implementations in DevOps workflows. Overall, these tasks illustrate the power of Kubernetes and monitoring tools in maintaining application availability and reliability.