Name:Advik Hegde Div:D15C Roll No.:15

1. Introduction

● Case Study Overview:

This case study focuses on deploying an Nginx server on a Kubernetes cluster using EC2 instances and monitoring its health with Nagios. The project aims to demonstrate basic monitoring of an application in a Kubernetes environment.

● Key Feature and Application:

The integration of Nagios with Kubernetes allows real-time monitoring of the Nginx pod, ensuring high availability. It provides alerts when the Nginx server experiences downtime or issues, enabling quick remediation.

Step by Step explanation

1.Firstly create a new IAM user and provide to that user both programmatic access and console access,after creation attach the necessary policies

Attach following AWS Managed Policies:

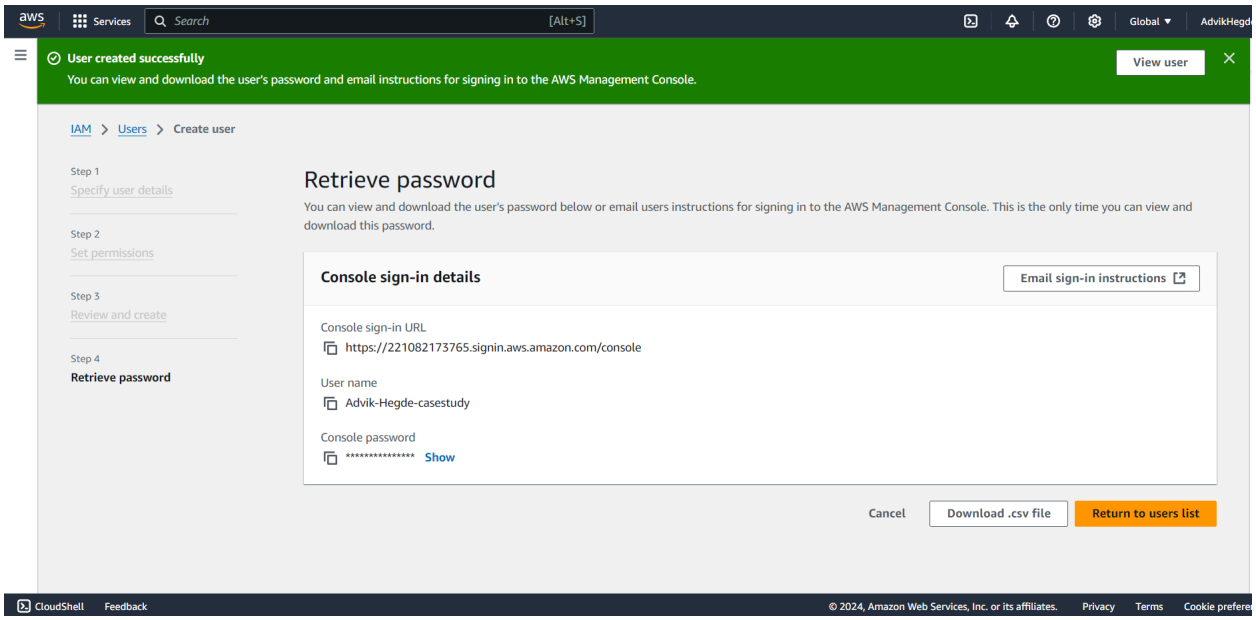
AmazonEKSClusterPolicy,AmazonEKSWorkerNodePolicy,AmazonEC2ContainerRegistryReadOnly,AmazonEC2FullAccess,IAMFullAccess.

These policies seemed insufficient as they were later giving permission based errors for the creation of the kubernetes cluster so I added a new user created policy with the following permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "eks:*",
                "ec2:*",
                "iam:*",
                "cloudformation:*",
                "ecr:*"
```

```
                ],
                "Resource": "*"
            }
        ]
}
```

Advik-Hegde-casestudy is the name of my IAM user



After successfully creating the IAM user,open your cmd in default directory.No need for admin rights for the below commands :

Enter the command aws configure,it will ask you for a series of details,for those details follow the following steps

Go to IAM (Identity and Access Management) from the services menu.

Select Users from the IAM dashboard and in the **User details** section, click on the **Security credentials** tab.

Scroll down to the **Access keys** section and click on **Create access key**.

Download or copy the access key ID and secret access key, as they will not be shown again. These credentials will be required in the next step.

Enter the command aws configure and proceed as shown below.

For the region,I went with what was recommended;if nothing is recommended enter the region where the instance is created and for the output format enter json.

After this open cmd again but with administrator access and enter the following commands :

powershell -Command "Set-ExecutionPolicy Bypass -Scope Process -Force"

powershell -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))"

choco install eksctl

eksctl version

If the version is being displayed the installation process has completed successfully.

We are using eksctl because **eksctl** is a command-line tool specifically designed for creating, managing, and maintaining **Amazon EKS (Elastic Kubernetes Service)** clusters.

It simplifies operations like creating Kubernetes clusters, managing node groups, and performing various cluster management tasks.

```
C:\Windows\system32>powershell -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))"
WARNING: 'choco' was found at 'C:\ProgramData\chocolatey\bin\choco.exe'.
WARNING: An existing Chocolatey installation was detected. Installation will not continue. This script will not
overwrite existing installations.
If there is no Chocolatey installation at 'C:\ProgramData\chocolatey', delete the folder and attempt the installation
again.

Please use choco upgrade chocolatey to handle upgrades of Chocolatey itself.
If the existing installation is not functional or a prior installation did not complete, follow these steps:
 - Backup the files at the path listed above so you can restore your previous installation if needed.
 - Remove the existing installation manually.
 - Rerun this installation script.
 - Reinstall any packages previously installed, if needed (refer to the lib folder in the backup).

Once installation is completed, the backup folder is no longer needed and can be deleted.

C:\Windows\system32>choco install eksctl
Chocolatey v2.3.0
Installing the following packages:
eksctl
By installing, you accept licenses for the packages.
Downloading package from source 'https://community.chocolatey.org/api/v2/'
Progress: Downloading eksctl 0.193.0... 100%

eksctl v0.193.0 [Approved]
eksctl package files install completed. Performing other installation steps.
The package eksctl wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N]o/[P]rint): A

eksctl is going to be installed in 'C:\ProgramData\chocolatey\lib\eksctl\tools'
Downloading eksctl 64 bit
  from 'https://github.com/eksctl-io/eksctl/releases/download/v0.193.0/eksctl_Windows_amd64.zip'
Progress: 100% - Completed download of C:\Users\ADMIN\AppData\Local\Temp\chocolatey\eksctl\0.193.0\eksctl_Windows_amd64.zip (34.96 MB).
Download of eksctl_Windows_amd64.zip (34.96 MB) completed.
Hashes match.
Extracting C:\Users\ADMIN\AppData\Local\Temp\chocolatey\eksctl\0.193.0\eksctl_Windows_amd64.zip to C:\ProgramData\chocolatey\lib\eksctl\tools...
C:\ProgramData\chocolatey\lib\eksctl\tools
 ShimGen has successfully created a shim for eksctl.exe
 The install of eksctl was successful.
  Deployed to 'C:\ProgramData\chocolatey\lib\eksctl\tools'

Chocolatey installed 1/1 packages.
 See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

C:\Windows\system32>eksctl version
0.193.0
```

After this proceed with the steps of creation of the cluster by entering the command

eksctl create cluster --name Advik-new-EKS-cluster-casestudy --region us-west-2 --nodegroup-name standard-nodes --node-type t3.micro --nodes 2 --nodes-min 1 --nodes-max 3 --managed

Be patient as this step can take some time,around 10-15 minutes for me.

```
2024-10-20 17:25:48 [ℹ]  all EKS cluster resources for "Adv-new-EKS-cluster-casestudy" have been created
2024-10-20 17:25:48 [ℹ]  created 0 nodegroup(s) in cluster "Adv-new-EKS-cluster-casestudy"
2024-10-20 17:25:50 [ℹ]  nodegroup "standard-nodes" has 2 node(s)
2024-10-20 17:25:50 [ℹ]  node "ip-192-168-11-172.us-west-2.compute.internal" is ready
2024-10-20 17:25:50 [ℹ]  node "ip-192-168-52-91.us-west-2.compute.internal" is ready
2024-10-20 17:25:50 [ℹ]  waiting for at least 1 node(s) to become ready in "standard-nodes"
2024-10-20 17:25:50 [ℹ]  nodegroup "standard-nodes" has 2 node(s)
2024-10-20 17:25:50 [ℹ]  node "ip-192-168-11-172.us-west-2.compute.internal" is ready
2024-10-20 17:25:50 [ℹ]  node "ip-192-168-52-91.us-west-2.compute.internal" is ready
2024-10-20 17:25:50 [ℹ]  created 1 managed nodegroup(s) in cluster "Adv-new-EKS-cluster-casestudy"
2024-10-20 17:25:54 [ℹ]  kubectl command should work with "C:\\Users\\ADMIN\\.kube\\config", try 'kubectl get nodes'
2024-10-20 17:25:54 [ℹ]  EKS cluster "Adv-new-EKS-cluster-casestudy" in "us-west-2" region is ready

C:\Windows\system32>
```

After successful creation of your kubernetes cluster , run the command of kubectl get nodes to view you nodes.

```
C:\Windows\system32>kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
ip-192-168-11-172.us-west-2.compute.internal   Ready    <none>   4m55s   v1.30.4-eks-a737599
ip-192-168-52-91.us-west-2.compute.internal    Ready    <none>   5m      v1.30.4-eks-a737599

C:\Windows\system32>
```

After that create your nginx deployment on this cluster by using th command

kubectl create deployment nginx --image=nginx

Expose this deployment using

kubectl expose deployment nginx --port=80 --type=LoadBalancer

```
C:\Windows\system32>kubectl create deployment nginx --image=nginx
deployment.apps/nginx created

C:\Windows\system32>kubectl expose deployment nginx --port=80 --type=LoadBalancer
service/nginx exposed

C:\Windows\system32>kubectl get services
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)        AGE
kubernetes   ClusterIP      10.100.0.1       <none>                                                                    443/TCP        15m
nginx        LoadBalancer   10.100.171.171   a2c6af33e08164418b81bf0752786c80-122325315.us-west-2.elb.amazonaws.com   80:31884/TCP   9s

C:\Windows\system32>
```

kubectl get services

kubectl get pods

```
C:\Windows\system32>kubectl get services
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP                                                               PORT(S)        AGE
kubernetes   ClusterIP      10.100.0.1       <none>                                                                    443/TCP        15m
nginx        LoadBalancer   10.100.171.171   a2c6af33e08164418b81bf0752786c80-122325315.us-west-2.elb.amazonaws.com   80:31884/TCP   9s

C:\Windows\system32>kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
nginx-bf5d5cf98-fdnxx    1/1     Running   0          77m
```

Finally use this command to view the status of tour nginx pod

- kubectl get pod nginx-bf5d5cf98-fdnxx -o wide

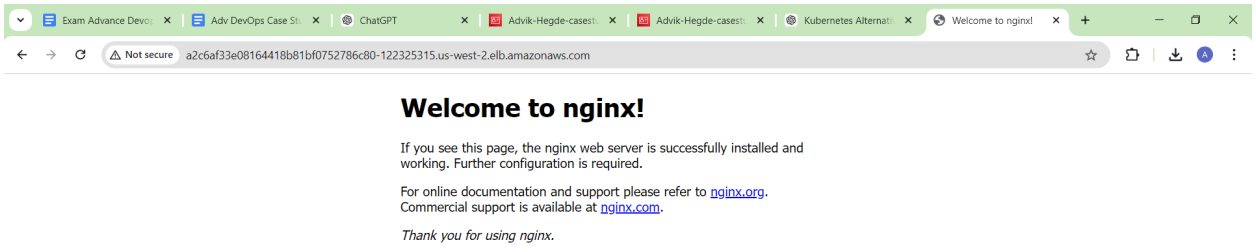
Replace the name with the name of your nginx pod you would have obtained in the previous step

Verify the service configurations after using this command

- kubectl describe service nginx
- http://<EXTERNAL-IP> [Enter the external ip here]

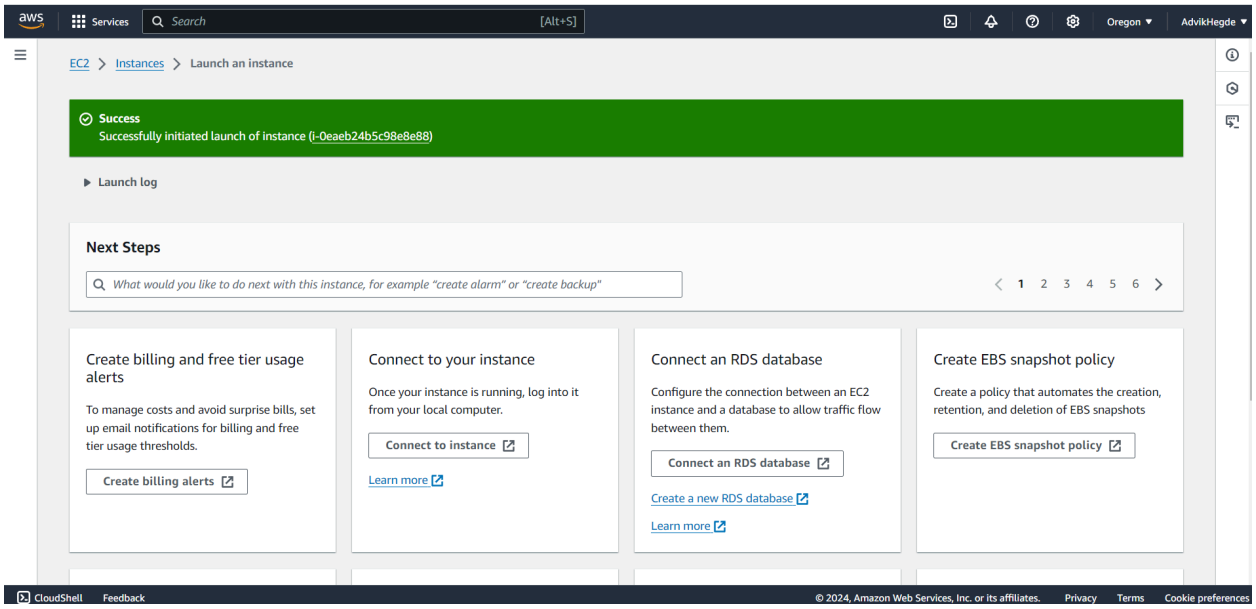
Access the web browser using the external ip of the nginx server that we got in the previous step of kubectl get services.

You should see a page like this



If you can see that page, it means that your Nginx server is up and running successfully on Kubernetes.

Now the next step is creating a new instance for Setting up nagios,connecting it to the nginx server and monitoring the health of the server using nagios.



Execute the following list of commands to ensure correct configuration of all nagios related files.

/usr/local/nagios/etc/objects/your-nginx-config.cfg.

cd /usr/local/nagios/etc/objects/

ls

sudo nano nginx.cfg

Example host definition:

```
define host {
    use                     linux-server      ; Inherit default values
    host_name               nginx-server      ; The name of the host
    alias           Nginx Server      ; A short description of the host
    address       <nginx-server-ip>  ; IP address of the Nginx server
    check_command    check-host-alive   ; Command to check host status
}
```

```
define service {
    use                 generic-service     ; Inherit default values
    host_name           nginx-server        ; The name of the host
    service_description HTTP                ; Service description
    check_command       check_http      ; Command to check HTTP service
}
```

Test the Configuration: After editing the configuration file,to test the Nagios configuration,Run the following command:

sudo nagios -v /usr/local/nagios/etc/nagios.cfg

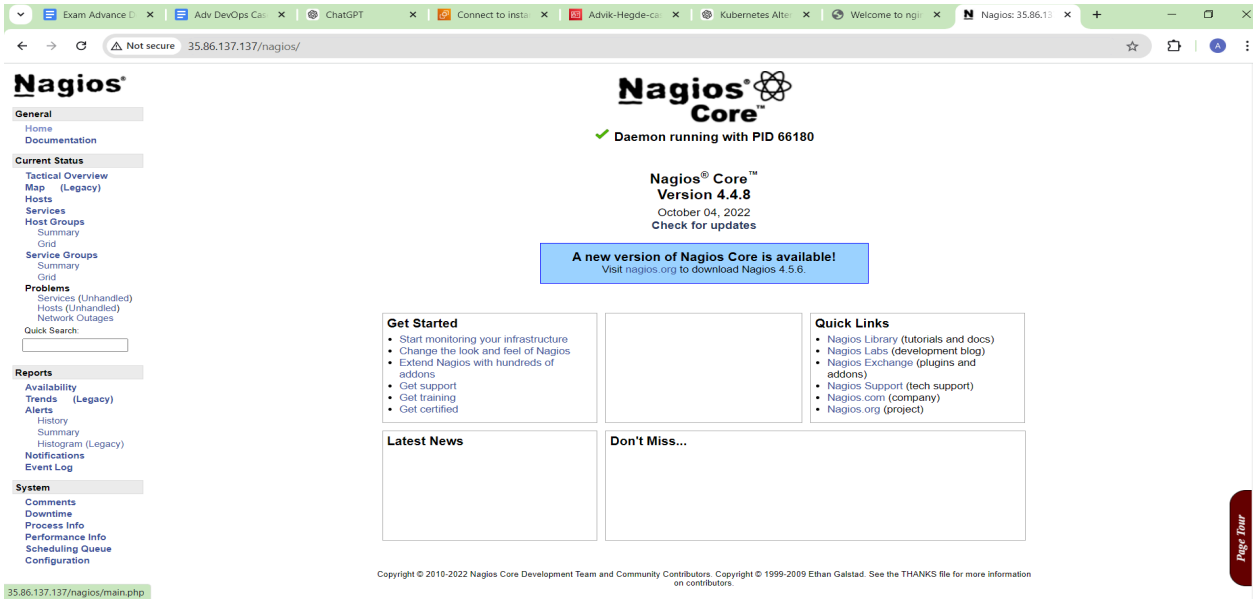
Test the Nagios check command manually to see if it returns the expected result:

/usr/local/nagios/libexec/check_ping -H <nginx-server-ip> -w 100,20% -c 500,60%

sudo systemctl restart nagios

Access the Nagios home page using the public-ip address of the instance where nagios was configured

The address for me to access nagios is **//http://35.86.137.137//nagios**



To get the nginx server to be monitored by nagios and display the nginx server as UP in the hosts section of nagios,we will need to set some settings for the load balancer that we are using.

Navigate to the health checks section if the ELB is associated with the nginx service.

You can do this by running a command to see if the Nginx pod is responding:

- kubectl exec -it nginx-bf5d5cf98-fdnxx -- curl -I http://localhost/

This command should return an HTTP 200 status if Nginx is configured correctly.

If your Nagios server is running at http://35.86.137.137/nagios, you should configure the inbound rules for the security group associated with your load balancer (or the EC2 instance running Nginx) to allow traffic from the source IP address 35.86.137.137 of Nagios server.

When adding an inbound rule for the security group, it would look something like this:

Type: HTTP , Protocol: TCP , Port: 80 , Source: 35.86.137.137/32 (This allows traffic only from the specific IP address of your Nagios server)

Adding Rule for Nagios Server

Click on Add rule again.Type: HTTP , Protocol: TCP , Port Range: 80 , Source: Choose Custom and enter the public IP address of your Nagios server (e.g., 35.86.137.137).

You can also use a CIDR notation to allow a range of IP addresses, like 35.86.137.0/24 if needed.You can also use the list of commands below to troubleshoot the problem of the server being down.

Check the Nagios logs for any errors or warnings that may provide insights into why the service is reported as down. Logs are typically found in

- /usr/local/nagios/var/nagios.log.

Test the Nagios check plugin manually to see if it can access the Nginx server. For example:

- /usr/local/nagios/libexec/check_http -H <external-ip>

This should return the HTTP response code. If it returns an error, that indicates a problem with the check configuration or connectivity.

- sudo systemctl restart nagios

Verify the Nagios configuration for the Nginx host to ensure it has the correct IP address and settings.

Conclusion :

In this experiment , we created 2 EC2 instances, 1 to create a kubernetes cluster to then deploy the Nginx server and another EC2 instance to install and configure Nagios to monitor the Nginx server.To ensure correct execution of these steps,we need to ensure that firstly, we access the nginx server using the correct external ip of the server and secondly , correctly configuring the nagios files and accessing the nagios home page using the public ip of the instance where Nagios was installed.

To be able to properly monitor the nginx server in the hosts section of the Nagios page , there may be a need to delete the kubernetes cluster and create it again as that can reset the server and change its HTTP status to OK which would earlier be in a critical state.

There may even be need to perform troubleshooting operations to ensure that the server status itself is displayed as UP.This would require to correctly configure inbound rules of the security group associated with the ELB used by the instance as well as the security group associate with the nagios server as well.

By the end of the case study , we were able to successfully create the 2 instances,create the nginx server and configure nagios to monitor the created server and then view the HTTP status of the server to have a status of OK.