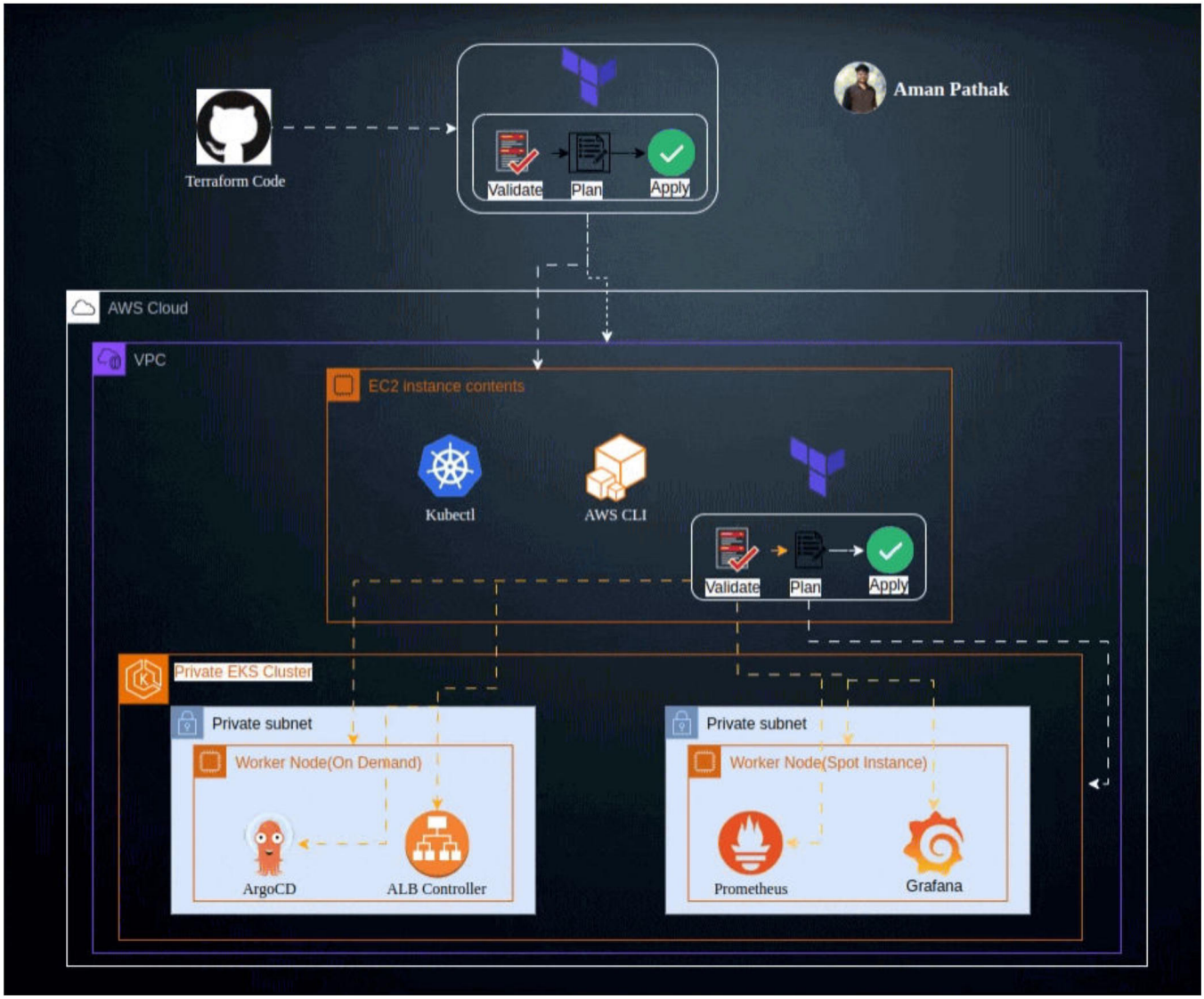
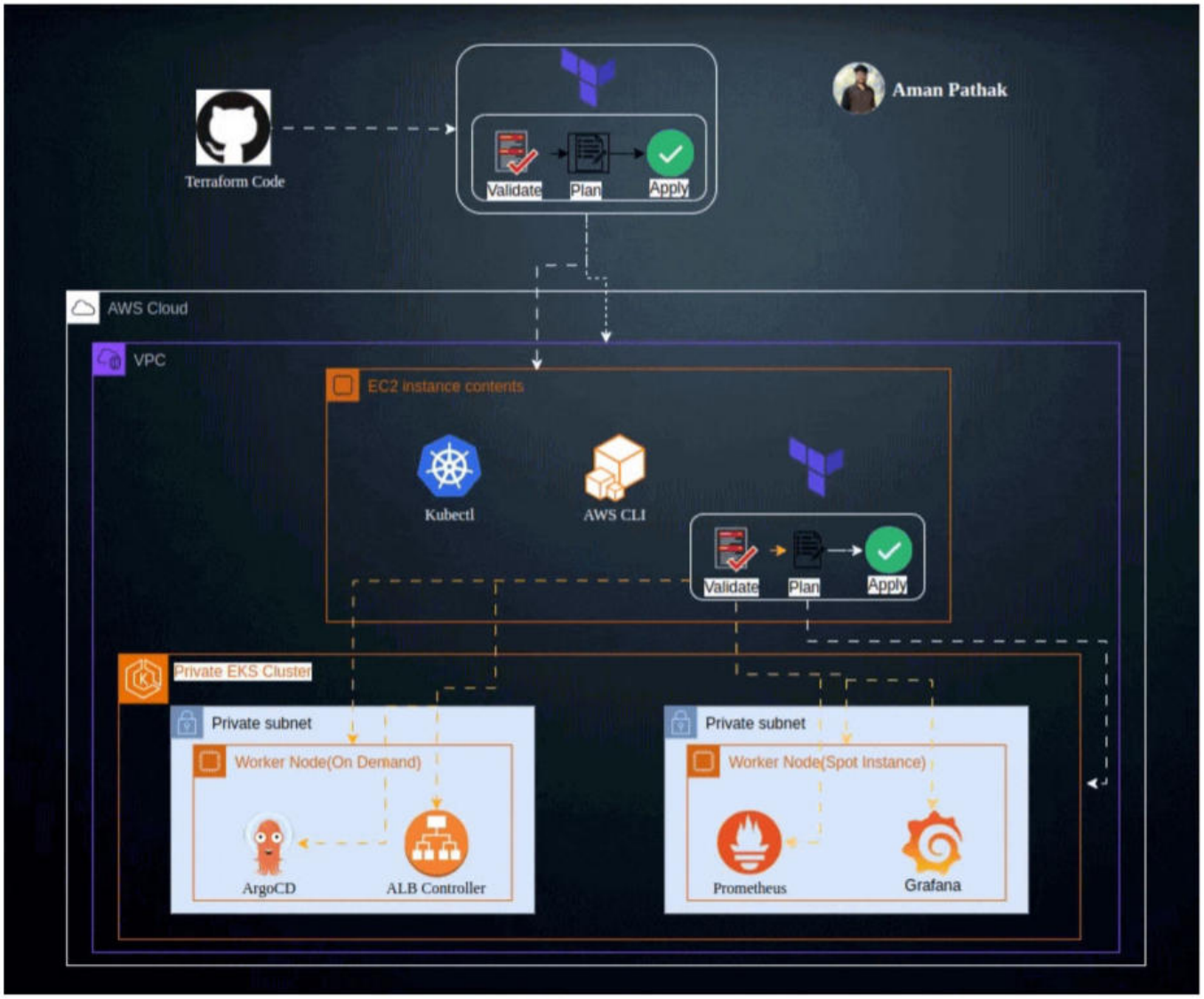


Configure ArgoCD, Prometheus, Grafana & AWS Load Balancer Controller on EKS Cluster using Terraform

Aman Pathak · Follow

Published in Towards Dev
9 min read · Sep 2, 2024

Listen Share



Introduction

In today's DevOps-driven world, automating infrastructure deployment is crucial for maintaining efficiency and scalability. Setting up a secure and robust EKS (Elastic Kubernetes Service) cluster, complete with essential tools like ArgoCD, Prometheus, and Grafana, requires careful planning and execution. This guide will walk you through the entire process, from configuring your environment to deploying your infrastructure using Terraform, ensuring that your private EKS cluster is up and running smoothly with all the necessary resources.

Why Use Terraform for Configuring ArgoCD, Prometheus, and Other Tools?

Configuring tools like ArgoCD, Prometheus, and Grafana through Terraform offers several key advantages:

1. **Infrastructure as Code (IaC)**: Terraform allows you to define your entire infrastructure as code, making it versionable, repeatable, and easier to manage.

This approach minimizes the risk of manual configuration errors and ensures consistency across environments.

2. **Automated and Scalable Deployments:** Terraform automates the deployment process, enabling you to scale your infrastructure efficiently. Whether you're deploying in a development, staging, or production environment, Terraform ensures that all resources are provisioned and configured correctly with minimal manual intervention.

3. **Easier Maintenance and Updates:** With Terraform, updating or modifying configurations for tools like ArgoCD and Prometheus is straightforward. You can manage changes centrally, apply updates consistently, and track modifications through version control, reducing the complexity of managing dynamic infrastructures.

4. **Improved Security and Compliance:** By using Terraform, you can enforce security policies and compliance standards across your infrastructure. Terraform configurations can be audited, and changes can be tracked, helping to maintain a secure and compliant environment.

Prerequisites

Before diving into the deployment process, ensure that you have the following prerequisites in place:

1. **AWS Account:** You should have access to an AWS account with sufficient permissions to create and manage resources like VPCs, EC2 instances, and EKS clusters.

2. **Terraform Installed:** Terraform must be installed on your local machine or the server from which you will manage the infrastructure.

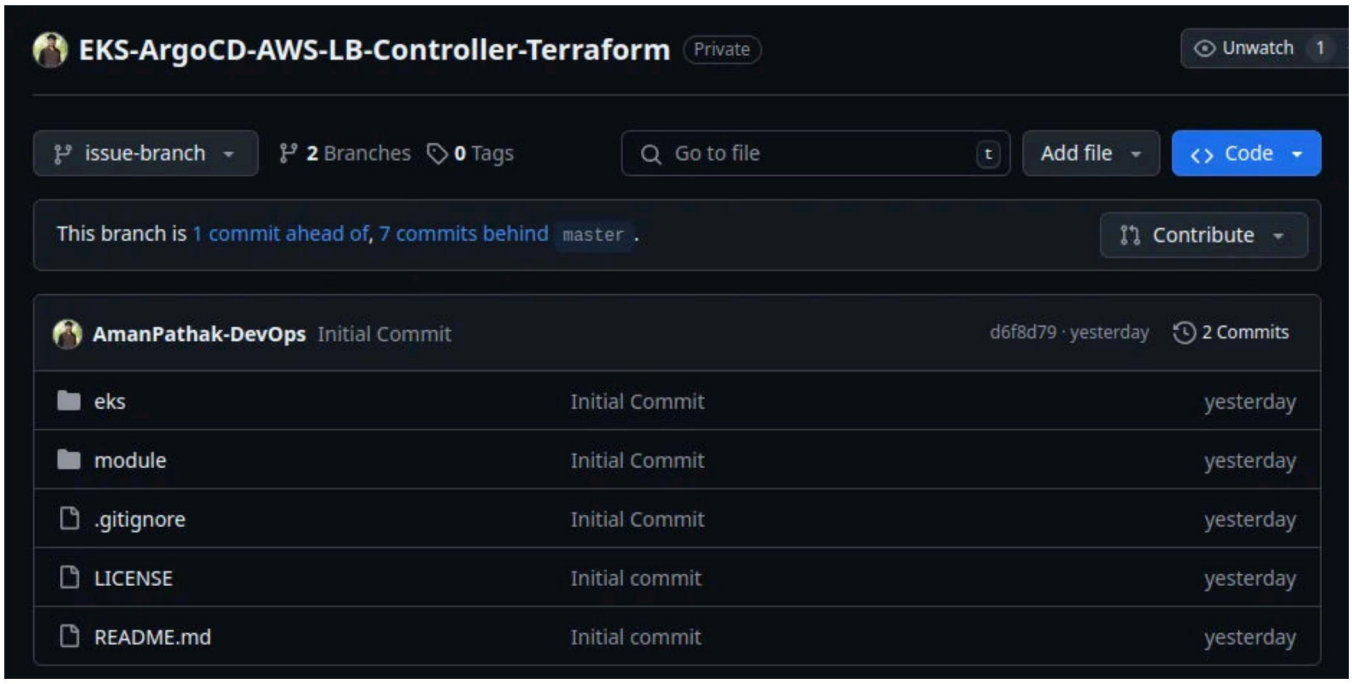
3. **Git:** Ensure Git is installed for cloning the necessary repositories.

4. **AWS CLI:** The AWS CLI tool should be installed and configured with appropriate credentials for deploying resources on AWS.

5. **Basic Knowledge of Terraform and Kubernetes:** Familiarity with Terraform and Kubernetes is essential to follow along with the steps and understand the infrastructure you're deploying.

GitHub Repository used in this demonstration- https://github.com/AmanPathak-DevOps/EKS-ArgoCD-AWS-LB-Controller-Terraform/tree/master

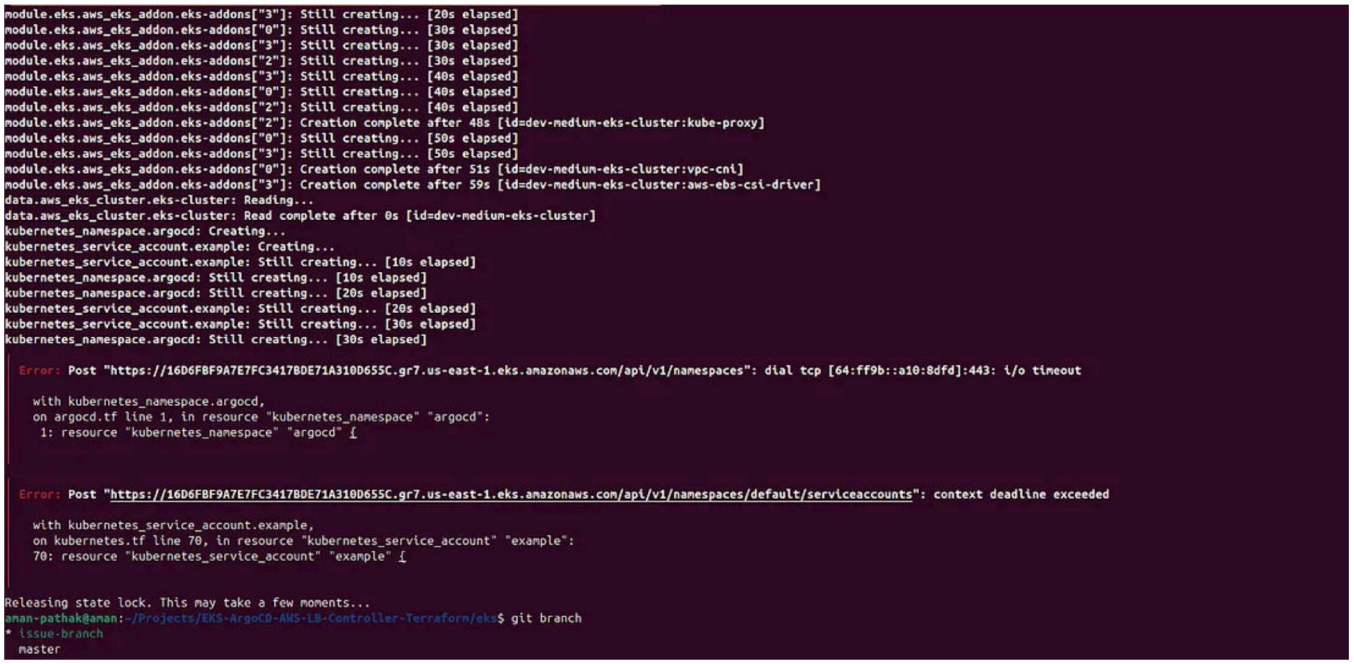
So, I have created two branches Issue-branch and the other one is default named as master. Issue branch is to let you understand, why you can't deploy argoCD, Prometheus, and any other resources in your EKS Cluster.



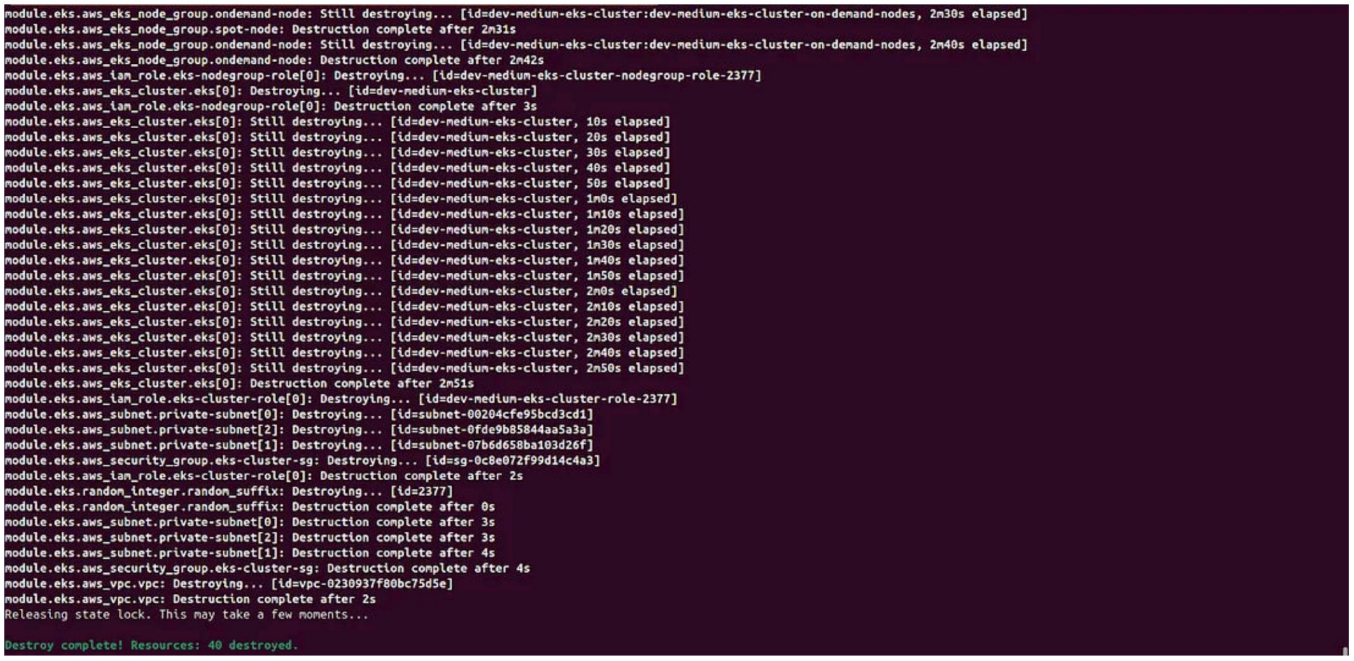
This is because your Cluster is Private. You need to be in the same network to enter your cluster and make any changes or configure something.

So, the Issue branch will create everything related to Cluster only such as EKS Cluster, Node groups, EKS Add-ons, etc. But when it tries to deploy argocd and other resources inside your cluster It will throw the error.

You can check the error in the below snippet.



P.S.: The EKS Cluster was configured as it was not visible in the above snippet.

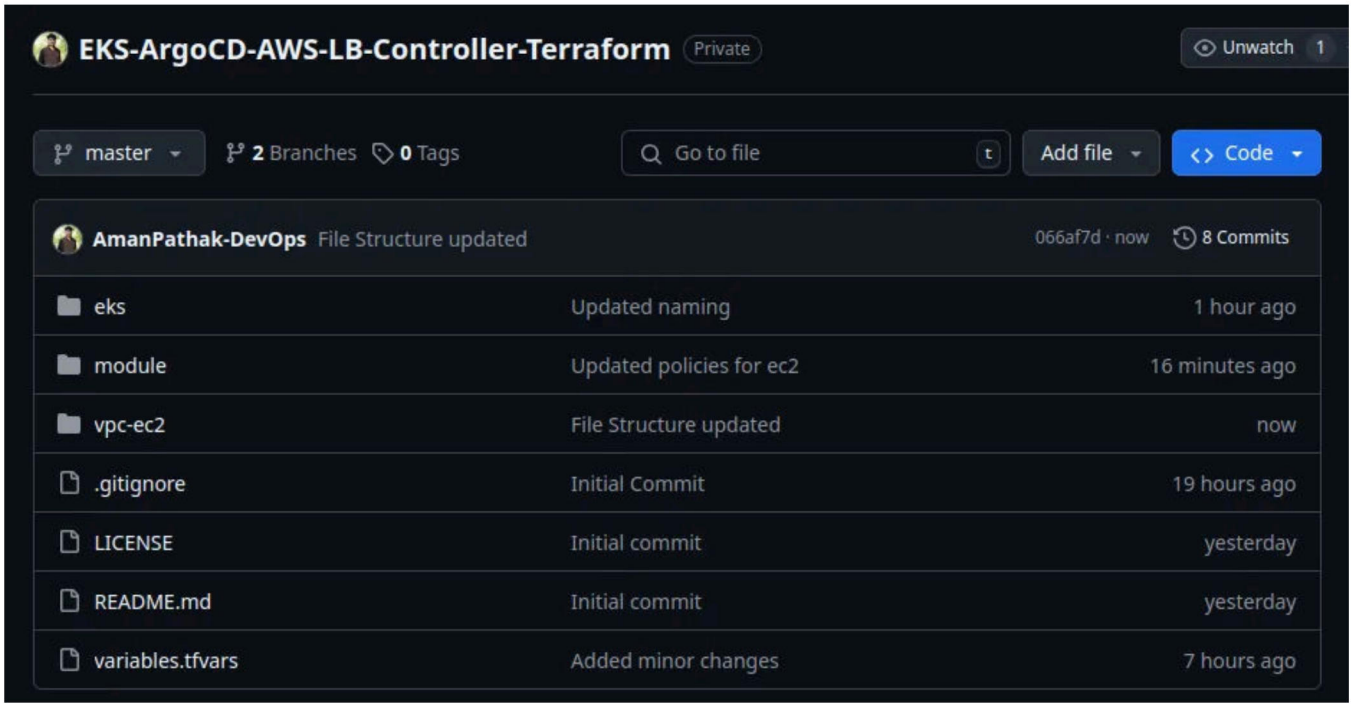


How to solve this?

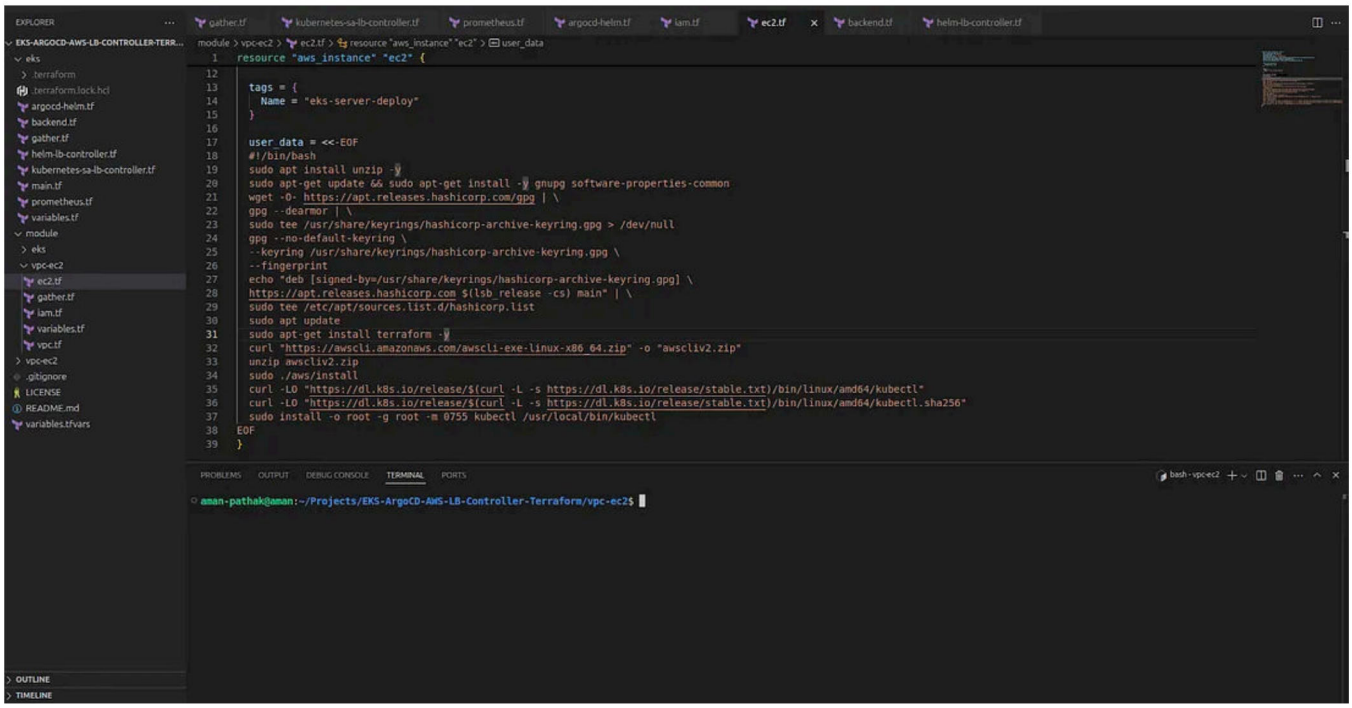
To deploy anything inside your EKS Cluster, we need an instance or server to deploy those configurations that have the same VPC.

So, we are going to deploy our VPC and one EC2 server first. Then from the ec2 server, we will deploy everything related to the EKS Cluster.

In the master branch of the repository, you will find two modules. The first module is vpc-ec2 which we need to apply first.

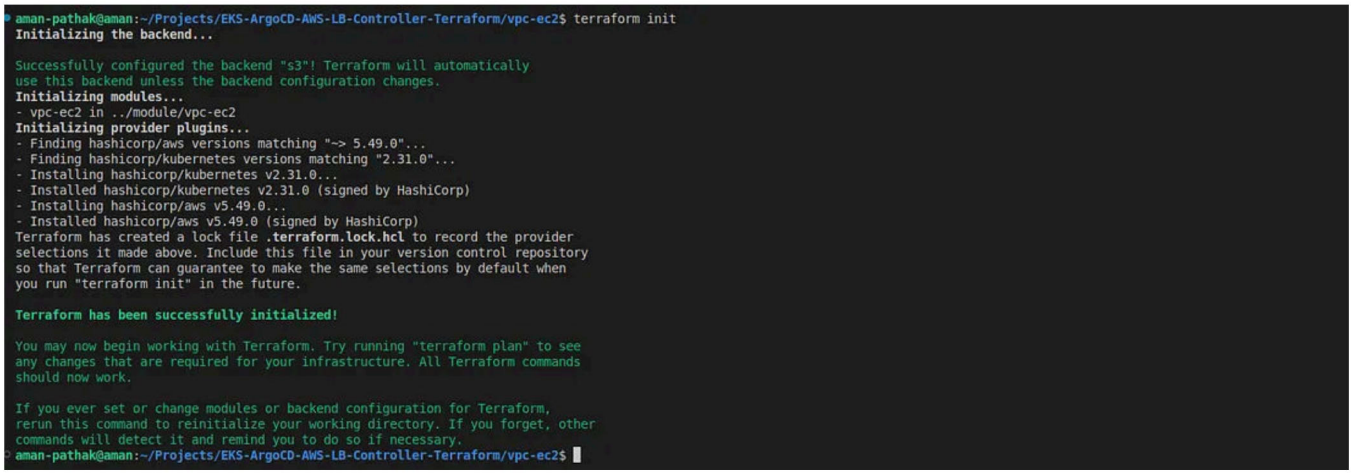


So, clone the repository with the master branch and navigate to the vpc-ec2 directory.



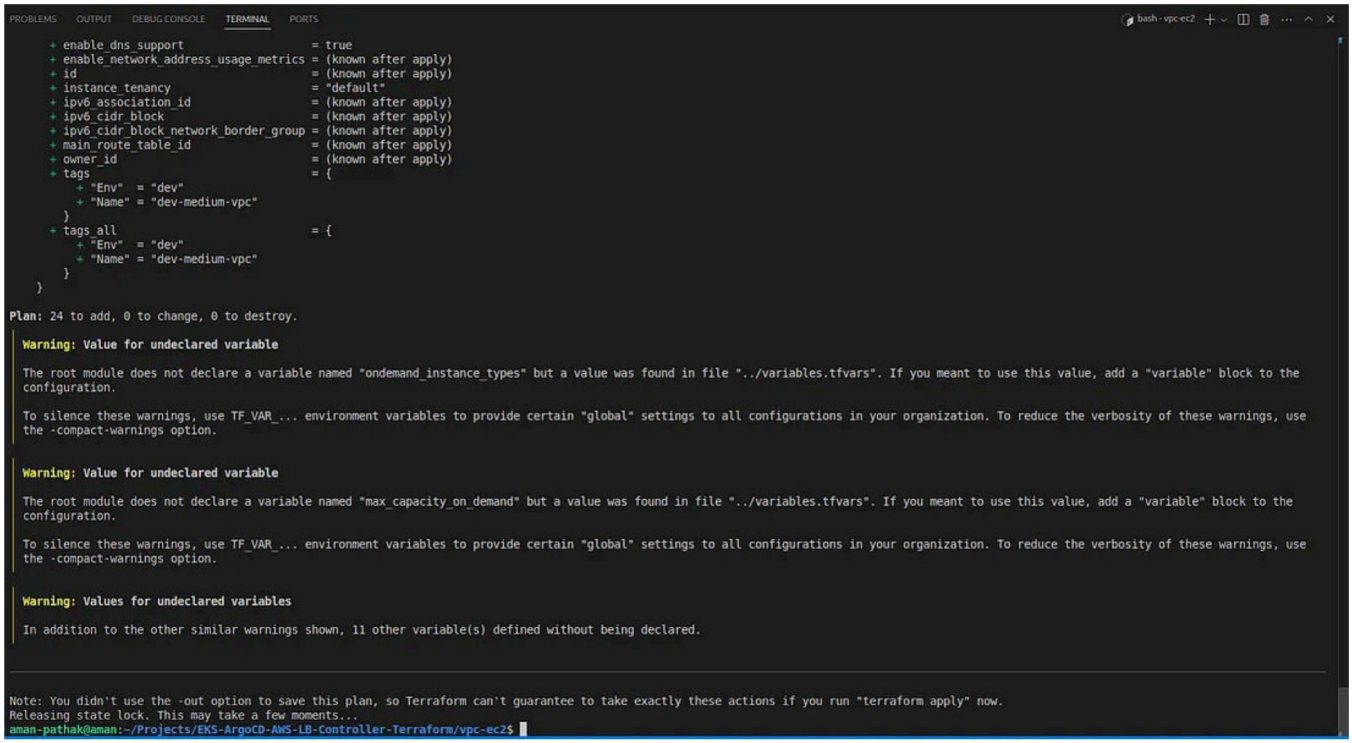
Run the below commands to deploy VPC(Other networking resources) & EC2 Server on AWS.

```
terraform init
```

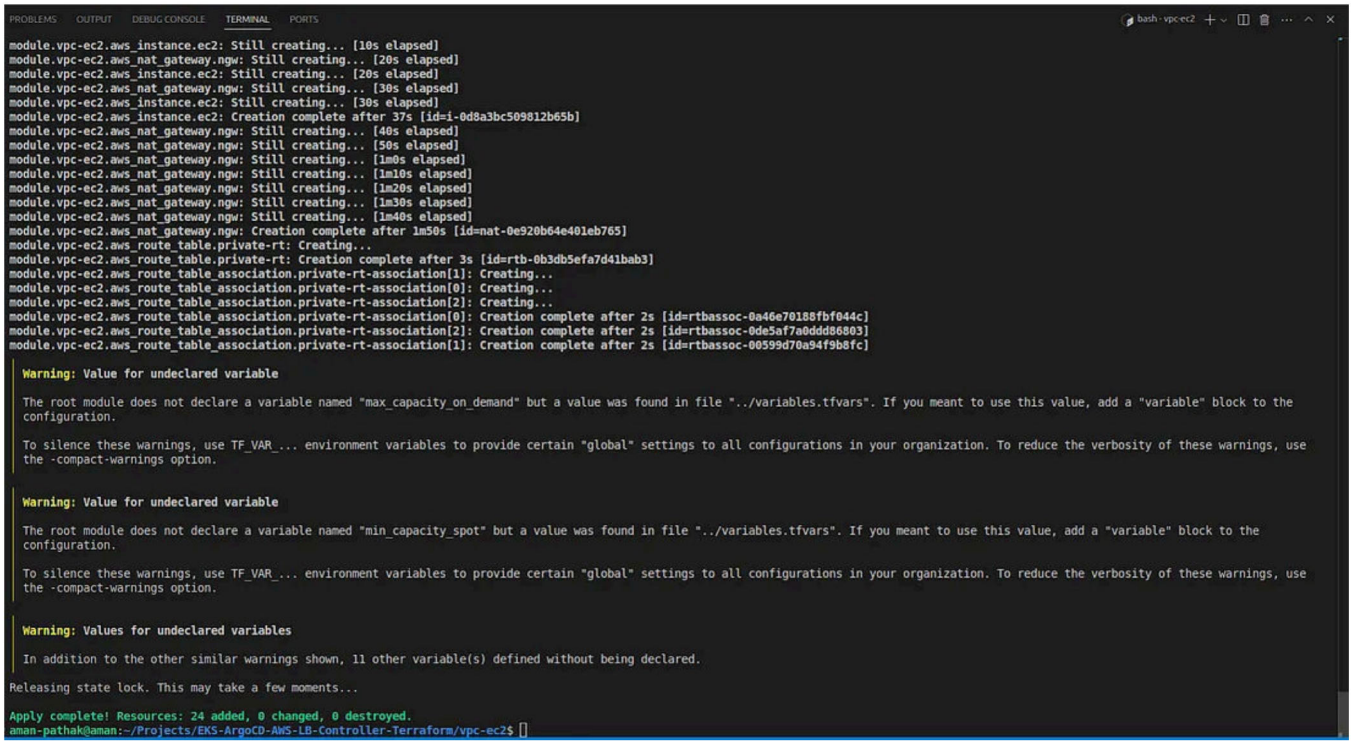



```
terraform validate
```


```
terraform plan -var-file=../variables.tfvars
```



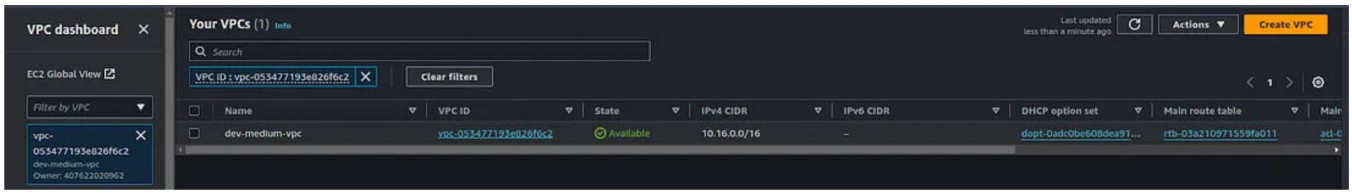
```
terraform apply -auto-approve -var-file=../variables.tfvars
```



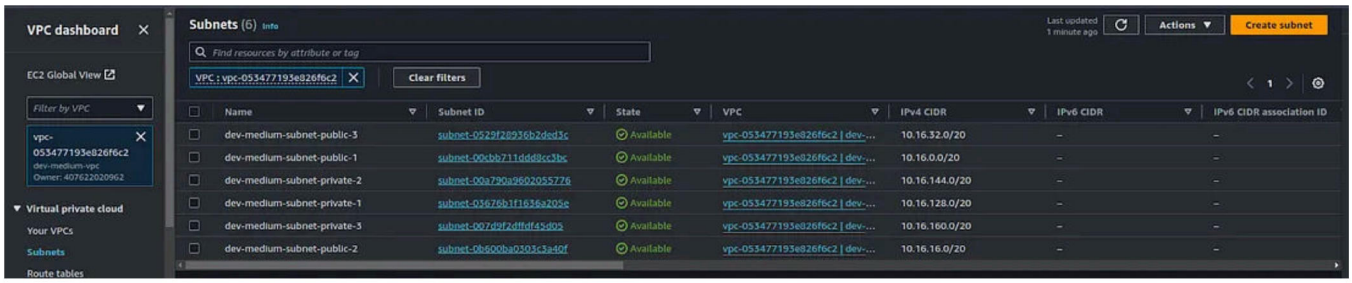
VPC & other services created through the above commands

Here are the snippets

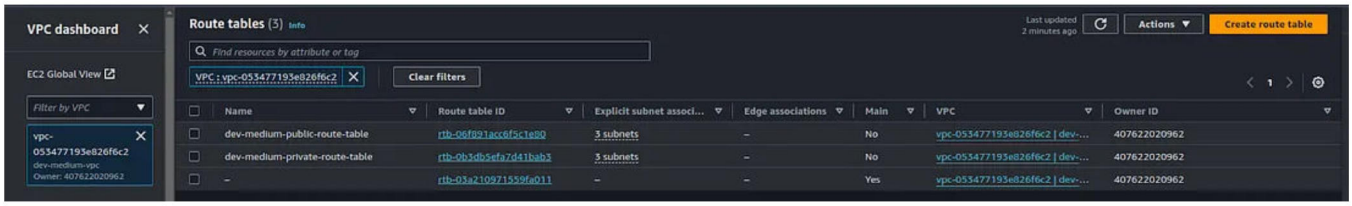
VPC



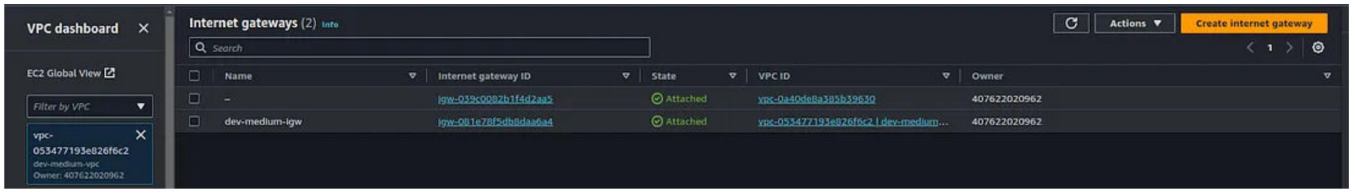
Subnets



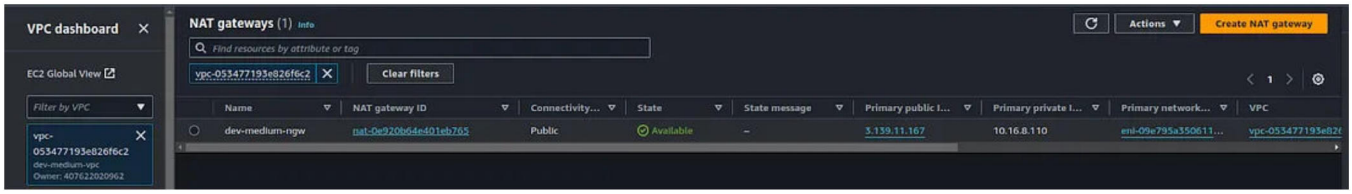
Route tables



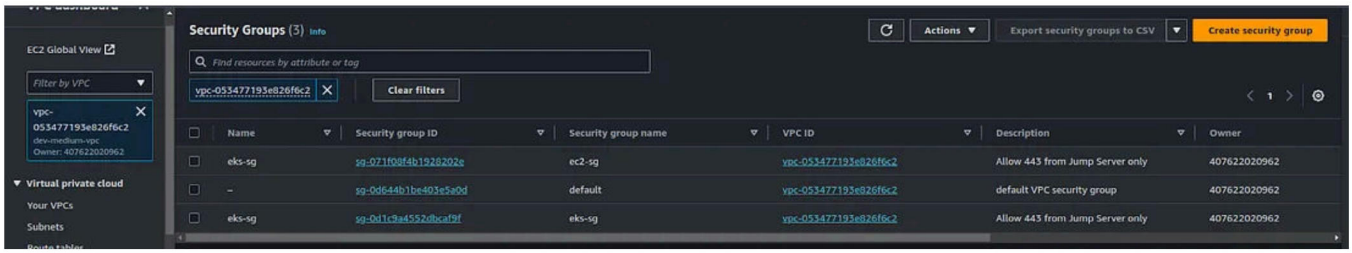
Internet Gateway



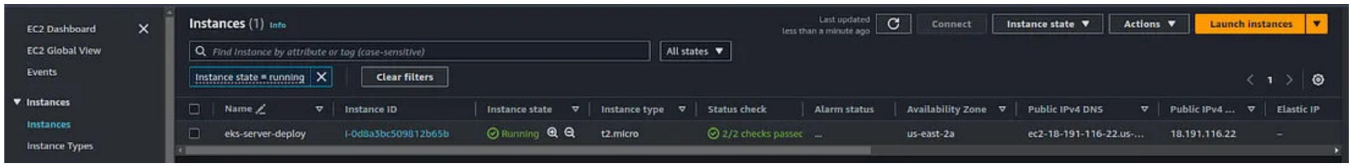
NAT Gateway



Security Group



EC2-Instance

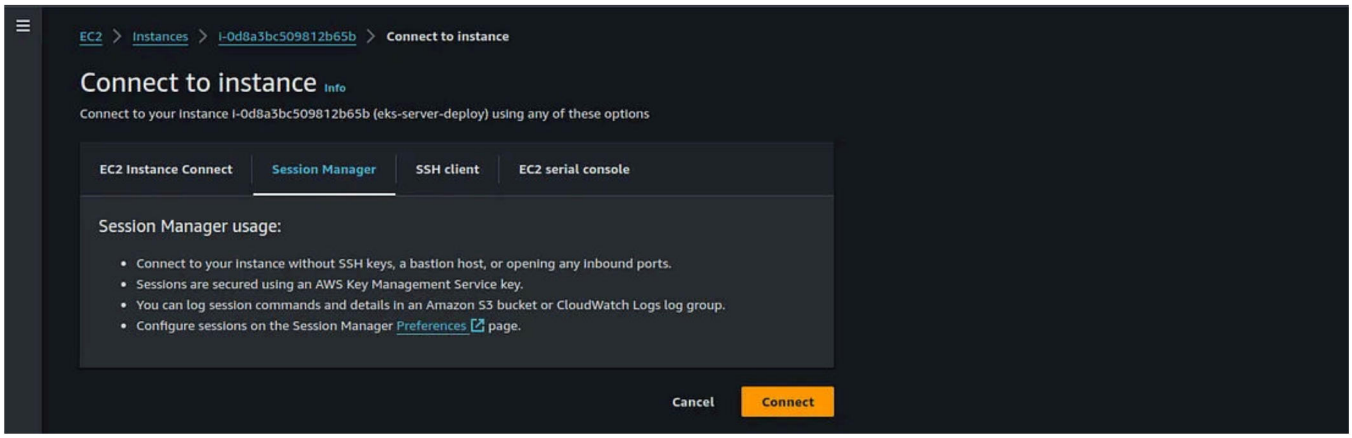


So, everything is created according to our requirements.

Now log in to the Server by selecting the created ec2 instance and click on **Connect**

You can log in without the Pem file as we did not follow that.

If Session Manager is struggling to connect to the instance, you can use EC2 Instance Connect to login to the server.



I am logged In to my server and switched to the Ubuntu user with the help of the below command

```
sudo su ubuntu
```

We need to install a few tools as our pre-requisites. We installed it through user data in the Terraform script

Run the below command to validate whether it's installed or not. If not, wait for 5–10 minutes,

```
terraform version
aws - version
kubectl version
```

Once all the tools are installed then, we need to configure AWS CLI as we need to deploy our infrastructure over AWS Cloud.

Run the below command to configure CLI and make sure you have sufficient permission with your credentials. For demonstration, you can utilize Administration Access user keys

P.S.: Don't use the same access keys as you will have your own keys in your AWS Account

```
aws configure
```

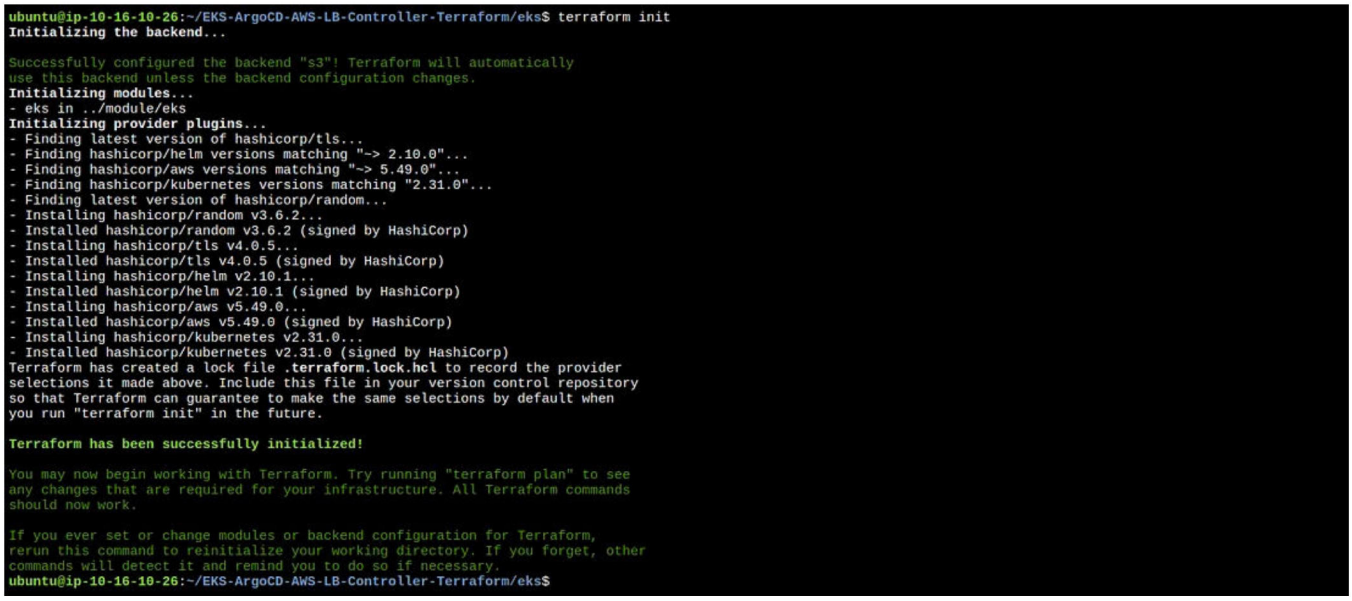
Now, clone the same repository where our Terraform code is present for EKS

```
git clone https://github.com/AmanPathak-DevOps/EKS-ArgoCD-AWS-LB-Controller-Ter
```

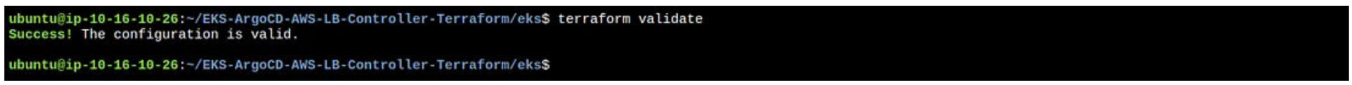
Now, we are ready to deploy our EKS Cluster, and other tools through Terraform.

Navigate to the eks directory and run the below commands to deploy it

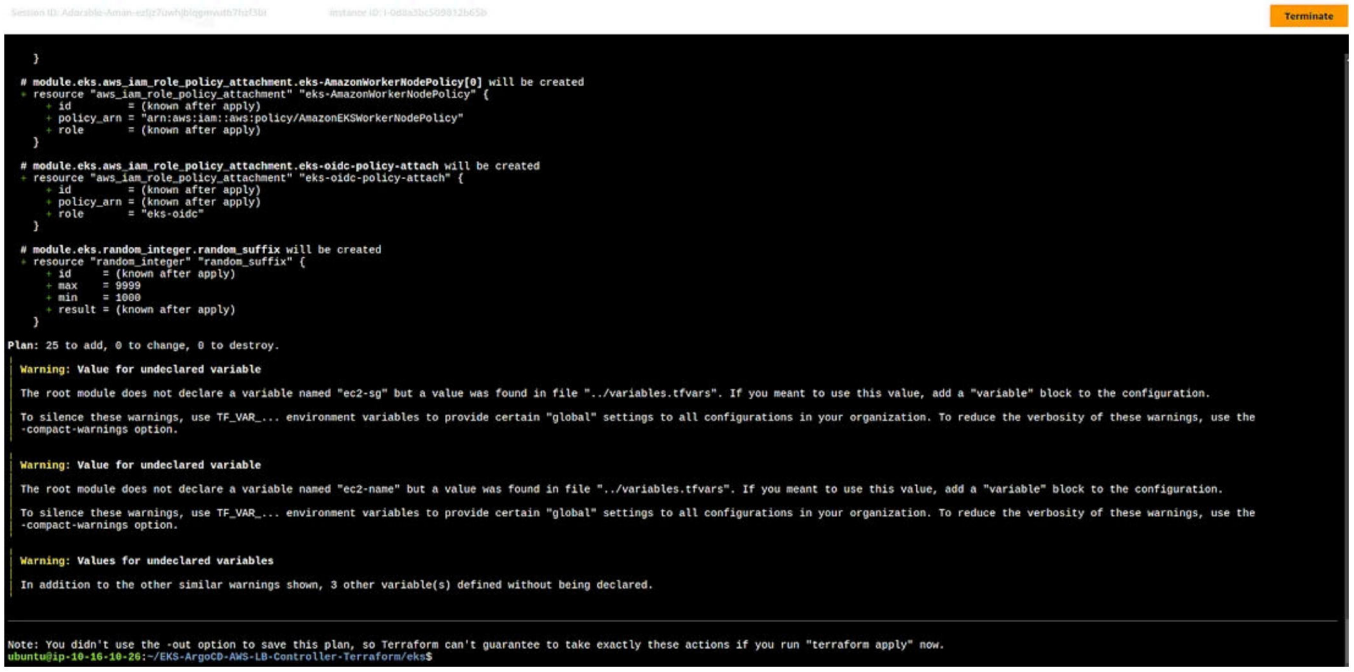
```
terraform init
```



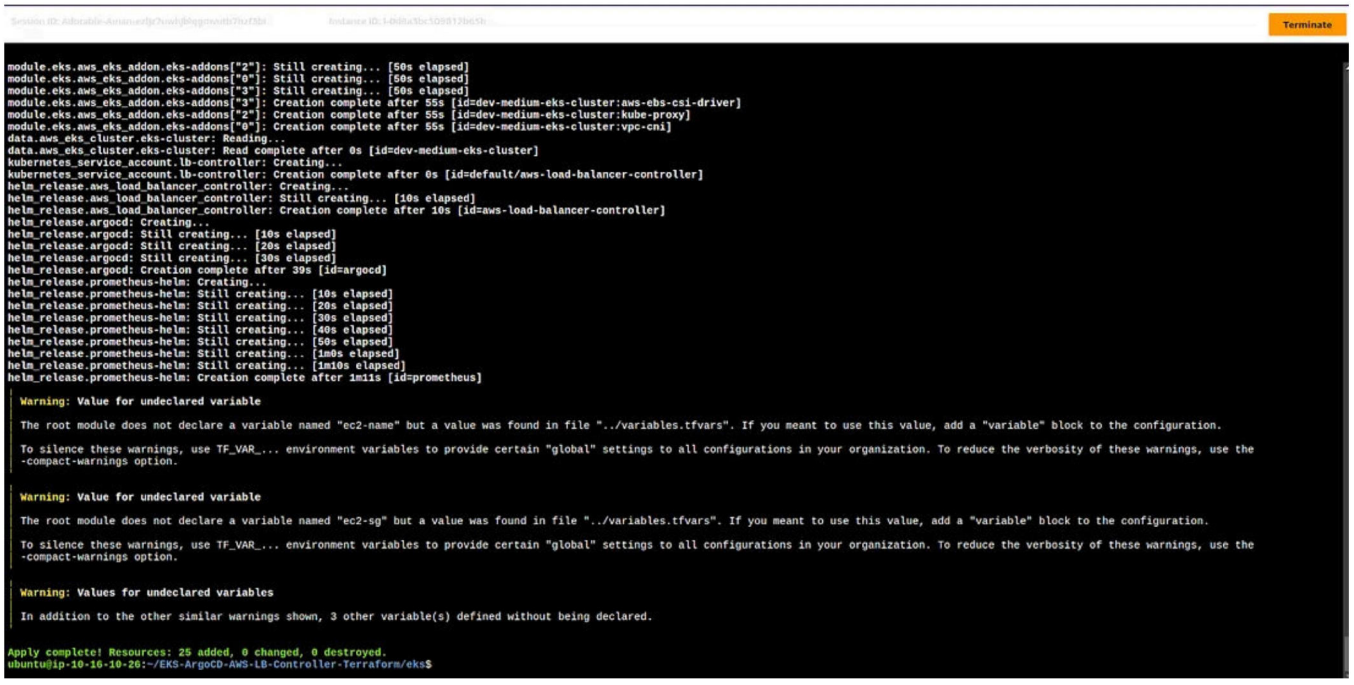

```
terraform validate
```




```
terraform plan -var-file=../variables.tfvars
```



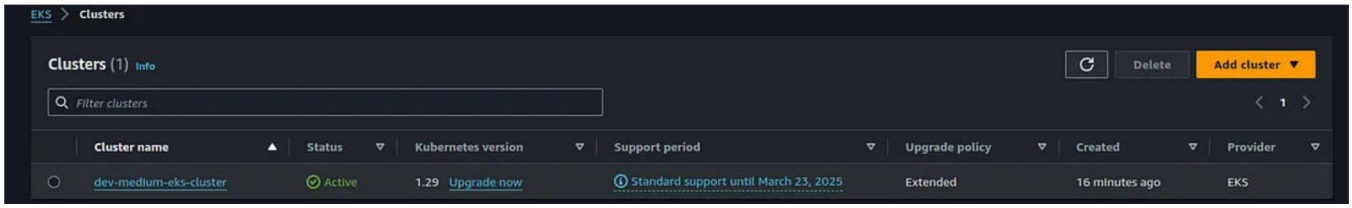

```
terraform apply -auto-approve -var-file=../variables.tfvars
```



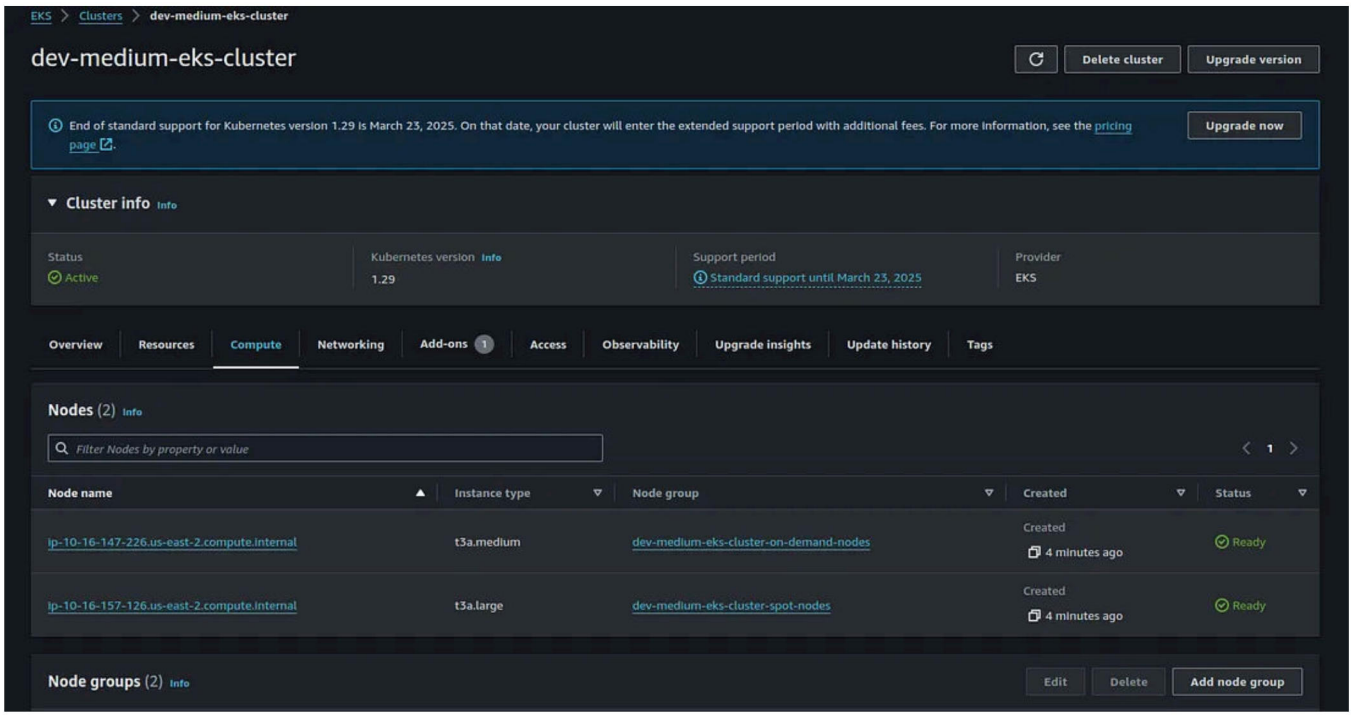
EKS Cluster & other services created through the above commands

Here are the snippets

EKS Cluster



Node Groups



To validate whether the other Kubernetes resources have been created or not. We need to update the kubconfig on our same ec2 server

Run the below command to update the kubeconfig

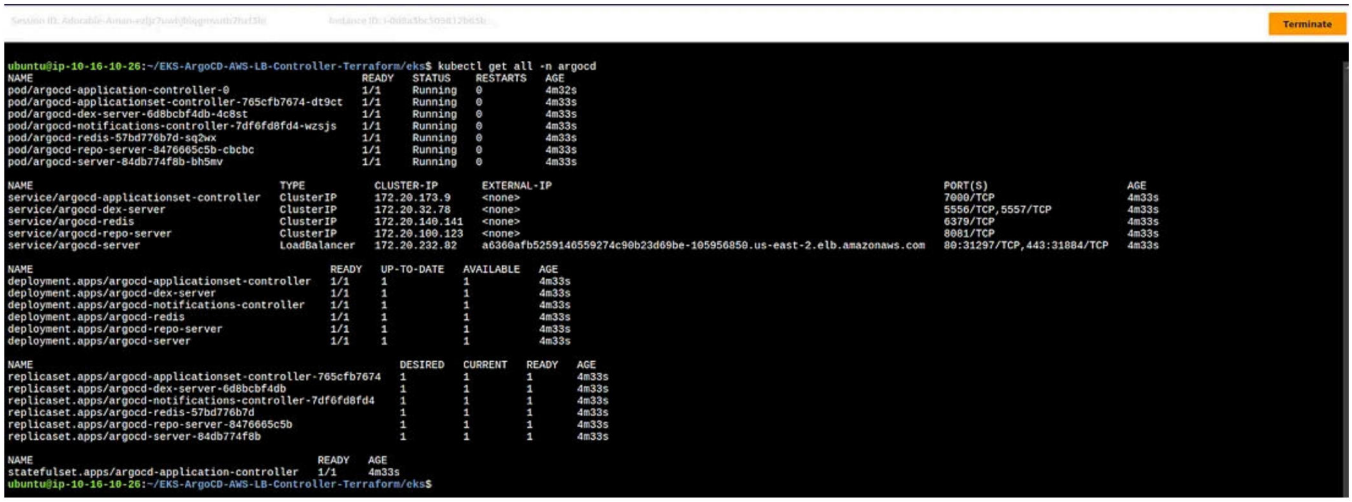
```
aws eks update-kubeconfig — name dev-medium-eks-cluster — region us-east-2
```

Run the below command to validate whether it's our cluster or not

```
kubectl get nodes
```

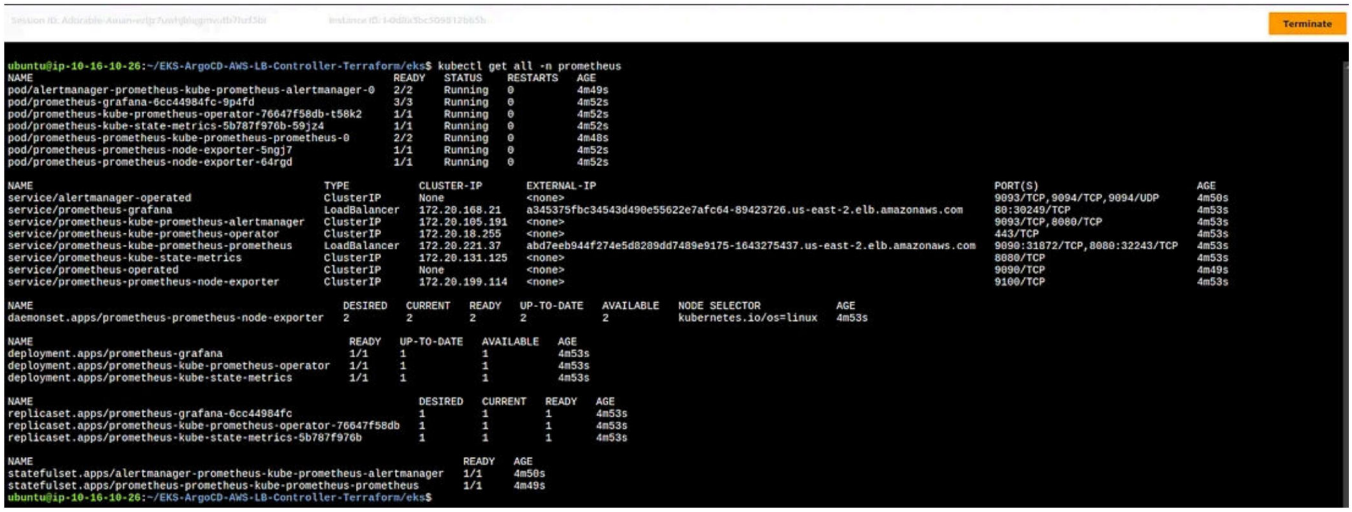
Check whether the resources are running in argocd namespace or not

```
kubectl get all -n argocd
```



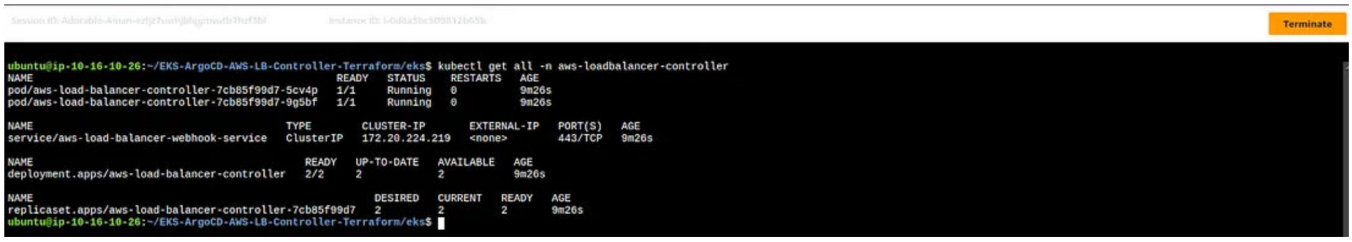
Check whether the resources are running in the Prometheus namespace or not

```
kubectl get all -n prometheus
```



Check whether the AWS Load Balancer controller pods are running in a aws-loadbalancer-controller namespace or not

```
kubectl get all -n aws-loadbalancer-controller
```

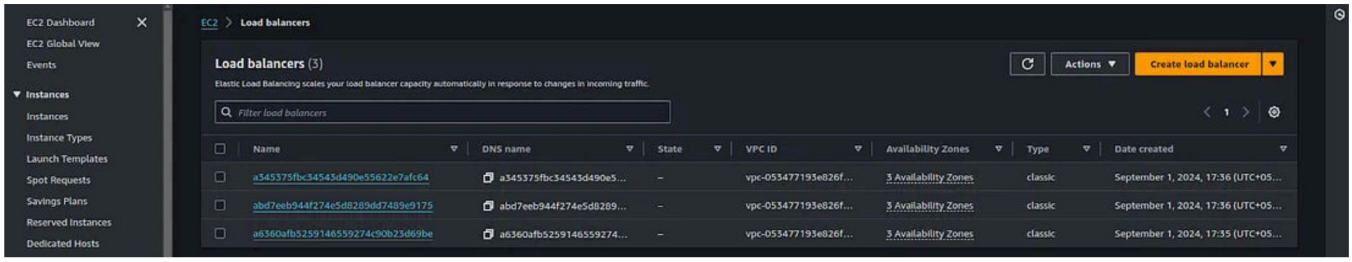


So, we have configured the AWS Load Balancer Controller, ArgoCD, Prometheus, and Grafana through Terraform.

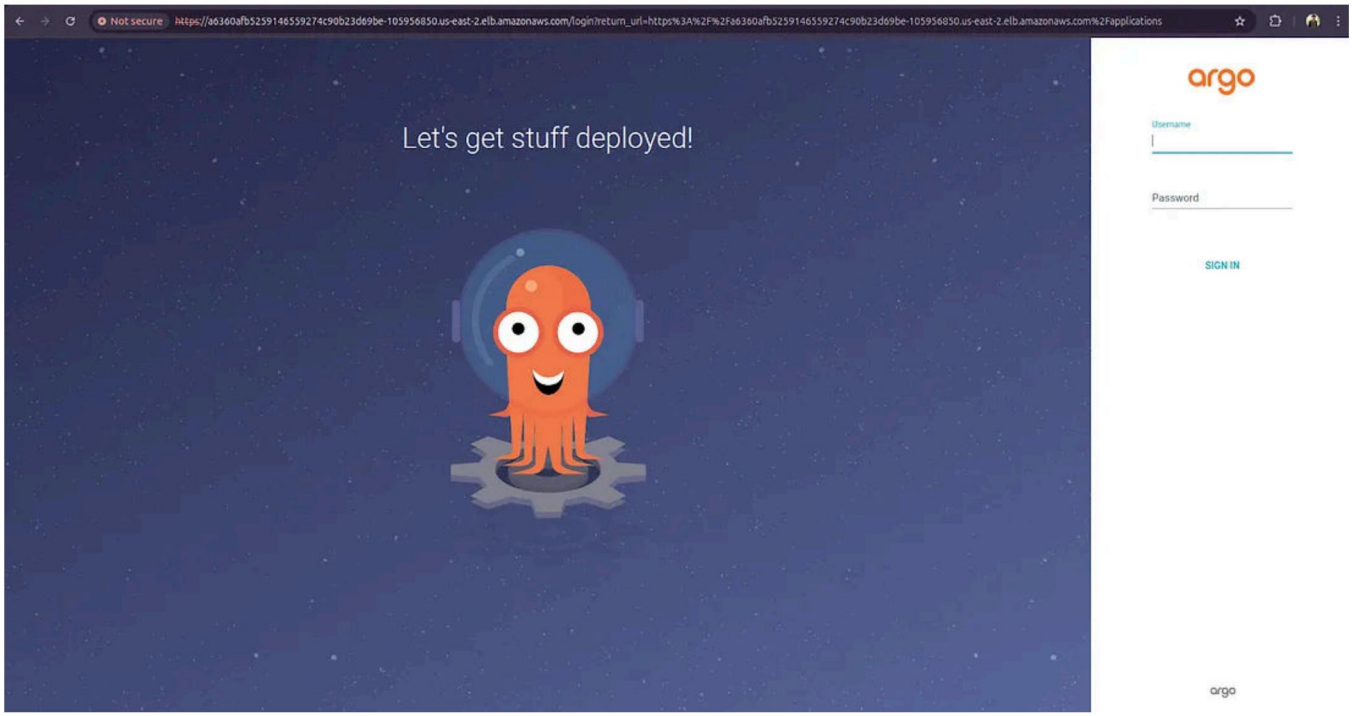
According to the terraform script, we have updated the service type for ArgoCD, Prometheus, and Grafana from ClusterIP to LoadBalacer.

Hence, navigate to Load Balancer on your AWS account.

You can check which LB is for which tool through the kubectl command that we ran in the above steps.



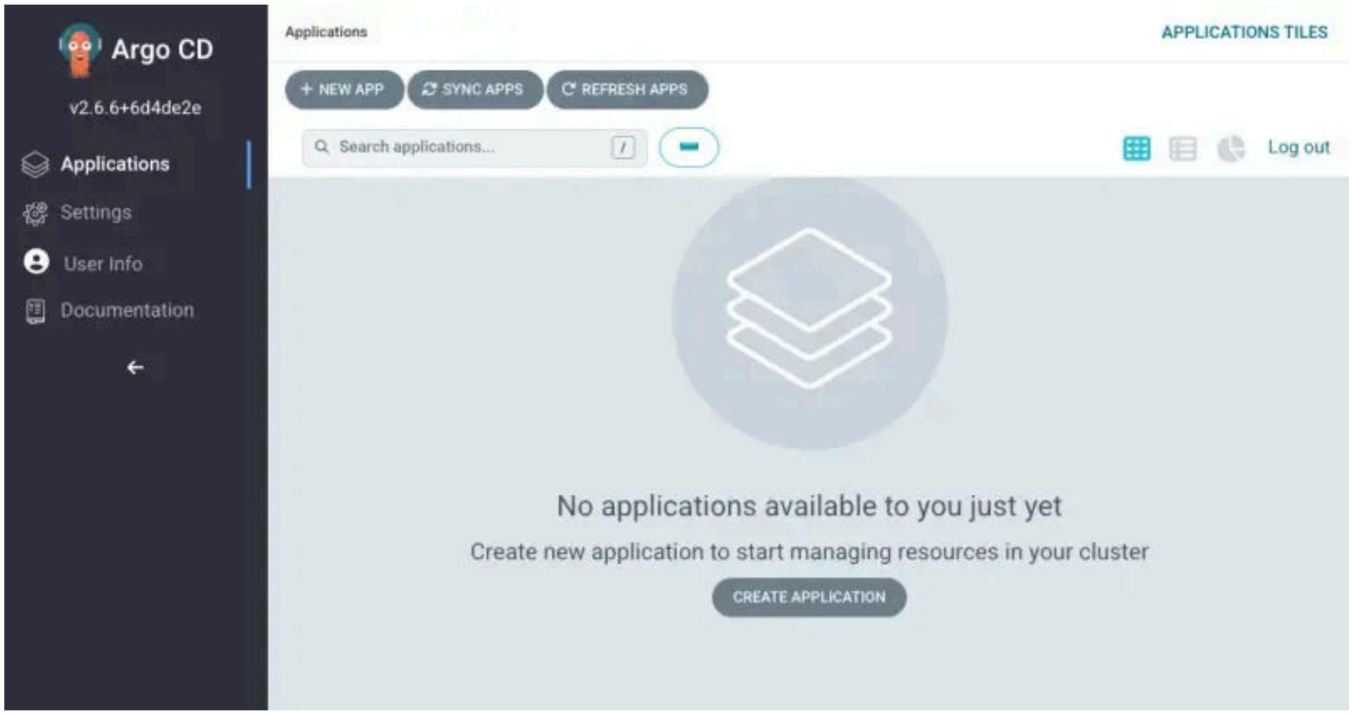
So, let's try to access argoCD first. Copy the DNS and paste it into your favorite browser



The username is admin but it needs the password which we don't know. So, we need to run a command on our ec2 server to get the password for the login

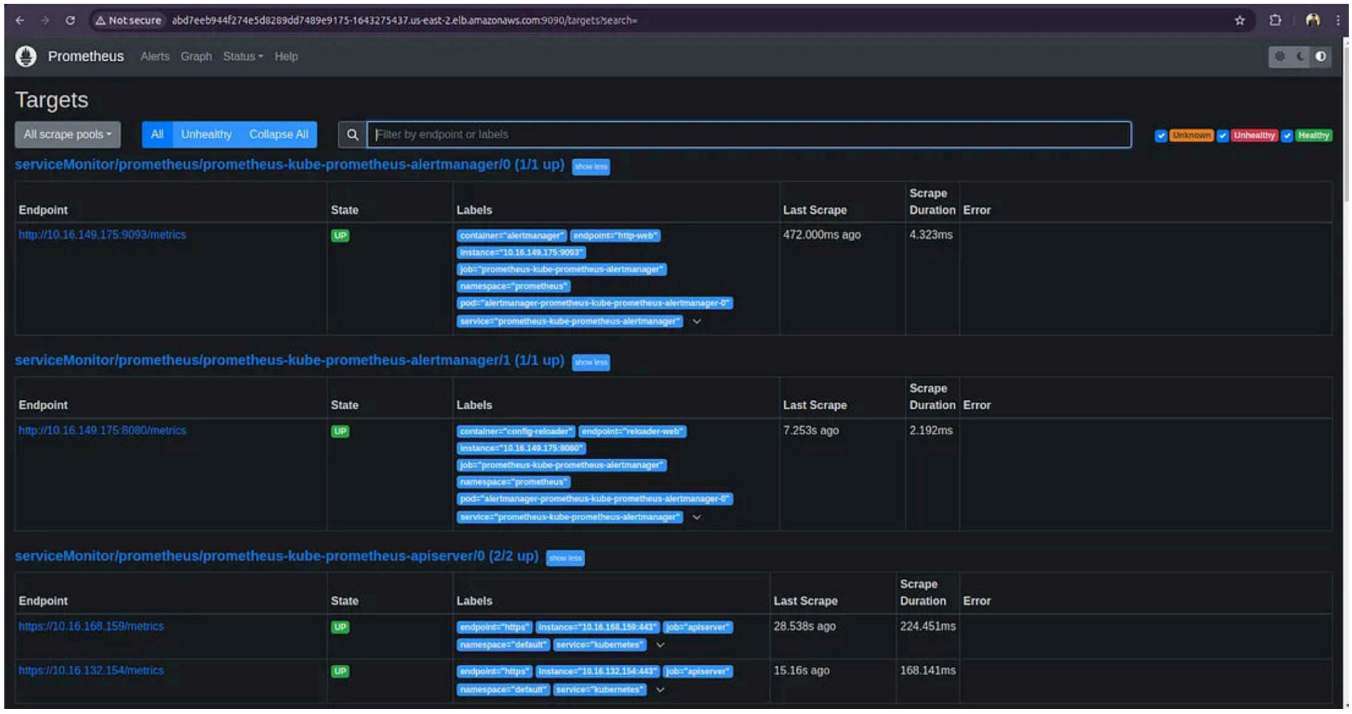
```
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.pa
```

It is accessible



Now, let's try to access Prometheus. Copy the DNS and paste it into your favorite browser with port 9090.

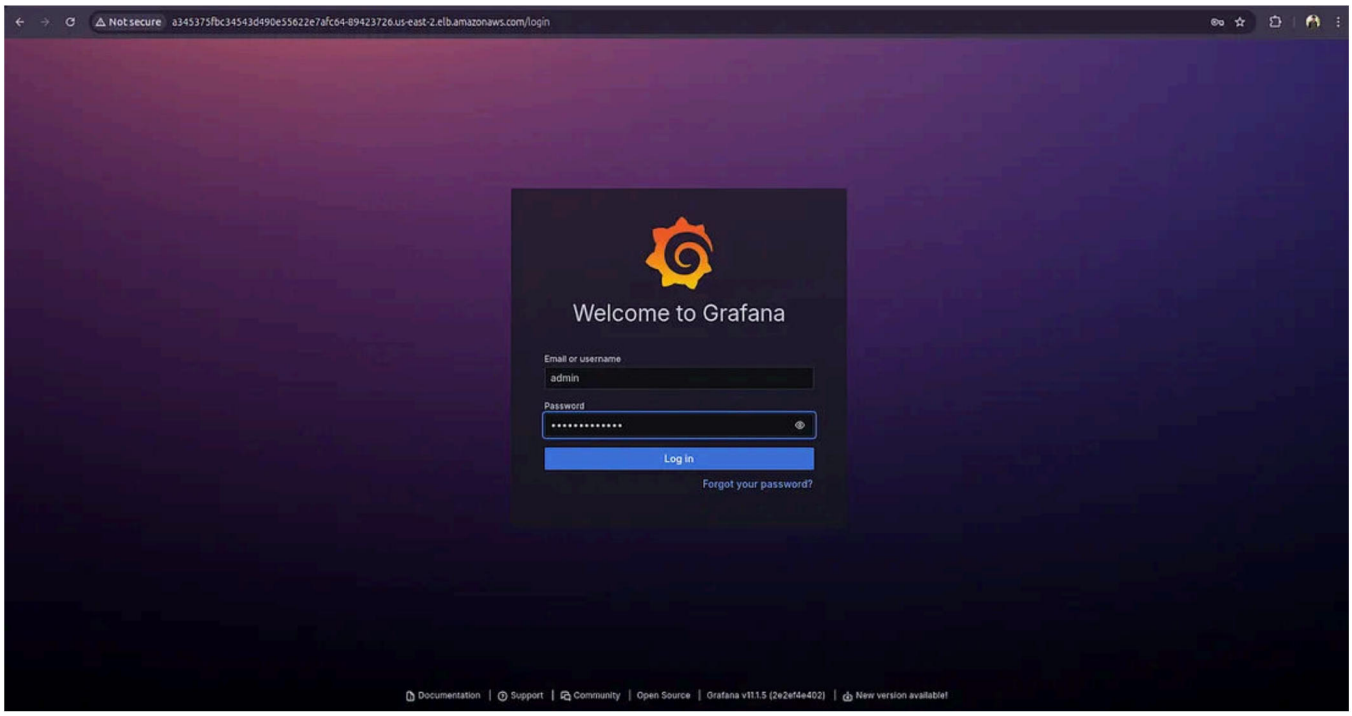
It is accessible



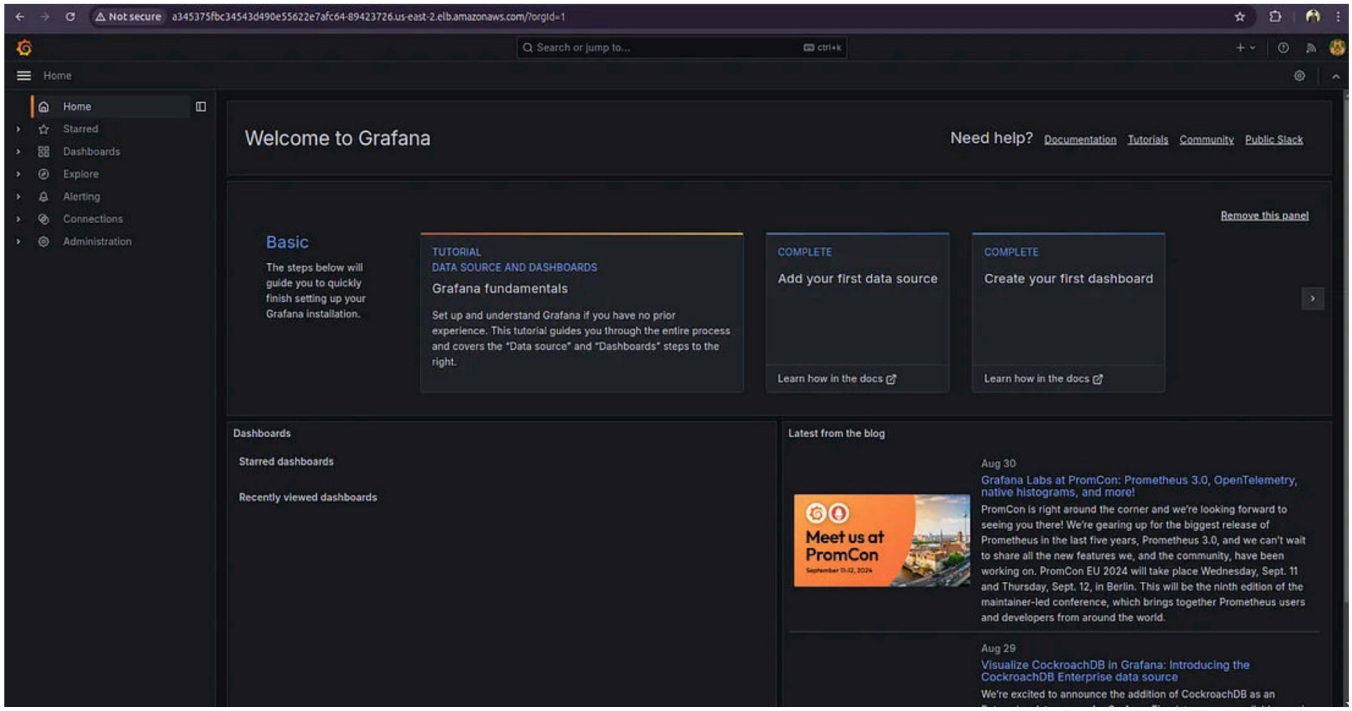
In the end, let's try to access our Grafana dashboard. Copy the DNS and paste it into your favorite browser

It is accessible.

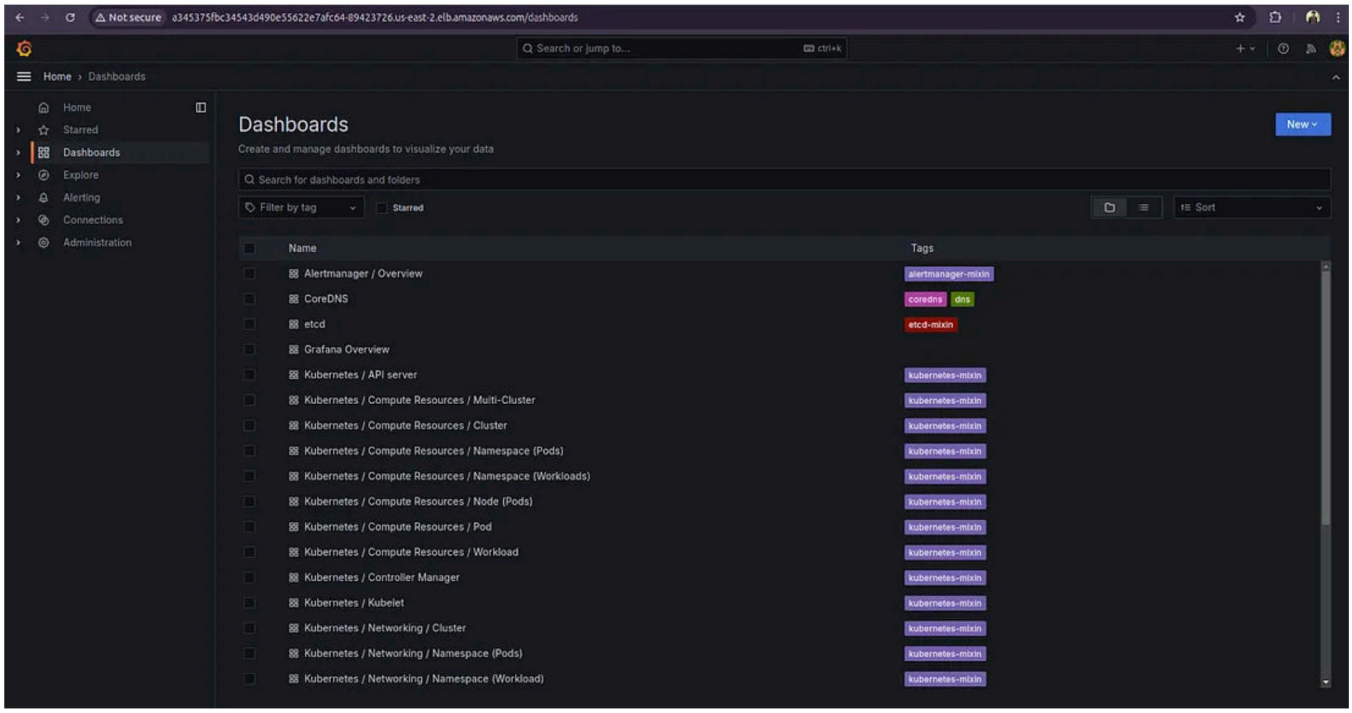
The username is **admin** and the password is **prom-operator** for the Grafana login



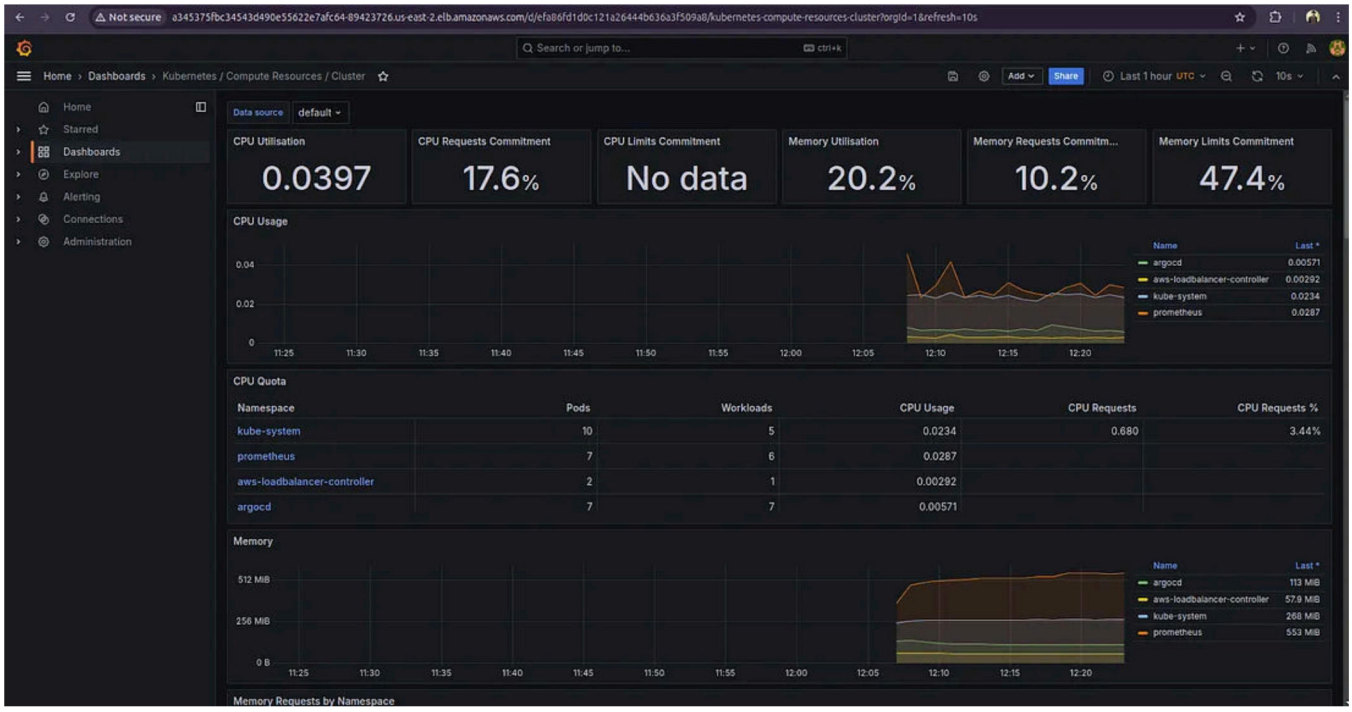
After login, the Grafana dashboard



There are a lot of dashboards already imported. You can click any of them to get insight about the EKS Cluster resources accordingly.



Here is one of them



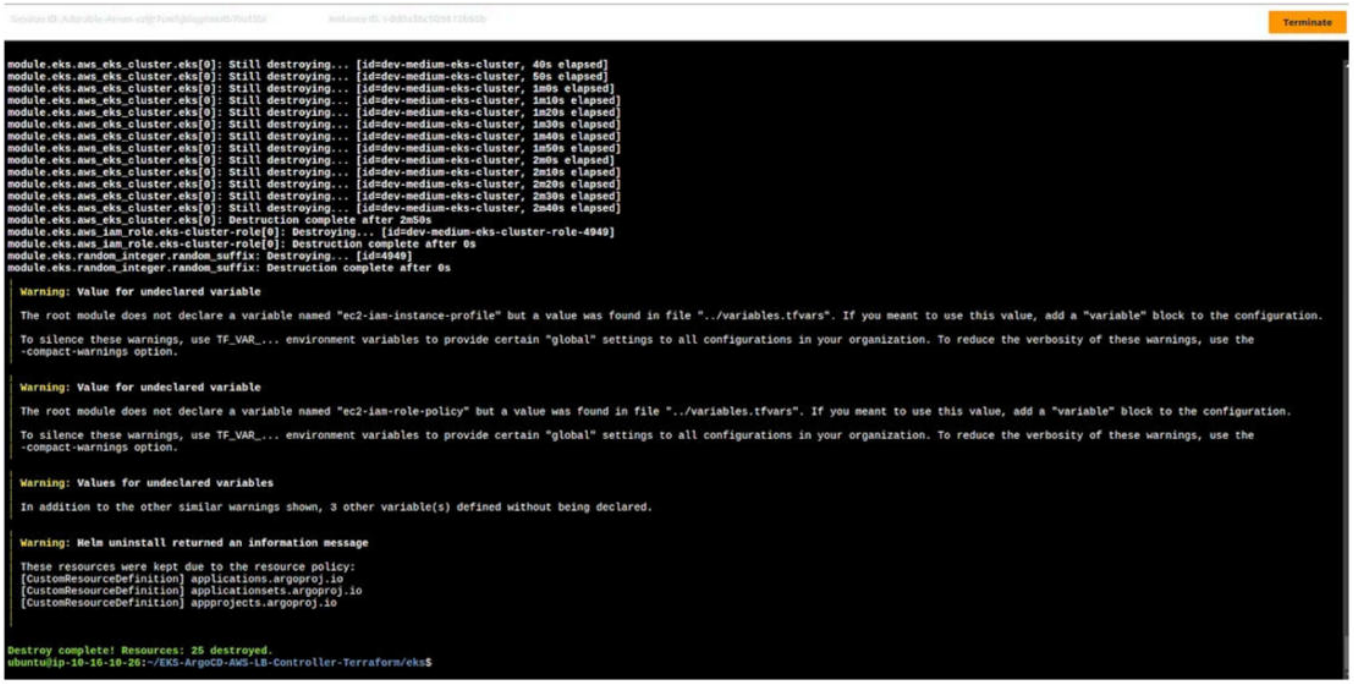
So, we have completed our demonstration for today.

Hope you learn something new today.

To prevent the cost of cloud. Don't forget to destroy all the resources.

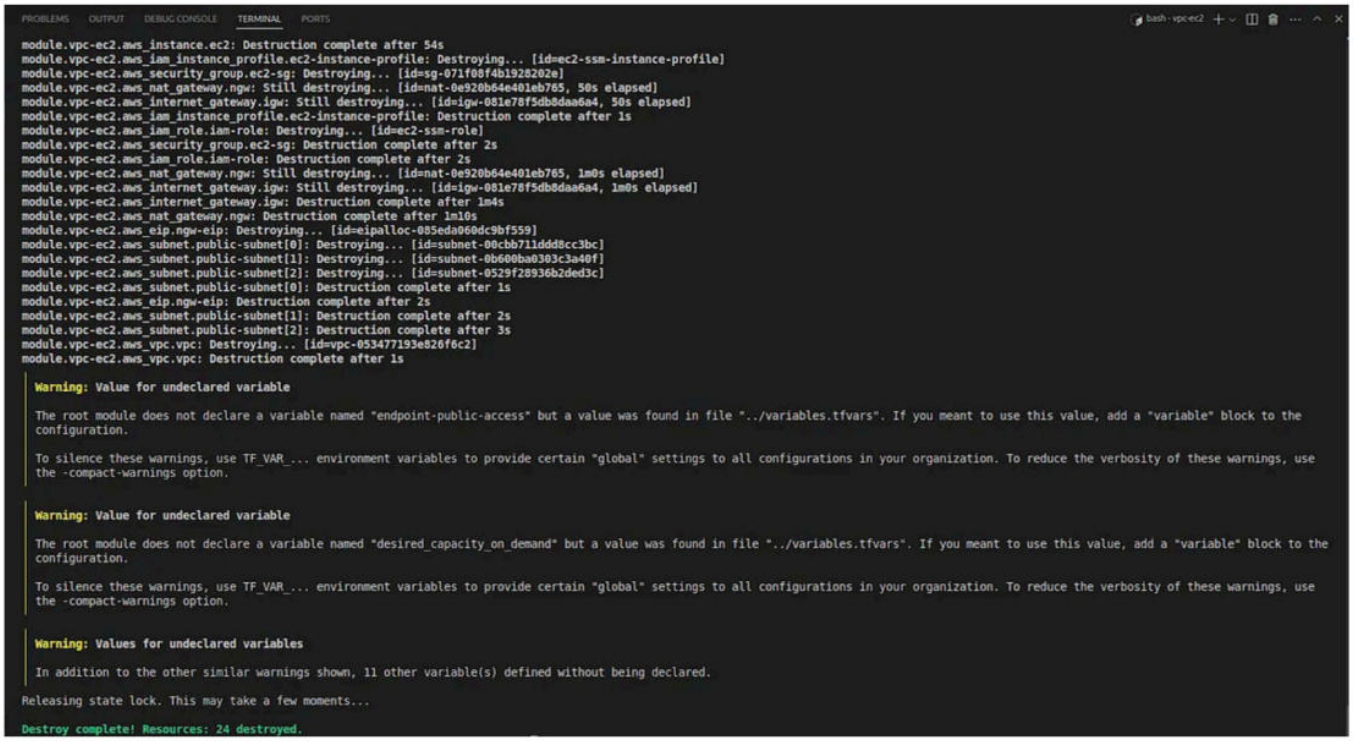
To do that, we need to run EKS Cluster first.

```
terraform destroy -auto-approve -var-file=../variables.tfvars
```



Now, destroy the vpc and an ec2 server. To do it, run the below command on your local server

```
terraform destroy -auto-approve -var-file=../variables.tfvars
```



I would recommend you to read the eks terraform files to get a better understanding of the resources. As in the repo, everything is straightforward if you worked on Terraform already.

Conclusion

In this comprehensive guide, we've explored how to deploy a private EKS cluster on AWS and configure essential Kubernetes tools such as ArgoCD, Prometheus, and Grafana using Terraform. By following these steps, you can efficiently manage your infrastructure and ensure that your applications are running smoothly in a secure, scalable environment. Remember to clean up your resources after the demonstration to avoid unnecessary costs. Continuous learning and hands-on practice are key to mastering these DevOps practices, so keep experimenting with different configurations and tools to enhance your skills.

Support my work-
https://www.buymeacoffee.com/aman.pathak

Stay connected on **LinkedIn**: LinkedIn Profile

Stay up-to-date with **GitHub**: GitHub Profile

Want to discuss trending technologies in DevOps & Cloud
Join the **Discord Server**- https://discord.gg/jdzF8kTtw2

Feel free to reach out to me, if you have any other queries.

Happy Learning!

AWS Kubernetes Terraform Cloud Computing DevOps

Follow

Published in Towards Dev

6.1K Followers · Last published 3 hours ago

A publication for sharing projects, ideas, codes, and new theories.

Follow

Written by Aman Pathak

9.7K Followers · 6 Following

DevOps & Cloud Engineer | AWS Community Builder | AWS Certified | CKA Certified | Azure | Terraform | Docker | Ansible | CI/CD Jenkins | Oracle Certified

Responses (1)

What are your thoughts?

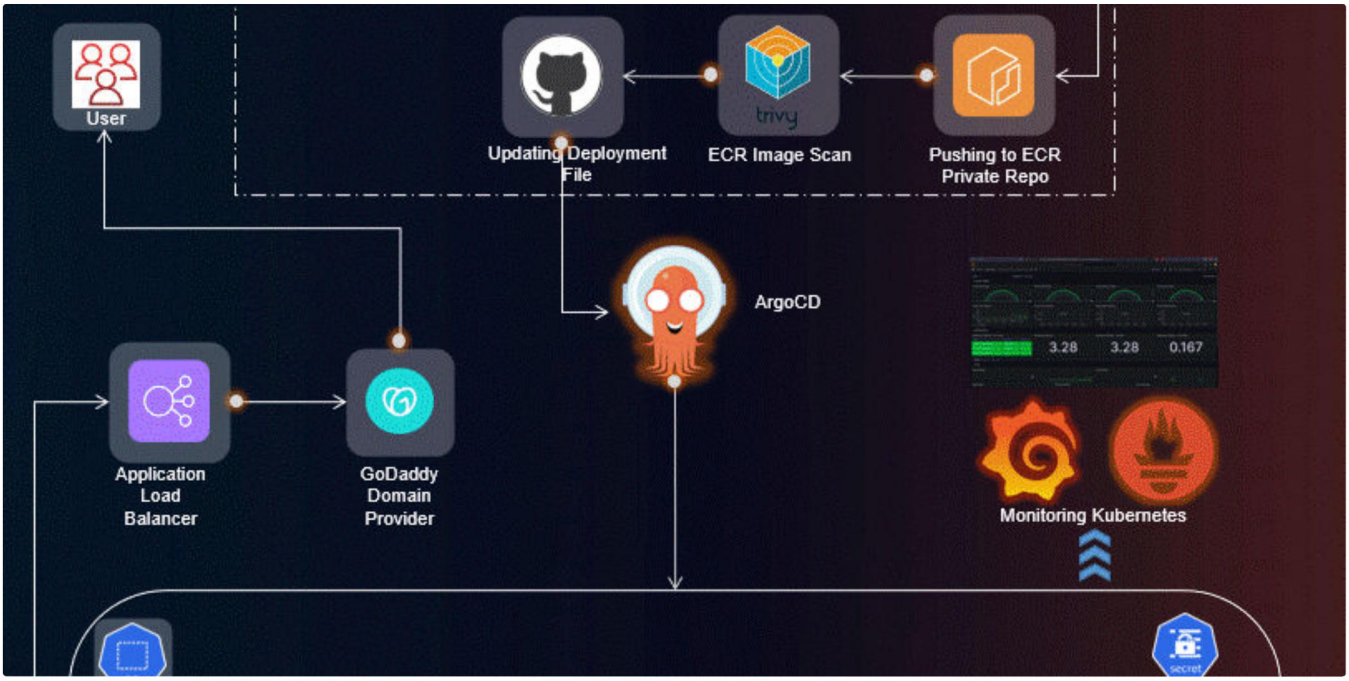
Respond

Santosh Garole
Sep 6, 2024

Good read.

8 Reply

More from Aman Pathak and Towards Dev



In Stackademic by Aman Pathak

Advanced End-to-End DevSecOps Kubernetes Three-Tier Project using AWS EKS, ArgoCD, Prometheus...

Project Introduction:

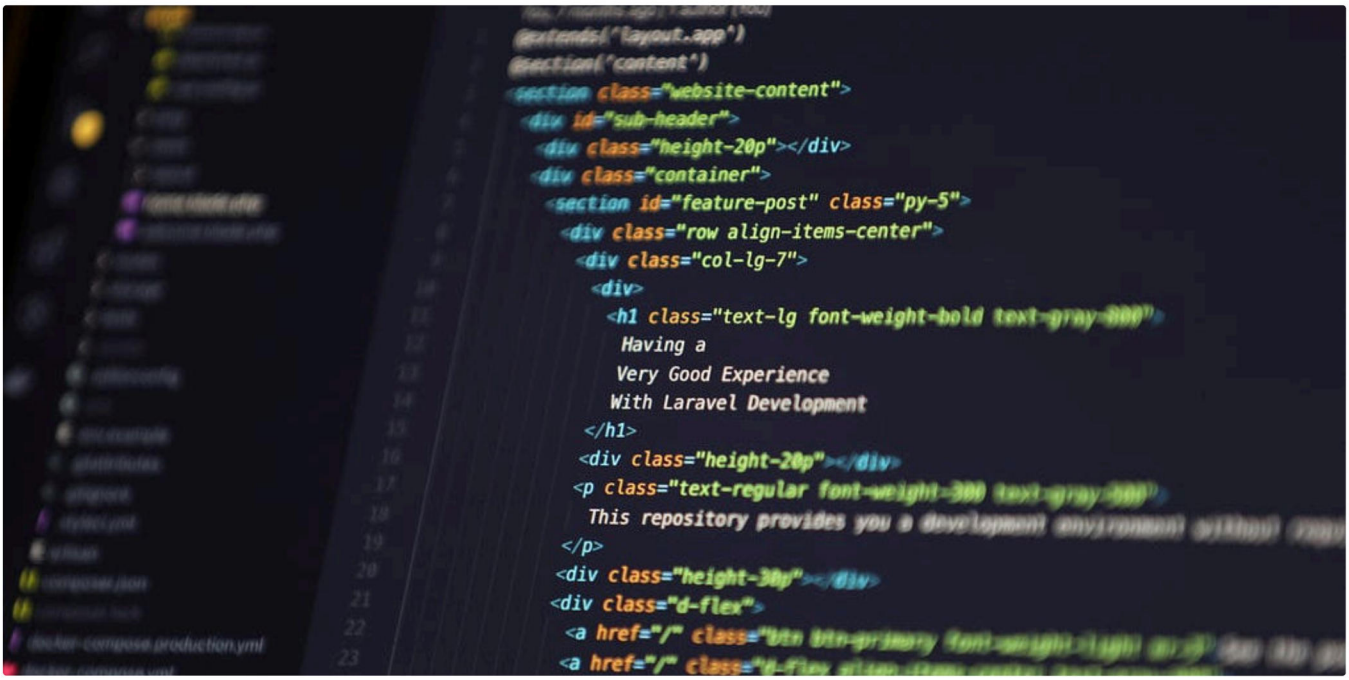
Jan 18, 2024 2K 26

In Towards Dev by DataScience Nexus

12 Python Libraries for Free Market Data That Everyone Should Know

Access to accurate and timely market data is crucial for traders, financial analysts, and data scientists. Whether you are building a...

Jan 2 345 6



In Towards Dev by Kevin Meneses González

This Is How I Scrape 99% of Websites (Step-by-Step Guide)

“Without data, you’re just another person with an opinion.” — W. Edwards Deming

Jan 15 489 3

In Stackademic by Aman Pathak

How to Containerize and Deploy a Three-Tier Application on AWS EKS with Kubernetes

Introduction

Aug 20, 2024 146 5

See all from Aman Pathak

See all from Towards Dev

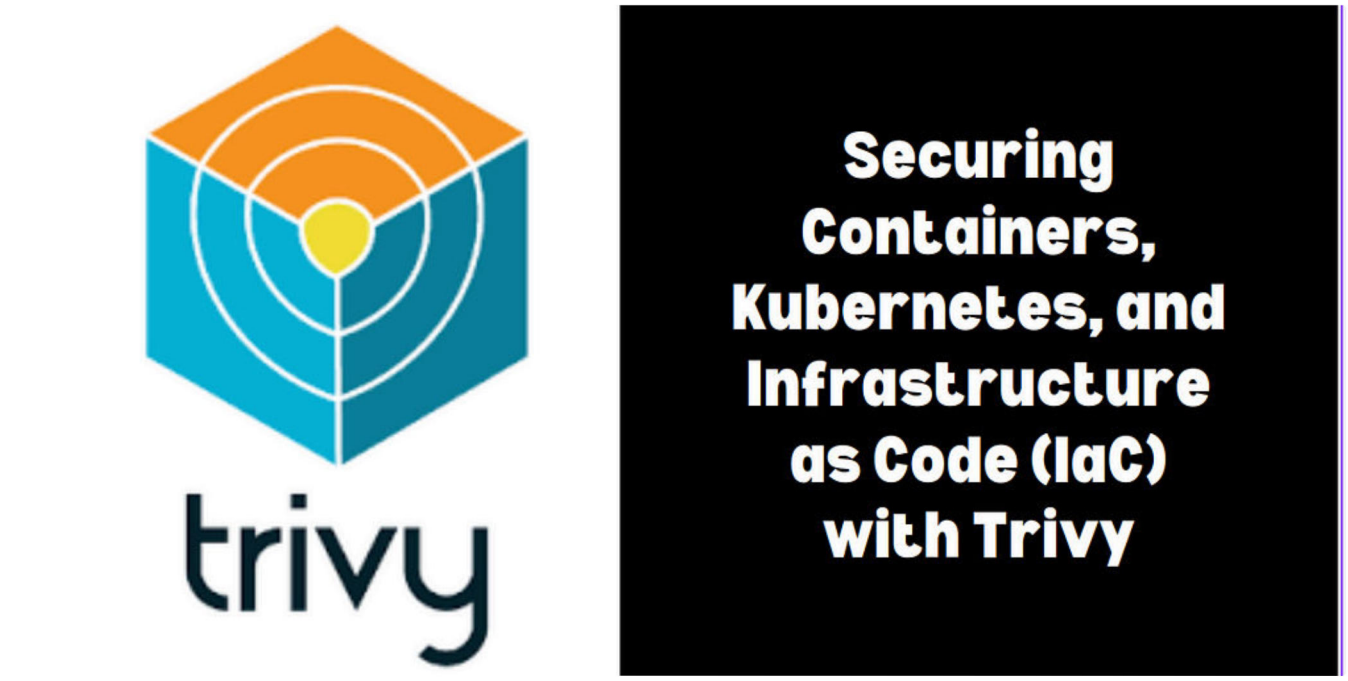
Recommended from Medium

Zudonu Osomudeya

10 Real-Life Problems DevOps Solves (and How)

Downtime Disasters—The 2 AM Pager Nightmare

Feb 1 92



Siddharth Singh

Securing Containers, Kubernetes, and Infrastructure as Code (IaC) with Trivy

Introduction

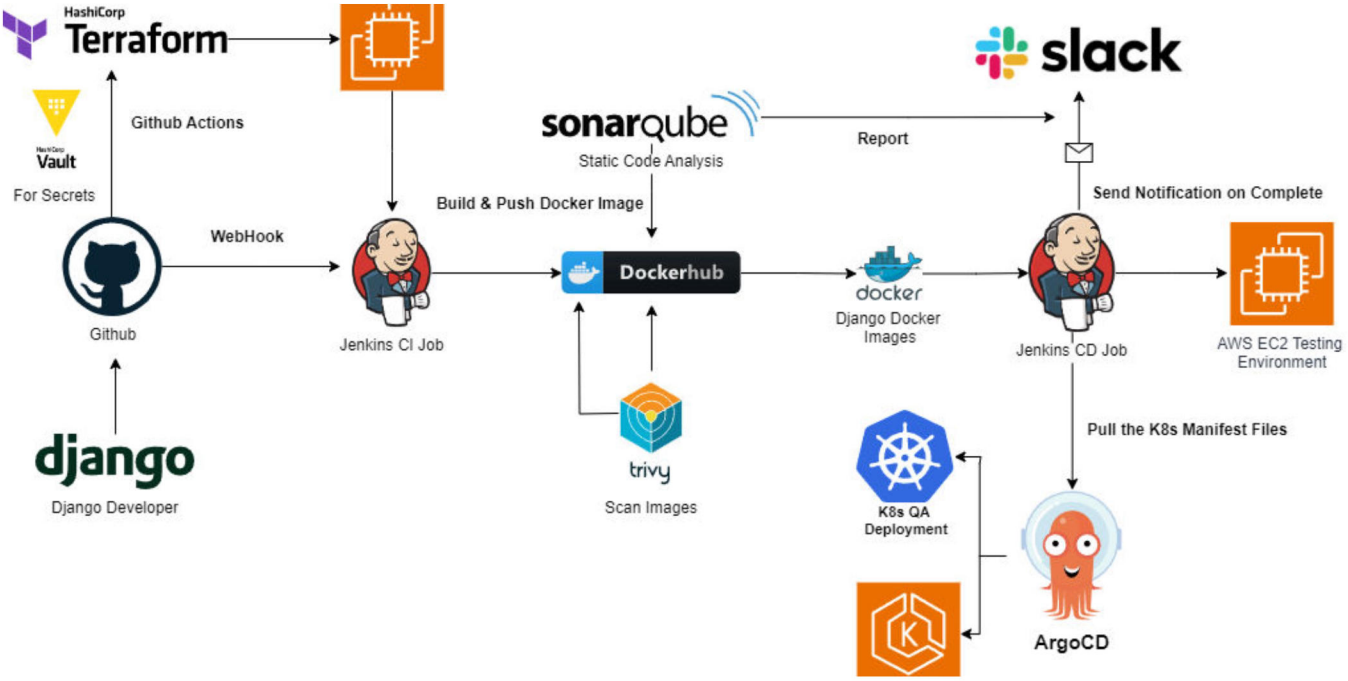
Nov 10, 2024 58

Lists

General Coding Knowledge
20 stories · 1918 saves

Natural Language Processing
1943 stories · 1597 saves

Productivity
244 stories · 685 saves

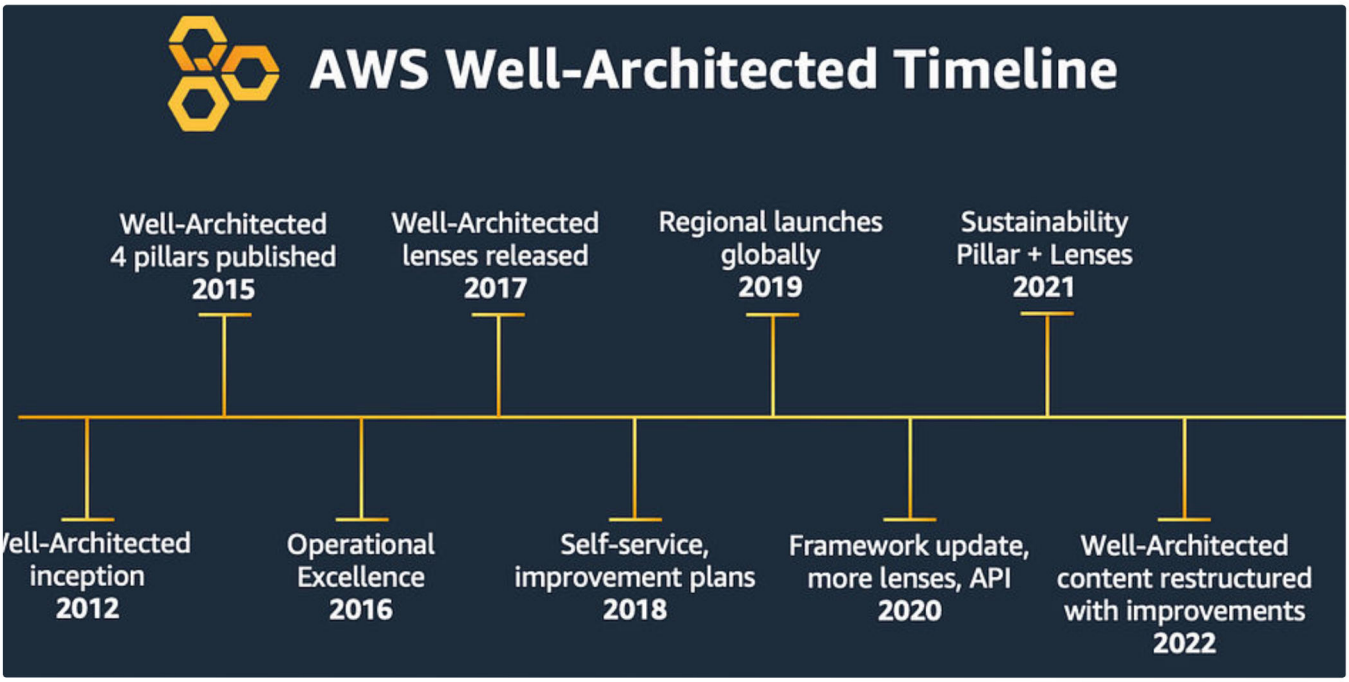


In Django Unleashed by Joel Wembo

Technical Guide: End-to-End CI/CD DevOps with Jenkins, Docker, Kubernetes, ArgoCD, Github Actions …

Building an end-to-end CI/CD pipeline for Django applications using Jenkins, Docker, Kubernetes, ArgoCD, AWS EKS, AWS EC2

Apr 12, 2024 1.1K 22

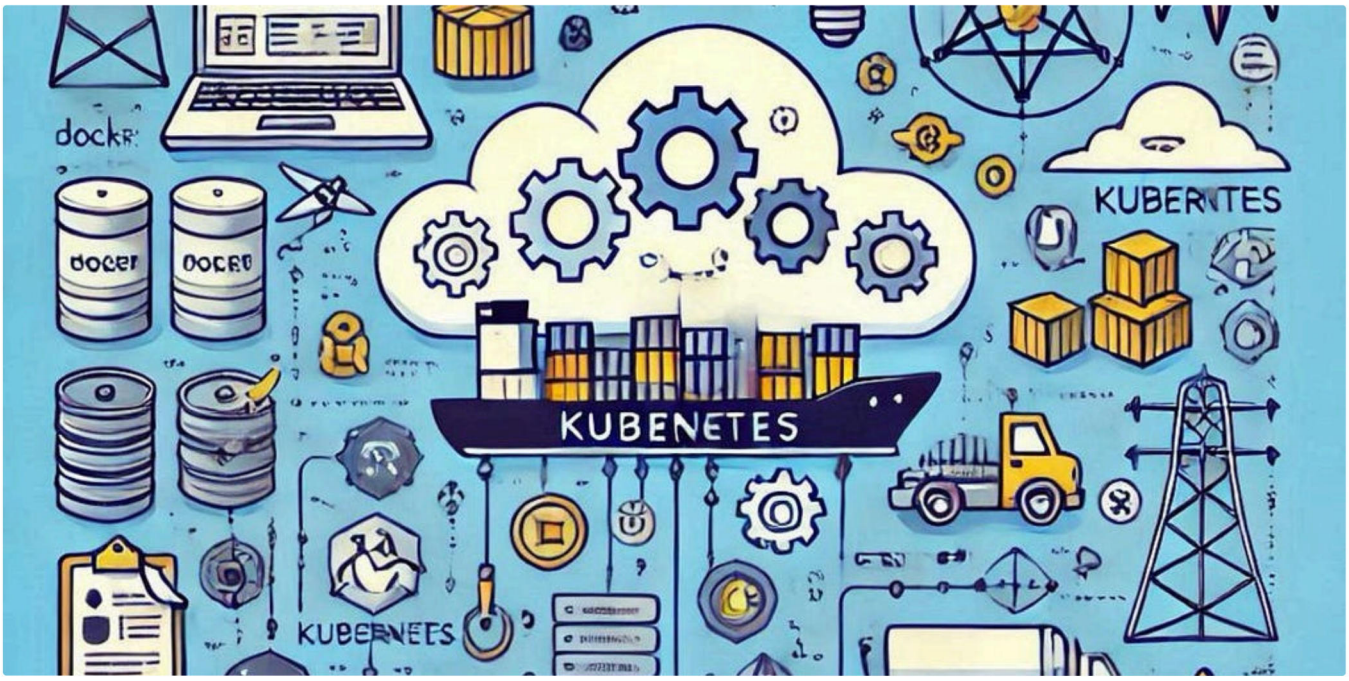


In MyCloudSeries by Ewere Diagboya

AWS Well-Architected EKS (Kubernetes) Cluster

Well Architected Framework for Kubernetes Workload

Dec 17, 2024 82



In DevPulse by Rsprasangi

Docker & Kubernetes: Basic To Advanced Interview Questions!

Have you ever been in a tech interview where you're asked about Docker or Kubernetes, and the questions feel like a test of your patience...

Dec 12, 2024 1

In Stackademic by Aman Pathak

How to Containerize and Deploy a Three-Tier Application on AWS EKS with Kubernetes

Introduction

Aug 20, 2024 146 5