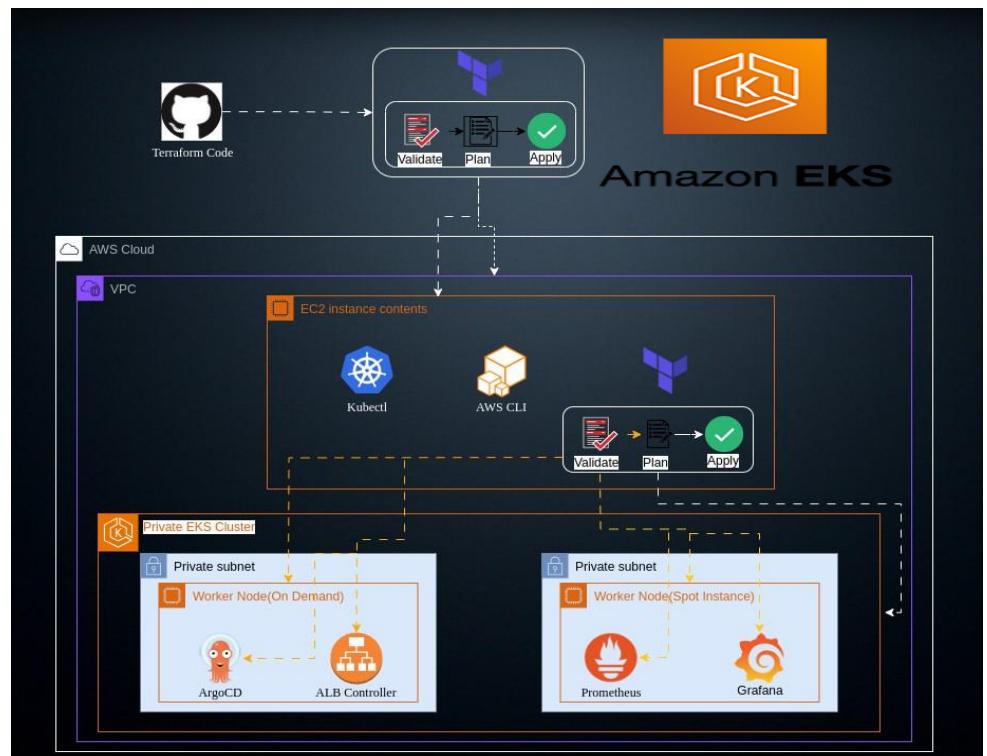


**Configure ArgoCD,  
Prometheus, Grafana &**

# AWS Load Balancer Con'l'oller on EKS Clus'er using Terraform



## 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.

---

## Demonstration

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.

```

module.eks.aws_eks_addon.eks-addons["3"]: Still creating... [28s elapsed]
module.eks.aws_eks_addon.eks-addons["0"]: Still creating... [30s elapsed]
module.eks.aws_eks_addon.eks-addons["3"]: Still creating... [30s elapsed]
module.eks.aws_eks_addon.eks-addons["2"]: Still creating... [30s elapsed]
module.eks.aws_eks_addon.eks-addons["3"]: Still creating... [40s elapsed]
module.eks.aws_eks_addon.eks-addons["0"]: Still creating... [40s elapsed]
module.eks.aws_eks_addon.eks-addons["2"]: Still creating... [40s elapsed]
module.eks.aws_eks_addon.eks-addons["2"]: Creation complete after 48s [id=dev-medium-eks-cluster:kube-proxy]
module.eks.aws_eks_addon.eks-addons["0"]: Still creating... [50s elapsed]
module.eks.aws_eks_addon.eks-addons["3"]: Still creating... [50s elapsed]
module.eks.aws_eks_addon.eks-addons["0"]: Creation complete after 51s [id=dev-medium-eks-cluster:vpc-cni]
module.eks.aws_eks_addon.eks-addons["3"]: Creation complete after 59s [id=dev-medium-eks-cluster:aws-ebs-csi-driver]
data.aws_eks_cluster.eks-cluster: Reading...
data.aws_eks_cluster.eks-cluster: Read complete after 0s [id=dev-medium-eks-cluster]
kubernetes_namespace.argocd: Creating...
kubernetes_service_account.example: Creating...
kubernetes_service_account.example: Still creating... [10s elapsed]
kubernetes_namespace.argocd: Still creating... [10s elapsed]
kubernetes_namespace.argocd: Still creating... [20s elapsed]
kubernetes_service_account.example: Still creating... [20s elapsed]
kubernetes_service_account.example: Still creating... [30s elapsed]
kubernetes_namespace.argocd: Still creating... [30s elapsed]

Error: Post "https://16D6FBF9A7E7FC3417B0E71A310D655C.gr7.us-east-1.eks.amazonaws.com/api/v1/namespaces": dial tcp [64:ff9b::a10:8dfd]:443: i/o timeout

with kubernetes_namespace.argocd,
on argocd.tf line 1, in resource "kubernetes_namespace" "argocd":
 1: resource "kubernetes_namespace" "argocd" {
```

```

Error: Post "https://16D6FBF9A7E7FC3417B0E71A310D655C.gr7.us-east-1.eks.amazonaws.com/api/v1/namespaces/default/serviceaccounts": context deadline exceeded

with kubernetes_service_account.example,
on kubernetes.tf line 70, in resource "kubernetes_service_account" "example":
 70: resource "kubernetes_service_account" "example" {
```

```

Releasing state lock. This may take a few moments...
aman-pathak@aman:~/Projects/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$ git branch
* issue-branch
  master
```

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

```

module.eks.aws_eks_node_group.on-demand-node: Still destroying... [id=dev-medium-eks-cluster:dev-medium-eks-cluster-on-demand-nodes, 2m30s elapsed]
module.eks.aws_eks_node_group.spot-node: Destruction complete after 2m31s
module.eks.aws_eks_node_group.on-demand-node: Still destroying... [id=dev-medium-eks-cluster:dev-medium-eks-cluster-on-demand-nodes, 2m40s elapsed]
module.eks.aws_eks_node_group.on-demand-node: Destruction complete after 2m42s
module.eks.aws_iam_role.eks-nodegroup-role[0]: Destroying... [id=dev-medium-eks-cluster-nodegroup-role-2377]
module.eks.aws_eks_cluster.eks[0]: Destroying... [id=dev-medium-eks-cluster]
module.eks.aws_eks_cluster.eks[0]: Destruction complete after 3s
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 10s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 20s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 30s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 40s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 50s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 1m0s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 1m10s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 1m20s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 1m30s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 1m40s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 1m50s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 2m0s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 2m10s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 2m20s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 2m30s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 2m40s elapsed]
module.eks.aws_eks_cluster.eks[0]: Still destroying... [id=dev-medium-eks-cluster, 2m50s elapsed]
Releasing state lock. This may take a few moments...

```

Destroy complete! Resources: 40 destroyed.

## How to solve this?

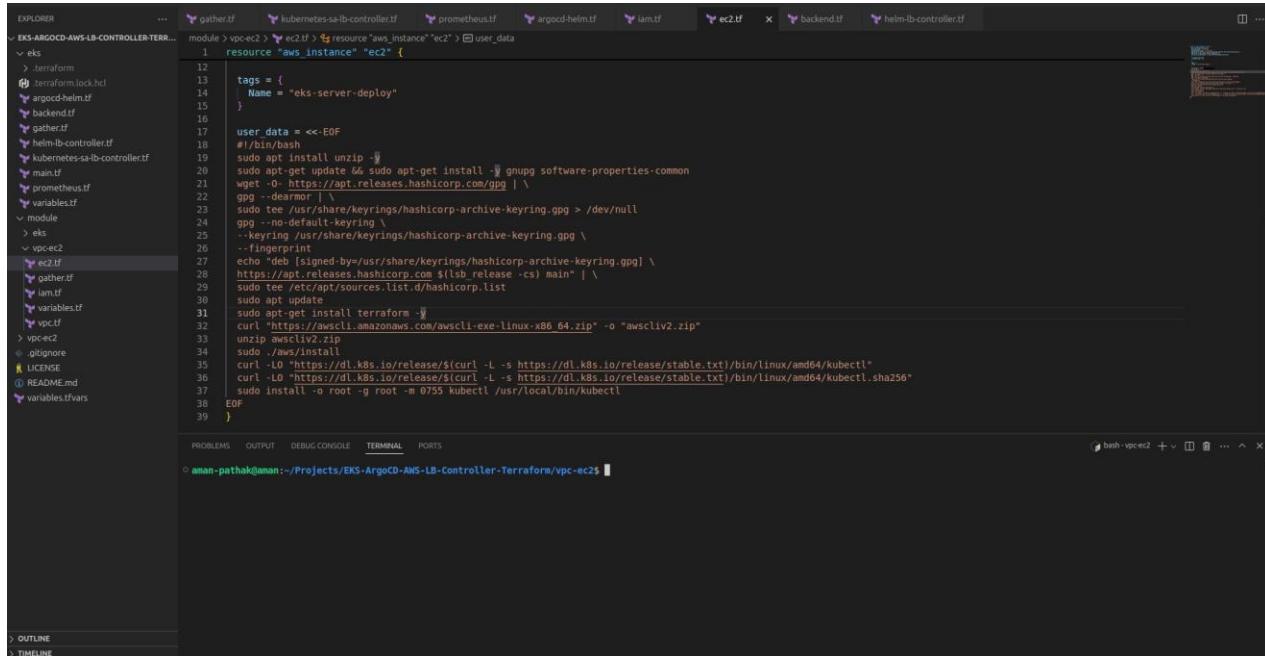
To deploy anything inside your EKS Cluster, we need an instance or server to deploy those configurations that have the same VPC.

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.

File	Description	Time Ago
eks	Updated naming	1 hour ago
module	Updated policies for ec2	16 minutes ago
vpc-ec2	File Structure updated	now
.gitignore	Initial Commit	19 hours ago
LICENSE	Initial commit	yesterday
README.md	Initial commit	yesterday
variables.tfvars	Added minor changes	7 hours ago

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



```
resource "aws_lambda_function" "lambda" {
    function_name = "lambda-function"
    runtime       = "nodejs14.x"
    handler      = "index.handler"
    role          = aws_iam_role.lambda_function.arn
    code          = base64encode(filebase64("lambda_function.zip"))
    timeout      = 30
}

resource "aws_lambda_layer_version" "nodejs14.x" {
    layer_name = "nodejs14.x"
    function   = aws_lambda_function.lambda.function_name
    code       = base64encode(filebase64("nodejs14.x.zip"))
}
```

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

> **terraform init**

```
amans-patnayak@amans-patnayak:~/Projects/EKS-ArgoCD-AWS-LB-Controller-Terraform/vpc-ec2$ terraform init
Initializing the backend...
Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing modules...
- vpc-ec2 in ./module/vpc-ec2
Initializing provider plugins...
- Finding hashicorp/aws versions matching "> 5.49.0"...
- Finding hashicorp/kubernetes versions matching "2.31.0"...
- Installing hashicorp/kubernetes v2.31.0...
- Installed hashicorp/kubernetes v2.31.0 (signed by HashiCorp)
- Installing hashicorp/aws v5.49.0...
- Installed hashicorp/aws v5.49.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

> **terraform validate**

```
▶ aman-pathak@aman:~/Projects/EKS-ArgoCD-AWS-LB-Controller-Terraform/vpc-ec2$ terraform validate
Success! The configuration is valid.
```

```
○ aman-pathak@aman:~/Projects/EKS-ArgoCD-AWS-LB-Controller-Terraform/vpc-ec2$
```

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ enable_dns_support = true
+ enable_network_address_usage_metrics = (known after apply)
+ id = (known after apply)
+ instance_tenancy = "default"
+ ipv6_association_id = (known after apply)
+ ipv6_cidr_block = (known after apply)
+ ipv6_cidr_block_network_border_group = (known after apply)
+ main_route_table_id = (known after apply)
+ owner_id = (known after apply)
+ tags = {
    + "Env" = "dev"
    + "Name" = "dev-medium-vpc"
}
+ tags_all = {
    + "Env" = "dev"
    + "Name" = "dev-medium-vpc"
}
}

Plan: 24 to add, 0 to change, 0 to destroy.

Warning: Value for undeclared variable
The root module does not declare a variable named "ondemand_instance_types" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.

To silence these warnings, use TF_VAR_... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Value for undeclared variable
The root module does not declare a variable named "max_capacity_on_demand" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.

To silence these warnings, use TF_VAR_... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Values for undeclared variables
In addition to the other similar warnings shown, 11 other variable(s) defined without being declared.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
Releasing state lock. This may take a few moments...
aman-pathak@aman:~/Projects/EKS-ArgoCD-AWS-LB-Controller-Terraform/vpc-ec2$
```

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

```

TERMINAL PORTS
bash-vpc-ec2 + 
module.vpc-ec2.aws_instance.ec2: Still creating... [10s elapsed]
module.vpc-ec2.aws_nat_gateway.ngw: Still creating... [20s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [20s elapsed]
module.vpc-ec2.aws_nat_gateway.ngw: Still creating... [30s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [30s elapsed]
module.vpc-ec2.aws_instance.ec2: Creation complete after 37s [id=1-0d8a3bc509812b65b]
module.vpc-ec2.aws_nat_gateway.ngw: Still creating... [40s elapsed]
module.vpc-ec2.aws_nat_gateway.ngw: Still creating... [50s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [50s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [1m0s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [1m10s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [1m20s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [1m30s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [1m40s elapsed]
module.vpc-ec2.aws_instance.ec2: Still creating... [1m50s elapsed]
module.vpc-ec2.aws_instance.ec2: Creation complete after 1m50s [id=nat-0e920b64e401eb765]
module.vpc-ec2.aws_route_table.private-rt: Creating...
module.vpc-ec2.aws_route_table.private-rt: Creation complete after 3s [id=rtb-0b3db5ef7d41bab3]
module.vpc-ec2.aws_route_table_association.private-rt-association[1]: Creating...
module.vpc-ec2.aws_route_table_association.private-rt-association[0]: Creating...
module.vpc-ec2.aws_route_table_association.private-rt-association[2]: Creating...
module.vpc-ec2.aws_route_table_association.private-rt-association[0]: Creation complete after 2s [id=rtbassoc-0a46e70188fb044c]
module.vpc-ec2.aws_route_table_association.private-rt-association[2]: Creation complete after 2s [id=rtbassoc-0d5e5f7a0dd6803]
module.vpc-ec2.aws_route_table_association.private-rt-association[1]: Creation complete after 2s [id=rtbassoc-00599d70a94fb8fc]

Warning: Value for undeclared variable
The root module does not declare a variable named "max_capacity_on_demand" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.

To silence these warnings, use TF_VAR_... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Value for undeclared variable
The root module does not declare a variable named "min_capacity_spot" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.

To silence these warnings, use TF_VAR_... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Values for undeclared variables
In addition to the other similar warnings shown, 11 other variable(s) defined without being declared.

Releasing state lock. This may take a few moments...
Apply complete! Resources: 24 added, 0 changed, 0 destroyed.
amans-patnak@aman:~/Projects/EKS-ArgoCD-AWS-LB-Controller-Terraform/vpc-ec2$ 

```

## VPC & other services created through the above commands

Here are the snippets

### VPC

Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR	DHCP option set	Main route table
dev-medium-vpc	vpc-053477193e826f6c2	Available	10.16.0.0/16	-	-	rtb-03a210971559af011

### Subnets

Name	Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	IPv6 CIDR association ID
dev-medium-subnet-public-3	subnet-0529f20336b2ded3c	Available	vpc-053477193e826f6c2	10.16.32.0/20	-	-
dev-medium-subnet-public-1	subnet-00cb711dddc8c3bc	Available	vpc-053477193e826f6c2	10.16.0.0/20	-	-
dev-medium-subnet-private-2	subnet-00a790a960255278	Available	vpc-053477193e826f6c2	10.16.144.0/20	-	-
dev-medium-subnet-private-1	subnet-05676b1f163ea205e	Available	vpc-053477193e826f6c2	10.16.128.0/20	-	-
dev-medium-subnet-private-3	subnet-007d9f2dfdf45d05	Available	vpc-053477193e826f6c2	10.16.160.0/20	-	-
dev-medium-subnet-public-2	subnet-0h600ba0303-3a40f	Available	vpc-053477193e826f6c2	10.16.16.0/20	-	-

### Route tables

Name	Route table ID	Explicit subnet associations	Edge associations	Main	VPC	Owner ID
dev-medium-public-route-table	rtb-06ff991accf5c1e100	3 subnets	-	No	vpc-053477193e826f6c2	40762020962
dev-medium-private-route-table	rtb-0b50b5fa7441bub3	3 subnets	-	No	vpc-053477193e826f6c2	40762020962
-	rtb-03a210971559af011	-	-	Yes	vpc-053477193e826f6c2	40762020962

## Internet Gateway

The screenshot shows the AWS VPC dashboard with the 'Internet gateways' section selected. It displays two internet gateways: 'igw-039c0082b1f4d2aa5' and 'igw-001e70f5db0dax634'. Both are attached to the 'dev-medium-vpc' VPC and are in an 'Attached' state.

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-039c0082b1f4d2aa5	Attached	vpc-0a40feba3b5b39630	407622020962
dev-medium-igw	igw-001e70f5db0dax634	Attached	vpc-053477193e026f6c2   dev-medium...	407622020962

## NAT Gateway

The screenshot shows the AWS VPC dashboard with the 'NAT gateways' section selected. It displays one NAT gateway: 'nat-0e920b6de401eb765'. It is in a 'Public' connectivity state and is available.

Name	NAT gateway ID	Connectivity...	State	Primary public I...	Primary private I...	Primary network...	VPC
dev-medium-ngw	nat-0e920b6de401eb765	Public	Available	5.139.11.167	10.16.8.110	eni-09e795a350611...	vpc-053477193e026f6c2

## Security Group

The screenshot shows the AWS EC2 Global View with the 'Security Groups' section selected. It displays three security groups: 'eks-sg' (sg-02710bf4b1928202e), 'default' (sg-0d644b1be403e5a0d), and another 'eks-sg' (sg-0d119a4552dbcafc9f). The first 'eks-sg' allows port 443 from the jump server.

Name	Security group ID	Security group name	VPC ID	Description	Owner
eks-sg	sg-02710bf4b1928202e	eks-sg	vpc-053477193e026f6c2	Allow 443 from Jump Server only	407622020962
-	sg-0d644b1be403e5a0d	default	vpc-053477193e026f6c2	default VPC security group	407622020962
eks-sg	sg-0d119a4552dbcafc9f	eks-sg	vpc-053477193e026f6c2	Allow 443 from Jump Server only	407622020962

## EC2-Instance

The screenshot shows the AWS EC2 Dashboard with the 'Instances' section selected. It displays one instance: 'eks-server-deploy' (i-068a3bc509812b65b). The instance is running and has a Public IPv4 address of 18.191.116.22.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
eks-server-deploy	i-068a3bc509812b65b	Running	t2.micro	2/2 checks passed	...	us-east-2a	ec2-18-191-116-22.us...	18.191.116.22	-

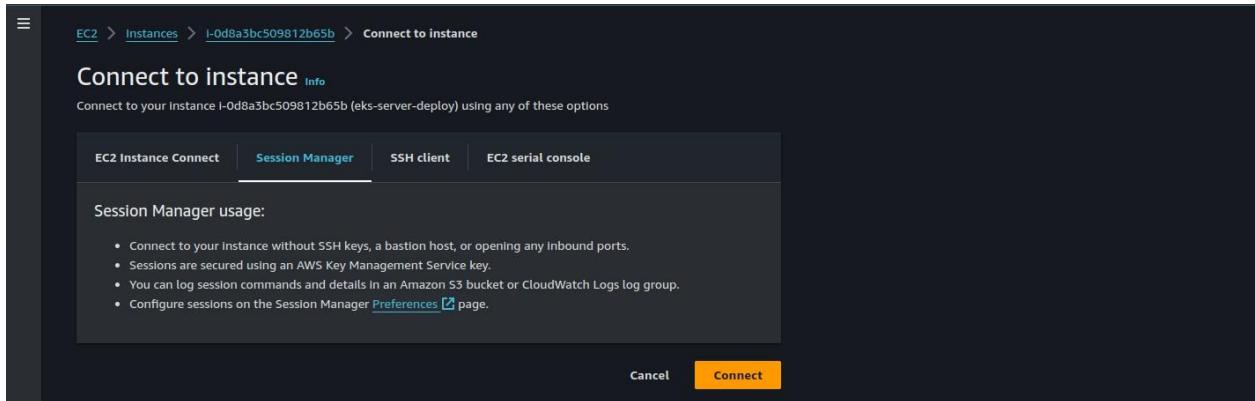
So, everything is created according to our requirements.

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
```

```
$ sudo su ubuntu
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-10-10-10-26:/var/snap/amazon-ssm-agent/7993$ cd
```

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
```

```
ubuntu@ip-10-10-10-26:~$ terraform version
Terraform v1.9.5
on linux_amd64
ubuntu@ip-10-10-10-26:~$ aws --version
aws-cli/2.3.2 Python/3.11.0 Linux/4.5.0-1024-aws exe/x86_64/ubuntu.22
ubuntu@ip-10-10-10-26:~$ kubectl version
Client Version: v1.31.0
Kustomize Version: v5.4.2
The connection to the server localhost:8080 was refused - did you specify the right host or port?
ubuntu@ip-10-10-10-26:~$
```

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 keys in your AWS Account

```
> aws configure
```

```
Session ID: 7edc4e6e-26e6-4e7f-93d9-0495e607fbef104  
Instance ID: i-0d8a1bc5d98121603  
Terminate  
  
ubuntu@ip-10-16-10-26:~$ aws configure  
AWS Access Key ID [None]: AKIAV52BKNSF2VG5R4H  
AWS Secret Access Key [None]: ez58Pn3rwKhtAMpwB0i/epCunsoomwqQZfLfghk  
Default region name [None]: us-east-2  
Default output format [None]: json  
ubuntu@ip-10-16-10-26:~$
```

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-Terraform.git
```

```
Session ID: 7edc4e6e-26e6-4e7f-93d9-0495e607fbef104  
Instance ID: i-0d8a1bc5d98121603  
Terminate  
  
ubuntu@ip-10-16-10-26:~$ git clone https://github.com/AmanPathak-DevOps/EKS-ArgoCD-AWS-LB-Controller-Terraform.git  
Cloning into 'EKS-ArgoCD-AWS-LB-Controller-Terraform'...  
Username for 'https://github.com': AmanPathak-DevOps  
Password for 'https://AmanPathak-DevOps@github.com':  
remote: Enumerating objects: 85, done.  
remote: Counting objects: 100% (85/85), done.  
remote: Compressing objects: 100% (66/66), done.  
remote: Total 85 (delta 40), reused 64 (delta 23), pack-reused 0 (from 0)  
Receiving objects: 100% (85/85), 19.01 KiB | 1.73 MiB/s, done.  
Resolving deltas: 100% (49/49), done.  
ubuntu@ip-10-16-10-26:~$ ls  
EKS-ArgoCD-AWS-LB-Controller-Terraform  
ubuntu@ip-10-16-10-26:~$
```

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
```

```
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$ terraform init  
Initializing the backend...  
  
Successfully configured the backend "s3"! Terraform will automatically  
use this backend unless the backend configuration changes.  
Initializing modules...  
- eks in ./module/eks  
Initializing provider plugins...  
- Finding latest version of hashicorp/tls...  
- Finding hashicorp/helm versions matching "> 2.10.0"...  
- Finding hashicorp/aws versions matching "> 5.49.0"...  
- Finding hashicorp/kubernetes versions matching "2.31.0"...  
- Finding latest version of hashicorp/random...  
- Installing hashicorp/random v3.6.2...  
- Installed hashicorp/random v3.6.2 (signed by HashiCorp)  
- Installing hashicorp/tls v4.0.5...  
- Installed hashicorp/tls v4.0.5 (signed by HashiCorp)  
- Installing hashicorp/helm v2.10.1...  
- Installed hashicorp/helm v2.10.1 (signed by HashiCorp)  
- Installing hashicorp/aws v5.49.0...  
- Installed hashicorp/aws v5.49.0 (signed by HashiCorp)  
- Installing hashicorp/kubernetes v2.31.0...  
- Installed hashicorp/kubernetes v2.31.0 (signed by HashiCorp)  
Terraform has created a lock file .terraform.lock.hcl to record the provider  
selections it made above. Include this file in your version control repository  
so that Terraform can guarantee to make the same selections by default when  
you run "terraform init" in the future.  
  
Terraform has been successfully initialized!  
  
You may now begin working with Terraform. Try running "terraform plan" to see  
any changes that are required for your infrastructure. All Terraform commands  
should now work.  
  
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.  
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$
```

```
> terraform validate
```

```
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$ terraform validate  
Success! The configuration is valid.  
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$
```

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

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

```
Session ID: Adorable-Amen-estj7uehglagenutb7hf3t8k
Instance ID: i-0d8a3bc5098f12b65b
Terminate

}

# module.eks.aws_iam_role_policy_attachment.eks AMAZONWorkerNodePolicy[0] will be created
+ resource "aws_iam_role_policy_attachment" "eks-AmazonWorkerNodePolicy" {
  + policy_arn = "arn:aws:iam::policy/AmazonEKSWorkerNodePolicy"
  + role       = (known after apply)
}

# module.eks.aws_iam_role_policy_attachment.eks-oidc-policy-attach will be created
+ resource "aws_iam_role_policy_attachment" "eks-oidc-policy-attach" {
  + id        = (known after apply)
  + policy_arn = (known after apply)
  + role       = "eks-oidc"
}

# module.eks.random_integer.random_suffix will be created
+ resource "random_integer" "random_suffix" {
  + id      = (known after apply)
  + max    = 1000
  + min    = 1000
  + result = (known after apply)
}

Plan: 25 to add, 0 to change, 0 to destroy.

Warning: Value for undeclared variable
The root module does not declare a variable named "ec2-sg" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Value for undeclared variable
The root module does not declare a variable named "ec2-name" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Values for undeclared variables
In addition to the other similar warnings shown, 3 other variable(s) defined without being declared.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$
```

```
Session ID: Adorable-Amen-estj7uehglagenutb7hf3t8k
Instance ID: i-0d8a3bc5098f12b65b
Terminate

module.eks.aws_eks.addon.eks-addons["2"]: Still creating... [50s elapsed]
module.eks.aws_eks.addon.eks-addons["0"]: Still creating... [50s elapsed]
module.eks.aws_eks.addon.eks-addons["3"]: Still creating... [50s elapsed]
module.eks.aws_eks.addon.eks-addons["1"]": Still creating... [50s elapsed]
[...]
module.release.aws.load_balancer_controller: Creation complete after 55s [id=dev-medium-eks-cluster:aws-ebs-csi-driver]
module.eks.aws_eks.addon.eks-addons["2"]": Creation complete after 55s [id=dev-medium-eks-cluster:kube-proxy]
module.eks.aws_eks.addon.eks-addons["0"]": Creation complete after 55s [id=dev-medium-eks-cluster:vpc-cni]
data.aws_eks_cluster.eks-cs-cluster: Reading...
data.aws_eks_cluster.eks-cs-cluster: Read complete after 0s [id=dev-medium-eks-cluster]
kubernetes service account.lb-controller: Creating...
kubernetes service account.lb-controller: Creation complete after 0s [id=default/aws-load-balancer-controller]
helm.release.aws.load_balancer_controller: Creating...
helm.release.aws.load_balancer_controller: Still creating... [10s elapsed]
helm.release.aws.load_balancer_controller: Creation complete after 10s [id=aws-load-balancer-controller]
helm.release.argo: Creating...
helm.release.argo: Still creating... [10s elapsed]
helm.release.argo: Still creating... [20s elapsed]
helm.release.argo: Still creating... [30s elapsed]
helm.release.argo: Creation complete after 39s [id=argocd]
helm.release.prometheus-helm: Creating...
helm.release.prometheus-helm: Still creating... [10s elapsed]
helm.release.prometheus-helm: Still creating... [20s elapsed]
helm.release.prometheus-helm: Still creating... [30s elapsed]
helm.release.prometheus-helm: Still creating... [40s elapsed]
helm.release.prometheus-helm: Still creating... [1m0s elapsed]
helm.release.prometheus-helm: Still creating... [1m10s elapsed]
helm.release.prometheus-helm: Creation complete after 1m10s [id=prometheus]

Warning: Value for undeclared variable
The root module does not declare a variable named "ec2-name" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Value for undeclared variable
The root module does not declare a variable named "ec2-sg" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Values for undeclared variables
In addition to the other similar warnings shown, 3 other variable(s) defined without being declared.

Apply complete! Resources: 25 added, 0 changed, 0 destroyed.
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$
```

**EKS Cluster & other services created through the above commands  
Here are the snippets**

## EKS Cluster

Clusters (1) <a href="#">Info</a>							
		Status	Kubernetes version	Support period	Upgrade policy	Created	Provider
<a href="#">dev-medium-eks-cluster</a>	<span>Active</span>	1.29	<a href="#">Upgrade now</a>	<span>Standard support until March 23, 2025</span>	Extended	16 minutes ago	EKS

## Node Groups

dev-medium-eks-cluster							
<a href="#">Cluster info</a> <a href="#">Info</a>							
Status	Kubernetes version	Support period	Provider	Overview	Resources	Compute	Networking
<span>Active</span>	1.29	<span>Standard support until March 23, 2025</span>	EKS				
<a href="#">Nodes (2) Info</a>							
<a href="#">Filter Nodes by property or value</a>							
Node name	Instance type	Node group	Created	Status			
<a href="#">ip-10-16-147-226.us-east-2.compute.internal</a>	t3a.medium	<a href="#">dev-medium-eks-cluster-on-demand-nodes</a>	Created <span>4 minutes ago</span>	<span>Ready</span>			
<a href="#">ip-10-16-157-126.us-east-2.compute.internal</a>	t3a.large	<a href="#">dev-medium-eks-cluster-spot-nodes</a>	Created <span>4 minutes ago</span>	<span>Ready</span>			
<a href="#">Node groups (2) Info</a>							
<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Add node group</a>							

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
```

Session ID: Adorable-Arcan-estp2wuhlgpmwth7uf3t8l	Instance ID: i-0d6fbcb50981256f3b	<a href="#">Terminate</a>
<code>ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks\$ aws eks update-kubeconfig --name dev-medium-eks-cluster --region us-east-2 Added new context arn:aws:eks:us-east-2:a07622920962:cluster/dev-medium-eks-cluster to /home/ubuntu/.kube/config ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks\$</code>		

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

```
> kubectl get nodes
```

Session ID: Adorable-Arcan-estp2wuhlgpmwth7uf3t8l	Instance ID: i-0d6fbcb50981256f3b	<a href="#">Terminate</a>
<code>ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks\$ kubectl get nodes NAME           STATUS   ROLES    AGE     VERSION ip-10-16-147-226.us-east-2.compute.internal   Ready   &lt;none&gt;   5m36s   v1.29.0-eks-1552ad0 ip-10-16-157-126.us-east-2.compute.internal     Ready   &lt;none&gt;   5m33s   v1.29.0-eks-1552ad0 ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks\$</code>		

Check whether the resources are running in argocd namespace or not

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

```

ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$ kubectl get all -n argocd
NAME                                         READY   STATUS    RESTARTS   AGE
pod/argocd-application-controller-0           1/1    Running   0          4m32s
pod/argocd-applicationset-controller-765cfb7674-dt9ct 1/1    Running   0          4m33s
pod/argocd-dex-server-6db8cbf4db-ac8ct      1/1    Running   0          4m33s
pod/argocd-notifications-controller-7df6fd8fd4-wzsjs 1/1    Running   0          4m33s
pod/argocd-redis-57bd78b7d-sq2wx            1/1    Running   0          4m33s
pod/argocd-repo-server-847666c5b-cbcfc       1/1    Running   0          4m33s
pod/argocd-server-84db774fb-b35m              1/1    Running   0          4m33s

NAME                                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/argocd-applicationset-controller ClusterIP  172.28.173.9 <none>        7089/TCP        4m33s
service/argocd-dex-server     ClusterIP  172.28.173.9 <none>        5358/TCP, 5557/TCP 4m33s
service/argocd-redis          ClusterIP  172.28.148.102 <none>       6379/TCP        4m33s
service/argocd-repo-server    ClusterIP  172.28.198.123 <none>       8081/TCP        4m33s
service/argocd-server         LoadBalancer 172.28.232.82  as360abfb5299146599274c90b23d69be-165956850.us-east-2.elb.amazonaws.com  80:31297/TCP, 443:31884/TCP 4m33s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/argocd-applicationset-controller 1/1    1           1           4m33s
deployment.apps/argocd-dex-server     1/1    1           1           4m33s
deployment.apps/argocd-notifications-controller 1/1    1           1           4m33s
deployment.apps/argocd-redis          1/1    1           1           4m33s
deployment.apps/argocd-repo-server    1/1    1           1           4m33s
deployment.apps/argocd-server         1/1    1           1           4m33s

NAME                               DESIRED   CURRENT   READY   AGE
replicaset.apps/argocd-applicationset-controller-765cfb7674 1         1         1         4m33s
replicaset.apps/argocd-dex-server-6db8cbf4db      1         1         1         4m33s
replicaset.apps/argocd-notifications-controller-7df6fd8fd4 1         1         1         4m33s
replicaset.apps/argocd-redis-57bd78b7d             1         1         1         4m33s
replicaset.apps/argocd-repo-server-847666c5b       1         1         1         4m33s
replicaset.apps/argocd-server-84db774fb             1         1         1         4m33s

NAME                               READY   AGE
statefulset.apps/argocd-application-controller-0           1/1    4m33s

```

Check whether the resources are running in Prometheus namespace or not

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

```

ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$ kubectl get all -n prometheus
NAME                                         READY   STATUS    RESTARTS   AGE
pod/alertmanager-prometheus-alertmanager-0   2/2     Running   0          4m50s
pod/prometheus-grafana-6cc44984fc-9p4fd      3/3     Running   0          4m52s
pod/prometheus-kube-prometheus-operator       1/1     Running   0          4m52s
pod/prometheus-kube-state-metrics-5b787f976b-59jz4 1/1     Running   0          4m52s
pod/prometheus-kube-prometheus-prometheus-0   2/2     Running   0          4m52s
pod/prometheus-kube-prometheus-node-exporter-0ngj7 1/1     Running   0          4m52s
pod/prometheus-kube-prometheus-node-exporter-4dr4g 1/1     Running   0          4m52s

NAME                                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
service/alertmanager-operated      ClusterIP  <none>        <none>        9092/TCP, 9094/TCP, 9094/UDP   4m53s
service/prometheus-grafana        LoadBalancer 172.28.108.21  a34537fbca3543d490e55622e7afc64-89423726.us-east-2.elb.amazonaws.com  80, 8024/TCP   4m53s
service/prometheus-kube-prometheus-alertmanager  ClusterIP  172.28.105.191  <none>          8093/TCP, 8088/TCP   4m53s
service/prometheus-kube-prometheus-operator    ClusterIP  172.28.10.255  <none>          443/TCP           4m53s
service/prometheus-kube-prometheus-prometheus  LoadBalancer 172.28.221.37  abd7eeb944f274e5d82899d7489e175-1643275437.us-east-2.elb.amazonaws.com  9090:31872/TCP, 8088:32243/TCP   4m53s
service/prometheus-kube-state-metrics      ClusterIP  172.28.131.125  <none>          8088/TCP           4m53s
service/prometheus-operated          ClusterIP  None           <none>          9090/TCP           4m49s
service/prometheus-kube-prometheus-node-exporter  ClusterIP  172.28.109.114  <none>          9109/TCP           4m53s

NAME              DESIRED   CURRENT  READY   UP-TO-DATE   AVAILABLE   AGE
daemonset.apps/prometheus-prometheus-node-exporter  2          2        2       2           2          2          kubernetes.io/os=linux   4m53s

NAME                                         DESIRED   CURRENT  READY   AGE
deployment.apps/prometheus-grafana            1/1      1        1       4m53s
deployment.apps/prometheus-kube-prometheus-operator 1/1      1        1       4m53s
deployment.apps/prometheus-kube-state-metrics   1/1      1        1       4m53s

NAME                                         DESIRED   CURRENT  READY   AGE
replicaset.apps/prometheus-grafana-6cc44984fc  1          1        1       4m53s
replicaset.apps/prometheus-kube-prometheus-operator-76647f58db  1          1        1       4m53s
replicaset.apps/prometheus-kube-state-metrics-5b787f976b  1          1        1       4m53s

NAME                                         READY   AGE
statefulset.apps/alertmanager-prometheus-kube-prometheus-alertmanager  1/1     4m50s
statefulset.apps/prometheus-kube-prometheus-prometheus   1/1     4m49s

ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$
```

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

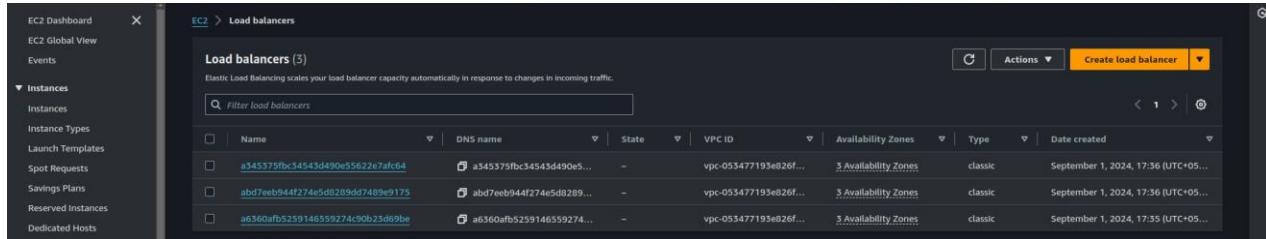
```
> kubectl get all -n aws-loadbalancer-controller
```

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.

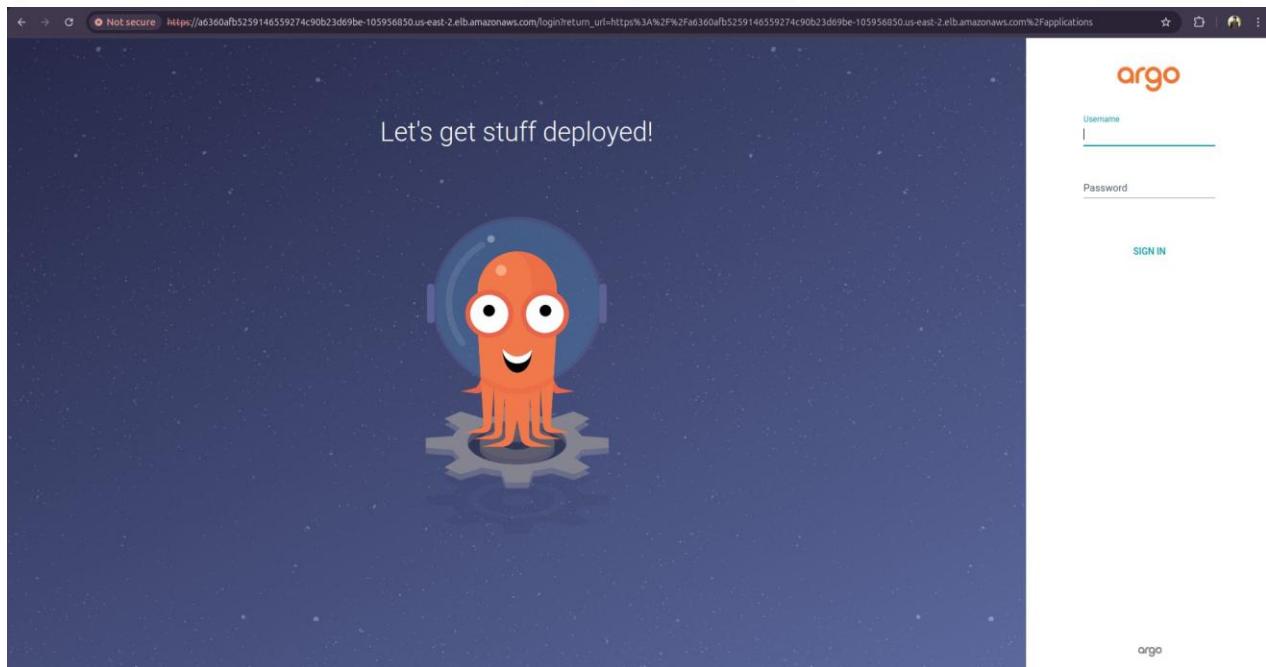
You can check which LB is for which tool through the kubectl command that we ran in the above steps.



The screenshot shows the AWS EC2 Load Balancers page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, and Dedicated Hosts. The main area is titled 'Load balancers (3)' and contains a table with columns: Name, DNS name, State, VPC ID, Availability Zones, Type, and Date created. The table lists three load balancers with their respective details.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
a345375fb34543d490e55622e7afc64	a345375fb34543d490e5...	-	vpc-053477193e026f...	3 Availability Zones	classic	September 1, 2024, 17:36 (UTC+05...)
abd7eeb944f274e5d9289dd7489e9175	abd7eeb944f274e5d9289...	-	vpc-053477193e026f...	3 Availability Zones	classic	September 1, 2024, 17:36 (UTC+05...)
a6360afb5259146559274c90b23d69be	a6360afb5259146559274...	-	vpc-053477193e026f...	3 Availability Zones	classic	September 1, 2024, 17:35 (UTC+05...)

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



The screenshot shows the ArgoCD login page. The URL is https://a6360afb5259146559274c90b23d69be-105956850.us-east-2.elb.amazonaws.com/login?return\_url=https%3A%2F%2Fa6360afb5259146559274c90b23d69be-105956850.us-east-2.elb.amazonaws.com%2FApplications. The page features a large orange cartoon character with a gear, and the text 'Let's get stuff deployed!'. On the right, there's a sign-in form with fields for 'Username' and 'Password', and a 'SIGN IN' button.

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.password}' | base64 -d
```

Session ID: Adorable-Aman-4107ushjblgmvnlt27hd3t8  
 Instance ID: i-0d8a3bc509812b45b  
Terminate

```
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$ kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=".data.password" | base64 -d
F8fbFBxsEJHjwRmUbuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$
```

It is accessible

The screenshot shows the Argo CD web interface. On the left, there's a sidebar with icons for Applications (selected), Settings, User Info, and Documentation. The main area has a large circular icon with three stacked rectangles inside, representing a stack of containers. Below it, the text "No applications available to you just yet" is displayed, followed by the instruction "Create new application to start managing resources in your cluster". A "CREATE APPLICATION" button is at the bottom. The top right corner has "APPLICATIONS TILES" and a "Log out" button.

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

It is accessible

The screenshot shows the Prometheus UI. At the top, there are tabs for Prometheus, Alerts, Graph, Status, and Help. Below that is a "Targets" section with three tables:

- serviceMonitor/prometheus/prometheus-kube-prometheus-alertmanager0 (1/1 up)**

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.16.149.175.9093/metrics	UP	container:="alertmanager" endpoint:="http-web" instance:="10.16.149.175.9093" job:="prometheus-kube-prometheus-alertmanager" namespace:="prometheus" pod:="alertmanager-prometheus-kube-prometheus-alertmanager-0" service:="prometheus-kube-prometheus-alertmanager"	472.000ms ago	4.323ms	
- serviceMonitor/prometheus/prometheus-kube-prometheus-alertmanager/1 (1/1 up)**

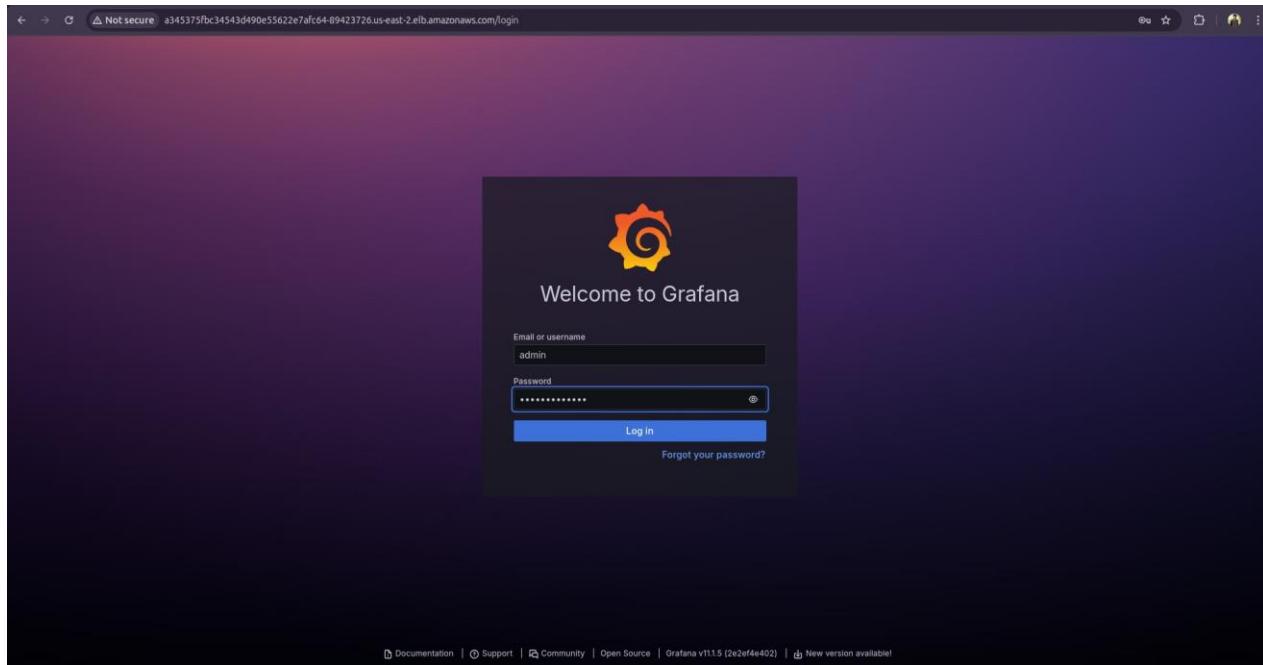
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.16.149.175.8080/metrics	UP	container:="config-reloader" endpoint:="reloader-web" instance:="10.16.149.175.8080" job:="prometheus-kube-prometheus-alertmanager" namespace:="prometheus" pod:="alertmanager-prometheus-kube-prometheus-alertmanager-0" service:="prometheus-kube-prometheus-alertmanager"	7.253s ago	2.192ms	
- serviceMonitor/prometheus/prometheus-kube-prometheus-apiserver/0 (2/2 up)**

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://10.16.168.159/metrics	UP	endpoint:="https" instance:="10.16.168.159:443" job:="apiserver" namespace:="default" service:="kubernetes"	28.538s ago	224.451ms	
https://10.16.132.154/metrics	UP	endpoint:="https" instance:="10.16.132.154:443" job:="apiserver" namespace:="default" service:="kubernetes"	15.16s ago	168.141ms	

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

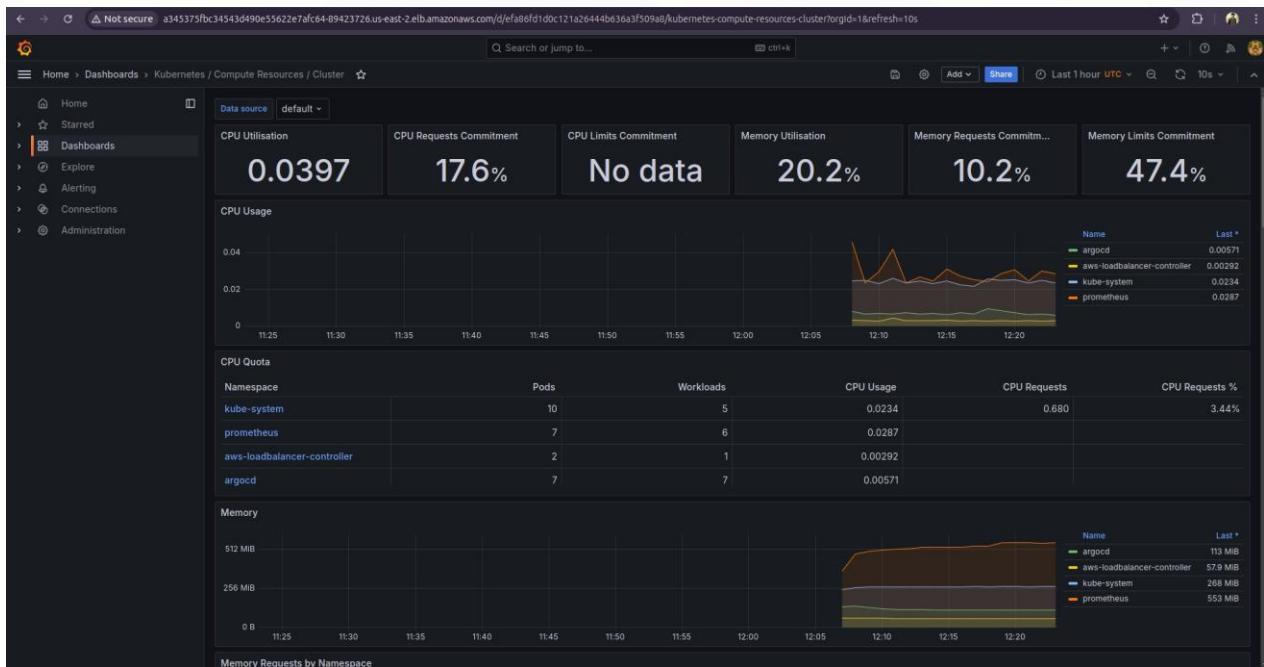
A screenshot of the Grafana dashboard after logging in. The URL in the address bar is 'Not secure a345373fbc34543d490e55622e7afc64-89423726.us-east-2.elb.amazonaws.com/org/d1'. The dashboard has a dark theme. On the left, there is a sidebar with a tree view: Home (selected), Starred, Dashboards, Explore, Alerting, Connections, and Administration. Below this is a 'Basic' section with a 'TUTORIAL DATA SOURCE AND DASHBOARDS' card for 'Grafana fundamentals'. This card includes a brief description and a link to 'Learn how in the docs'. To the right of this are three 'COMPLETE' cards: 'Add your first data source', 'Create your first dashboard', and another 'Create your first dashboard' card. Below the sidebar is a 'Dashboards' section with links for 'Starred dashboards' and 'Recently viewed dashboards'. On the right side, there is a 'Latest from the blog' section with two posts: one about 'Meet us at PromCon' dated Aug 30, and another about 'Visualize CockroachDB in Grafana: Introducing the CockroachDB Enterprise data source' dated Aug 29. The overall layout is clean and organized, providing quick access to essential tools and information.

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

The screenshot shows the Grafana interface with the URL <https://a345375fbc34543d490e55622e7afc64-89423726.us-east-2.elb.amazonaws.com/dashboards>. The left sidebar has 'Dashboards' selected. The main area lists various dashboards with their names and tags:

- Alertmanager / Overview (Tags: alertmanager-mixin)
- CoreDNS (Tags: coredns, dns)
- etc (Tags: etcd-mixin)
- Grafana Overview (Tags: kubernetes-mixin)
- Kubernetes / API server (Tags: kubernetes-mixin)
- Kubernetes / Compute Resources / Multi-Cluster (Tags: kubernetes-mixin)
- Kubernetes / Compute Resources / Cluster (Tags: kubernetes-mixin)
- Kubernetes / Compute Resources / Namespace (Pods) (Tags: kubernetes-mixin)
- Kubernetes / Compute Resources / Namespace (Workloads) (Tags: kubernetes-mixin)
- Kubernetes / Compute Resources / Node (Pods) (Tags: kubernetes-mixin)
- Kubernetes / Compute Resources / Pod (Tags: kubernetes-mixin)
- Kubernetes / Compute Resources / Workload (Tags: kubernetes-mixin)
- Kubernetes / Controller Manager (Tags: kubernetes-mixin)
- Kubernetes / Kubelet (Tags: kubernetes-mixin)
- Kubernetes / Networking / Cluster (Tags: kubernetes-mixin)
- Kubernetes / Networking / Namespace (Pods) (Tags: kubernetes-mixin)
- Kubernetes / Networking / Namespace (Workload) (Tags: kubernetes-mixin)

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
```

```
Session ID: 528a6e0-4080-390c-5098f7290530
Instance ID: i-0108a290c5098f7290530
Terminate

module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 40s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 50s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 1m0s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 1m10s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 1m20s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 1m30s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 1m40s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 1m50s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 2m0s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 2m10s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 2m20s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 2m30s elapsed]
module.eks.aws.eks_cluster.eks[0]: Still destroying... [id=ddev-medium-eks-cluster, 2m40s elapsed]
module.eks.aws.eks_cluster.eks[0]: Destruction complete after 2m56s
module.eks.aws.iam_role.eks-cluster-role[0]: Destroying... [id=ddev-medium-eks-cluster-role-4949]
module.eks.aws.iam_role.eks-cluster-role[0]: Destruction complete after 0s
module.eks.random_integer.random_suffix: Destroying... [id=4949]
module.eks.random_integer.random_suffix: Destruction complete after 0s

Warning: Value for undeclared variable
The root module does not declare a variable named "ec2-iam-instance-profile" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Value for undeclared variable
The root module does not declare a variable named "ec2-iam-role-policy" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Values for undeclared variables
In addition to the other similar warnings shown, 3 other variable(s) defined without being declared.

Warning: Helm uninstall returned an information message
These resources were kept due to the resource policy:
[CustomResourceDefinition] applications.argoproj.io
[CustomResourceDefinition] applicationsets.argoproj.io
[CustomResourceDefinition] appprojects.argoproj.io

Destroy complete! Resources: 25 destroyed.
ubuntu@ip-10-16-10-26:~/EKS-ArgoCD-AWS-LB-Controller-Terraform/eks$
```

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
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
hash-vpc-ec2 + × ☰ … ^ ×

module.vpc-ec2.aws.instance.ec2: Destruction complete after 54s
module.vpc-ec2.aws.iam_instance_profile.ec2-instance-profile: Destroying... [id=ec2-ssm-instance-profile]
module.vpc-ec2.aws.security_group.ec2-sg: Destroying... [id=sg-071f08f4b1928202e]
module.vpc-ec2.aws.nat.gateway.ngw: Still destroying... [id=nat-0e920b64e401eb765, 50s elapsed]
module.vpc-ec2.aws.internet_gateway.igw: Still destroying... [id=igw-081e78f5d08daa6a4, 50s elapsed]
module.vpc-ec2.aws.iam.role.iam-role: Destruction complete after 1s
module.vpc-ec2.aws.iam.role.iam-role: Destroying... [id=ec2-ssm-role]
module.vpc-ec2.aws.security_group.ec2-sg: Destruction complete after 2s
module.vpc-ec2.aws.iam.role.iam-role: Destroying... [id=nat-0e920b64e401eb765, 1m0s elapsed]
module.vpc-ec2.aws.internet_gateway.igw: Still destroying... [id=igw-081e78f5d08daa6a4, 1m0s elapsed]
module.vpc-ec2.aws.internet_gateway.igw: Destruction complete after 1m4s
module.vpc-ec2.aws.nat.gateway.ngw: Destruction complete after 1m10s
module.vpc-ec2.aws.eip.ngw-eip: Destroying... [id=ipalloc-085eda060d90f559]
module.vpc-ec2.aws.subnet.public.subnet[0]: Destroying... [id=subnet-00ccb711dd8cc3bc]
module.vpc-ec2.aws.subnet.public.subnet[1]: Destroying... [id=subnet-0b600ba0303c3a40f]
module.vpc-ec2.aws.subnet.public.subnet[2]: Destroying... [id=subnet-0529728936b2ded3c]
module.vpc-ec2.aws.eip.ngw-eip: Destruction complete after 1s
module.vpc-ec2.aws.eip.ngw-eip: Destroying... [id=eip-053477193e826f6c2]
module.vpc-ec2.aws.vpc.vpc: Destroying... [id=vpc-053477193e826f6c2]
module.vpc-ec2.aws.vpc.vpc: Destruction complete after 1s

Warning: Value for undeclared variable
The root module does not declare a variable named "endpoint-public-access" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Value for undeclared variable
The root module does not declare a variable named "desired_capacity_on_demand" but a value was found in file "../variables.tfvars". If you meant to use this value, add a "variable" block to the configuration.
To silence these warnings, use TF_VAR... environment variables to provide certain "global" settings to all configurations in your organization. To reduce the verbosity of these warnings, use the -compact-warnings option.

Warning: Values for undeclared variables
In addition to the other similar warnings shown, 11 other variable(s) defined without being declared.

Releasing state lock. This may take a few moments...
Destroy complete! Resources: 24 destroyed.
```

I would recommend you to read the eks terraform files to get a better understanding of the resources.

## 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.

