# 😊 AWS Elastic Kubernetes Service

1. In this lab you are going to learn how to launch Kubernetes cluster using Terraform.
2. Now the code used in this lab is available on GitHub. You can clone the source code from there in your VS Code.
3. After that simply open your gitbash and move to the location where you have cloned the code and then do a listing of files there. You will see these files.

```
$ ls
eks-cluster.tf  main.tf  outputs.tf  terraform.tf  variables.tf  vpc.tf
```

4. Then you are going to initialize it. So, this might take some time. There was a lot to initialize.
5. After that you are going to run the plan and apply command to execute your code.

```
Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.
Initializing modules...
Downloading registry.terraform.io/terraform-aws-modules/eks/aws 19.0.4 for eks...
- eks in .terraform\modules\eks
- eks.eks_managed_node_group in .terraform\modules\eks\modules\eks-managed-node-group
- eks.eks_managed_node_group.user_data in .terraform\modules\eks\modules\_user_data
- eks.fargate_profile in .terraform\modules\eks\modules\fargate-profile
Downloading registry.terraform.io/terraform-aws-modules/kms/aws 1.1.0 for eks.kms...
- eks.kms in .terraform\modules\eks.kms
- eks.self_managed_node_group in .terraform\modules\eks\modules\self-managed-node-group
- eks.self_managed_node_group.user_data in .terraform\modules\eks\modules\_user_data
Downloading registry.terraform.io/terraform-aws-modules/vpc/aws 3.14.2 for vpc...
- vpc in .terraform\modules\vpc

Initializing provider plugins...
- Finding hashicorp/random versions matching "~> 3.4.3"...
- Finding hashicorp/tls versions matching ">= 3.0.0, ~> 4.0.4"...
- Finding hashicorp/cloudinit versions matching ">= 2.0.0, ~> 2.2.0"...
- Finding hashicorp/kubernetes versions matching ">= 2.10.0, ~> 2.16.1"...
- Finding hashicorp/aws versions matching ">= 3.63.0, >= 3.72.0, >= 4.45.0, ~> 4.46.0"...
- Installing hashicorp/cloudinit v2.2.0...
- Installed hashicorp/cloudinit v2.2.0 (signed by HashiCorp)
- Installing hashicorp/kubernetes v2.16.1...
- Installed hashicorp/kubernetes v2.16.1 (signed by HashiCorp)
- Installing hashicorp/aws v4.46.0...
- Installed hashicorp/aws v4.46.0 (signed by HashiCorp)
- Installing hashicorp/random v3.4.3...
- Installed hashicorp/random v3.4.3 (signed by HashiCorp)
- Installing hashicorp/tls v4.0.5...
- Installed hashicorp/tls v4.0.5 (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.
```

6. As you can see that there are 53 resources to add so, when you'll run the apply command it will take some time.
7. Moreover, you can also read the structure which plan command provides to know what it will be doing.

```
Plan: 53 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + cluster_endpoint          = (known after apply)
  + cluster_name              = "demo-eks"
  + cluster_security_group_id = (known after apply)
  + region                    = "us-east-1"
```

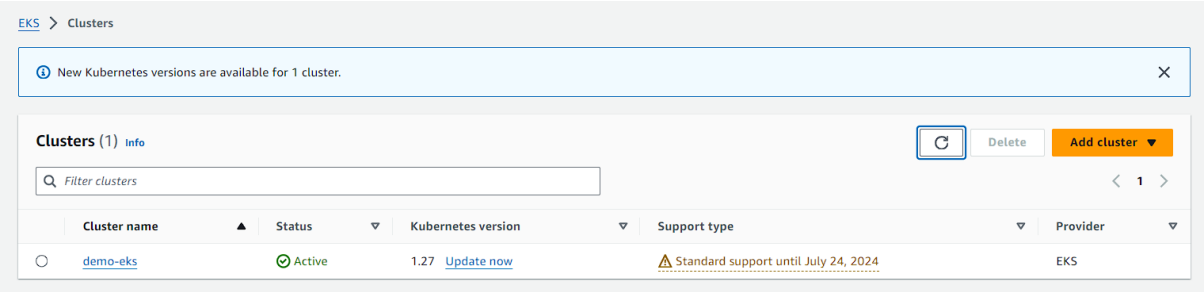
8. Here you can see that the cluster has been created several minutes. And you can also see its endpoint.

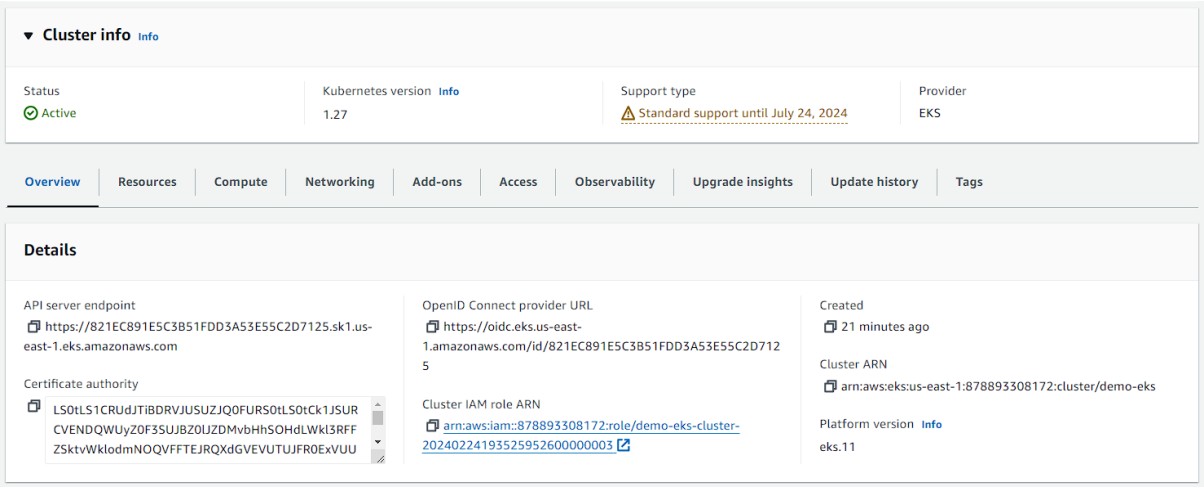
```
Apply complete! Resources: 53 added, 0 changed, 0 destroyed.

Outputs:

cluster_endpoint = "https://821EC891E5C3B51FDD3A53E55C2D7125.sk1.us-east-1.eks.amazonaws.com"
cluster_name = "demo-eks"
cluster_security_group_id = "sg-09a5321cc65a968a2"
region = "us-east-1"
```

9. This is your cluster do not delete it from here delete it from terraform.




10. In order to generate the EKS config file you need to run this code in gitbash. So, it will give you a location you can view this config file too.

    **aws eks update-kubeconfig --region us-east-1 --name demo-eks**
    **cat ~/.kube/config**

```
$ aws eks update-kubeconfig --region us-east-1 --name demo-eks
Added new context arn:aws:eks:us-east-1:878893308172:cluster/demo-eks to C:\Users\      \.kube\config
```


11. After that you need to run the Kubernetes commands and if you don't have Kubernetes installed on your local machine then you can download it from chocolaty.
12. Just open your PowerShell as administrator and run this command.

    **choco install kubernetes-cli**
13. Now if you will this command you can see the nodes.

    **kubectl get nodes**

```
$ kubectl get nodes
NAME                           STATUS   ROLES    AGE   VERSION
ip-172-20-2-47.ec2.internal    Ready    <none>   20m   v1.27.9-eks-5e0fdde
ip-172-20-3-46.ec2.internal    Ready    <none>   20m   v1.27.9-eks-5e0fdde
ip-172-20-3-91.ec2.internal    Ready    <none>   20m   v1.27.9-eks-5e0fdde
```

14. Once you are done just delete your cluster immediately or it will generate bill on your account.