**Provision an EKS Cluster (AWS)**

**Using Terraform**

The Amazon Elastic Kubernetes Service (EKS) is the AWS service for deploying, managing and scaling containerized applications with Kubernetes.

### Why deploy with Terraform?

While you could use the built-in AWS provisioning processes (UI, CLI, CloudFormation) for EKS clusters, Terraform provides you with several benefits:

* **Unified Workflow** - If you are already deploying infrastructure to AWS with Terraform, your EKS cluster can fit into that workflow. You can also deploy applications into your EKS cluster using Terraform.
* **Full Lifecycle Management** - Terraform doesn't only create resources, it updates, and deletes tracked resources without requiring you to inspect the API to identify those resources.
* **Graph of Relationships** - Terraform understands dependency relationships between resources. For example, if an AWS Kubernetes cluster needs a specific VPC and subnet configurations, Terraform won't attempt to create the cluster if the VPC and subnets failed to create with the proper configuration.

STEP 1: First, we must configure the AWS CLI in VS Code.

AWS Access Key ID [None]: YOUR\_AWS\_ACCESS\_KEY\_ID

AWS Secret Access Key [None]: YOUR\_AWS\_SECRET\_ACCESS\_KEY

Default region name [None]: YOUR\_AWS\_REGION

STEP 2: In your terminal, clone the following repository.

git clone <https://github.com/hashicorp/learn-terraform-provision-eks-cluster>

STEP 3: You can explore this repository by changing directories or navigating in your UI.

cd learn-terraform-provision-eks-cluster

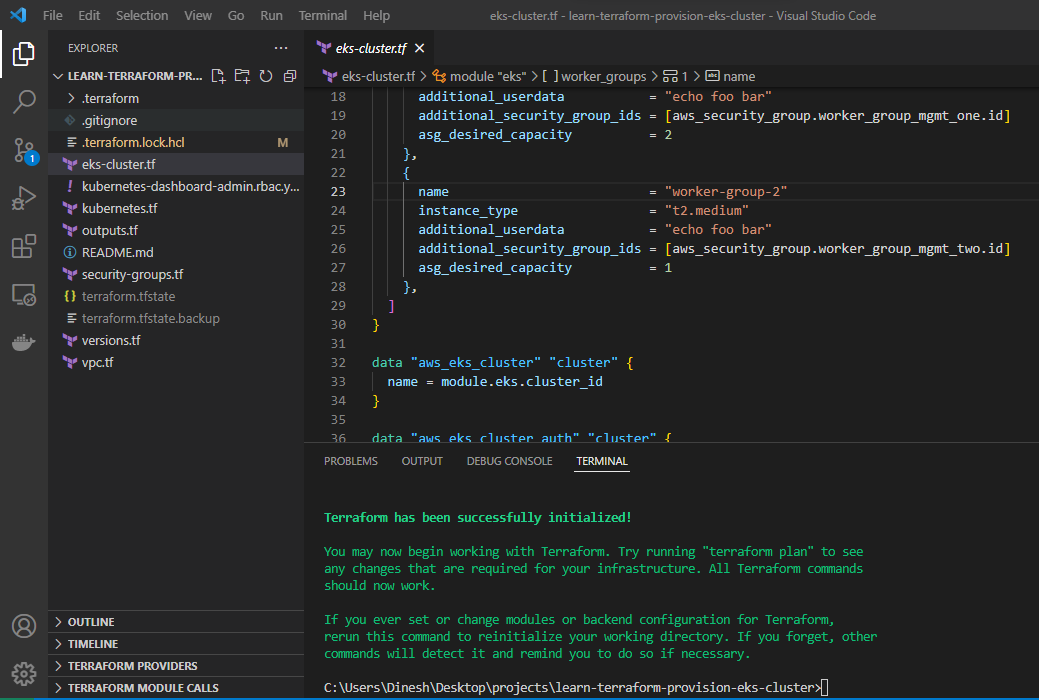
STEP 4: In here, you will find six files used to provision a VPC, security groups, and an EKS cluster. The final product should be similar to this:

Graphical user interface, application

Description automatically generated

### STEP 5: Initialize Terraform workspace, Once you have cloned the repository, initialize your Terraform workspace, which will download and configure the providers.

🡪terraform init using this command we can initialize the workspace.



## STEP 6: Provision the EKS cluster, In your initialized directory, run terraform apply and review the planned actions. Your terminal output should indicate the plan is running and what resources will be created.

🡪terraform apply using this command we can review the planned actions.

Text

Description automatically generated

STEP 7: Configure kubectl, Now that you've provisioned your EKS cluster, you need to configure kubectl.

Run the following command to retrieve the access credentials for your cluster and automatically configure kubectl.

aws eks --region $(terraform output -raw region) update-kubeconfig --name $(terraform output -raw cluster\_name)

aws eks --region us-east-2 update-kubeconfig --name education-eks-V2r5Ppcx



## STEP 8: Deploy and access Kubernetes Dashboard

To verify that your cluster is configured correctly and running, you will deploy the Kubernetes dashboard and navigate to it in your local browser.

### Deploy Kubernetes Metrics Server

The Kubernetes Metrics Server used to gather metrics such as cluster CPU and memory usage over time, is not deployed by default in EKS clusters.

1. Deploy the Metrics Server with the following command

kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>

Text

Description automatically generated

1. Verify that the metrics-server deployment is running the desired number of pods with the following command.

kubectl get deployment metrics-server -n kube-system

Text

Description automatically generated with low confidence

STEP 9: Now, create a proxy server that will allow you to navigate to the dashboard from the browser on your local machine. This will continue running until you stop the process by pressing CTRL + C

🡪kubectl proxy Using this command we can see the dashboard from the browser.



You should be able to access the Kubernetes dashboard [here](http://127.0.0.1:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/) (<http://127.0.0.1:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/>).

Graphical user interface, text, application, email

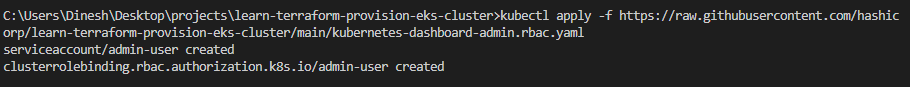
Description automatically generated

## STEP 10: Authenticate the dashboard

To use the Kubernetes dashboard, you need to create a ClusterRoleBinding and provide an authorization token. This gives the cluster-admin permission to access the Kubernetes dashboard. Authenticating using kubeconfig is **not** an option. You can read more about it in the [Kubernetes documentation](https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/#accessing-the-dashboard-ui).

In another terminal (do not close the kubectl proxy process), create the ClusterRoleBinding resource.

1.kubectl apply -f <https://raw.githubusercontent.com/hashicorp/learn-terraform-provision-eks-cluster/main/kubernetes-dashboard-admin.rbac.yaml>



2.kubectl get secrets -n kube-system

Text

Description automatically generated

Note:(We should copy the service controller token only)

3.kubectl describe secret service-controller-token-5gj8p -n kube-system

Text

Description automatically generated

STEP 11: Select "Token" on the Dashboard UI then copy and paste the entire token you receive into the [dashboard authentication screen](http://127.0.0.1:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/) to sign in. You are now signed in to the dashboard for your Kubernetes cluster.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

STEP 11: Navigate to the "Cluster" page by clicking on "Cluster" in the left navigation bar. You should see a list of nodes in your cluster.

Graphical user interface, application

Description automatically generated

STEP 11: If not, remember to destroy any resources. Run the destroy command and confirm with yes in your terminal.

🡪terraform destroy