For this tutorial, you will need:

* an AWS account with the IAM permissions listed on the EKS module documentation,
* a configured AWS CLI
* AWS IAM Authenticator
* kubectl
* wget (required for the eks module)

**Installations:**

Configure AWS Cli:

**Choco install awscli**

AWS IAM Authenticator:

**choco install aws-iam-authenticator**

kubectl:

**choco install kubernetes-cli**

wget:

**choco install wget**

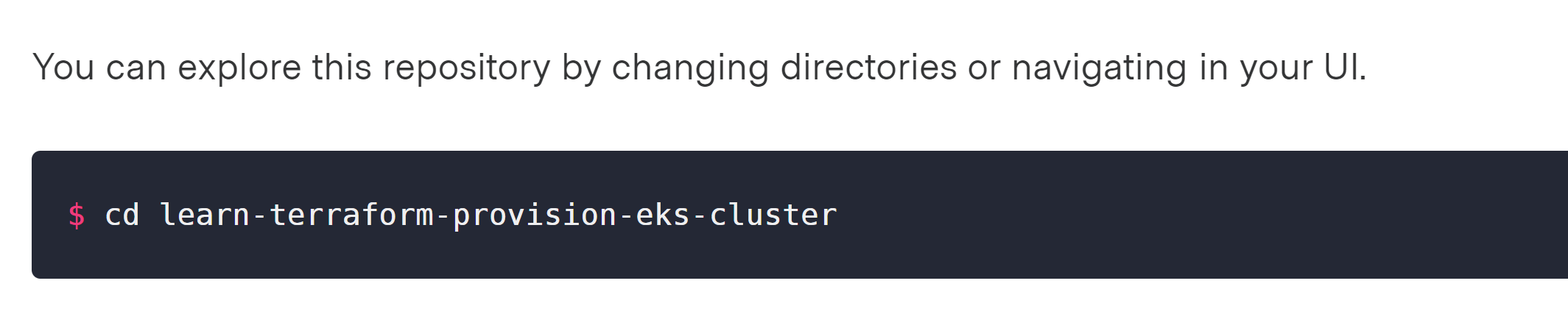
In case of RedHat use **Yum** instead of **choco** in the above

**Set up and initialize your Terraform workspace**

Copy paste the folder provided.

Unzip it.

Open terminal (Linux) or Powershell(Windows)



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:

**vpc.tf** provisions a VPC, subnets and availability zones using the AWS VPC Module. A new VPC is created for this tutorial so it doesn't impact your existing cloud environment and resources.

**security-groups.tf** provisions the security groups used by the EKS cluster.

**eks-cluster.tf** provisions all the resources (AutoScaling Groups, etc...) required to set up an EKS cluster using the AWS EKS Module.

On line 14, the AutoScaling group configuration contains three nodes.

worker\_groups = [

{

name = "worker-group-1"

instance\_type = "t2.small"

additional\_userdata = "echo foo bar"

asg\_desired\_capacity = 2

additional\_security\_group\_ids = [aws\_security\_group.worker\_group\_mgmt\_one.id]

},

{

name = "worker-group-2"

instance\_type = "t2.medium"

additional\_userdata = "echo foo bar"

additional\_security\_group\_ids = [aws\_security\_group.worker\_group\_mgmt\_two.id]

asg\_desired\_capacity = 1

},

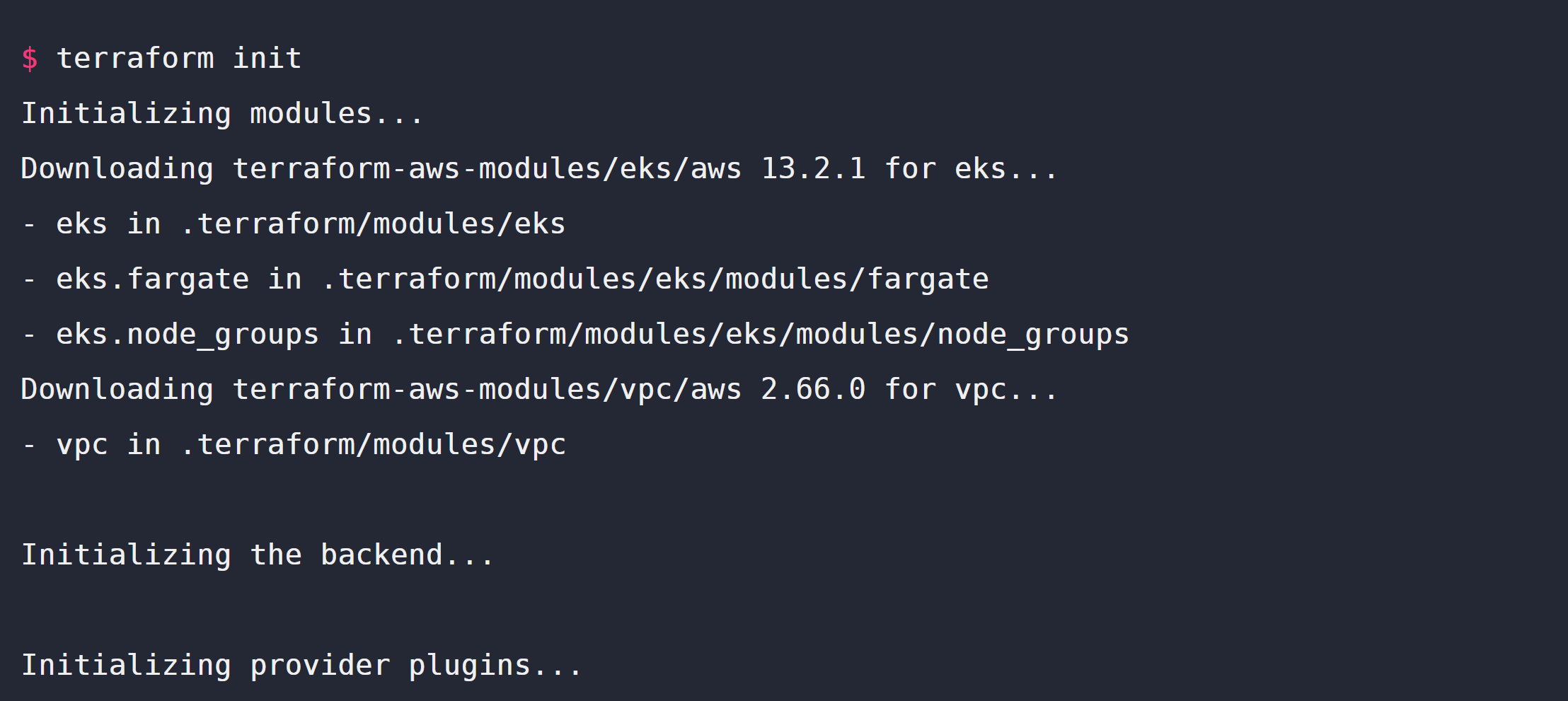
]

**outputs.tf** defines the output configuration.

**versions.tf** sets the Terraform version to at least 0.14. It also sets versions for the providers used in this sample.

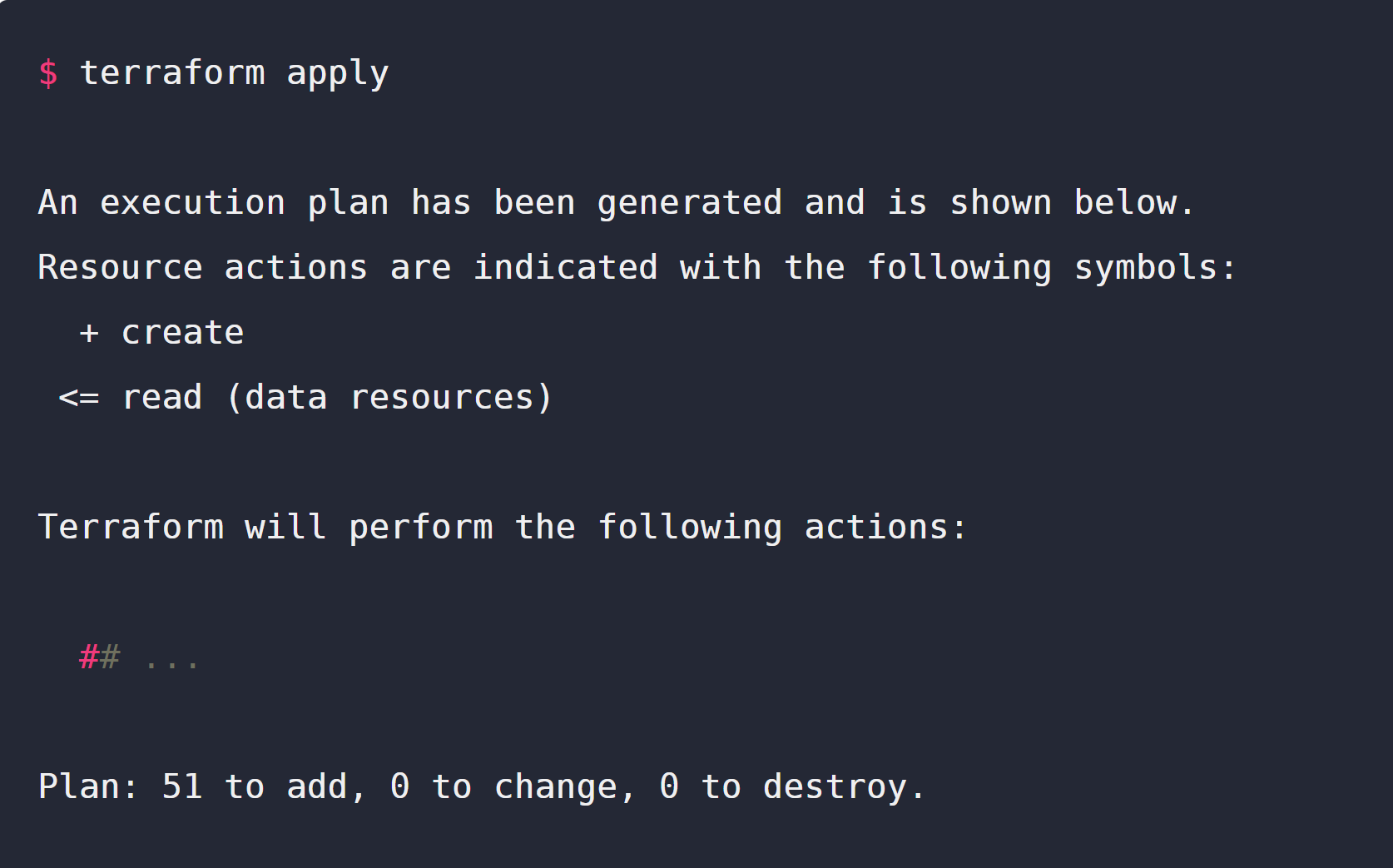
### **Initialize Terraform workspace**

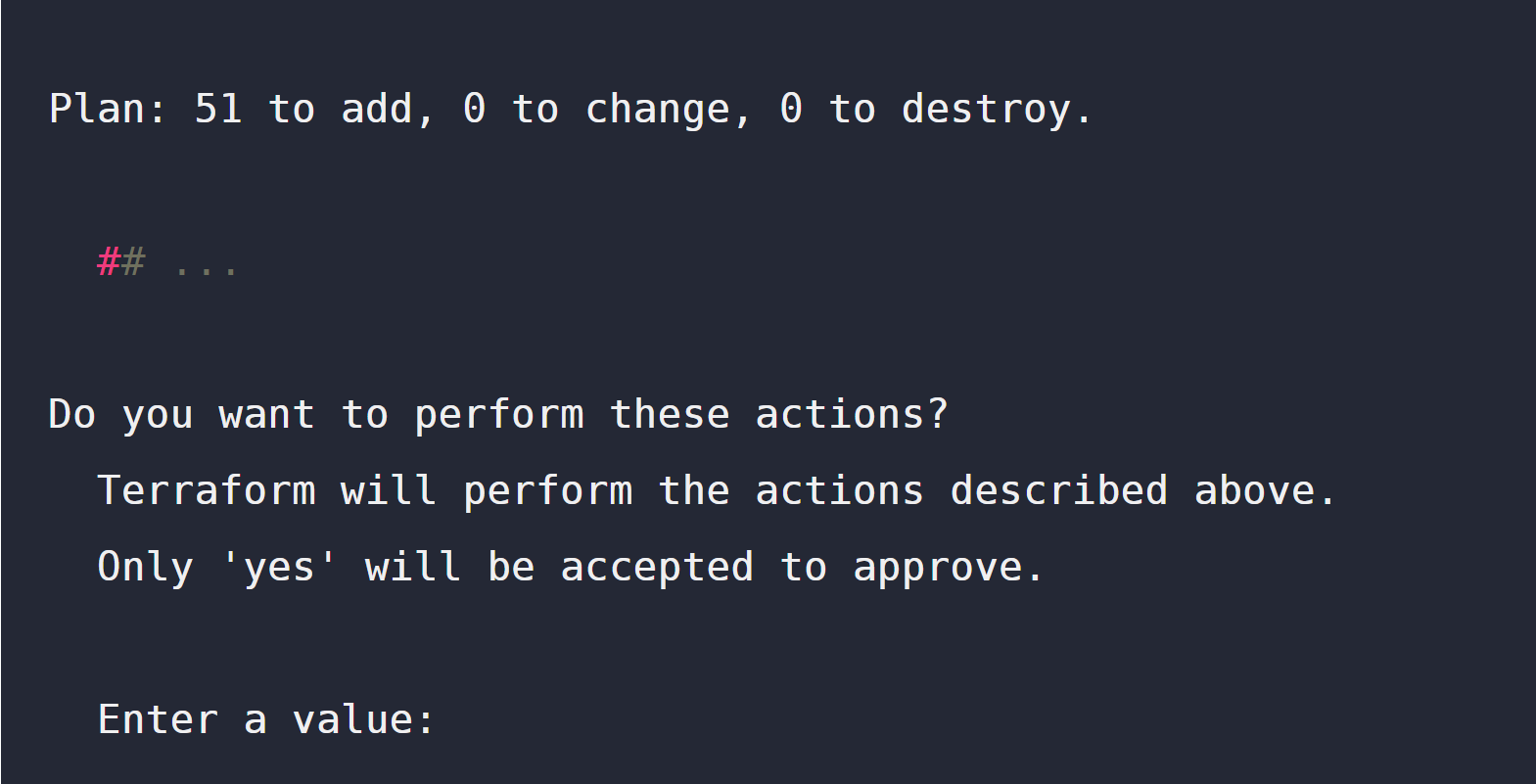
Once you have cloned the repository, initialize your Terraform workspace, which will download and configure the providers.



## Provision the EKS cluster

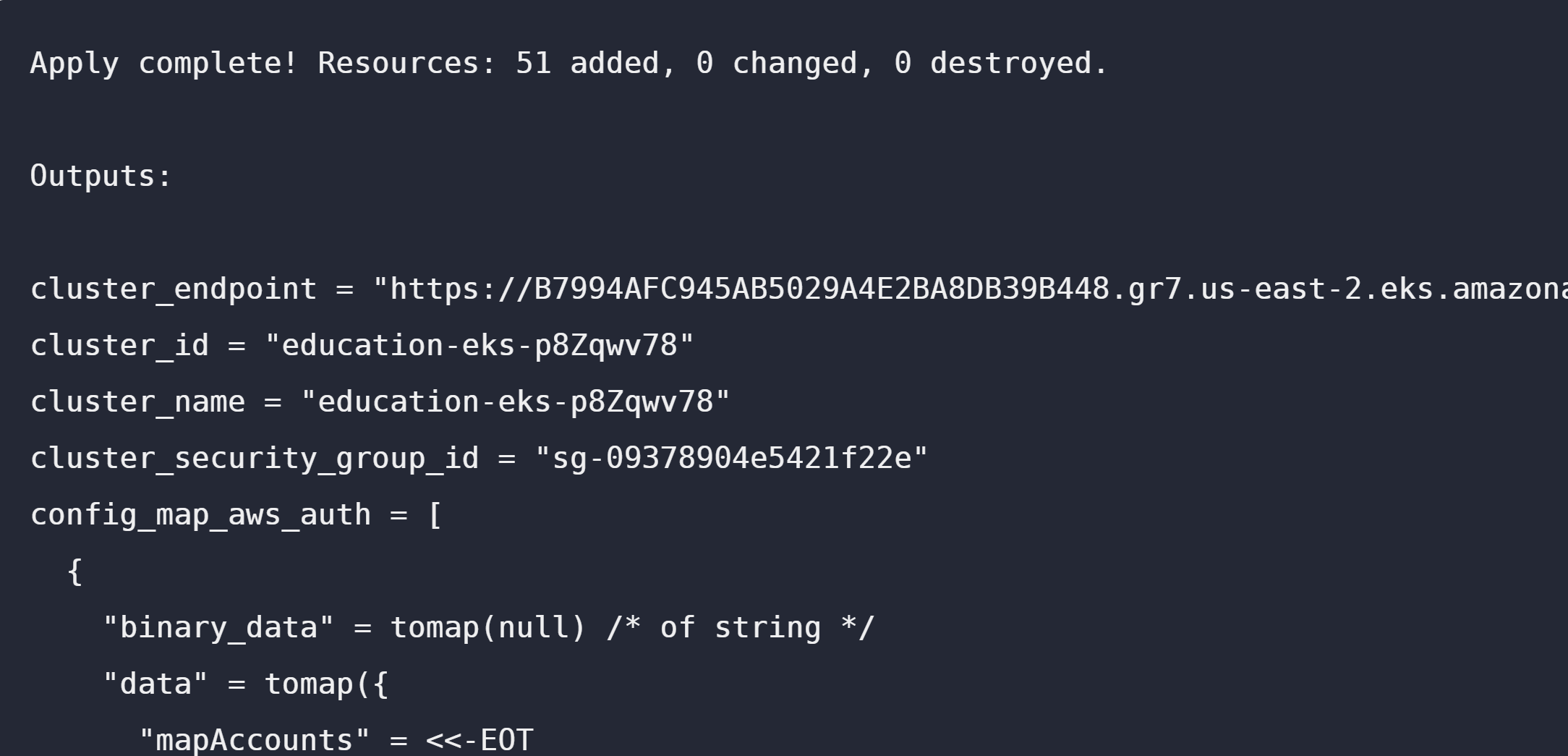
In your initialized directory, run terraform apply and review the planned actions. Your terminal output should indicate the plan is running and what resources will be created.





This terraform apply will provision a total of 51 resources (VPC, Security Groups, AutoScaling Groups, EKS Cluster, etc...). Confirm the apply with a yes.

This process should take approximately 10 minutes. Upon successful application, your terminal prints the outputs defined in outputs.tf.



Apply complete! Resources: 51 added, 0 changed, 0 destroyed.

Outputs:

cluster\_endpoint = "https://B7994AFC945AB5029A4E2BA8DB39B448.gr7.us-east-2.eks.amazonaws.com"

cluster\_id = "education-eks-p8Zqwv78"

cluster\_name = "education-eks-p8Zqwv78"

cluster\_security\_group\_id = "sg-09378904e5421f22e"

config\_map\_aws\_auth = [

{

"binary\_data" = tomap(null) /\* of string \*/

"data" = tomap({

"mapAccounts" = <<-EOT

[]

EOT

"mapRoles" = <<-EOT

- "groups":

- "system:bootstrappers"

- "system:nodes"

"rolearn": "arn:aws:iam::561656980159:role/education-eks-p8Zqwv782021011204012796010000000c"

"username": "system:node:{{EC2PrivateDNSName}}"

EOT

"mapUsers" = <<-EOT

[]

EOT

})

"id" = "kube-system/aws-auth"

"metadata" = tolist([

{

"annotations" = tomap(null) /\* of string \*/

"generate\_name" = ""

"generation" = 0

"labels" = tomap({

"app.kubernetes.io/managed-by" = "Terraform"

"terraform.io/module" = "terraform-aws-modules.eks.aws"

})

"name" = "aws-auth"

"namespace" = "kube-system"

"resource\_version" = "942"

"self\_link" = "/api/v1/namespaces/kube-system/configmaps/aws-auth"

"uid" = "f328b2d0-f099-4e72-a1b1-760781045f10"

},

])

},

]

kubectl\_config = ...

## ...

## 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)***

The Kubernetes cluster name and region correspond to the output variables showed after the successful Terraform run.

## 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.

While you can deploy the Kubernetes metrics server and dashboard using Terraform, kubectl is used in this tutorial so you don't need to configure your Terraform Kubernetes Provider.

**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.

Download and unzip the metrics server by running the following command.

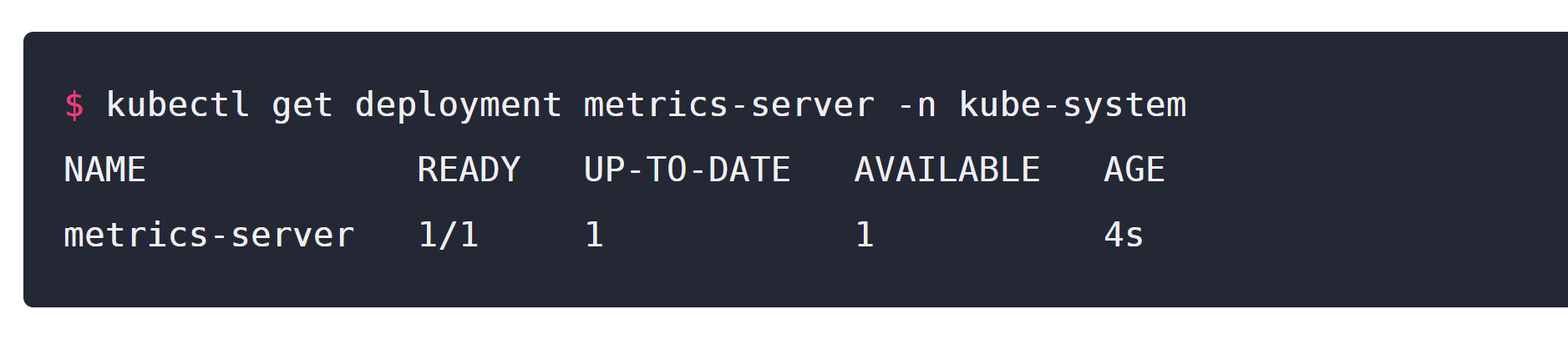
wget -O v0.3.6.tar.gz [https://codeload.github.com/kubernetes-sigs/metrics-server/tar.gz/v0.3.6 && tar -xzf v0.3.6.tar.gz](https://codeload.github.com/kubernetes-sigs/metrics-server/tar.gz/v0.3.6%20&&%20tar%20-xzf%20v0.3.6.tar.gz)

Deploy the metrics server to the cluster by running the following command.

kubectl apply -f metrics-server-0.3.6/deploy/1.8+/

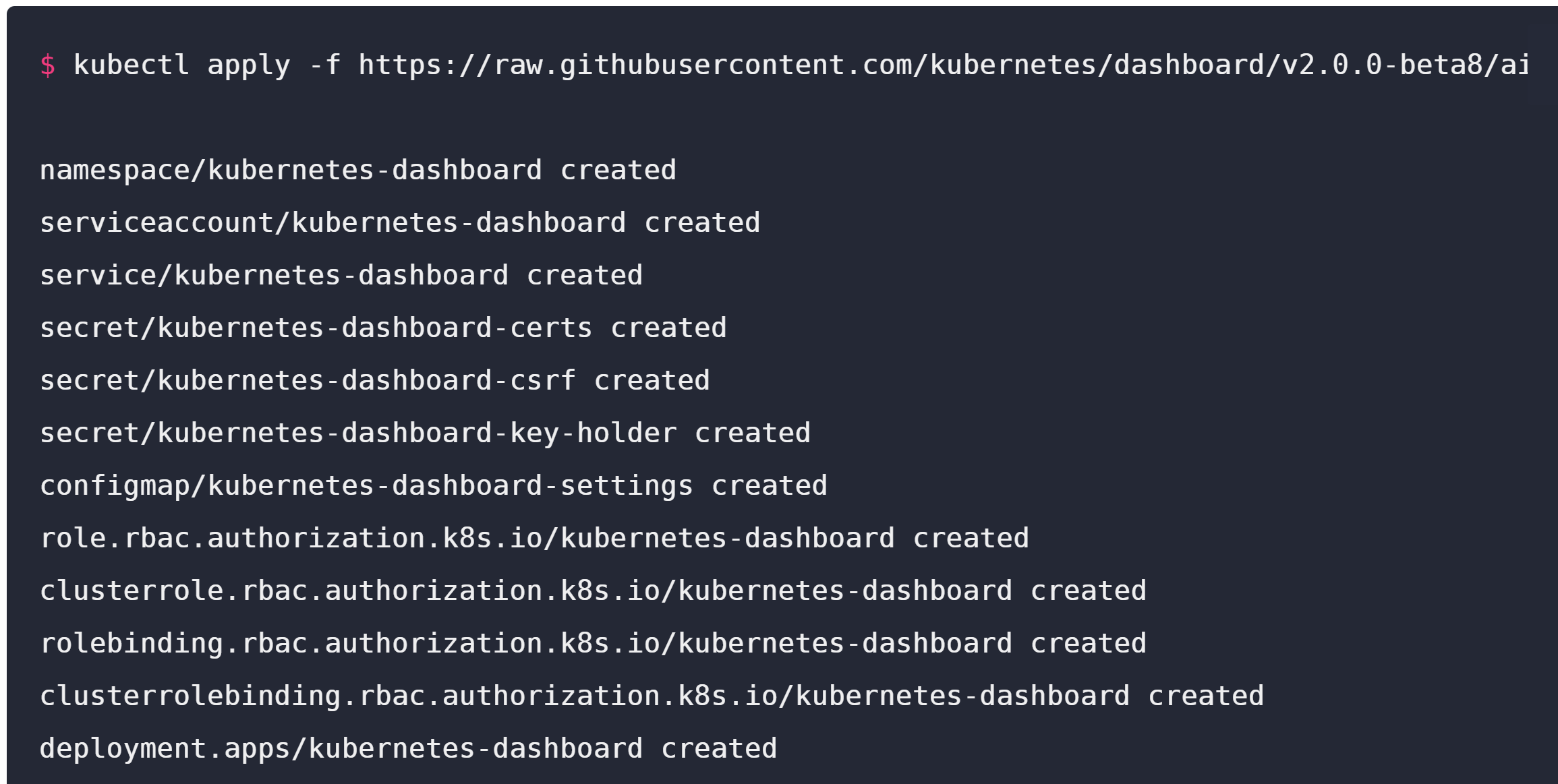
Verify that the metrics server has been deployed. If successful, you should see something like this.

kubectl get deployment metrics-server -n kube-system



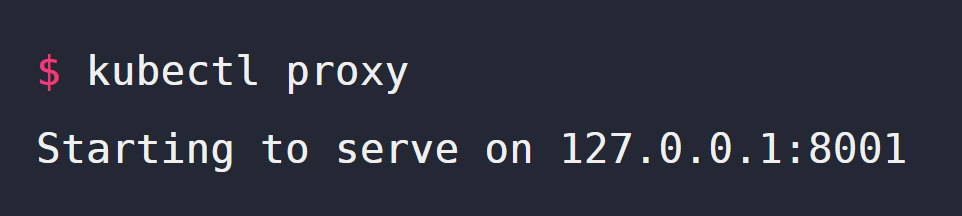
**Deploy Kubernetes Dashboard**

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-beta8/aio/deploy/recommended.yaml>

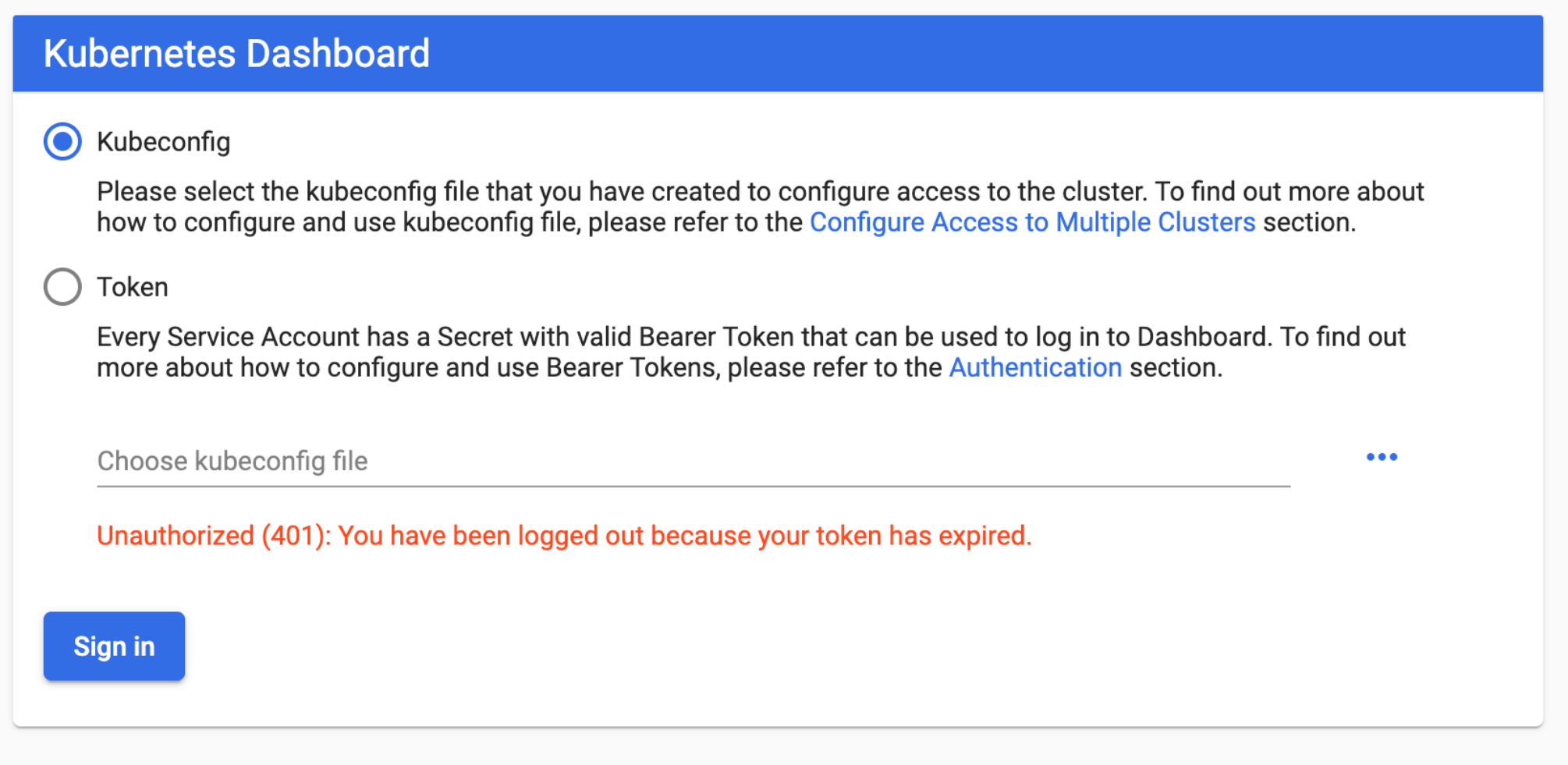


Now, create a proxy server that will allow you to navigate to the dashboard from the browser on your local machine. This will continue running until you stop the process by pressing CTRL + C.

kubectl proxy



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/).



## Authenticate the dashboard

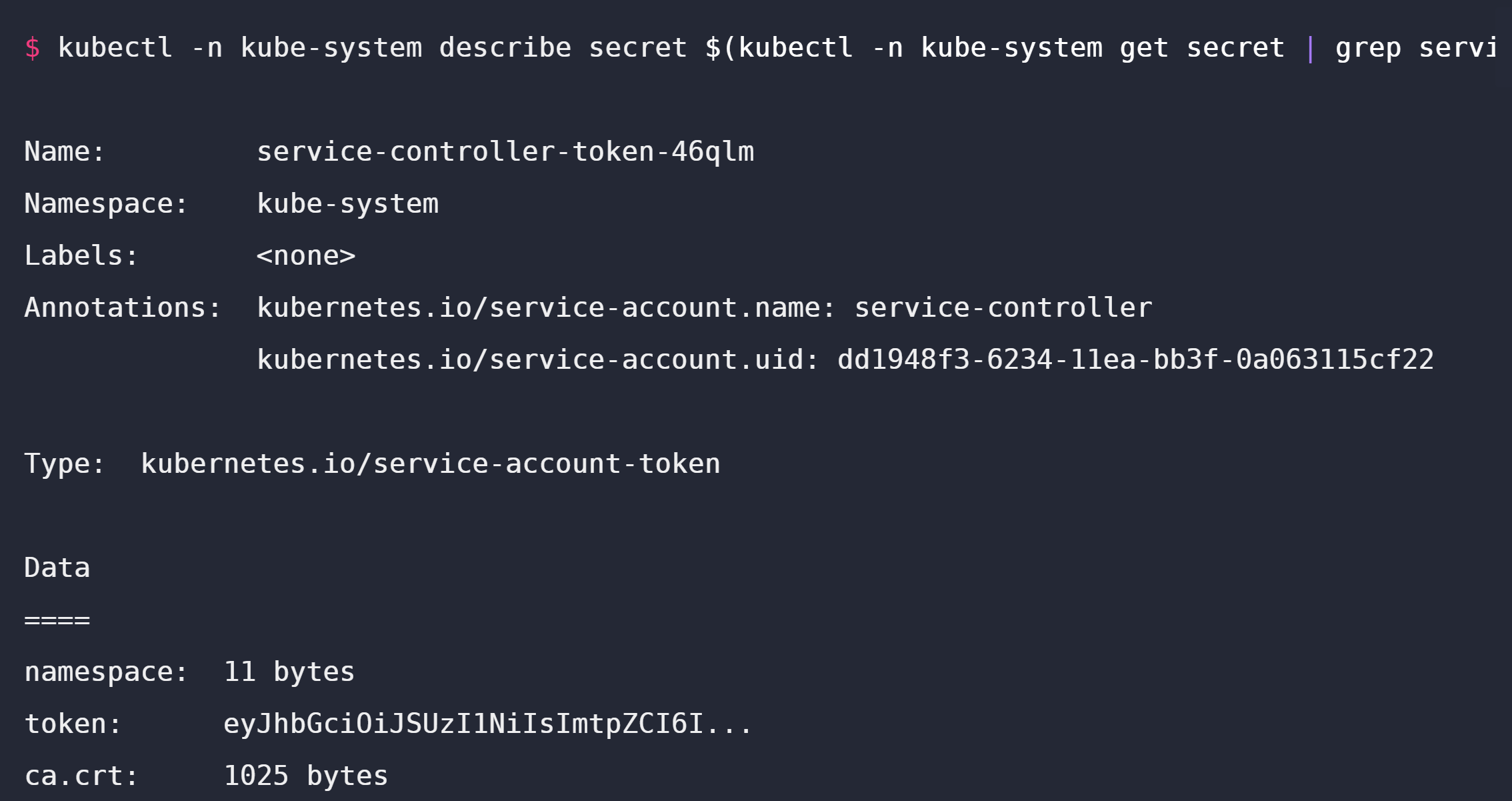
To use the Kubernetes dashboard, you need to create a ClusterRoleBinding and provide an authorization token. This gives the cluster-admin permission to access the kubernetes-dashboard. Authenticating using kubeconfig is not an option. You can read more about it in the Kubernetes documentation.

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

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

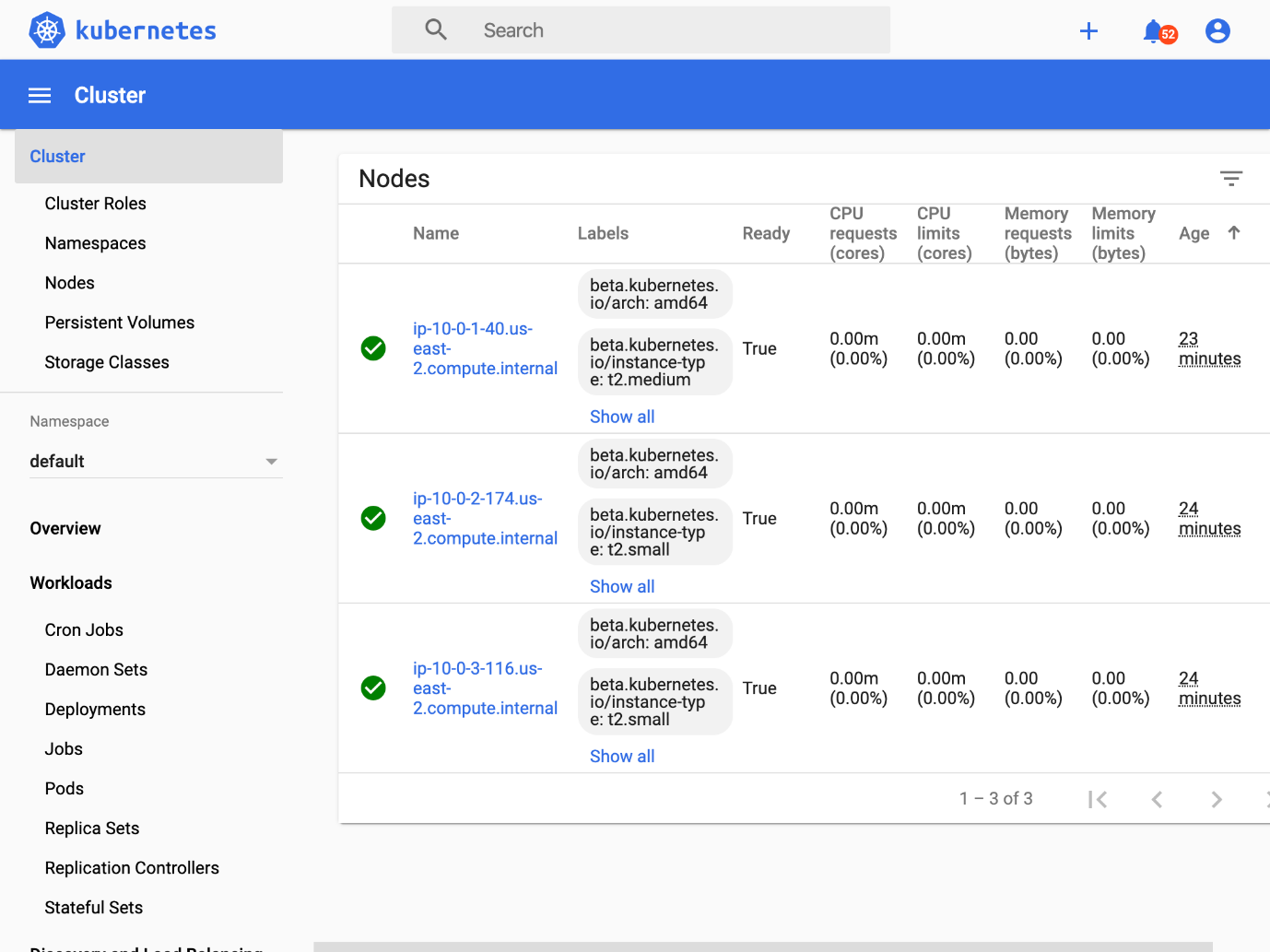
Then, generate the authorization token.

kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep service-controller-token | awk '{print $1}')

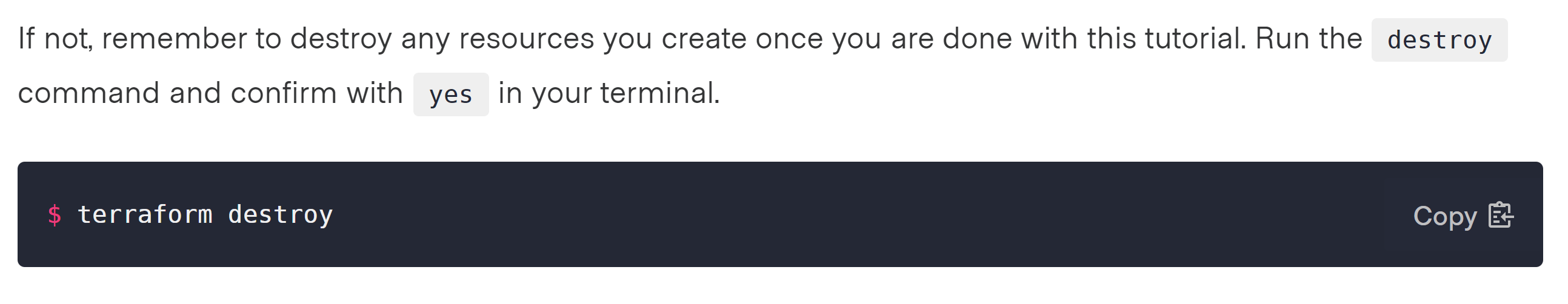


Select "Token" on the Dashboard UI then copy and paste the entire token you receive into the dashboard authentication screen to sign in. You are now signed in to the dashboard for your Kubernetes cluster.

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



## Clean up your workspace



**The basic Architecture that has been deployed is as follows:**

