

Home / AWS / Guided Lab / Create an Amazon EKS Cluster and install kubectl using Terraform

Create an Amazon EKS Cluster and install kubectl using Terraform

Level: Fundamental

Amazon Web Services Amazon Elastic Kubernetes Service Terraform



0h 56m 3s left



End Lab

Open Console

Validation

Lab Credentials

User Name ⓘ

Whiz_User_80425.56819031



Password ⓘ

783d739c-c65b-4618-b181-e32c7639e930



Access Key ⓘ

AKIAQUY64LTY4WCZSFBT



Secret Key ⓘ

L6/vQB1x1W26P2uIOZ7snTQl4rVj98zy+CQ+ox6/






Lab Resources


No Lab Resources Found

Support Documents

1. FAQs and Troubleshooting

Need help?

-  How to use Hands on Lab
-  Troubleshooting Lab
-  FAQs

[Submit Feedback](#)[Share](#)[Lab Overview](#)[Lab Steps](#)[Lab Validation](#)[Lab FAQs](#) Cloud Developer, Cloud DevOps Engineer Containers, Infrastructure

Lab Steps

Task 1: Sign in to AWS Management Console

1. Click on the **Open Console** button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,
 - Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
 - Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign in** button
3. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1**.

Info : If you haven't installed Terraform in you local, please use the [link](#) provided in the **prerequisites** section to follow the Cloud9 setup steps.

Task 2: Setup Visual Studio Code

In this task, we will set up Visual Studio Code for the lab.

1. Open the visual studio code.
2. If you have already installed and using Visual Studio code, open a new window.
3. A new window will open a new file and release notes page (only if you have installed or updated Visual Studio Code recently). Close the Release notes tab.
4. Open Terminal by selecting **View** from the Menu bar and choose Terminal.
5. It may take up to 2 minutes to open the terminal window.
6. Once the terminal is ready, let us navigate to the **Desktop**. (Ignore this if you are using cloud9)

```
cd Desktop
```



7. Create a new folder by running the below command.

```
mkdir task_10132
```



8. Change your present working directory to use the newly created folder by running the below command:

```
cd task_10132
```



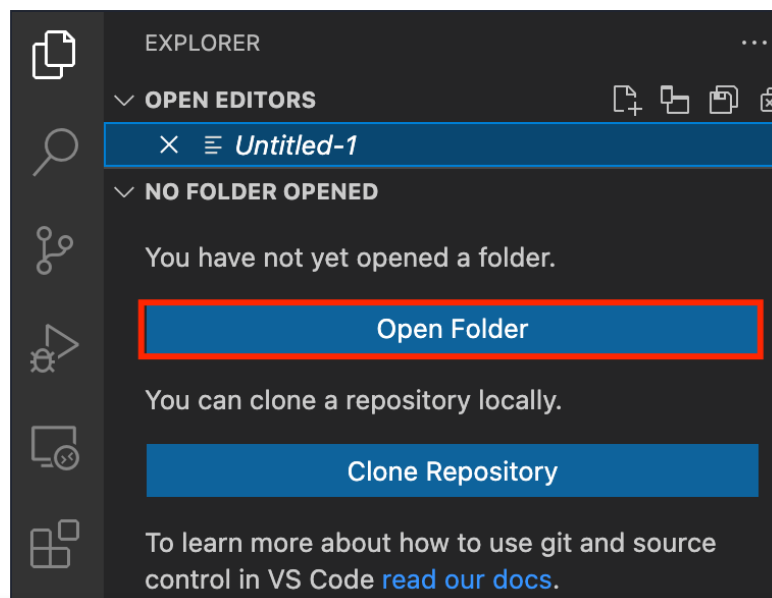
9. Get the location of the present working directory by running the below command:

```
pwd
```



10. Note down the location, as you will open the same in the next steps.
11. Now click on the first icon Explorer present on the left sidebar.
12. Click on the button called Open folder and navigate to the location of folder **task_10132**.





for a while as you are allowing a new folder to be accessed by VSC.

14. Visual Studio Code is now ready to use.

Task 3: Create a variables file

In this task, you will create variable files where you will declare all the global variables with a short description and a default value.

1. To create a variable file, expand the folder **task_10132** and click on the **New File** icon to add the file.
2. Name the file as **variables.tf** and press **Enter** to save it.
3. **Note:** Don't change the location of the new file, keep it default, i.e. inside the **task_10132** folder.
4. Paste the below contents in **variables.tf** file.

```
variable "access_key" {  
    description = "Access key to AWS console"  
}  
  
variable "secret_key" {  
    description = "Secret key to AWS console"  
}  
  
variable "region" {  
    description = "AWS region"  
}
```



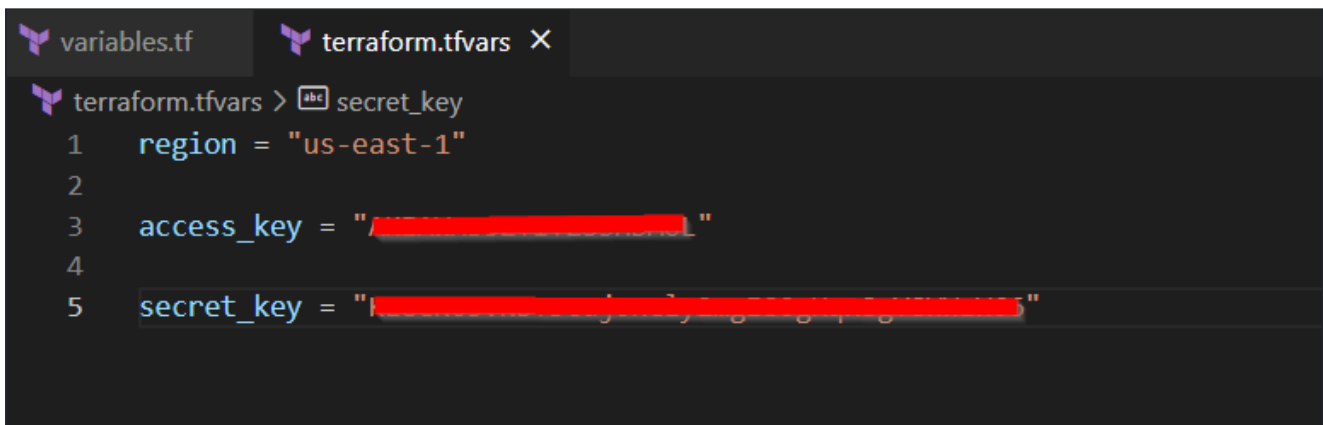
5. In the above content, you are declaring a variable called, `access_key`, `secret_key`, and `region` with a short description of all 3.

6. After pasting the above contents, save the file by pressing **ctrl + S**.
7. Now expand the folder **task_10132** and click on the **New File** icon to add the file.
8. Name the file as **terraform.tfvars** and press **Enter** to save it.
9. Paste the below content into the **terraform.tfvars** file.

```
region = "us-east-1"
access_key = "<YOUR AWS CONSOLE ACCESS ID>"
secret_key = "<YOUR AWS CONSOLE SECRET KEY>"
```



10. In the above code, you are defining the dynamic values of variables declared earlier.
11. Replace the values of `access_key` and `secret_key` by copying from the lab page.
12. After replacing the values of `access_key` and `secret_key`, save the file by pressing **Ctrl + S**.



Task 4: Create an EKS cluster in the main.tf file

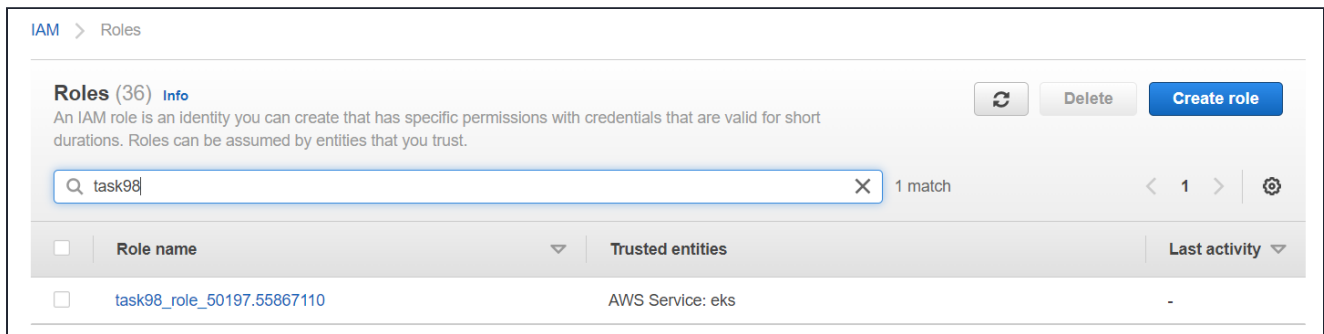
In this task, you will create a **main.tf** file where you will add details of the provider and resources.

1. To create a **main.tf** file, expand the folder **task_10132**, and click on the **New File** icon to add the file.
2. Name the file as **main.tf** and press **Enter** to save it.
3. Paste the below content into the **main.tf** file.

```
provider "aws" {
  region      = "${var.region}"
  access_key  = "${var.access_key}"
  secret_key  = "${var.secret_key}"
}
```



4. In the above code, you are defining the provider as aws.
5. Next, we want to tell Terraform to create an EKS cluster. For that, IAM Role ARNs and Subnet IDs are required.
6. To get IAM Role ARN, navigate to **IAM** by clicking on **Services** on the top under **Security, Identity, & Compliance**.
7. Click on the **Roles** on the left navigation panel and search **task98**.



8. Click on **task98_role_<RANDOM_NUMBER>** role and copy the **ARN** and paste it in the notepad



9. Now to get Subnet IDs, navigate to **VPC** by clicking on **Services** on the top, then click on **VPC** under **Networking & Content delivery**.
10. Click on the **Subnets** on the left navigation panel.
11. Copy the **Subnet ID** of subnets available in **us-east-1a** and **us-east-1b** and paste it in the notepad.

Subnet ID	State	VPC	IPv4 CIDR	IPv6 CIDR	Available IPv4 addresses	Availability Zone
subnet-be7326e1	Available	vpc-18c6a365 Default VPC	172.31.32.0/20	-	4091	us-east-1d
subnet-81263ecc	Available	vpc-18c6a365 Default VPC	172.31.16.0/20	-	4091	us-east-1c
subnet-b41e1cba	Available	vpc-18c6a365 Default VPC	172.31.64.0/20	-	4091	us-east-1f
subnet-9c5f17fa	Available	vpc-18c6a365 Default VPC	172.31.0.0/20	-	4091	us-east-1a
subnet-dd65f9ec	Available	vpc-18c6a365 Default VPC	172.31.48.0/20	-	4091	us-east-1e
subnet-d6f7a0f7	Available	vpc-18c6a365 Default VPC	172.31.80.0/20	-	4091	us-east-1b

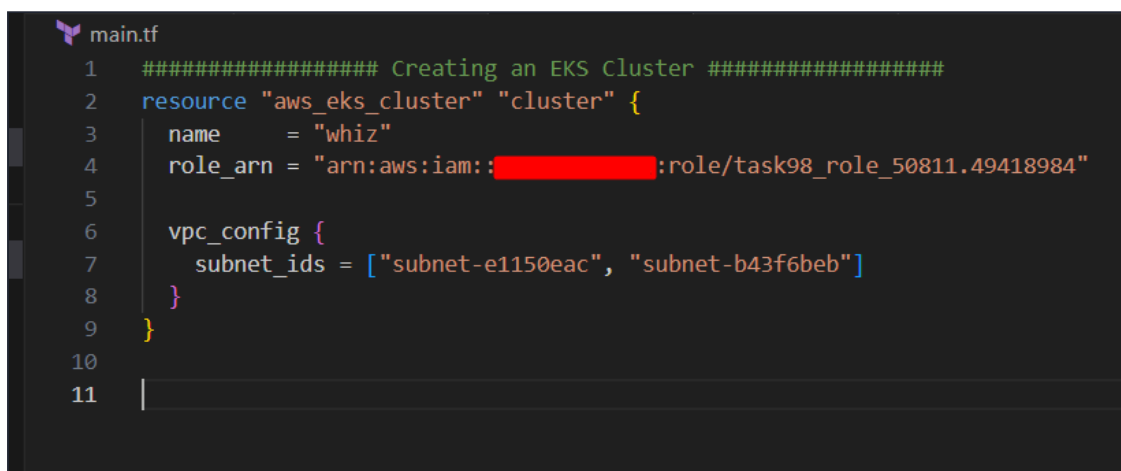
12. To create an EKS Cluster, Paste the below content into the **main.tf** file after the provider.

```
##### Creating an EKS Cluster #####
resource "aws_eks_cluster" "cluster" {
  name      = "whiz"
  role_arn  = "<YOUR_IAM_ROLE_ARN>"

  vpc_config {
    subnet_ids = ["SUBNET-ID 1", "SUBNET-ID 2"]
  }
}
```



13. Replace the **role_arn** with **task98_role** ARN and **subnet_ids** with subnet ids copied.



```
main.tf
1 ##### Creating an EKS Cluster #####
2 resource "aws_eks_cluster" "cluster" {
3   name      = "whiz"
4   role_arn  = "arn:aws:iam::[REDACTED]:role/task98_role_50811.49418984"
5
6   vpc_config {
7     subnet_ids = ["subnet-e1150eac", "subnet-b43f6beb"]
8   }
9 }
10
11 |
```

14. Save the file by pressing **Ctrl + S**.

Task 5: Create an Output file

In this task, you will create an **output.tf** file where you will add details of the provider and resources.

1. To create an **output.tf** file, expand the folder **task_10132**, and click on the **New File** icon to add the file.
2. Name the file as **output.tf** and press **Enter** to save it.
3. Paste the below content into the **output.tf** file.

```
output "cluster" {
  value = aws_eks_cluster.cluster.endpoint
}
```



4. In the above code, we will extract details of resources created to confirm that they are created.

Task 6: Confirm the installation of Terraform by checking the version

Info : If you haven't installed Terraform in you local, run below commands in Cloud9 terminal.

1. In the Visual Studio Code, open Terminal by selecting **View** from the Menu bar and choose **Terminal**.
2. If you are not in the newly created folder change your present working directory by running the below command.

```
cd task_10132
```



3. To confirm the installation of Terraform, run the below command to check the version:

```
terraform version
```



4. If you are getting output as command not found: terraform, this means that terraform is not installed on your system, To install terraform follow the official guide link provided in the Prerequisite section above.

Task 7: Apply terraform configurations

1. Initialize Terraform by running the below command,

```
terraform init
```



```
PS C:\Users\red\Desktop\Task_10132> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.48.0...
- Installed hashicorp/aws v4.48.0 (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.
```


Note: terraform init will check for all the plugin dependencies and download them if required, this will be used for creating a deployment plan

2. To generate the action plans run the below command,

```
terraform plan
```



```
Plan: 1 to add, 0 to change, 0 to destroy.
```

```
Changes to Outputs:
```

```
+ cluster = (known after apply)
```

3. To create all the resources declared in the main.tf configuration file, run the below command:

```
terraform apply
```



4. Approve the creation of all the resources by entering **yes**.

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

```
Outputs:
```

```
cluster = "https://B73AD1739C6573320D1207E88E05759F.gr7.us-east-1.eks.amazonaws.com"
```

5. It may take up to 10-15 minutes for the terraform apply command to create the resources.
6. The endpoint of the cluster created by terraform will be visible there.

Task 8: Check the resources in AWS Console

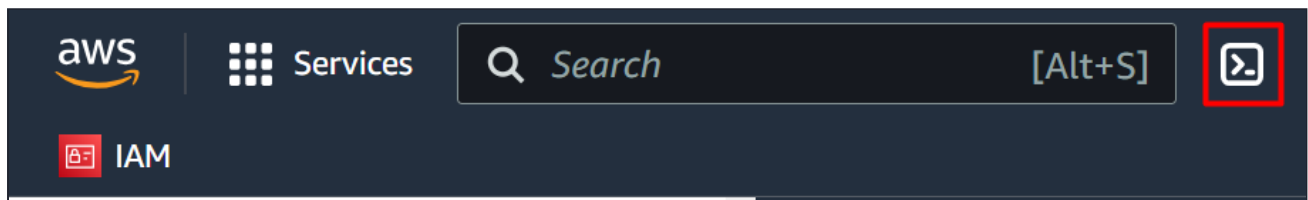
1. Make sure you are in the **US East (N. Virginia) us-east-1** Region.
2. Navigate to **Elastic Kubernetes Services** by clicking on **Services** on the top, then click on **Elastic Kubernetes Services** in the **Containers** section.
3. Click on the **Clusters** on the left navigation panel. You can see the cluster created successfully.



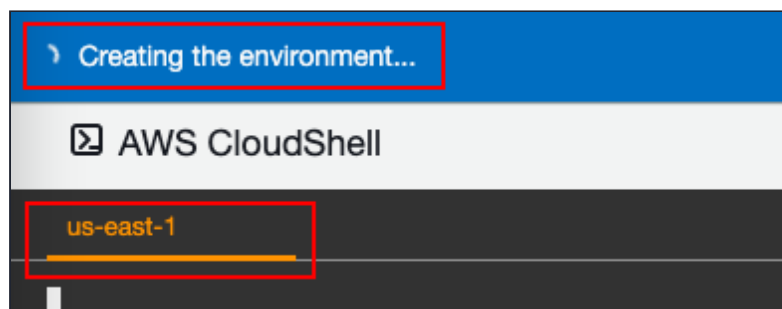
Clusters (1) Info					Refresh	Delete	Add cluster ▼
<input type="text" value="Filter cluster by name, status, kubernetes version, or provider"/>					< 1 >		
Cluster name	▲	Status	▼	Kubernetes version	▼	Provider	▼
<input type="radio"/> whiz		Active		1.23 Update now		EKS	

Task 9: Create an Environment in CloudShell

1. Make sure you are in the **N.Virginia** Region.
2. Click on the Cloud Shell icon on the top right AWS menu bar.



3. A new tab in your browser opens and if you see a welcome message to cloud shell then click on the Close button in that message.
4. You will see a creating environment message on the screen.



5. Wait for a few minutes to complete the environment creation. Once the environment is created, You are ready to use the terminal.



Task 10: Install kubectl on AWS CloudShell

In this task, we will download the specific Amazon EKS kubectl binary for your cluster's Kubernetes version from Amazon S3 using the curl command. Then, give execute permissions

to the binary with `chmod`, and copy it to a folder in your `PATH` with `mkdir` and `cp` commands.

1. Once the environment is ready on CloudShell, Download the Amazon EKS vended kubectl binary for your cluster's Kubernetes version from Amazon S3. To do so, run the following command:

```
curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
```



2. Grant execution permissions to the binary.

```
chmod +x ./kubectl
```



3. Copy the binary to a folder in your `PATH`. If you have already installed a version of kubectl, then we recommend creating a `$HOME/bin/kubectl` and ensuring that `$HOME/bin` comes first in your `$PATH`.

```
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl &&  
export PATH=$PATH:$HOME/bin
```



4. After you install kubectl, you can verify its version with the following command:

```
kubectl version --short --client
```



Task 11: Configure your AWS CloudShell to communicate with your cluster

1. Once the environment is ready on CloudShell, you create a kubeconfig file for your cluster. The settings in this file enable the kubectl CLI to communicate with your cluster.
2. To create a kubeconfig file, run the following command:

```
aws eks update-kubeconfig --region us-east-1 --name whiz
```



Note: If your cluster name is not `whiz`, Update your cluster name in the above command.

3. Test your configuration, with the following command:

```
kubectl get svc
```

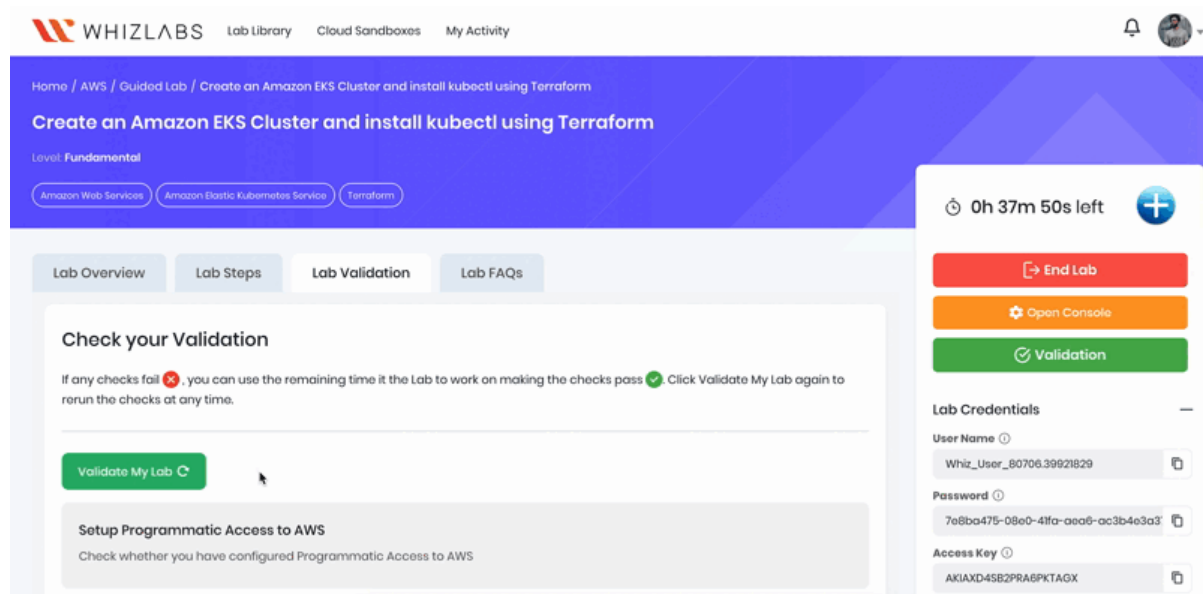


```
[cloudshell-user@ip-10-0-105-161 ~]$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	7m21s

Task 12: Validation Test

1. Once the lab steps are completed, please click on the **Validation** button on the left side panel.
2. This will validate the resources in the AWS account and shows you whether you have completed this lab successfully or not.
3. Sample output :



Task 13: Delete AWS Resources

1. To delete the resources, open Terminal again.
2. Run the below command to delete all the resources.

```
terraform destroy
```



3. Approve the creation of all the resources by entering **yes**. You can see the **Destroy complete!** message.

Enter a value: yes

```
aws_eks_cluster.cluster: Destroying... [id=whiz]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 10s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 20s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 30s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 40s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 50s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 1m0s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 1m10s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 1m20s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 1m30s elapsed]
aws_eks_cluster.cluster: Still destroying... [id=whiz, 1m40s elapsed]
aws_eks_cluster.cluster: Destruction complete after 1m49s

Destroy complete! Resources: 1 destroyed.
```

Completion and Conclusion

- You have successfully set up the Visual Studio Code editor.
- You have successfully created variables.tf and terraform.tfvars files.
- You have successfully created an EKS cluster using terraform
- You have successfully created output.tf
- You have successfully executed the terraform configuration commands to create the resources.
- You have successfully checked all the resources created by opening the Console.
- You have successfully installed Kubectl in AWS Cloudshell.
- You have successfully configured AWS Cloudshell to communicate with AWS EKS Cluster.
- You have successfully deleted all the resources.

End Lab

1. Sign out of AWS Account.
2. You have successfully completed the lab.
3. Once you have completed the steps, click on **End Lab** from your whizlabs dashboard.



