**ntroduction**

Hey there, Gurus! Welcome to the lab! This lab will demonstrate how to set up a small Kubernetes cluster and deploy two NGINX nodes using Terraform. You, as an admin, are being asked to deploy a simple Kubernetes cluster running two NGINX nodes to test out a new web application. You will need to, first, set up the cluster using kind and kubectl, and then use Terraform to deploy the NGINX nodes.

**Solution**

Log in to the terminal using the credentials provided in the lab details:

ssh cloud\_user@<PUBLIC\_IP\_ADDRESS>

**Create Your Kubernetes Cluster**

1. View the contents of your current directory:

ls

1. Change into the *lab-deploy-kubernetes* directory:

cd lab-deploy-kubernetes/

1. View the contents of the *lab-deploy-kubernetes* directory:

ls

You should see three files: kind-config,yaml, kubernetes.tf, and terraform.tfvars.

1. Create your Kubernetes cluster:

kind create cluster --name lab-terraform-kubernetes --config kind-config.yaml

When the cluster is successfully created, you should see all the creation steps have green checkmarks, and you receive a 'Have a nice day!' message.

1. Copy the provided command and paste it in the terminal:

kubectl cluster-info --context kind-lab-terraform-kubernetes

1. Verify your cluster was created:

kind get clusters

**Configure Terraform for Use with the Kubernetes Cluster**

1. Run kubectl to get information about your cluster:

kubectl config view --minify --flatten --context=kind-lab-terraform-kubernetes

1. Add the server address to your terraform.tfvars file:
   * Scroll up to the *server* line and copy the server address.
   * Edit the terraform.tfvars file:

vim terraform.tfvars

* + On the *host* line, enter insert mode and replace DUMMY VALUE with your copied server address.
  + Press **ESC** to exit insert mode, then write and quit to save your changes:

:wq!

1. Add the client certificate data to your terraform.tfvars file:
   * Scroll up to the *client-certificate-data* line and copy the full certificate details.
   * Edit the terraform.tfvars file:

vim terraform.tfvars

* + On the *client\_certificate* line, enter insert mode and replace DUMMY VALUE with your copied client certificate data.
  + Press **ESC** to exit insert mode, then write and quit to save your changes:

:wq!

1. Add the client key data to your terraform.tfvars file:
   * Scroll up to the *client-key-data* line and copy the full client key details.
   * Edit the terraform.tfvars file:

vim terraform.tfvars

* + On the *client\_key* line, enter insert mode and replace DUMMY VALUE with your copied client key data.
  + Press **ESC** to exist insert mode, then write and quit to save your changes:

:wq!

1. Add the certificate authority data to your terraform.tfvars file:
   * Scroll up to the *certificate-authority-data* line and copy the full certificate authority details.
   * Edit the terraform.tfvars file:

vim terraform.tfvars

* + On the *cluster\_ca\_certificate* line, enter insert mode and replace DUMMY VALUE with your copied client authority data.
  + Press **ESC** to exit insert mode, then write and quit to save your changes:

:wq!

1. View your kubernetes.tf file:

vim kubernetes.tf

You can see this configuration file pulls from the terraform.tfvars file to declare the variables and then pass them to the Kubernetes provider.

1. Quit out of the file when you're finished reviewing it:

:q

**Deploy Resources to the Kubernetes Cluster**

1. Initialize your working directory:

terraform init

1. Download the Terraform configuration file provided in the lab's resources:

wget https://raw.githubusercontent.com/linuxacademy/content-terraform-2021/main/lab\_kubernetes\_resources.tf

1. View the contents of your current directory:

ls

You should see a new configuration file called lab\_kubernetes\_resources.tf.

1. View your lab\_kubernetes\_resources.tf file:

vim lab\_kubernetes\_resources.tf

You have two resources: "kubernetes\_deployment" and "nginx", with the name "long-live-the-bat". You also have two replicas. You're using the NGINX 1.7.8 image and a container named "batman".

1. Quit out of the file when you're finished reviewing it:

:q

1. Plan your configuration changes:

terraform plan

1. Apply your configuration changes:

terraform apply

1. On the *Enter a value* line, type "yes" to confirm the apply. The application takes a moment to complete.
2. View your deployment details:

kubectl get deployments

You should see your deployment, "long-live-the-bat" has two nodes up and running.

# Manage Kubernetes Resources with Terraform

## Introduction

Hey there, Gurus! Welcome to the lab! This lab will demonstrate how to change, manage, and delete resources in Kubernetes using Terraform. You, as an admin, are being asked to deploy a NodePort service to your Kubernetes cluster to go along with your NGINX nodes. You will need to first set up the cluster using kind and kubectl, and deploy the NGINX nodes using Terraform. You will then use Terraform to deploy the NodePort service, and scale your NGINX nodes from 2 to 4 nodes. You will finish by destroying your resource and deleting your cluster.

## Solution

Log in to the terminal using the credentials provided for the lab:

ssh cloud\_user@[PUBLIC\_IP\_ADDRESS]

### Set Up the Lab Environment

1. View the contents of your current directory:

ls

You should see lab-manage-kubernetes and lab-manage-kubernetes.zip.

1. Change into the lab-manage-kubernetes directory:

cd lab-manage-kubernetes/

1. View the contents of that directory:

ls

You should see 4 files: kind-config.yaml, kubernetes.tf, lab\_kubernetes\_resources.tf, and terraform.tfvars.

1. Create a cluster:

kind create cluster --name lab-terraform-kubernetes --config kind-config.yaml

1. After the cluster is created, copy and run the command provided in the terminal to configure kubetctl:

kubectl cluster-info --context kind-lab-terraform-kubernetes

1. Confirm the lab-terraform-kubernetes cluster was created:

kind get clusters

1. Edit the cluster's host address:
   * Run kubectl to view the cluster's host address:

kubectl config view --minify --flatten --context=kind-lab-terraform-kubernetes

* + Copy the server address.
  + Edit the variables file:

vim terraform.tfvars

* + On the host line, replace [DUMMY VALUE] with your copied server address.
  + Write and quit to save the file:

:wq!

1. Edit the cluster's SSL certificate:
   * Copy the full client-certificate-data details.
   * Edit the variables file:

vim terraform.tfvars

* + On the client\_certificate line, replace [DUMMY VALUE] with your copied client-certificate-data details.
  + Write and quit to save the file:

:wq!

1. Edit the cluster's client key data:
   * Copy the full client-key-data details.
   * Edit the variables file:

vim terraform.tfvars

* + On the client\_key line, replace [DUMMY VALUE] with your copied client-key-data details.
  + Write and quit to save the file:

:wq!

1. Edit the cluster's certificate authority data:
   * Copy the full certificate-authority-data details.
   * Edit the variables file:

vim terraform.tfvars

* + On the cluster\_ca\_certificate line, replace [DUMMY VALUE] with your copied certificate-authority-data details.
  + Write and quit to save the file:

:wq!

1. Initialize your working directory:

terraform init

1. Deploy your resources to your Kubernetes cluster:

terraform apply

1. When prompted, type "yes" to confirm the deployment. The deployment takes a moment to complete.

### Add a Service

1. Download the Terraform configuration file provided in the lab's resources:

wget https://raw.githubusercontent.com/linuxacademy/content-terraform-2021/main/lab\_kubernetes\_service.tf

1. View the contents of that directory:

ls

In addition to the previous 4 files, you should now see 2 additional files: lab\_kubernetes\_service.tf and terraform.tfstate.

1. View the lab\_kubernetes\_service.tf file:

vim lab\_kubernetes\_service.tf

1. Quit out of the file when you're finished reviewing it:

:q

1. Apply the added configuration:

terraform apply

1. On the Enter a value line, type "yes" to confirm the apply. The application takes a moment to complete.
2. Verify the NodePort service was applied successfully:

kubectl get services

You should see the NodePort service named robin listed in your services.

### Scale the Nodes

1. Edit the lab\_kubernetes\_resources.tf file:

vim lab\_kubernetes\_resources.tf

1. Modify the replicas from 2 to 4:
2. spec {
3. replicas = 4
4. selector {
5. match\_labels = {
6. App = "longlivethebat"
7. }

}

1. Write and quit to save your change to the file:

:wq!

1. Apply your changes:

terraform apply

1. Confirm the replicas were changed from 2 to 4, then type "yes" on the Enter a value line to confirm the apply.
2. Verify your deployment is now using 4 pods:

kubectl get deployments

### Delete Your Cluster

1. Destroy your resources, including your NGINX deployment and NodePort service:

terraform destroy

1. On the Enter a value line, type "yes" to confirm the destroy. The destroy takes a moment to complete.
2. Delete your cluster:

kind delete cluster --name lab-terraform-kubernetes

1. Verify the cluster was deleted:

kind get clusters