**Using Terraform**

DANGER

Creating the workshop cluster with Terraform is currently in preview. Please raise any issues encountered in the [GitHub repository](https://github.com/aws-samples/eks-workshop-v2/issues).

This section outlines how to build a cluster for the lab exercises using the [Hashicorp Terraform](https://developer.hashicorp.com/terraform" \t "_blank). This is intent to be for learners that are used work with Terraform infrastructure-as-code.

The terraform CLI has been pre-installed in your Amazon Cloud9 Environment, so we can immediately create the cluster. Lets take a look at the main Terraform configuration files that will be used to build the cluster and its supporting infrastructure.

The providers.tf file configures the Terraform providers that will be needed to build the infrastructure. In our case we use the aws, kubernetes and helm providers:

provider "aws" {  
 default\_tags {  
 tags = local.tags  
 }  
}  
  
terraform {  
 required\_providers {  
 aws = {  
 source = "hashicorp/aws"  
 version = ">= 4.67.0"  
 }  
 }  
  
 required\_version = ">= 1.4.2"  
}

The main.tf file sets up some Terraform data sources so we can retrieve the current AWS account and region being used, as well as some default tags:

locals {  
 tags = {  
 created-by = "eks-workshop-v2"  
 env = var.cluster\_name  
 }  
}

The vpc.tf configuration will make sure our VPC infrastructure is created:

locals {  
 private\_subnets = [for k, v in local.azs : cidrsubnet(var.vpc\_cidr, 3, k + 3)]  
 public\_subnets = [for k, v in local.azs : cidrsubnet(var.vpc\_cidr, 3, k)]  
 azs = slice(data.aws\_availability\_zones.available.names, 0, 3)  
}  
  
data "aws\_availability\_zones" "available" {  
 state = "available"  
}  
  
module "vpc" {  
 source = "terraform-aws-modules/vpc/aws"  
 version = "~> 5.1"  
  
 name = var.cluster\_name  
 cidr = var.vpc\_cidr  
  
 azs = local.azs  
 public\_subnets = local.public\_subnets  
 private\_subnets = local.private\_subnets  
 public\_subnet\_suffix = "SubnetPublic"  
 private\_subnet\_suffix = "SubnetPrivate"  
  
 enable\_nat\_gateway = true  
 create\_igw = true  
 enable\_dns\_hostnames = true  
 single\_nat\_gateway = true  
  
 # Manage so we can name  
 manage\_default\_network\_acl = true  
 default\_network\_acl\_tags = { Name = "${var.cluster\_name}-default" }  
 manage\_default\_route\_table = true  
 default\_route\_table\_tags = { Name = "${var.cluster\_name}-default" }  
 manage\_default\_security\_group = true  
 default\_security\_group\_tags = { Name = "${var.cluster\_name}-default" }  
  
 public\_subnet\_tags = merge(local.tags, {  
 "kubernetes.io/role/elb" = "1"  
 })  
 private\_subnet\_tags = merge(local.tags, {  
 "karpenter.sh/discovery" = var.cluster\_name  
 })  
  
 tags = local.tags  
}

Finally the eks.tf file specifies our EKS cluster configuration, including a Managed Node Group:

module "eks" {  
 source = "terraform-aws-modules/eks/aws"  
 version = "~> 19.16"  
  
 cluster\_name = var.cluster\_name  
 cluster\_version = var.cluster\_version  
 cluster\_endpoint\_public\_access = true  
  
 cluster\_addons = {  
 vpc-cni = {  
 before\_compute = true  
 most\_recent = true  
 configuration\_values = jsonencode({  
 env = {  
 ENABLE\_POD\_ENI = "true"  
 ENABLE\_PREFIX\_DELEGATION = "true"  
 POD\_SECURITY\_GROUP\_ENFORCING\_MODE = "standard"  
 }  
  
 enableNetworkPolicy = "true"  
 })  
 }  
 }  
  
 vpc\_id = module.vpc.vpc\_id  
 subnet\_ids = module.vpc.private\_subnets  
  
 create\_cluster\_security\_group = false  
 create\_node\_security\_group = false  
  
 eks\_managed\_node\_groups = {  
 default = {  
 instance\_types = ["m5.large"]  
 force\_update\_version = true  
 release\_version = var.ami\_release\_version  
  
 min\_size = 3  
 max\_size = 6  
 desired\_size = 3  
   
 update\_config = {  
 max\_unavailable\_percentage = 50  
 }  
  
 labels = {  
 workshop-default = "yes"  
 }  
 }  
 }  
  
 tags = merge(local.tags, {  
 "karpenter.sh/discovery" = var.cluster\_name  
 })  
}

For the given configuration, terraform will create the Workshop environment with the following:

* Create a VPC across three availability zones
* Create an EKS cluster
* Create an IAM OIDC provider
* Add a managed node group named default
* Configure the VPC CNI to use prefix delegation

Download the Terraform files:

~$mkdir -p ~/environment/terraform; cd ~/environment/terraform

~$curl --remote-name-all https://raw.githubusercontent.com/aws-samples/eks-workshop-v2/stable/cluster/terraform/{main.tf,variables.tf,providers.tf,vpc.tf,eks.tf}

Run the following Terraform commands to deploy your workshop environment.

~$terraform init

~$terraform apply -var="cluster\_name=$EKS\_CLUSTER\_NAME" -auto-approve

This generally takes 20-25 minutes to complete. Once the cluster is created run this command to use the cluster for the lab exercises:

~$use-cluster $EKS\_CLUSTER\_NAME

**Next Steps**[**​**](https://www.eksworkshop.com/docs/introduction/setup/your-account/using-terraform#next-steps)

Now that the cluster is ready, head to the [Getting Started](https://www.eksworkshop.com/docs/introduction/getting-started) module or skip ahead to any module in the workshop with the top navigation bar. Once you're completed with the workshop, follow the steps below to clean-up your environment.

**Cleaning Up**[**​**](https://www.eksworkshop.com/docs/introduction/setup/your-account/using-terraform#cleaning-up)

DANGER

The following demonstrates how you will later clean up resources once you have completed your desired lab exercises. These steps will delete all provisioned infrastructure.

Before deleting the Cloud9 environment we need to clean up the cluster that we set up above.

First use delete-environment to ensure that the sample application and any left-over lab infrastructure is removed:

~$delete-environment

Next delete the cluster with terraform:

~$cd ~/environment/terraform

~$terraform destroy -var="cluster\_name=$EKS\_CLUSTER\_NAME" -auto-approve