Using Terraform

DANGER

Creating the workshop cluster with Terraform is currently in preview. Please raise any issues encountered in the GitHub repository.

This section outlines how to build a cluster for the lab exercises using the Hashicorp Terraform. 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

Now that the cluster is ready, head to the 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

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
```