Setup Kubernetes cluster, services and deployment using terraform

Feature/Task JIRA	Setup Kubernetses cluster services and deployment using terraform
Document Status	Completed
Assignee	Nidhi Airi
Review	Mr. Mukul Srivastava
Approval	Mr Mukul Srivastava

Description

This task is related to set up kubernetes cluster, services and deployments using terraform.

Findings/Prerequisite

The given task requires some prerequisites that are as follows:

- The terraform needs to be installed in your system
- Then the operator must have access to Gcloud account and Gcloud successfully installed in the system
- The operator must have .JSON file that have required credentials for the project.
- The operator must have a host IP and cluster_ca_certificate of the cluster.

Strategy

- First we will write main.tf file that will be entry point in order to create kubernetes cluster after creating a VPC network and subnet associated
- Then we will run the following commands:

```
terraform init  # will initialise terraform in the repository
terraform plan   # will give all the innformation of the resources created or destroyed
terraform apply # will apply the above the resources to be added or destroyed
```

Then we will look at our gcloud account and we will be able to view our cluster running
After connecting our cluster with the cloud shell we will be able to fetch our cluster-endpoints and entry generated for our cluster
Then we will create the service and deployments associated with the cluster

- Then we will run the following commands:

```
kubectl get svc
```

```
kubectl get pods
```

We will able to view the service created for the cluster and view our running pods

Changes Implemented on Present Infra

Implementation Details

- **Setting up Terraform**

At first we will install terraform in our system using the following commands

```
wget
https://releases.hashicorp.com/terraform/0.13.6/terraform_0
```

```
   .13.6_linux_amd64.zip
2  sudo apt-get install zip -y
3  unzip terraform*.zip
4  sudo mv terraform /usr/local/bin
```

- ## Fetching the credentials of Google Cloud API

➤ In the Cloud Console, go to the **Create service account** page.
➤ Select a project.
➤ In the Service account name field, enter a name. The Cloud Console fills in the Service account ID field based on this name.
➤ In the Service account description field, enter a description. For example, Service account for quickstart.
➤ Click Create and continue.
➤ Click the Select a role field.
➤ Under Quick access, click Basic, then click Owner.
➤ Click Continue.
➤ Click Done to finish creating the service account.

Create a service account key:

➤ In the Cloud Console, click the email address for the service account that you created.
➤ Click Keys.
➤ Click **Add key**, then click Create new key.
➤ Click Create. A JSON key file is downloaded to your computer.
➤ Click Close.
➤ You need to add this JSON key file in the terraform repository

- ## Installing Gcloud on the operator system

We will also require the gcloud to be installed in our system by running the following commands

```
1  sudo snap install google-cloud-sdk --classic
2  gcloud auth login
3  sudo snap install kubectl
```

- ## Creation of files

1. First we will create main.tf file

In main .tf file first we use create a network associated with the cluster. For this we will require google provider.

Then we will create our VPC network and subnet associated with the VPC
And then we will create a cluster associated with the VPC network.

```
# Setting up providers
provider "google" {
 project     = "${var.gcp_project_id}"
 credentials = "${file("${var.credentials}")}" #file that contains cluster credentials
 region      = "${var.region}"   # region assosciated
}

# data google_container_cluster
data "google_container_cluster" "primary" {
  name     = "${var.env_name}-gke"
  location = var.region
}

# creating vpc network
resource "google_compute_network" "vpc" {
  name                    = "${var.env_name}-vpc"
  auto_create_subnetworks = "false"
}

# creating Subnet associated with vpc
resource "google_compute_subnetwork" "subnet" {
  name          = "${var.env_name}-subnet"
  region        = var.region
  network       = "${google_compute_network.vpc.name}"
  ip_cidr_range = var.cidr_range
}


# Kubernetes Cluster Creation for the Vpc network
resource "google_container_cluster" "primary" {
  name     = "${var.env_name}-gke"
  location = var.region
```

```
  project  = "${var.gcp_project_id}"
  # We can't create a cluster with no node pool defined, but we want to only use
  # separately managed node pools. So we create the smallest possible default
  # node pool and immediately delete it.
  remove_default_node_pool = true
  initial_node_count       = var.nodecount
  network    = "${google_compute_network.vpc.name}"
  subnetwork = "${google_compute_subnetwork.subnet.name}"
}

# Node pool creation
resource "google_container_node_pool" "primary_preemptible_nodes" {
  name       = "node-pool"
  location   = var.region
  cluster    = google_container_cluster.primary.name
  project     = "${var.gcp_project_id}"
  node_count = var.nodecount

  node_config {
    preemptible  = true
    machine_type = var.machine
  }
}

#setting up credentials
resource "null_resource" "get-credentials" {

 depends_on = [
   google_container_cluster.primary,
  ]
 provisioner "local-exec" {
   command = "gcloud container clusters get-credentials ${google_container_cluster.primary.name} --region=${var.region} --
```

```
    project=${var.gcp_project_id}"
 }
}

```

The variables.tf file and terraform.tfvars for this main file is below

variables.tf file

```

variable "region" {
    type =  string
    description = "region for the network"
}
variable "gcp_project_id" {
    type = string
}
variable "credentials" {
    description = " will contain the .JSON file for connecting to the Google Cloud API "
}

variable "env_name"{
    type = string
    description = "will contains the enviornment variable required to create resources"
}
variable "cidr_range" {
   type        = string
   description = "cidr ranges for subnet creation"
}
variable "nodecount"{
    type = number
    description = "will count number of nodes"
}
variable "machine" {
```

```
26     type = string
27     description = "machine name for nodes"
28 }
```

terrraform.tfvar file

```
1  gcp_project_id   = "infra-333911"
2  region           = "us-west2"
3  credentials      = "infra-33.json"
4  env_name         = "test"
5  cidr_range       = "10.10.0.0/24"
6  nodecount       = "1"
7  machine          = "e2-medium"
```

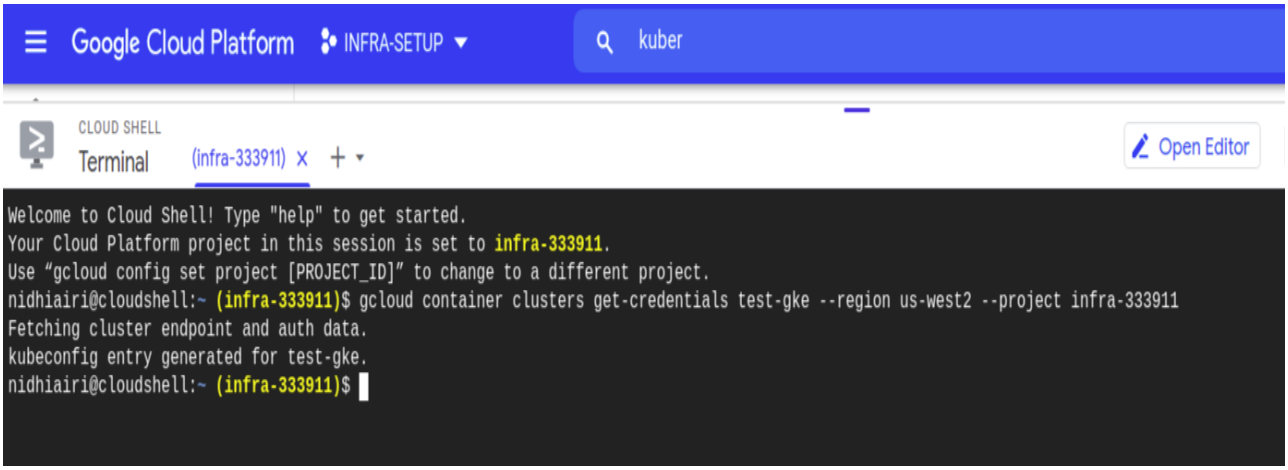
After configuring our main.tf , variables.tf and terraform.tfvars we will run the following terraform commands .

```
1  terraform init
2  terraform  plan
3  terraform apply
```

After successfully creating the vpc network we will verify whether the network is created in our project

After verifying for the Vpc network creation we will verify for the kubernetes cluster

After successful creation of the cluster we would be able to connect with our cluster by using cloud shell



With the help of the above image we can see that we are able to fetch our cluster endpoints and also the entry generated for our running cluster.

Now we will create the service for our cluster and the deployments for our cluster
Now after adding the resources kubernetes_service and kubernetes_pods the complete main.tf file would look like

```
# Setting up providers
provider "google" {
 project     = "${var.gcp_project_id}"
 credentials = "${file("${var.credentials}")}" #file that contains cluster credentials
 region      = "${var.region}"   # region assosciated
}


# data google_container_cluster
data "google_container_cluster" "primary" {
  name     = "${var.env_name}-gke"
  location = var.region
}

# creating vpc network
resource "google_compute_network" "vpc" {
  name                    = "${var.env_name}-vpc"
  auto_create_subnetworks = "false"
}

# creating Subnet assosciated with vpc
resource "google_compute_subnetwork" "subnet" {
  name          = "${var.env_name}-subnet"
  region        = var.region
  network       = "${google_compute_network.vpc.name}"
  ip_cidr_range = var.cidr_range
}

```

```
# Kubernetes Cluster Creation for the Vpc network
resource "google_container_cluster" "primary" {
  name     = "${var.env_name}-gke"
  location = var.region
  project  = "${var.gcp_project_id}"
  # We can't create a cluster with no node pool defined, but we want to only use
  # separately managed node pools. So we create the smallest possible default
  # node pool and immediately delete it.
  remove_default_node_pool = true
  initial_node_count       = var.nodecount
  network    = "${google_compute_network.vpc.name}"
  subnetwork = "${google_compute_subnetwork.subnet.name}"
}

# Node pool creation
resource "google_container_node_pool" "primary_preemptible_nodes" {
  name       = "node-pool"
  location   = var.region
  cluster    = google_container_cluster.primary.name
  project     = "${var.gcp_project_id}"
  node_count = var.nodecount

  node_config {
    preemptible  = true
    machine_type = var.machine
  }
}

#setting up credentials
resource "null_resource" "get-credentials" {

 depends_on = [
   google_container_cluster.primary,
```

```
  ]
 provisioner "local-exec" {
   command = "gcloud container clusters get-credentials ${google_container_cluster.primary.name} --region=${var.region} --project=${var.gcp_project_id}"
 }
}

// In order to create the services and deployements for cluster creation we need to uncomment the below lines

provider "kubernetes" {
   config_path    = "~/.kube/config"
    host                 = "https://${data.google_container_cluster.primary.endpoint}"
    cluster_ca_certificate = base64decode(data.google_container_cluster.primary.master_auth[0].cluster_ca_certificate)
}

#service creation
resource "kubernetes_service" "service-testing" {
  metadata {
    name = "service-test"
  }
   spec {
    port {
      port        = var.ports
      target_port = var.targetport
    }
    type = "LoadBalancer"
  }
   depends_on = [
      null_resource.get-credentials,
   ]
```

```
90 }
91 #pod creation assosciated with service
92 resource "kubernetes_pod" "pod-test" {
93   metadata {
94     name = "test-nginx"
95     labels = {
96      App = "nginx"
97     }
98   }
99   spec {
100      container {
101       image = var.imagename
102       name  = "nginx"
103       port {
104         container_port = var.containerport
105       }
106     }
107   }
108 }
```

.

After successfully adding the above resources we will run the command

```
1 terraform init
2 terraform plan
3 terraform apply
```

Validating the above steps

We will be able to view the service and the pod creation associated with our cluster.

We will run the command

```
1 kubectl get svc
```

We will be able to view our service

```
1 kubectl get pods
```

we will be able to view our running pods

Pricing

In setting terraform

- No cost involved in installing terraform

In setting up Gcloud service account

We have used Gcloud free account which will give $300 for 90 days

Questions

Questions	Outcome
How can we connect the g cloud with the terraform?	We will require the google providers as well .JSON file saved in our repository where terraform is initialized so that after running terraform init we will able to provide necessary plugins to the terraform and terraform will be able to access the gcloud account
How to create a cluster within a network?	By using the google_container_cluster we will be able to configure the cluster for our gcloud
How to connect the cluster from the system?	By runnning the command gcloud container clusters get-credentials ${google_container_cluster.primary.name} --region=${var.region} --project=${var.gcp_project}
How to create the service associated with our cluster	By using the kubernetes_service we will be able to create service associated with our cluster
How to create the pods associated with	By using the kubernetes_pods we will be

our cluster	able to create the pods assosciated with the cluster

Challenges/Blockers or Not Doing

- **Challenge 1**: We were not able to connect our system with the Gcloud account

Resolved: We have resolved the issue by running the below command

```
1 gcloud container clusters get-credentials
  ${google_container_cluster.primary.name} --
  region=${var.region} --project=${var.gcp_project}
```

- **Challenge 2:** We were not able to setup services for kubernetes cluster

Resolved: We have resolved this issue by upgrading the version of the terraform

Referred Links

- https://stackoverflow.com/questions/46434348/error-permission-denied-on-resource-project-when-launching-dataproc-cluster
- https://learn.hashicorp.com/tutorials/terraform/kubernetes-provider?in=terraform/kubernetes
- https://github.com/terraform-google-modules/terraform-google-kubernetes-engine/blob/master/examples/safer_cluster/main.tf
- https://registry.terraformhttps://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/container_cluster.io/providers/hashicorp/kubernetes/latest/docs
- https://discuss.kubernetes.io/t/deployment-failing-on-kubernetes-with-terraform-the-server-could-not-find-the-requested-resource/14557
- https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs/resources/service
- https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs/resources/pod
- https://stackoverflow.com/questions/51180147/determine-what-resource-was-not-found-from-error-from-server-notfound-the-se
- https://www.terraform.io/downloads.html