Ensure Access & Identity in Google Cloud Challenge Lab

Author: Vedant Kakde | **GitHub Profile:** github.com/vedant-kakde | **LinkedIn Profile:** linkedin.com/in/vedant-kakde/

Set a zone

```
gcloud config set compute/zone us-east1-b
```

Task 1: Create a custom security role

You will create a new custom IAM security role called `orca_storage_update` in this task. The custom role needs to include the permissions necessary to add and update objects in Google Cloud Storage buckets.

The following steps show how to create a custom role using a YAML file:

1. Create a YAML file in the Cloud Shell environment.
2. Copy the following to the YAML file.

```
title: "Edirca Storage Update"

description: "Add and update objects in Google Cloud Storage buckets"

includedPermissions:

- storage.buckets.get

- storage.objects.get

- storage.objects.list

- storage.objects.update

- storage.objects.create
```

3. Run the following command to create the custom role using the definition in the YAML file.

```
gcloud iam roles create orca_storage_update --project $DEVSHELL_PROJECT_ID --file role-definition.yaml
```

Task 2: Create a service account

This task only requires you to create a new service account named `orca-private-cluster-sa`.

Run the following command line in the Cloud Shell:

```
gcloud iam service-accounts create orca-private-cluster-sa \

   --display-name "Orca Private Cluster Service Account"
```

Task 3: Bind a custom security role to a service account

The lab requires you to bind these three built-in roles to the service account:

- `roles/monitoring.viewer`
- `roles/monitoring.metricWriter`
- `roles/logging.logWriter`

Run the following to bind them to `orca-private-cluster-sa`:

```
gcloud projects add-iam-policy-binding $DEVSHELL_PROJECT_ID \

   --member serviceAccount:orca-private-cluster-sa@$DEVSHELL_PROJECT_ID.iam.gserviceaccount.com
--role roles/monitoring.viewer




gcloud projects add-iam-policy-binding $DEVSHELL_PROJECT_ID \

   --member serviceAccount:orca-private-cluster-sa@$DEVSHELL_PROJECT_ID.iam.gserviceaccount.com
--role roles/monitoring.metricWriter




gcloud projects add-iam-policy-binding $DEVSHELL_PROJECT_ID \

   --member serviceAccount:orca-private-cluster-sa@$DEVSHELL_PROJECT_ID.iam.gserviceaccount.com
--role roles/logging.logWriter
```

Also, bind the custom role `orca_storage_update` created in Task 1 by running the following command:

```
gcloud projects add-iam-policy-binding $DEVSHELL_PROJECT_ID \

   --member serviceAccount:orca-private-cluster-sa@$DEVSHELL_PROJECT_ID.iam.gserviceaccount.com
--role projects/$DEVSHELL_PROJECT_ID/roles/orca_storage_update
```

Task 4: Create and configure a new Kubernetes Engine private cluster

The new Kubernetes Engine cluster needs to fulfil the following requirements:

- The cluster must be called `orca-test-cluster`
- The cluster must be deployed to the subnet `orca-build-subnet`
- The cluster must be configured to use the `orca-private-cluster-sa` service account.
- The private cluster options `enable-master-authorized-networks`, `enable-ip-alias`, `enable-private-nodes`, and `enable-private-endpoint` must be enabled.

1. Navigate to **Compute Engine** in the Cloud Console, note down the **Internal IP** address of the `orca-jumphost` instance.

   Alternatively, you can use the following command to obtain the IP address:

```
JUMPHOST_IP=$(gcloud compute instances describe orca-jumphost \

--format='get(networkInterfaces[0].networkIP)')
```

2. Navigate to **VPC network** in the Cloud Console, note down the IP address range for the regional subnet. You may also lookup the IP range from https://cloud.google.com/vpc/docs/vpc. The IP address for `us-east1` is `10.142.0.0`.

```
SUBNET_IP_RANGE="10.142.0.0/20"
```

3. Run the following `gcloud` command to create the cluster with the required configrations:

```
gcloud beta container clusters create orca-test-cluster \

   --network orca-build-vpc

   --subnetwork orca-build-subnet \

   --service-account=SERVICE_ACCOUNT orca-private-cluster-
sa@$DEVSHELL_PROJECT_ID.iam.gserviceaccount.com \

   --enable-master-authorized-networks $JUMPHOST_IP/32 \

   --master-authorized-networks \

   --enable-private-nodes \

   --master-ipv4-cidr $SUBNET_IP_RANGE

   --enable-ip-alias \
```

```
    --enable-private-endpoint
```

Task 5: Deploy an application to a private Kubernetes Engine cluster

The last task is to deploy a simple test application `hello-server` to the Kubernetes Engine cluster. The Cloud Shell cannot directly access the private cluster. So, you have to access it via the jumphost.

1. Navigate to **Compute Engine** in the Cloud Console.
2. Click on the **SSH** button for the `orca-jumphost` instance.
3. In the SSH window, connect to the private cluster by running the following:

```
gcloud container clusters get-credentials orca-test-cluster --internal-ip
```

4. Deploy the `hello-server` to the private cluster using the following `kubectl` command:

```
kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
```

5. Finally, run the following to expose the application using a load balancer service with mapping from port 80 to 8080:

```
kubectl expose deployment hello-server --name orca-hello-service \

    --type LoadBalancer --port 80 --target-port 8080
```

Congratulations! You completed this challenge lab.