Google Cloud Shell

Google Cloud Shell :

Cloud shell provides the following:

- Temporary Compute Engine VM
- Command-line access to the instance via a browser
- 5 GB of persistent disk storage ($HOME dir)
- Pre-installed Cloud SDK and other tools
- gcloud: for working with Compute Engine and many Google Cloud services
- gsutil: for working with Cloud Storage
- kubectl: for working with Google Kubernetes Engine and Kubernetes
- bq: for working with BigQuery
- Language support for Java, Go, Python, Node.js, PHP, and Ruby
- Web preview functionality
- Built-in authorization for access to resources and instances

Définir des variables :

```
export PROJECT_ID=XXXXXXXX
export ZONE=us-central1-a
```

Quelques notes :

LINUX :

Voici un exemple de clé RSA à entrer dans Google Cloud pour, par exemple, utiliser PUTTY sur une VM Compute Engine dans Google Cloud.

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAQEAnD6qhcMV+JPyHIeU4817qk2F5I5HeK8JcjzWsIdCx925QZ1bZ7o8Peo3AuDYTit
k0gFiLmfaJrKr3Gtt1731CsFDbOLkA/mwfdAXk9zpuEviIaODy9pOVLGmqMafsQFeETEzTxUrbkkaGqL1ELJy0OYTy+FV
```

```
sRzKWQdpnbC7DptKnLDaVsuO3TQBjOnNJk1EeXIs6LlxxRb4gQE7z+onyjIDqoscC4K/9sOdzoqk/V/ZfUVNha1K+ePM
p3pcF27ikRslhHUK384X/akrPXAmJ58S5pun54XeRAAqENrsGBX/qO4lxmd9NzDgbhtqSj3DOzM3K4gI5hJva1EYrN39
hQ== nehemie
```

Windows :

Activer le compte Administrateur :

Si vous souhaitez déployer le rôle ADDS sur un Windows Serveur dans Google Cloud, vous devez procéder aux étapes ci-dessous :

```
net user Administrator /passwordreq:yes
net user Administrator *
net user Administrator /active:yes
```

Renommer une VM :

```
gcloud beta compute instances set-name OLD --new-name=NEW
```

Afficher les zones :

```
gcloud compute zones list | grep us-central1
```

Créer une instance de VM :

```
gcloud compute instances create VM1 --machine-type n1-standard-2 --zone $ZONE --project $PROJECT_ID
gcloud compute instances describe VM1 --zone $ZONE --project $PROJECT_ID
gcloud compute ssh VM1 --zone $ZONE --project $PROJECT_ID
```

Créer un cluster :

```
gcloud container clusters create mon-cluster --project $PROJECT_ID
gcloud container clusters get-credentials mon-cluster --project $PROJECT_ID
```

```
kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
kubectl expose deployment hello-server --type=LoadBalancer --port 8080
```

Supprimer un cluster :

```
gcloud container clusters delete mon-cluster --project $PROJECT_ID
```

Équilibrage de charge HTTP et Réseau :

```
gcloud config set project IDPROJET

gcloud config set compute/zone us-central1-a
gcloud config set compute/region us-central1

gcloud compute instances create www1 \
  --image-family debian-9 \
  --image-project debian-cloud \
  --zone us-central1-a \
  --tags network-lb-tag \
  --metadata startup-script="#! /bin/bash
    sudo apt-get update
    sudo apt-get install apache2 -y
    sudo service apache2 restart
    echo '<!doctype html><html><body><h1>www1</h1></body></html>' | tee /var/www/html/index.html"


gcloud compute instances create www2 \
  --image-family debian-9 \
  --image-project debian-cloud \
  --zone us-central1-a \
  --tags network-lb-tag \
  --metadata startup-script="#! /bin/bash
    sudo apt-get update
    sudo apt-get install apache2 -y
    sudo service apache2 restart
    echo '<!doctype html><html><body><h1>www2</h1></body></html>' | tee /var/www/html/index.html"
```

```
gcloud compute instances create www3 \
  --image-family debian-9 \
  --image-project debian-cloud \
  --zone us-central1-a \
  --tags network-lb-tag \
  --metadata startup-script="#! /bin/bash
    sudo apt-get update
    sudo apt-get install apache2 -y
    sudo service apache2 restart
    echo '<!doctype html><html><body><h1>www3</h1></body></html>' | tee /var/www/html/index.html"
```

Règle de parefeu pour autorisé le trafique entrant :

```
gcloud compute firewall-rules create www-firewall-network-lb \
    --target-tags network-lb-tag --allow tcp:80
```

Afficher les instances :

```
gcloud compute instances list
```

www1 : 34.133.245.113
www2 : 104.154.73.144
www3 : 34.71.227.111

Vérification du bon fonctionnement :

```
curl 34.133.245.113
curl 104.154.73.144
curl 34.71.227.111
```

Tâche 3 : Configurer le service d'équilibrage de charge

Création d'une adresse IP Externe pour l'équilibrage ::

```
gcloud compute addresses create network-lb-ip-1 \
 --region us-central1
```

Ajout d'une ancienne ressource :

```
gcloud compute http-health-checks create basic-check
```

Ajout du pool cible :

```
gcloud compute target-pools create www-pool \
    --region us-central1 --http-health-check basic-check
```

Ajoutez les instances au pool :

```
gcloud compute target-pools add-instances www-pool \
    --instances www1,www2,www3
```

Ajout d'une règle de transfet :

```
gcloud compute forwarding-rules create www-rule \
    --region us-central1 \
    --ports 80 \
    --address network-lb-ip-1 \
    --target-pool www-pool
```

Saisissez la commande suivante pour afficher l'adresse IP externe définie dans la règle de transfert www-rule utilisée par l'équilibreur de charge :

```
gcloud compute forwarding-rules describe www-rule --region us-central1
```

```
while true; do curl -m1 35.225.251.108; done
```

Tâche 5 : Créer un équilibreur de charge HTTP

```
gcloud compute instance-templates create lb-backend-template \
   --region=us-central1 \
   --network=default \
   --subnet=default \
```

```
  --tags=allow-health-check \
  --image-family=debian-9 \
  --image-project=debian-cloud \
  --metadata=startup-script='#! /bin/bash
    apt-get update
    apt-get install apache2 -y
    a2ensite default-ssl
    a2enmod ssl
    vm_hostname="$(curl -H "Metadata-Flavor:Google" \
    http://169.254.169.254/computeMetadata/v1/instance/name)"
    echo "Page served from: $vm_hostname" | \
    tee /var/www/html/index.html
    systemctl restart apache2'
```

Commencez par créer le modèle de l'équilibreur de charge :

```
gcloud compute instance-groups managed create lb-backend-group \
  --template=lb-backend-template --size=2 --zone=us-central1-a
```

Créez la règle de pare-feu fw-allow-health-check. Il s'agit d'une règle d'entrée qui autorise le trafic provenant des systèmes de vérification d'état Google Cloud (130.211.0.0/22 et 35.191.0.0/16). Cet exemple utilise le tag cible allow-health-check pour identifier les VM.

```
gcloud compute firewall-rules create fw-allow-health-check \
    --network=default \
    --action=allow \
    --direction=ingress \
    --source-ranges=130.211.0.0/22,35.191.0.0/16 \
    --target-tags=allow-health-check \
    --rules=tcp:80
```

Maintenant que vos instances sont opérationnelles, configurez une adresse IP externe statique globale que vos clients utiliseront pour accéder à votre équilibreur de charge.

```
gcloud compute addresses create lb-ipv4-1 \
    --ip-version=IPV4 \
    --global
```

Savoir quelle adresse :

```
gcloud compute addresses describe lb-ipv4-1 \
    --format="get(address)" \
```

```
    --global
```

= 34.117.115.143

Créez une vérification d'état pour l'équilibreur de charge :

```
    gcloud compute health-checks create http http-basic-check \
        --port 80
```

Créer un service de backend :

```
    gcloud compute backend-services create web-backend-service \
        --protocol=HTTP \
        --port-name=http \
        --health-checks=http-basic-check \
        --global
```

Ajoutez votre groupe d'instances en tant que backend au service de backend :

```
    gcloud compute backend-services add-backend web-backend-service \
        --instance-group=lb-backend-group \
        --instance-group-zone=us-central1-a \
        --global
```

Créez un mappage d'URL pour acheminer les requêtes entrantes vers le service de backend par défaut :

```
    gcloud compute url-maps create web-map-http \
        --default-service web-backend-service
```

Créez un proxy HTTP cible, qui va acheminer les requêtes vers votre mappage d'URL :

```
    gcloud compute target-http-proxies create http-lb-proxy \
        --url-map web-map-http
```

Créez une règle de transfert globale pour acheminer les requêtes entrantes vers le proxy :

```
    gcloud compute forwarding-rules create http-content-rule \
        --address=lb-ipv4-1\
        --global \
        --target-http-proxy=http-lb-proxy \
        --ports=80
```

Google Kubernetes Engine :

Définition de la zone de préférence GCP (Google Cloud Project) :

```
export MY_ZONE=us-central1-a
```

Création d'un cluster avec deux noeuds :

```
gcloud container clusters create webfrontend --zone $MY_ZONE --num-nodes 2
```

Déploiement d'une image :

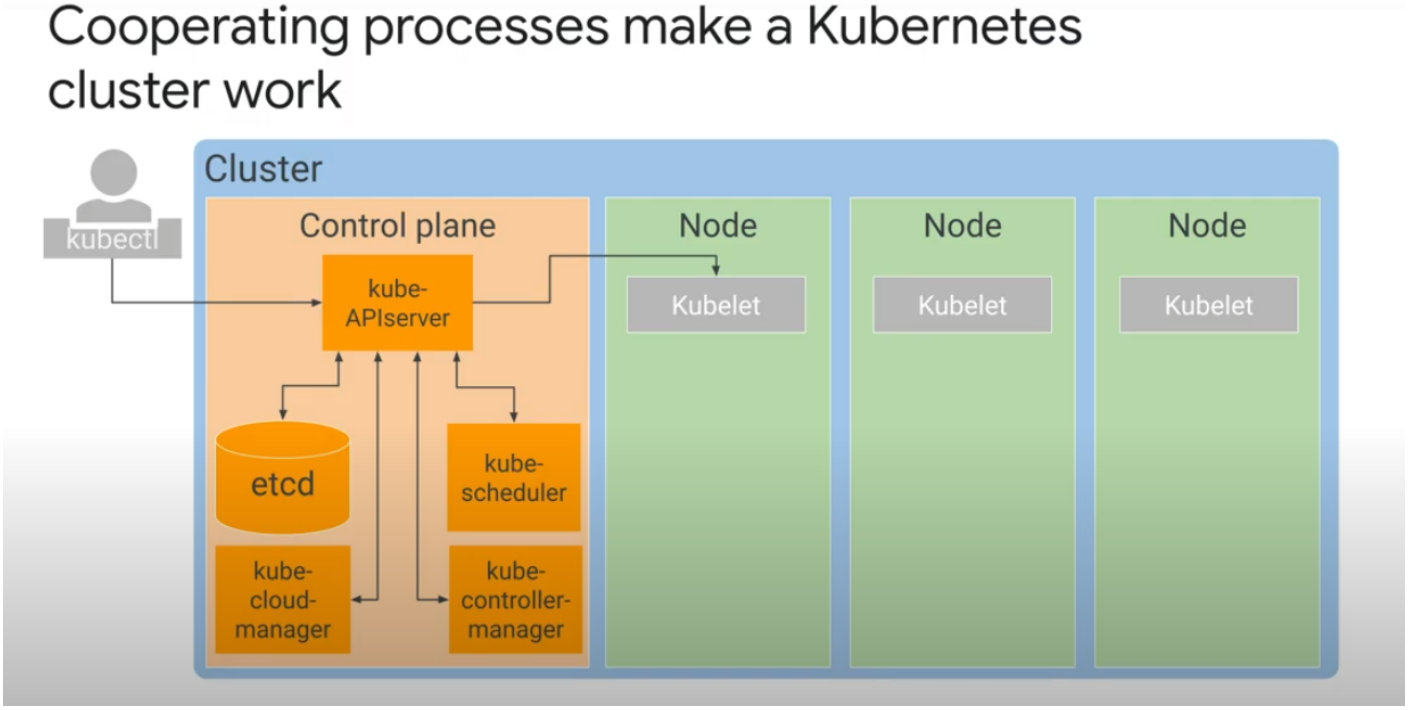
```
kubectl create deploy nginx --image=nginx:1.17.10
```

Exposition du service :

```
kubectl expose deployment nginx --port 80 --type LoadBalancer
```

Définition des nombre de PODs qui s'exécutent sur le service :

```
kubectl scale deployment nginx --replicas 3
```



Deployment Manager and Stackdriver :

Deployment Manager = Création d'un déploiement et maintien de celui-ci

Stackdriver = Monitoring

Créer un déploiement :

Fichier mydeploy.yml

```
resources:
- name: my-vm
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType: zones/us-central1-a/machineTypes/n1-standard-1
    metadata:
      items:
      - key: startup-script
        value: "apt-get update; apt-get install nginx-light -y"
    disks:
    - deviceName: boot
      type: PERSISTENT
      boot: true
      autoDelete: true
      initializeParams:
        sourceImage: https://www.googleapis.com/compute/v1/projects/debian-cloud/global/images/debian-9-
stretch-v20201216
    networkInterfaces:
    - network: global/networks/default
```

Lancer le déploiement :

```
gcloud deployment-manager deployments create my-first-depl --config mydeploy.yaml
```

Mettre à jour un déploiement :

```
gcloud deployment-manager deployments update my-first-depl --config mydeploy.yaml
```

Stackdriver :

Eteindre la machine virtuelle, définir Allow full access to all Cloud APIs dans le menu Édition de celle-ci en utilisant le Service account dropdown. Puis rallumer la machine.

Installation des agents sur la machine :

```
curl -sSO https://dl.google.com/cloudagents/install-monitoring-agent.sh
sudo bash install-monitoring-agent.sh

curl -sSO https://dl.google.com/cloudagents/install-logging-agent.sh
sudo bash install-logging-agent.sh
```

Puis, dans le menu Monitoring, nous pouvons allez dans les **metrics**, les données de la VM (processeur, ram, etc) sont automatiquement remontées.

Procéder à un déploiement automatique avec deployment-manager :

```
nano instance-template.jinja
```

```
resources:
- name: {{ env["name"] }}
  type: compute.v1.instance
  properties:
    machineType: zones/{{ properties["zone"] }}/machineTypes/{{ properties["machineType"] }}
    zone: {{ properties["zone"] }}
    networkInterfaces:
     - network: {{ properties["network"] }}
       subnetwork: {{ properties["subnetwork"] }}
```

```
      accessConfigs:
      - name: External NAT
        type: ONE_TO_ONE_NAT
    disks:
     - deviceName: {{ env["name"] }}
       type: PERSISTENT
       boot: true
       autoDelete: true
       initializeParams:
         sourceImage: https://www.googleapis.com/compute/v1/projects/debian-
cloud/global/images/family/debian-9
```

```
nano config.yaml
```

```
  imports:
  - path: instance-template.jinja

  resources:
  # Create the auto-mode network
  - name: mynetwork
    type: compute.v1.network
    properties:
      autoCreateSubnetworks: true

  # Create the firewall rule
  - name: mynetwork-allow-http-ssh-rdp-icmp
    type: compute.v1.firewall
    properties:
      network: $(ref.mynetwork.selfLink)
      sourceRanges: ["0.0.0.0/0"]
      allowed:
      - IPProtocol: TCP
        ports: [22, 80, 3389]
      - IPProtocol: ICMP

   # Create the mynet-us-vm instance
  - name: mynet-us-vm
    type: instance-template.jinja
    properties:
      zone: us-central1-a
```

```
      machineType: n1-standard-1
      network: $(ref.mynetwork.selfLink)
      subnetwork: regions/us-central1/subnetworks/mynetwork

  # Create the mynet-eu-vm instance
  - name: mynet-eu-vm
    type: instance-template.jinja
    properties:
      zone: europe-west1-d
      machineType: n1-standard-1
      network: $(ref.mynetwork.selfLink)
      subnetwork: regions/europe-west1/subnetworks/mynetwork
```

Préparation au déploiement :

```
gcloud deployment-manager deployments create dminfra --config=config.yaml --preview
```

Déploiement :

```
gcloud deployment-manager deployments update dminfra
```

Procéder à un déploiement automatique avec Terraform :

Création du dossier pour la configuration :

```
mkdir tfinfra ; cd tfinfra
```

Définition du provider :

```
nano provider.tf
```

```
provider "google" {}
```

Définition du réseau à créer :

```
nano mynetwork.tf
```

```
# Create the mynetwork network
resource "google_compute_network" "mynetwork" {
  name                    = "mynetwork"
  auto_create_subnetworks = true
}

# Add a firewall rule to allow HTTP, SSH, RDP, and ICMP traffic on mynetwork
resource "google_compute_firewall" "mynetwork-allow-http-ssh-rdp-icmp" {
  name    = "mynetwork-allow-http-ssh-rdp-icmp"
  network = google_compute_network.mynetwork.self_link

  allow {
    protocol = "tcp"
    ports    = ["22", "80", "3389"]
  }
  allow {
    protocol = "icmp"
  }
  source_ranges = ["0.0.0.0/0"]
}

# Create the mynet-us-vm instance
module "mynet-us-vm" {
  source           = "./instance"
  instance_name    = "mynet-us-vm"
  instance_zone    = "us-central1-a"
  instance_network = google_compute_network.mynetwork.self_link
}

# Create the mynet-eu-vm" instance
module "mynet-eu-vm" {
  source           = "./instance"
  instance_name    = "mynet-eu-vm"
  instance_zone    = "europe-west1-d"
  instance_network = google_compute_network.mynetwork.self_link
}
```

Définition de l'instance à créer :

```
nano instances/main.tf
```

```
variable "instance_name" {}
variable "instance_zone" {}
variable "instance_type" {
  default = "n1-standard-1"
  }
variable "instance_network" {}

resource "google_compute_instance" "vm_instance" {
  name         = "${var.instance_name}"
  zone         = "${var.instance_zone}"
  machine_type = "${var.instance_type}"
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-9"
      }
  }
  network_interface {
    network = "${var.instance_network}"
    access_config {
      # Allocate a one-to-one NAT IP to the instance
    }
  }
}
```

Initialisation de terraforma :

```
terraform init
```

Affichage du plan :

```
terraform plan
```

Réaliser le déploiement :

```
terraform apply
```

LA CLOUD SHELL NE FONCTIONNE PAS ?

--> Un bloqueur de pub peut l'handicaper !!!

Désactivez addblock !

Revision #4
Created 5 August 2022 09:22:58 by Nehemie
Updated 19 March 2023 14:47:48 by Nehemie