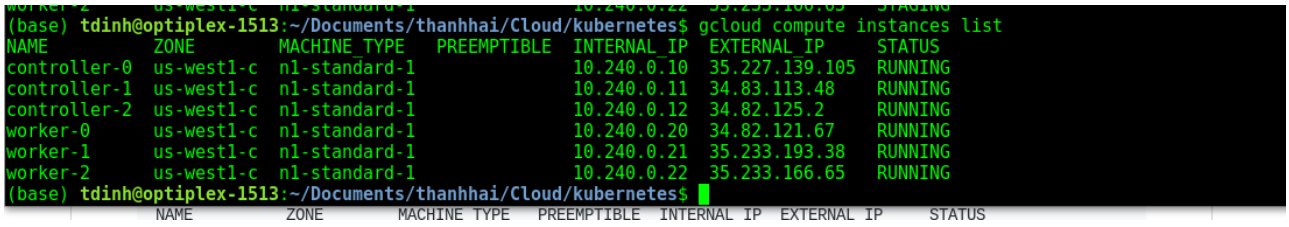
TD Kubernetes

DINH THANH HAI ASL 5A STI

Questions

1. Listez vos instances

```
gcloud compute instances list
```



2. Quels certificats créez-vous et pourquoi ?

   J'ai crée des certificats comme ça:

   - **Certificate Authority:** est utilisé pour générer des certificats TLS supplémentaires.
   - **Client and Server Certificates**: sont utilisé pour identifier un serveur et un client.
   - **The Kubelet Client Certificates**: créer un certificat pour chaque kubernetes worker node qui répond aux exigences de Node Authorizer.
   - **The Controller Manager Client Certificate**: controller le client management
   - **The Kube Proxy Client Certificate**: permettre manager le proxy
   - **The Scheduler Client Certificate**: permettre le planificateur du client.
   - **The Kubernetes API Server Certificate**: ensurer le certificat Kubernetes API Server qui peut etre validé par client distance.

3. Quelle est l'adress publique de votre cluster?

```
echo ${KUBERNETES_PUBLIC_ADDRESS}
```

   Résultat: 34.82.0.73

4. A quoi servent les fichiers kubeconfig ?

   Les fichiers kubeconfig sont ce qui configurer l'accès à plusieurs clusters. Notre clusters, utilisateurs et contextes sont défini dans un ou plusieurs fichiers configurations, on peut rapidement changer entre les clusters en utilisant kubeconfig.

```
(base) tdinh@optiplex-1513:~/Documents/thanhhai/Cloud/kubernetes$ ls -l *.kubeconfig
-rw------- 1 tdinh tdinh 6261 déc.   3 14:07 admin.kubeconfig
-rw------- 1 tdinh tdinh 6391 déc.   3 14:07 kube-controller-manager.kubeconfig
-rw------- 1 tdinh tdinh 6318 déc.   3 14:06 kube-proxy.kubeconfig
-rw------- 1 tdinh tdinh 6341 déc.   3 14:07 kube-scheduler.kubeconfig
-rw------- 1 tdinh tdinh 6380 déc.   3 14:06 worker-0.kubeconfig
-rw------- 1 tdinh tdinh 6380 déc.   3 14:06 worker-1.kubeconfig
-rw------- 1 tdinh tdinh 6380 déc.   3 14:06 worker-2.kubeconfig
(base) tdinh@optiplex-1513:~/Documents/thanhhai/Cloud/kubernetes$
```

5. Donnez en exemple le kubeconfig pour l'un de vos worker nodes.

   Générer un kubeconfig pour le node worker *worker-0*:

```
kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --embed-certs=true \
    --server=https://${KUBERNETES_PUBLIC_ADDRESS}:6443 \
    --kubeconfig=worker-0.kubeconfig

kubectl config set-credentials system:node:${instance} \
    --client-certificate=${instance}.pem \
    --client-key=worker-0-key.pem \
    --embed-certs=true \
    --kubeconfig=worker-0.kubeconfig

kubectl config set-context default \
    --cluster=kubernetes-the-hard-way \
    --user=system:node:worker-0 \
    --kubeconfig=worker-0.kubeconfig

kubectl config use-context default --kubeconfig=worker-0.kubeconfig
```

   - **kubectl config set-cluster:** Définit un cluster dans kubeconfig.
   - **kubectl config set-credentials:** Définit un utilisateur dans kubeconfig.
   - **kubectl config set-context:** Définit un contexte dans kubeconfig.

6. Que sont des Kubernetes Secrets ? A quoi servent-ils ?

   Kubernetes Secret nous permet stocker et manager des sensibles information comme le mot de pass, OAuth tokens, et ssh keys. Dans notre TD, on génére une clé de chiffrement base64 au hasard par le fonction urandom.

```
base) tdinh@optiplex-1513:~/Documents/thanhhai/Cloud/kubernetes$ ENCRYPTION_KEY=$(head -c 32 /dev/urandom | base64)
```

```
(base) tdinh@optiplex-1513:~/Documents/thanhhai/Cloud/kubernetes$ echo $ENCRYPTION_KEY=
D6T2k1HBdTuRWOC1xXPeGmVKTqXfs5BH9bUiDRcnFjY==
```

7. Qu'est-ce qu'un cluster etcd ? A quoi sert-il pour un cluster k8s ?

   Un cluster etcd est un système distribué, open-source pour faire le stockage des values key-value. K8s utilise le etcd pour stocker, mettre à jour et dupliquer ses données. Etcd est trop important, toutes des données de cluster sont stocké là-bas.

8. Quels services installez-vous sur le Control Plane ? Quel est leur rôle ?

Les services on a installé sur le Control Plane sont:

- Kubernetes API Server: valide et configure les données pour l'api objet qui contient le pods, les services, etc ... Il donne l'acces à Kubernetes API où la ressource Kubernetes est publiée.
- Scheduler: après qu'un utilisateur a crée un pod, le k8s scheduler affectera assigner le pod à un Node sur la base de certaines critièrers, i.e la disponible des ressources, comment les ressources sont nécessaires, ...
- Controller Manager: est un démon qui intègre les principales boucles de contrôle fournies avec Kubernetes. Il fonctionne des controlleurs pour manager des clusters.

9. Pourquoi devez vous installer nginx?

   Parce que load balancer de réseau prend en charge tout seul le HTTP health check mais notre server API utilise la protocole HTTPS donc ce "load balancer" ça marche pas. Pour résoudre ce problème, on déploye le nginx (est un web server fort).

```
> }
tdinh@controller-0:~$ kubectl get componentstatuses --kubeconfig admin.kubeconfig
NAME                 STATUS    MESSAGE             ERROR
controller-manager   Healthy   ok
etcd-0               Healthy   {"health":"true"}
scheduler            Healthy   ok
etcd-1               Healthy   {"health":"true"}
etcd-2               Healthy   {"health":"true"}
tdinh@controller-0:~$
```

```
[Install]
WantedBy=multi-user.target
tdinh@controller-1:~$ {
>   sudo systemctl daemon-reload
>   sudo systemctl enable kube-apiserver kube-controller-manager kube-scheduler
>   sudo systemctl start kube-apiserver kube-controller-manager kube-scheduler
> }
tdinh@controller-1:~$ kubectl get componentstatuses --kubeconfig admin.kubeconfig
NAME                 STATUS    MESSAGE             ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-1               Healthy   {"health":"true"}
etcd-2               Healthy   {"health":"true"}
etcd-0               Healthy   {"health":"true"}
tdinh@controller-1:~$
```

```
[Install]
WantedBy=multi-user.target
tdinh@controller-2:~$ {
>   sudo systemctl daemon-reload
>   sudo systemctl enable kube-apiserver kube-controller-manager kube-scheduler
>   sudo systemctl start kube-apiserver kube-controller-manager kube-scheduler
> }
tdinh@controller-2:~$ kubectl get componentstatuses --kubeconfig admin.kubeconfig
NAME                 STATUS    MESSAGE             ERROR
controller-manager   Healthy   ok
scheduler            Healthy   ok
etcd-0               Healthy   {"health":"true"}
etcd-2               Healthy   {"health":"true"}
etcd-1               Healthy   {"health":"true"}
tdinh@controller-2:~$
```

10. Dans la section "Provision a load balancer", que font les différentes commandes ?

    Les commandes sont:

-

```
KUBERNETES_PUBLIC_ADDRESS=$(gcloud compute addresses describe kubernetes-
the-hard-way \
    --region $(gcloud config get-value compute/region) \
    --format 'value(address)')
```

Créer un variable environnement qui stocke l'adresse publique de kubernetes

-

```
gcloud compute http-health-checks create kubernetes \
    --description "Kubernetes Health Check" \
    --host "kubernetes.default.svc.cluster.local" \
    --request-path "/healthz"
```

create an HTTP legacy health check to monitor load balanced instances

-

```
gcloud compute firewall-rules create kubernetes-the-hard-way-allow-health-
check \
    --network kubernetes-the-hard-way \
    --source-ranges 209.85.152.0/22,209.85.204.0/22,35.191.0.0/16 \
    --allow tcp
```

create a Google Compute Engine firewall rule *

```
gcloud compute target-pools create kubernetes-target-pool \
    --http-health-check kubernetes
```

Définit un load-balanced de machine virtuelle instances.

-

```
gcloud compute target-pools add-instances kubernetes-target-pool \
--instances controller-0,controller-1,controller-2
```

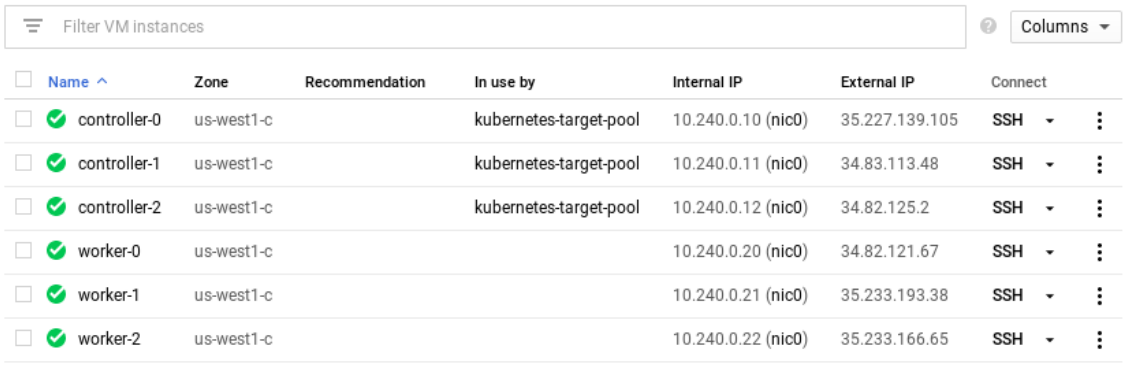
add instances to a target pool

-

```
gcloud compute forwarding-rules create kubernetes-forwarding-rule \
    --address ${KUBERNETES_PUBLIC_ADDRESS} \
    --ports 6443 \
    --region $(gcloud config get-value compute/region) \
    --target-pool kubernetes-target-pool
```

create a forwarding rule to direct network traffic to a load balancer

```
(base) tdinh@optiplex-1513:~/Documents/thanhhai/Cloud/kubernetes$ curl --cacert ca.pem https://${KUBERNETES_PUBLIC_ADDRESS}:6443/version
{
  "major": "1",
  "minor": "15",
  "gitVersion": "v1.15.3",
  "gitCommit": "2d3c76f9091b6bec110a5e63777c332469e0cba2",
  "gitTreeState": "clean",
  "buildDate": "2019-08-19T11:05:50Z",
  "goVersion": "go1.12.9",
  "compiler": "gc",
  "platform": "linux/amd64"
}(base) tdinh@optiplex-1513:~/Documents/thanhhai/Cloud/kubernetes$
```



11. Qu'est-ce que CoreDNS ?

    CoreDNS est un serveur DNS flexible et extensible pouvant servir de serveur DNS de cluster Kubernetes. Comme Kubernetes, le projet CoreDNS est hébergé par le CNCF. Il est développé par Go. Il peut être utilisé dans une multitude d'environnements en raison de sa flexibilité.