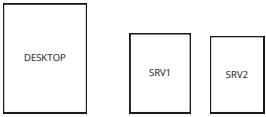
# KUBERNETES




VMWare non lancé :
- dupliquez SRV1 et SRV2
  en SRV3 et SRV4

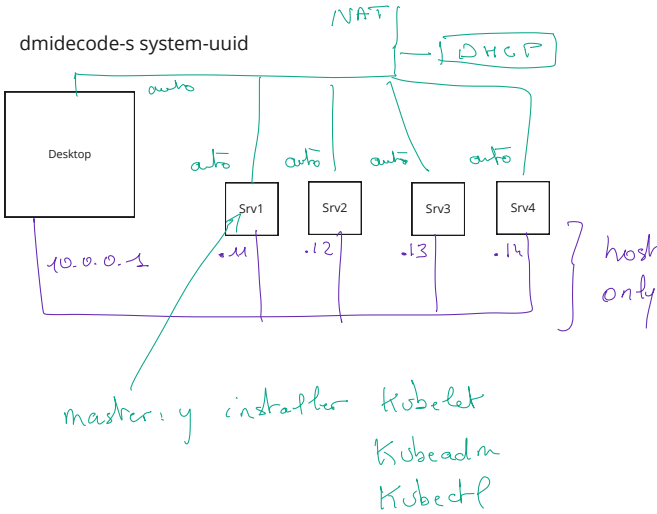
- Lancez VMWare
- ouvrez les VM
- choisissez absolument "I Copied it"
- modifiez les IP de SRV3 et SRV4 :
  editez /etc/network/interfaces
  remplacez 11 → 13 (SRV3)
  12 → 14 (SRV4)

Objectif : Depuis DESKTOP
Ping SRV1.stagiaire.lan → IP 11 OK
SRV2 → IP 12 OK
SRV3 → IP 13 OK
SRV4 → IP 14 OK

Puis sur SRV3 et SRV4
hostnamectl set-hostname SRV3|4

- Eteignez toutes vos VMs.
- Faites un snapshot pour chaque

→ Sur Desktop, assurez-vous que l'IP de la seconde carte soit bien fixe 10.0.0.1
DNS 10.0.0.1




Installation du cluster :

- Désactiver le swap ( nano /etc/fstab )
- installer docker dernière version
- Initialiser le cluster :
  - Installer kubeadm etc
    - https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
  - kubeadm init --apiserver-advertise-address=10.0.0.11 --pod-network-cidr=10.244.0.0/24
- La liste des opération :
  - https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm/
- Installer la couche réseau internoeuds, par exemple :
  - kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
  - Pour une implémentation Flannel
  - Chaque fournisseur a sa propre méthode d'installation
- Installer les binaires Kubernetes sur chaque noeud
- Executer la commande kubeadm join sur chaque noeud

Si vous n'avez pas noté la commande kubeadm join fournie par kubeadm init :
Lancez : kubeadm token create --print-join-command

Si problème d'accès aux fonctionnalités :
root@srv1:~# cp /etc/kubernetes/admin.conf ~
root@srv1:~# export KUBECONFIG=$HOME/admin.conf
puis : kubectl get pods

root@srv1:~# kubectl get nodes
NAME STATUS ROLES AGE VERSION
srv1 NotReady master 48m v1.16.3

annuler un cluster pour le reconfigurer :
kubeadm reset
rm -rf ~/.kube

puis à nouveau :
kubeadm init --apiserver-advertise-address=10.0.0.11 --pod-network-cidr=10.244.0.0/24

puis
cp /etc/kubernetes/admin.conf ~
export KUBECONFIG=$HOME/admin.conf ( a mettre dans votre bashrc )
( a faire si vous avez une erreur de connexion impossible )

puis appliquer flannel

puis contrôler :
kubectl get nodes

pour rerécupérer le hash :
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.* //'

pour le token :
kubeadm token list

la commande de base :
kubeadm join 10.0.0.11:6443 --token z0f2tc.kkxag4ja89hb743e --discovery-token-ca-cert-hash
sha256:92296560833a0e97f4613c04452f35c7dbc791efb8c18daa8b53c385e2663e27

Si une partie de votre infrastructure se plante :

```
root@master:~# kubectl get pods -A
NAMESPACE     NAME                              READY   STATUS             RESTARTS   AGE
kube-system   coredns-5644d7b6d9-c72pd          1/1     Running            0          72m
kube-system   coredns-5644d7b6d9-szjvl          1/1     Running            0          72m
kube-system   etcd-srv1                         1/1     Running            0          71m
kube-system   kube-apiserver-srv1               1/1     Running            0          71m
kube-system   kube-controller-manager-srv1      1/1     Running            0          71m
kube-system   kube-flannel-ds-amd64-drpt4       0/1     CrashLoopBackOff   6          8m10s
kube-system   kube-flannel-ds-amd64-ht67t       0/1     CrashLoopBackOff   6          8m4s
kube-system   kube-flannel-ds-amd64-l7nxx       1/1     Running            0          70m
kube-system   kube-flannel-ds-amd64-vdv6s       0/1     CrashLoopBackOff   6          8m44s
kube-system   kube-proxy-bnl26                  1/1     Running            0          8m44s
kube-system   kube-proxy-kch2f                  1/1     Running            0          72m
kube-system   kube-proxy-q2kdp                  1/1     Running            0          8m4s
kube-system   kube-proxy-tns4f                  1/1     Running            0          8m10s
kube-system   kube-scheduler-srv1               1/1     Running            0          71m
```

sur le master, pour chaque noeud :

```
kubectl patch node srv2 -p '{"spec":{"podCIDR":"10.244.0.0/24"}}'
```

puis sur chaque noeud :

```
systemctl restart kubelet
```

```
root@master:~# kubectl get pods -A
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   coredns-5644d7b6d9-c72pd          1/1     Running   0          78m
kube-system   coredns-5644d7b6d9-szjvl          1/1     Running   0          78m
kube-system   etcd-srv1                         1/1     Running   0          77m
kube-system   kube-apiserver-srv1               1/1     Running   0          77m
kube-system   kube-controller-manager-srv1      1/1     Running   0          77m
kube-system   kube-flannel-ds-amd64-drpt4       1/1     Running   8          14m
kube-system   kube-flannel-ds-amd64-ht67t       1/1     Running   8          14m
kube-system   kube-flannel-ds-amd64-l7nxx       1/1     Running   0          76m
kube-system   kube-flannel-ds-amd64-vdv6s       1/1     Running   8          15m
kube-system   kube-proxy-bnl26                  1/1     Running   0          15m
kube-system   kube-proxy-kch2f                  1/1     Running   0          78m
kube-system   kube-proxy-q2kdp                  1/1     Running   0          14m
kube-system   kube-proxy-tns4f                  1/1     Running   0          14m
kube-system   kube-scheduler-srv1               1/1     Running   0          77m
```

A garder sous la main :
https://kubernetes.io/docs/reference/kubectl/cheatsheet/

**Commandes utiles :**
kubectl get nodes
kubectl get pods -o wide
kubectl get services
kubectl describe node srv2
delete svc kubia-http

Accès à toutes les resources :
kubectl api-resources

modifier un facteur de réplication sur un déploiement :
kubectl scale --replicas=3 deployment/kubia

obtenir le détail du comportement d'un POD :
kubectl describe pod kubia-9d785b578-rcg79
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
node.kubernetes.io/unreachable:NoExecute for 300s
Events:

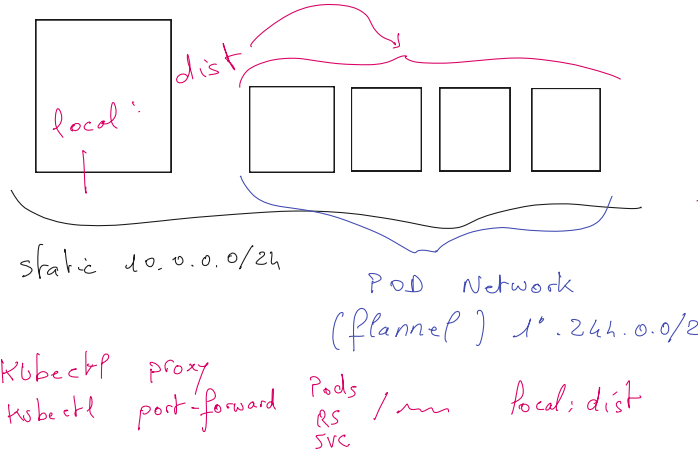
| Type | Reason | Age | From | Message |
|---|---|---|---|---|
| Normal | Scheduled | <unknown> | default-scheduler | Successfully assigned default/kubia-9d785b578-rcg79 to srv2 |
| Normal | Pulling | 8m3s | kubelet, srv2 | Pulling image "luksa/kubia" |
| Normal | Pulled | 30s | kubelet, srv2 | Successfully pulled image "luksa/kubia" |
| Normal | Created | 30s | kubelet, srv2 | Created container kubia |
| Normal | Started | 30s | kubelet, srv2 | Started container kubia |

La version de l'API Kubernetes :
https://kubernetes.io/docs/reference/

appliquer des ressources :
kubectl apply -f <res.yml>

Suppression des ressources :
kubectl delete -f <res.yml>

**IMPORTANT**
**ACCEDER A VOS APPLICATIONS DEPUIS UN HOST HORS CLUSTER ( ip n'est pas dans la gamme )**
**Vous devez appliquer de la redirection de port locale :**
**kubectl port-forward pods/<nom de pod> <port local>:<port de service> , ou :**
**kubectl port-forward deployment/<nom de déploiement> <port local>:<port de service> , ou :**
**kubectl port-forward rs/<nom de réplicaset> <port local>:<port de service> , ou :**
**kubectl port-forward svc/<nom de service> <port local>:<port de service>**




Accès au dashboard depuis un noeud distant :

1) modifier la configuration en node port :
kubectl -n kube-system edit service kubernetes-dashboard
( sinon dans le ns : kubernetes-dashboard :
kubectl -n kubernetes-dashboard edit service kubernetes-dashboard )
et changer la section type:

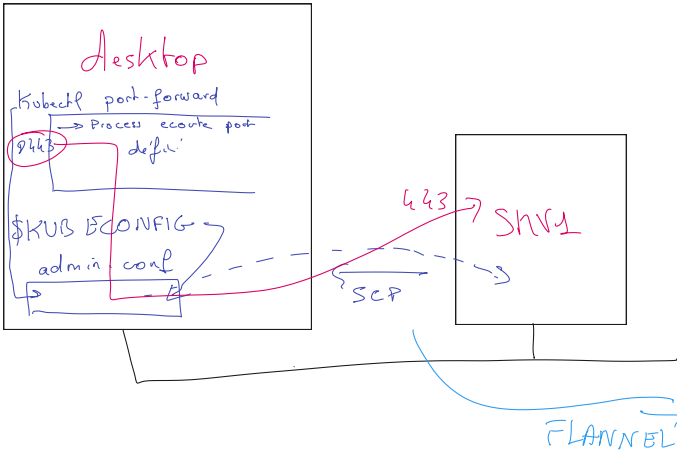
```
sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

3) rediriger le port :
kubectl get all -A ( rechercher )
export KUBE_EDITOR="nano"
kubectl port-forward -n kubernetes-dashboard service/kubernetes-dashboard
**8443**:443

4) se connecter :
https://localhost:**8443**

5) obtenez le jeton :
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')

puis copiez/collez ce jeton dans la page d'autentification via token




Vous devez créer un utilisateur avec jeton pour y accéder :
Créez le fichier puis appliquez :
[lien vers le site]

1) :
```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kube-system
```

2)
```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kube-system
```

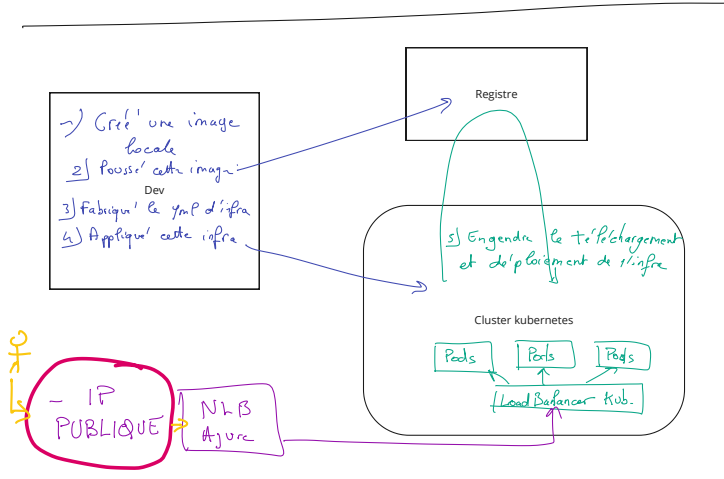
```
Récupérez le jeton :
kubectl -n kube-system describe secret $(kubectl -n kube-system get secret |
grep admin-user | awk '{print $1}')

et connectez-vous
https://localhost:8443
```

3ième jour :
Doc concernant la mise en oeuvre déclarative :
https://kubernetes.io/docs/tasks/manage-kubernetes-objects/declarative-config/#how-to-create-objects

HELM :
https://helm.sh/docs/intro/quickstart/

Installez HELM
Installez un repo
Installez mysql
Faites la procédure proposée pour vous y connecter.



Obtenir la doc concernant le format descripteur ( déclaratif ) :
**kubectl explain pod.metadata**

Changer votre namespace par défaut pour votre contexte actuel :
**kubectl config set-context --current --namespace=myns**

Exercice :
Implémentez un ReplicaSet et modifiez ensuite le facteur de réplication :
https://kubernetes.io/fr/docs/concepts/workloads/controllers/replicaset/#exemple

Mise en oeuvre de DaemonSet :

tagguer 2 noeuds de type disk=ssd
Créez le DaemonSet pour déployer l'application sur ces noeuds

Solution de monitoring Prometheus :
https://medium.com/faun/production-grade-kubernetes-monitoring-using-prometheus-78144b835b60