Version - **2024.01**

Dernière mise-à-jour : 2024/12/18 13:32

DOF302 - Gestion des PODs, Contrôleurs de Réplication, ReplicaSets, Deployments, de la Maintenance et des Mises-à-jour du Cluster

Contenu du Module

Ressources

Lab #1

- https://www.dropbox.com/scl/fi/n8iwjrfich5af7vtoezac/pod-definition.yaml?rlkey=hs3nwcczi1zucs3l7cge3bc9s&dl=0

Lab #2

- https://www.dropbox.com/scl/fi/2atlsvs2oi21fp6xudh46/cr_definition.yaml?rlkey=35lzzkg6qah55pjw34ccaval7&dl=0
- https://www.dropbox.com/scl/fi/j8ww8mdm82cec71lnz023/replicaset-definition.yaml?rlkey=2iypd89690ipt6lik3h3bae3f&dl=0

Lab #3

- https://www.dropbox.com/scl/fi/h057eynmqvlaoytph372r/deployment-definition.yaml?rlkey=lrs8llq89ffn1a5megm317f5y&dl=0

LAB #1 - Création d'un POD

1.1 - Présentation d'un POD

Un POD est un objet qui encapsule un conteneur. Le conteneur est un instance d'une application. La relation entre un POD et un conteneur d'application est en règle générale 1:1, c'est-à-dire que dans le cas d'une augmentation de la charge, des PODs additionnels sont créés, chacun contenant un conteneur d'application au lieu de créer plusieurs conteneurs dans le même POD.

A l'inverse, dans le cas d'une réduction de la charge, des PODs sont détruits. Avec Kubernetes, on ne crée pas de conteneurs multiples du même type dans le même POD. Par contre, il est possible d'avoir des conteneurs de types différents dans le même POD.

Dans ce cas on parle d'un conteneur d'application et un ou des conteneur(s) **Helper**. Le conteneur d'application et le conteneur Helper peuvent communiquer directement parce qu'ils partagent le même **espace réseau**. De même, ils ont accès au même **espace de stockage**.

Un POD permet donc de dispenser l'administrateur de la gestion de **liens** Docker ainsi que des **volumes**.

Lors da la création d'un POD avec la commande **kubectl**, celle-ci télécharge l'image Docker nécessaire à la création du conteneur à partir du Docker Hub.

1.2 - Création Manuelle d'un POD

Commencez par créer un POD dénommé **nginx** à partir de l'image nginx :

```
root@kubemaster:~# kubectl run  nginx --image=nginx
pod/nginx created
```

Visualisez le POD avec la commande **kubectl** :

```
root@kubemaster:~# kubectl get pods
NAME    READY   STATUS              RESTARTS   AGE
nginx   0/1     ContainerCreating   0          20s
```

```
root@kubemaster:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          44s
```

Consultez les informations concernant ce POD :

```
root@kubemaster:~# kubectl describe pods
Name:         nginx
Namespace:    default
Priority:     0
Node:         kubenode1.ittraining.loc/192.168.56.3
Start Time:   Wed, 13 Jul 2022 05:09:12 +0200
Labels:       run=nginx
Annotations:  cni.projectcalico.org/containerID: b401002d2766b402d37143c1fa4da7b87c1fc332324e841a9532c3814320ff83
              cni.projectcalico.org/podIP: 192.168.239.1/32
              cni.projectcalico.org/podIPs: 192.168.239.1/32
Status:       Running
IP:           192.168.239.1
IPs:
  IP:  192.168.239.1
Containers:
  nginx:
    Container ID:   containerd://7976f5c10f7884c02d862c69bb21115f47bf8cd22646a76aed51ede70214371e
    Image:          nginx
    Image ID:
docker.io/library/nginx@sha256:dbe677093f569cc0afe2a149c529645f255aac959490ef11fb19ac6418b815d3
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Wed, 13 Jul 2022 05:09:55 +0200
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
```

```
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-pmfww (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-pmfww:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  23m   default-scheduler  Successfully assigned default/nginx to kubenode1.ittraining.loc
  Normal  Pulling    23m   kubelet            Pulling image "nginx"
  Normal  Pulled     22m   kubelet            Successfully pulled image "nginx" in 41.16449179s
  Normal  Created    22m   kubelet            Created container nginx
  Normal  Started    22m   kubelet            Started container nginx
```

Important : Notez que la première ligne de la section **Events** indique clairement que dans le cas de cet exemple, le kubemaster a schedulé le POD sur kubenode1.

Utilisez maintenant le commande kubectl avec l'option **-o wide** :

```
root@kubemaster:~# kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP              NODE                        NOMINATED NODE   READINESS
GATES
nginx   1/1     Running   0          24m   192.168.239.1   kubenode1.ittraining.loc   <none>           <none>
```

Important : Notez que l'adresse IP du POD est la **192.168.239.1**. Cette adresse est **dynamique**. Si le POD s'arrête et un autre démarre, l'adresse IP du nouveau POD sera différente.

Important : Notez que dans la colonne **NOMINATED NODE** il est marqué **<none>**. En effet il est possible d'assigner un POD à un nœud spécifique grâce à l'utilisation d'une étiquette définie pour le ou les nœuds nominés. Pour plus d'information maintenant, consultez le lien **https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/**.

Important : Notez que dans la colonne **READINESS GATES** il est marqué **<none>**. En effet il est possible d'assigner à un POD des conditions spécifiques pour que Kubenetes considère que le POD soit dans un état de **ready**. Pour plus d'information maintenant, consultez le lien **https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#pod-readiness-gate**.

1.3 - Création d'un POD à l'aide d'un fichier YAML

Kubernetes utilise des fichiers YAML pour créer des objets. Par conséquent, la définition du POD à créer est décrite dans un fichier YAML. Créez le fichier **pod-definition.yaml** :

```
root@kubemaster:~# vi pod-definition.yaml
root@kubemaster:~# cat pod-definition.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Dans ce fichier on trouve les champs suivants :

apiVersion

- Ce champs est **obligatoire**,
- La version de l'API diffère selon le type d'objet qui est créé,
- La valeur du champs est sous la forme d'une chaîne.

kind	apiVersion
Pod	v1
Service	v1
ReplicaSet	apps/v1

kind	apiVersion
Deployment	apps/v1

kind

- Ce champs est **obligatoire**,
- La valeur de l'apiServer par rapport au type d'objet est :

kind	apiVersion
Pod	v1
Service	v1
ReplicaSet	apps/v1
Deployment	apps/v1

metadata

- Ce champs est **obligatoire**,
- Il contient des informations telles le nom et les étiquettes,
- Les informations sont sous la forme d'un **dictionnaire** YAML :

```
metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end
```

spec

- Ce champs est **obligatoire**,
- Il contient des informations pour Kubernetes spécifiques au type d'objet à créer,
- Les informations sont sous la forme d'un **liste** YAML :

```
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Utilisation du Fichier YAML

Utilisez maintenant le fichier YAML afin de créer un POD :

```
root@kubemaster:~# kubectl create -f pod-definition.yaml
pod/myapp-pod created
```

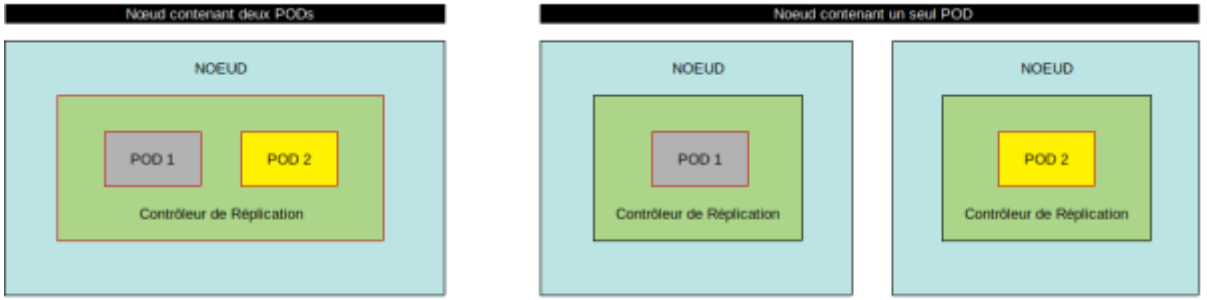
```
root@kubemaster:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
myapp-pod   1/1     Running   0          23s
nginx       1/1     Running   0          29m
```

LAB #2 - Utilisation de Contrôleurs de Réplication et ReplicaSets

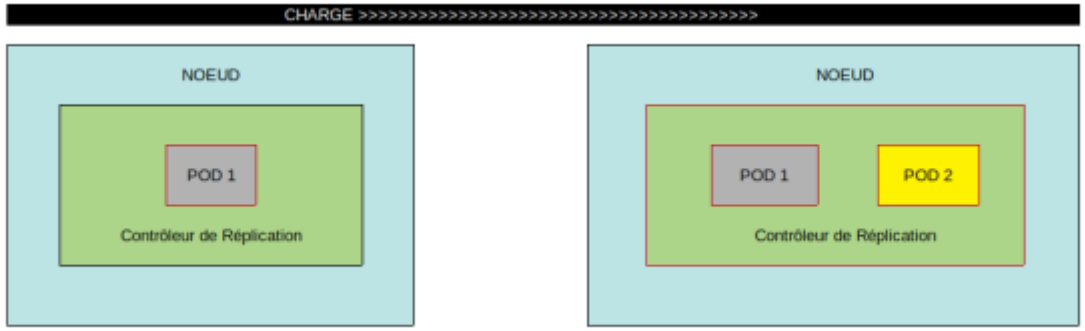
2.1 - Contrôleurs de Réplication

Présentation d'un Contrôleur de Réplication

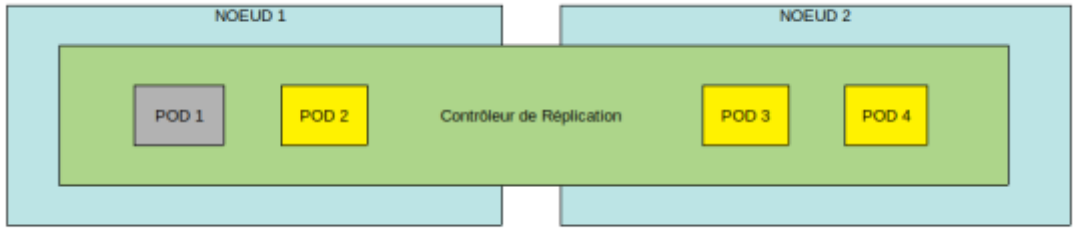
Un Contrôleur de Réplication permet d'exécuter plusieurs instances du même POD de façon à offrir de la **haute disponibilité** au cas où l'application crash et le POD se met en échec. Même dans le cas où il n'y a qu'un seul POD, le Contrôleur de Réplication peut démarrer automatiquement un autre POD contenant l'application :



Un Contrôleur de Réplication permet aussi de démarrer de nouveaux PODs en cas d'augmentation de la charge ainsi que d'assurer l'équilibrage de la charge entre les PODs :



Dans le cas où le premier nœud venait à court de ressources, un Contrôleur de Réplication est capable de démarrer de nouveaux PODs sur un deuxième noeud :



Mise en Application

Pour créer un Contrôleur de Réplication, il convient de créer un fichier YAML. Créez donc le fichier **cr-definition.yaml** :

```
root@kubemaster:~# vi cr-definition.yaml
root@kubemaster:~# cat cr-definition.yaml
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp-cr
  labels:
    app: myapp
    type: front-end
spec:
  template:

    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
```

```
      - name: nginx-container
        image: nginx

  replicas: 3
```

Dans ce fichier est placée une section appelée **template**. Cette section est un gabarit pour la création de PODs supplémentaires et est identique au contenu du fichier **pod-definition.yaml** sans les champs apiVersion et kind :

```
root@kubemaster:~# cat pod-definition.yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Le champs **replicas** qui se trouve au même niveau de **template** indique le nombre de PODs à créer.

Utilisez le fichier rc-definition.yaml pour créer le Contrôleur de Réplication :

```
root@kubemaster:~# kubectl create -f cr-definition.yaml
replicationcontroller/myapp-cr created
```

Pour visualiser le Contrôleur de Réplication, utilisez la commande suivante :

```
root@kubemaster:~# kubectl get replicationcontroller
NAME       DESIRED   CURRENT   READY   AGE
myapp-cr   3         3         3       71s
```

Pour visualiser les PODs créés par le Contrôleur de Réplication, utilisez la commande suivante :

```
root@kubemaster:~# kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
myapp-cr-6gxg6   1/1     Running   0          90s
myapp-cr-78frz   1/1     Running   0          90s
myapp-pod        1/1     Running   0          3m53s
nginx            1/1     Running   0          32m
```

Important : Notez que le Contrôleur de Réplication a créé deux replicas **myapp-cr-6gxg6** et **myapp-cr-78frz** car le premier existait déjà **myapp-pod**. Pour identifier un POD du même type déjà en place, le Contrôleur de Réplication se fie au champ **labels** dans la section **template**.

Supprimez maintenant le POD **myapp-pod** :

```
root@kubemaster:~# kubectl delete pod myapp-pod
pod "myapp-pod" deleted
```

Constatez ensuite la réaction du Contrôleur de Réplication :

```
root@kubemaster:~# kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
myapp-cr-6gxg6   1/1     Running   0          3m5s
myapp-cr-78frz   1/1     Running   0          3m5s
myapp-cr-pt4zt   1/1     Running   0          27s
nginx            1/1     Running   0          34m
```

Important : Notez que le Contrôleur de Réplication a créé le POD **myapp-cr-pt4zt**.

Pour consulter le statut d'un Contrôleur de Réplication, utilisez la commande suivante :

```
root@kubemaster:~# kubectl describe replicationcontrollers/myapp-cr
Name:         myapp-cr
Namespace:    default
Selector:     app=myapp,type=front-end
Labels:       app=myapp
              type=front-end
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason            Age    From                    Message
  ----    ------            ----   ----                    -------
  Normal  SuccessfulCreate  3m51s  replication-controller  Created pod: myapp-cr-78frz
  Normal  SuccessfulCreate  3m51s  replication-controller  Created pod: myapp-cr-6gxg6
  Normal  SuccessfulCreate  72s    replication-controller  Created pod: myapp-cr-pt4zt
```

Pour supprimer un Contrôleur de Réplication, utilisez la commande suivante :

```
root@kubemaster:~# kubectl delete replicationcontroller myapp-cr
replicationcontroller "myapp-cr" deleted
```

2.2 - ReplicaSets

Présentation d'un ReplicaSet

Un ReplicaSet remplit la même fonction qu'un Contrôleur de Réplication. ReplicaSets sont la façon la plus récente de gérer la réplication.

Mise en Application

Pour créer un ReplicaSet, créez le fichier **replicaset-definition.yaml** :

```
root@kubemaster:~# vi replicaset-definition.yaml
root@kubemaster:~# cat replicaset-definition.yaml
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
    app: myapp
    type: front-end
spec:
  template:

    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
```

```
      image: nginx

  replicas: 3
  selector:
    matchLabels:
      type: front-end
```

Important : Notez que dans le cas d'un ReplicaSet, celui-ci identifie les PODs sous son contrôle par la valeur du champ **matchLabels**..

Utilisez le fichier replicaset-definition.yaml pour créer le ReplicaSet :

```
root@kubemaster:~# kubectl create -f replicaset-definition.yaml
replicaset.apps/myapp-replicaset created
```

Pour visualiser le ReplicaSet, utilisez la commande suivante :

```
root@kubemaster:~# kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
myapp-replicaset   3         3         3       12s
```

Pour visualiser les PODs créés par le ReplicaSet, utilisez la commande suivante :

```
root@kubemaster:~# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-replicaset-56gwv   1/1     Running   0          29s
myapp-replicaset-gh8gl   1/1     Running   0          29s
myapp-replicaset-kz742   1/1     Running   0          29s
nginx                    1/1     Running   0          60m
```

Modifiez maintenant le fichier **replicaset-definition.yaml** en augmentant le nombre de replicas de 3 à **6** :

```
root@kubemaster:~# vi replicaset-definition.yaml
root@kubemaster:~# cat replicaset-definition.yaml
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
    app: myapp
    type: front-end
spec:
  template:

    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx

  replicas: 6
  selector:
    matchLabels:
      type: front-end
```

Exécutez ensuite la commande **kubectl replace** :

```
root@kubemaster:~# kubectl replace -f replicaset-definition.yaml
replicaset.apps/myapp-replicaset replaced
```

Visualiser le ReplicaSet :

```
root@kubemaster:~# kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
myapp-replicaset   6         6         3       95s
root@kubemaster:~# kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
myapp-replicaset   6         6         5       98s
root@kubemaster:~# kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
myapp-replicaset   6         6         6       99s
```

Visualiser les PODs créés par le ReplicaSet :

```
root@kubemaster:~# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-replicaset-56gwv   1/1     Running   0          2m14s
myapp-replicaset-7g6r4   1/1     Running   0          49s
myapp-replicaset-7rsnc   1/1     Running   0          49s
myapp-replicaset-gh8gl   1/1     Running   0          2m14s
myapp-replicaset-kz742   1/1     Running   0          2m14s
myapp-replicaset-twcwg   1/1     Running   0          49s
nginx                    1/1     Running   0          62m
```

Exécutez ensuite la commande suivante :

```
root@kubemaster:~# kubectl scale --replicas=9 -f replicaset-definition.yaml
replicaset.apps/myapp-replicaset scaled
```

Visualiser le ReplicaSet :

```
root@kubemaster:~# kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
myapp-replicaset   9         9         9       3m6s
```

Visualiser les PODs créés par le ReplicaSet :

```
root@kubemaster:~# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-replicaset-56gwv   1/1     Running   0          3m12s
myapp-replicaset-7g6r4   1/1     Running   0          107s
myapp-replicaset-7rsnc   1/1     Running   0          107s
myapp-replicaset-gh8gl   1/1     Running   0          3m12s
myapp-replicaset-klsvp   1/1     Running   0          33s
myapp-replicaset-kz742   1/1     Running   0          3m12s
myapp-replicaset-twcwg   1/1     Running   0          107s
myapp-replicaset-vqsxc   1/1     Running   0          33s
myapp-replicaset-z9l65   1/1     Running   0          33s
nginx                    1/1     Running   0          63m
```

Notez que dans ce cas, la valeur des **replicas** dans le fichier **replicaset-definition.yaml** n'a pas été modifiée :

```
root@kubemaster:~# cat replicaset-definition.yaml
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
  labels:
    app: myapp
    type: front-end
spec:
  template:

    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
```

```
    spec:
      containers:
      - name: nginx-container
        image: nginx

  replicas: 6
  selector:
    matchLabels:
      type: front-end
```

Dernièrement, exécutez la commande suivante :

```
root@kubemaster:~# kubectl scale --replicas=3 replicaset myapp-replicaset
replicaset.extensions/myapp-replicaset scaled
```

Visualiser le ReplicaSet :

```
root@kubemaster:~# kubectl get replicaset
NAME               DESIRED   CURRENT   READY   AGE
myapp-replicaset   3         3         3       4m12s
```

Visualiser les PODs créés par le ReplicaSet :

```
root@kubemaster:~# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
myapp-replicaset-56gwv   1/1     Running   0          4m4s
myapp-replicaset-7g6r4   1/1     Running   0          2m39s
myapp-replicaset-gh8gl   1/1     Running   0          4m4s
nginx                    1/1     Running   0          64m
```

Créez maintenant un POD en dehors du ReplicaSet :

```
root@kubemaster:~# kubectl create -f pod-definition.yaml
```

```
pod/myapp-pod created
```

Consultez la liste des PODs :

```
root@kubemaster:~# kubectl get pods
NAME                    READY   STATUS        RESTARTS   AGE
myapp-pod               0/1     Terminating   0          2s
myapp-replicaset-56gwv  1/1     Running       0          5m58s
myapp-replicaset-7g6r4  1/1     Running       0          4m33s
myapp-replicaset-gh8gl  1/1     Running       0          5m58s
nginx                   1/1     Running       0          66m
```

Important : Notez que **myapp-pod** est dans un état **Terminating**. En effet le ReplicaSet ne permet pas la création d'un POD ayant la même étiquette que celle spécifiée par le champ **matchLabels** du fichier **replicaset-definition.yaml**.

Pour supprimer le ReplicaSet, utilisez la commande suivante :

```
root@kubemaster:~# kubectl delete replicaset myapp-replicaset
replicaset.extensions "myapp-replicaset" deleted
```

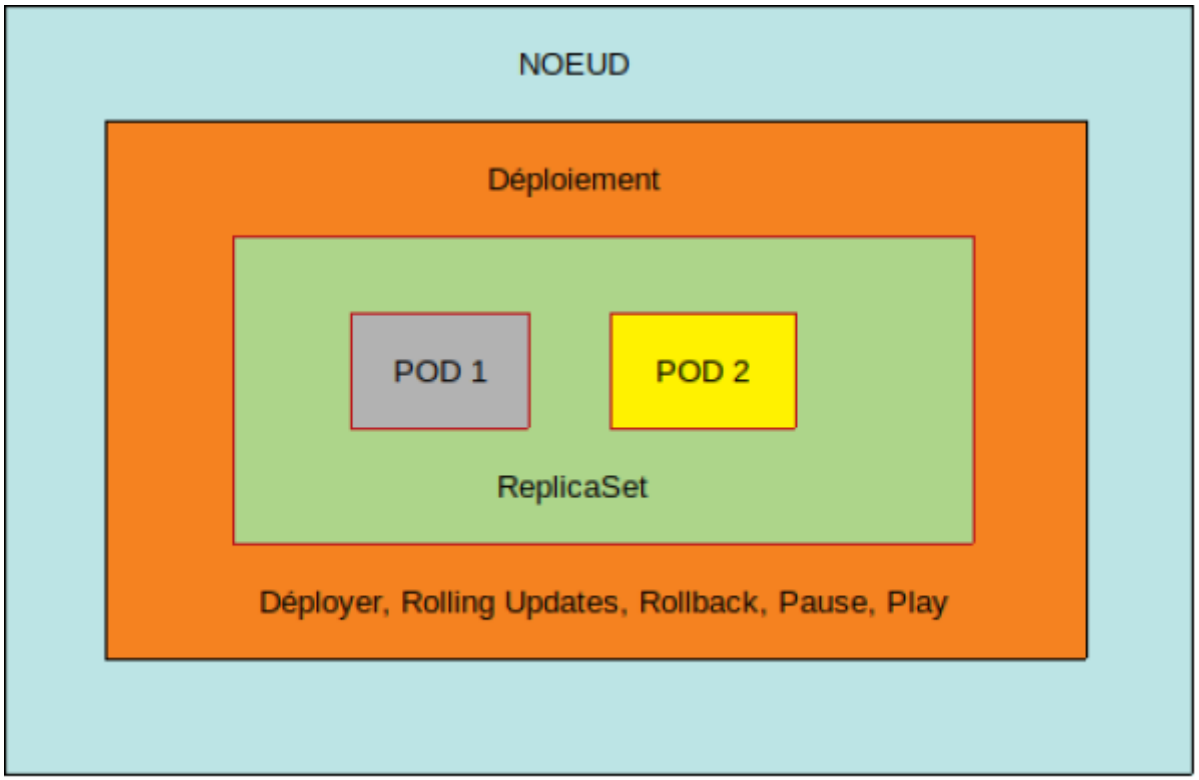
Consultez maintenant tous les objets du cluster :

```
root@kubemaster:~# kubectl get all
NAME        READY   STATUS    RESTARTS   AGE
pod/nginx   1/1     Running   0          67m

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   16h
```

LAB #3 - Gestion des Deployments

3.1 - Présentation d'un Deployment

Un **Deployment** sous Kubernetes est un objet hiérarchiquement supérieur à un ReplicaSet :



Le Deployment permet la gestion des :

- déploiements de PODs (Rollouts),
- mises à jour roulantes (Rolling Updates),

- retours en arrière (Rollbacks).

3.2 - Mise en Application

Rollouts

Pour créer un Deployment, il convient de créer un fichier YAML. Créez donc le fichier **deployment-definition.yaml** :

```
root@kubemaster:~# vi deployment-definition.yaml
root@kubemaster:~# cat deployment-definition.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
    app: myapp
    type: front-end
spec:
  template:

    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx

  replicas: 3
```

```
  selector:
    matchLabels:
      type: front-end
```

Utilisez la commande suivante pour créer le Deployment :

```
root@kubemaster:~# kubectl create -f deployment-definition.yaml
deployment.apps/myapp-deployment created
```

Constatez la création de celui-ci :

```
root@kubemaster:~# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment   3/3     3            3           17s
```

Notez que la création du Deployment a également créé un ReplicaSet :

```
root@kubemaster:~# kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-689f9d59   3         3         3       41s
```

Important : Notez que la valeur **689f9d59** est générée d'une manière aléatoire en interne par Kubernetes.

Bien entendu, la création de Deployment a créé le nombre de PODs indiqué dans le fichier YAML :

```
root@kubemaster:~# kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
myapp-deployment-689f9d59-cmxlm   1/1     Running   0          98s
myapp-deployment-689f9d59-kt88s   1/1     Running   0          98s
```

```
myapp-deployment-689f9d59-zlwp4   1/1     Running   0          98s
```

Pour voir tous ces objets en même temps, utilisez la commande **kubectl get all** :

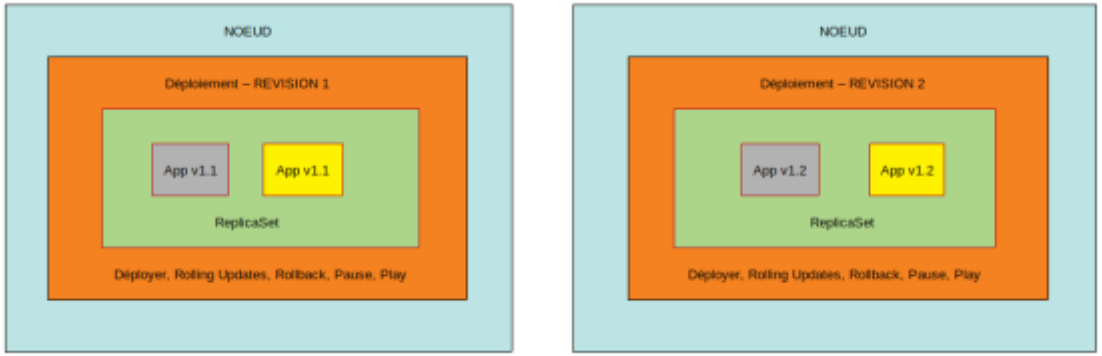
```
root@kubemaster:~# kubectl get all
NAME                                   READY   STATUS    RESTARTS   AGE
pod/myapp-deployment-689f9d59-cmxlm   1/1     Running   0          2m10s
pod/myapp-deployment-689f9d59-kt88s   1/1     Running   0          2m10s
pod/myapp-deployment-689f9d59-zlwp4   1/1     Running   0          2m10s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   16h

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/myapp-deployment   3/3     3            3           2m10s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/myapp-deployment-689f9d59   3         3         3       2m10s
```

Pour obtenir plus d'informations concernant le Deployment, utilisez la commande **kubectl describe** :

```
root@kubemaster:~# kubectl describe deployments
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Wed, 13 Jul 2022 06:18:11 +0200
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               type=front-end
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
```

```
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   myapp-deployment-689f9d59 (3/3 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  2m48s  deployment-controller  Scaled up replica set myapp-deployment-689f9d59 to 3
```

Lors du Rollout du Deployment une **Révision** est créée. Cette Révision est incrémentée lors de chaque mise-à-jour :



Pour consulter le statut du Rollout, il convient d'utiliser la commande suivante :

```
root@kubemaster:~# kubectl rollout status deployment/myapp-deployment
deployment "myapp-deployment" successfully rolled out
```

Pour consulter la liste des Révisions, utilisez la commande suivante :

```
root@kubemaster:~# kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
1         <none>
```

Important : Notez que la valeur de **CHANGE-CAUSE** est **<none>** parce que l'option **-record** n'a pas été spécifiée sur la ligne de commande. Il est possible de modifier la valeur **CHANGE-CAUSE** avec la commande **kubectl annotate deployment <deployment> kubernetes.io/change-cause="<Message>" -record=false -overwrite=true**.

Supprimez donc le Deployment avec la commande suivante :

```
root@kubemaster:~# kubectl delete deployment myapp-deployment
deployment.extensions "myapp-deployment" deleted
```

Vérifiez la suppression du Deployment :

```
root@kubemaster:~# kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   18h
```

Créez le Deployment de nouveau en ajoutant l'option **–record** :

```
root@kubemaster:~# kubectl create -f deployment-definition.yaml --record
deployment.apps/myapp-deployment created
```

Consultez le statut du Rollout :

```
root@kubemaster:~# kubectl rollout status deployment/myapp-deployment
deployment "myapp-deployment" successfully rolled out
```

Important : Notez qu'un Deployment peut être mis en pause avec la commande **kubectl rollout pause deployment <deployment>** et peut être repris avec la commande **kubectl rollout resume deployment <deployment>**.

Consultez la liste des Révisions :

```
root@kubemaster:~# kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
1         kubectl create --filename=deployment-definition.yaml --record=true
```

Important : Notez que la valeur de **CHANGE-CAUSE** est la commande qui a été saisie.

Rolling Updates

Il existe deux méthodes de Deployment en cas de mise-à-jours :

- **Recreate**,
  - Dans ce cas tous les PODs existants sont détruits en même temps et des PODs contenant la mise-à-jour sont créés dans un deuxième temps. L'inconvénient de cette méthode est évident - entre la déstruction des PODs et la re-création des nouveaux PODs, l'application n'est pas disponible,
- **Rolling Update**
  - Dans ce cas, les PODs sont détruits un-par-un. Après chaque destruction, un nouveau POD est créé contenant la mise-à-jour. De cette façon, l'application reste disponible.

Important : Notez que **Rolling Update** est la méthode par défaut.

Modifiez maintenant le fichier **deployment-description.yaml** en spécifiant la version **1.12** de **nginx** :

```
root@kubemaster:~# vi deployment-definition.yaml
root@kubemaster:~# cat deployment-definition.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-deployment
  labels:
    app: myapp
```

```
    type: front-end
spec:
  template:

    metadata:
      name: myapp-pod
      labels:
        app: myapp
        type: front-end
    spec:
      containers:
      - name: nginx-container
        image: nginx:1.12

  replicas: 3
  selector:
    matchLabels:
      type: front-end
```

Appliquez ce changement :

```
root@kubemaster:~# kubectl apply -f deployment-definition.yaml --record
Flag --record has been deprecated, --record will be removed in the future
Warning: resource deployments/myapp-deployment is missing the kubectl.kubernetes.io/last-applied-configuration
annotation which is required by kubectl apply. kubectl apply should only be used on resources created
declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched
automatically.
deployment.apps/myapp-deployment configured
```

Consultez le statut du Deployment :

```
root@kubemaster:~# kubectl rollout status deployment/myapp-deployment
Waiting for deployment "myapp-deployment" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "myapp-deployment" rollout to finish: 1 old replicas are pending termination...
```

```
deployment "myapp-deployment" successfully rolled out
```

Notez qu'il y a maintenant une **Révision** supplémentaire :

```
root@kubemaster:~# kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
1         kubectl create --filename=deployment-definition.yaml --record=true
2         kubectl apply --filename=deployment-definition.yaml --record=true
```

Consultez les détails du Deployment **myapp-deployment** :

```
root@kubemaster:~# kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Wed, 13 Jul 2022 07:44:43 +0200
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision: 2
                        kubernetes.io/change-cause: kubectl apply --filename=deployment-definition.yaml --
record=true
Selector:               type=front-end
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx:1.12
    Port:         <none>
    Host Port:    <none>
```

```
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   myapp-deployment-57c6cb89d9 (3/3 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  7m46s  deployment-controller  Scaled up replica set myapp-deployment-689f9d59 to 3
  Normal  ScalingReplicaSet  4m45s  deployment-controller  Scaled up replica set myapp-deployment-57c6cb89d9 to 1
  Normal  ScalingReplicaSet  4m20s  deployment-controller  Scaled down replica set myapp-deployment-689f9d59 to 2
  Normal  ScalingReplicaSet  4m19s  deployment-controller  Scaled up replica set myapp-deployment-57c6cb89d9 to 2
  Normal  ScalingReplicaSet  3m43s  deployment-controller  Scaled down replica set myapp-deployment-689f9d59 to 1
  Normal  ScalingReplicaSet  3m42s  deployment-controller  Scaled up replica set myapp-deployment-57c6cb89d9 to 3
  Normal  ScalingReplicaSet  2m10s  deployment-controller  Scaled down replica set myapp-deployment-689f9d59 to 0
```

Important : Notez que l'image utilisée est bien la **nginx:1.12**. Notez ensuite que dans la section **Events**, les PODs ont été **Scaled down** un-par-un et **Scaled up** un-par-un. Notez aussi que la valeur de **StrategyType** peut être soit **Recreate** soit **RollingUpdate**. Dernièrement, notez la valeur de **RollingUpdateStrategy**. **25% max unavailable** indique qu'à un instant “t” 75% des PODs doivent être disponibles tandis que **25% max surge** indique le nombre total des PODs ne peut pas dépasser 1,25 fois la valeur du champ **Replicas**. Ces valeurs peuvent être modifiées. Consultez la page **https://kubernetes.io/docs/concepts/workloads/controllers/deployment/**.

Lors de la mise-à-jour le Deployment crée un autre ReplicaSet contenant les PODs mis-à-jour en suivant la méthode Rolling Update. Ceci peut être vu

en regardant la sortie de la commande **kubectl get replicasets** :

```
root@kubemaster:~# kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-57c6cb89d9   3         3         3       5m41s
myapp-deployment-689f9d59     0         0         0       8m42s
```

Important : Notez que le nombre d'anciens ReplicaSets retenu est de **10** par défaut. Cette valeur peut être modifiée. Consultez la page **https://kubernetes.io/docs/concepts/workloads/controllers/deployment/**.

La modification de la version de l'image peut aussi être effectuée sur la ligne de commande :

```
root@kubemaster:~# kubectl set image deployment/myapp-deployment nginx-container=nginx:1.14 --record
Flag --record has been deprecated, --record will be removed in the future
deployment.apps/myapp-deployment image updated
```

Le nom du conteneur **nginx-container** est défini dans le fichier de définition du POD :

```
root@kubemaster:~# cat pod-definition.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
    type: front-end
spec:
  containers:
    - name: nginx-container
```

```
      image: nginx
```

Consultez le statut du Deployment :

```
root@kubemaster:~# kubectl rollout status deployment/myapp-deployment
deployment "myapp-deployment" successfully rolled out
```

Notez qu'il y a maintenant une **Révision** supplémentaire :

```
root@kubemaster:~# kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
1         kubectl create --filename=deployment-definition.yaml --record=true
2         kubectl apply --filename=deployment-definition.yaml --record=true
3         kubectl set image deployment/myapp-deployment nginx-container=nginx:1.14 --record=true
```

Lors de la mise-à-jour le Deployment crée un autre ReplicaSet contenant les PODs mis-à-jour en suivant la méthode Rolling Update. Ceci peut être vu en regardant la sortie de la commande **kubectl get replicasets** :

```
root@kubemaster:~# kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-57c6cb89d9   0         0         0       22m
myapp-deployment-689f9d59     0         0         0       25m
myapp-deployment-6c95f449f5   3         3         3       16m
```

Consultez les détails du Deployment **myapp-deployment** :

```
root@kubemaster:~# kubectl describe deployment myapp-deployment
Name:                   myapp-deployment
Namespace:              default
CreationTimestamp:      Wed, 13 Jul 2022 07:44:43 +0200
Labels:                 app=myapp
                        type=front-end
Annotations:            deployment.kubernetes.io/revision: 3
```

```
                        kubernetes.io/change-cause: kubectl set image deployment/myapp-deployment nginx-
container=nginx:1.14 --record=true
Selector:               type=front-end
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=myapp
           type=front-end
  Containers:
   nginx-container:
    Image:        nginx:1.14
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   myapp-deployment-6c95f449f5 (3/3 replicas created)
Events:
  Type    Reason             Age                From                   Message
  ----    ------             ----               ----                   -------
  Normal  ScalingReplicaSet  26m                deployment-controller  Scaled up replica set myapp-
deployment-689f9d59 to 3
  Normal  ScalingReplicaSet  23m                deployment-controller  Scaled up replica set myapp-
deployment-57c6cb89d9 to 1
  Normal  ScalingReplicaSet  22m                deployment-controller  Scaled down replica set myapp-
deployment-689f9d59 to 2
```

```
  Normal  ScalingReplicaSet  22m                deployment-controller  Scaled up replica set myapp-
deployment-57c6cb89d9 to 2
  Normal  ScalingReplicaSet  22m                deployment-controller  Scaled down replica set myapp-
deployment-689f9d59 to 1
  Normal  ScalingReplicaSet  22m                deployment-controller  Scaled up replica set myapp-
deployment-57c6cb89d9 to 3
  Normal  ScalingReplicaSet  20m                deployment-controller  Scaled down replica set myapp-
deployment-689f9d59 to 0
  Normal  ScalingReplicaSet  16m                deployment-controller  Scaled up replica set myapp-
deployment-6c95f449f5 to 1
  Normal  ScalingReplicaSet  16m                deployment-controller  Scaled down replica set myapp-
deployment-57c6cb89d9 to 2
  Normal  ScalingReplicaSet  14m (x4 over 16m)  deployment-controller  (combined from similar events): Scaled
down replica set myapp-deployment-57c6cb89d9 to 0
```

Important : Notez que l'image utilisée est bien la **nginx:1.14**.

Rollbacks

Grâce au système des **Révisions**, il est possible de revenir en arrière vers la version précédente **N-1** de l'application. Saisissez la commande suivante :

```
root@kubemaster:~# kubectl rollout undo deployment/myapp-deployment
deployment.extensions/myapp-deployment rolled back
```

Important : Notez qu'il est possible de revenir en arrière vers une version précédente spécifique avec la commande **kubectl rollout undo deployment <deployment> -to-revision=<revision>**.

Saisissez la commande **kubectl get replicasets** :

```
root@kubemaster:~# kubectl get replicasets
NAME                          DESIRED   CURRENT   READY   AGE
myapp-deployment-57c6cb89d9   3         3         3       24m
myapp-deployment-689f9d59     0         0         0       27m
myapp-deployment-6c95f449f5   0         0         0       18m
```

Important : Notez que l'application est revenue à la version précédente.

Utilisez la commande **kubectl rollout history** :

```
root@kubemaster:~# kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
1         kubectl create --filename=deployment-definition.yaml --record=true
3         kubectl set image deployment/myapp-deployment nginx-container=nginx:1.14 --record=true
4         kubectl apply --filename=deployment-definition.yaml --record=true
```

Important : Notez que Révision 2 est devenue la Révision 4 démontrant ainsi le Rollback.

Créez maintenant une erreur d'un Rollout :

```
root@kubemaster:~# kubectl set image deployment/myapp-deployment nginx-container=nginx1.14 --record
deployment.extensions/myapp-deployment image updated
```

Important : Notez que l'erreur est **nginx1.14** qui devrait être **nginx:1.14**.

Constatez le statut du Deployment :

```
root@kubemaster:~# kubectl rollout status deployment/myapp-deployment
Waiting for deployment "myapp-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
^C
```

Important : Notez que le Rollout est bloqué. L'erreur **error: deployment “myapp-deployment” exceeded its progress deadline** va être retournée au bout d'une dizaine de minutes !

Pour visualiser ce qui se passe, utilisez la commande **kubectl get deployments** :

```
root@kubemaster:~# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment   3/3     1            3           15m
```

La commande **kubectl get pods** démontre un statut de **ImagePullBackOff** pour le premier POD dans le nouveau ReplicaSet qui indique que Kubernetes ne peut pas effectuer le **pull** de l'image à partir de Docker Hub :

```
root@kubemaster:~# kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
myapp-deployment-57c6cb89d9-dh4cb   1/1     Running            0          7m24s
myapp-deployment-57c6cb89d9-f69nk   1/1     Running            0          7m30s
myapp-deployment-57c6cb89d9-q7d4p   1/1     Running            0          7m19s
myapp-deployment-74f697676f-2z95l   0/1     ImagePullBackOff   0          4m1s
```

En consultant l'historique du Rollout, une Révision supplémentaire a été ajoutée suite à la commande en erreur :

```
root@kubemaster:~# kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
```

```
1         kubectl create --filename=deployment-definition.yaml --record=true
3         kubectl set image deployment/myapp-deployment nginx-container=nginx:1.14 --record=true
4         kubectl apply --filename=deployment-definition.yaml --record=true
5         kubectl set image deployment/myapp-deployment nginx-container=nginx1.14 --record=true
```

Pour rectifier cette erreur il convient de faire un Rollback :

```
root@kubemaster:~# kubectl rollout undo deployment/myapp-deployment
deployment.extensions/myapp-deployment rolled back
```

Constatez ensuite la réussite de la commande :

```
root@kubemaster:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
myapp-deployment-57c6cb89d9-dh4cb   1/1     Running   0          9m38s
myapp-deployment-57c6cb89d9-f69nk   1/1     Running   0          9m44s
myapp-deployment-57c6cb89d9-q7d4p   1/1     Running   0          9m33s

root@kubemaster:~# kubectl rollout history deployment/myapp-deployment
deployment.apps/myapp-deployment
REVISION  CHANGE-CAUSE
1         kubectl create --filename=deployment-definition.yaml --record=true
3         kubectl set image deployment/myapp-deployment nginx-container=nginx:1.14 --record=true
5         kubectl set image deployment/myapp-deployment nginx-container=nginx1.14 --record=true
6         kubectl apply --filename=deployment-definition.yaml --record=true
```

LAB #4 - Gestion de la Maintenance

Afin de procéder à la maintenance d'un noeud, il est souvent nécessaire de le sortir du cluster. Cette opération s'appelle un **drain**.

4.1 - La Commande drain

Constatez l'état des pods :

```
root@kubemaster:~# kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                                READY   STATUS    RESTARTS      AGE   IP
NODE                        NOMINATED NODE   READINESS GATES
default       myapp-deployment-57c6cb89d9-dh4cb                   1/1     Running   0             27m
192.168.150.2    kubenode2.ittraining.loc    <none>           <none>
default       myapp-deployment-57c6cb89d9-q7d4p                   1/1     Running   0             27m
192.168.239.2    kubenode1.ittraining.loc    <none>           <none>
default       myapp-deployment-57c6cb89d9-f69nk                   1/1     Running   0             27m
192.168.150.3    kubenode2.ittraining.loc    <none>           <none>
default       nginx                                               1/1     Running   0             32m
192.168.239.1    kubenode1.ittraining.loc    <none>           <none>
kube-system   calico-kube-controllers-6799f5f4b4-zk298            1/1     Running   0             60m
192.168.55.195   kubemaster.ittraining.loc   <none>           <none>
kube-system   calico-node-5htrc                                   1/1     Running   0             50m
192.168.56.3     kubenode1.ittraining.loc    <none>           <none>
kube-system   calico-node-dc7hd                                   1/1     Running   0             60m   10.0.2.65
kubemaster.ittraining.loc   <none>           <none>
kube-system   calico-node-qk5kt                                   1/1     Running   0             52m
192.168.56.4     kubenode2.ittraining.loc    <none>           <none>
kube-system   coredns-6d4b75cb6d-kxtqk                            1/1     Running   0             62m
192.168.55.194   kubemaster.ittraining.loc   <none>           <none>
kube-system   coredns-6d4b75cb6d-td7cf                            1/1     Running   0             62m
192.168.55.193   kubemaster.ittraining.loc   <none>           <none>
kube-system   etcd-kubemaster.ittraining.loc                      1/1     Running   1 (57m ago)   63m   10.0.2.65
kubemaster.ittraining.loc   <none>           <none>
kube-system   kube-apiserver-kubemaster.ittraining.loc            1/1     Running   2 (55m ago)   63m   10.0.2.65
kubemaster.ittraining.loc   <none>           <none>
kube-system   kube-controller-manager-kubemaster.ittraining.loc   1/1     Running   5 (50m ago)   63m   10.0.2.65
kubemaster.ittraining.loc   <none>           <none>
```

```
kube-system   kube-proxy-fpksg                                  1/1     Running   0             62m   10.0.2.65
kubemaster.ittraining.loc   <none>           <none>
kube-system   kube-proxy-sn26v                                  1/1     Running   0             50m
192.168.56.3     kubenode1.ittraining.loc    <none>           <none>
kube-system   kube-proxy-wxm4z                                  1/1     Running   0             52m
192.168.56.4     kubenode2.ittraining.loc    <none>           <none>
kube-system   kube-scheduler-kubemaster.ittraining.loc          1/1     Running   5 (51m ago)   63m   10.0.2.65
kubemaster.ittraining.loc   <none>           <none>
```

Important : Notez que sur **kubenode1.ittraining.loc**, il y a 4 pods, à savoir **myapp-deployment-57c6cb89d9-q7d4p**, **nginx**, **calico-node-5htrc** et **kube-proxy-sn26v**.

Procédez maintenant au drain de kubenode1.ittraining.loc :

```
root@kubemaster:~# kubectl drain kubenode1.ittraining.loc
node/kubenode1.ittraining.loc cordoned
error: unable to drain node "kubenode1.ittraining.loc" due to error:[cannot delete Pods declare no controller
(use --force to override): default/nginx, cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to
ignore): kube-system/calico-node-5htrc, kube-system/kube-proxy-sn26v], continuing command...
There are pending nodes to be drained:
 kubenode1.ittraining.loc
cannot delete Pods declare no controller (use --force to override): default/nginx
cannot delete DaemonSet-managed Pods (use --ignore-daemonsets to ignore): kube-system/calico-node-5htrc, kube-
system/kube-proxy-sn26v
```

Notez que la commande retourne deux erreurs :

- cannot delete Pods declare no controller (use –force to override): default/nginx
- cannot delete DaemonSet-managed Pods (use –ignore-daemonsets to ignore): kube-system/calico-node-5htrc, kube-system/kube-proxy-sn26v

La première erreur est due au fait que l'opération ne peux pas déplacer un pod isolé, autrement dit un pod qui n'est pas géré par un Controller d'un

vers un autre noeud. Dans ce cas, le drain ne peut que supprimer le pod **nginx** et refuse donc de le faire sans l'utilisation de l'option **–force**.

Important : Le mot Controller implique un ReplicationController, un ReplicaSet, un Job, un DaemonSet et un StatefulSet.

La deuxième erreur est due au fait que l'opération ne peut pas traiter les DaemonSets.

Important : Un DaemonSet contient des pods qui sont **liés** à des noeuds **spécifiques**.

Exécutez donc la commande de nouveau en ajoutant les deux options **–ignore-daemonsets** et **–force** :

```
root@kubemaster:~# kubectl drain kubenode1.ittraining.loc --ignore-daemonsets --force
node/kubenode1.ittraining.loc already cordoned
WARNING: deleting Pods that declare no controller: default/nginx; ignoring DaemonSet-managed Pods: kube-
system/calico-node-5htrc, kube-system/kube-proxy-sn26v
evicting pod default/nginx
evicting pod default/myapp-deployment-57c6cb89d9-f69nk
pod/nginx evicted
pod/myapp-deployment-57c6cb89d9-f69nk evicted
node/kubenode1.ittraining.loc drained
```

Important : Notez que la commande n'a pas retourné d'erreurs.

Consultez de nouveau l'état des pods :

```
root@kubemaster:~# kubectl get pods -o wide --all-namespaces
```

```
NAMESPACE     NAME                                                  READY   STATUS    RESTARTS      AGE     IP
NODE                        NOMINATED NODE   READINESS GATES
default       myapp-deployment-57c6cb89d9-dh4cb                     1/1     Running   0             45m
192.168.150.2    kubenode2.ittraining.loc    <none>           <none>
default       myapp-deployment-57c6cb89d9-f69nk                     1/1     Running   0             45m
192.168.150.3    kubenode2.ittraining.loc    <none>           <none>
default       myapp-deployment-57c6cb89d9-l7lkd                     1/1     Running   0             6m22s
192.168.150.4    kubenode2.ittraining.loc    <none>           <none>
kube-system   calico-kube-controllers-6799f5f4b4-zk298              1/1     Running   0             77m
192.168.55.195   kubemaster.ittraining.loc   <none>           <none>
kube-system   calico-node-5htrc                                     1/1     Running   0             68m
192.168.56.3     kubenode1.ittraining.loc    <none>           <none>
kube-system   calico-node-dc7hd                                     1/1     Running   0             77m
10.0.2.65        kubemaster.ittraining.loc   <none>           <none>
kube-system   calico-node-qk5kt                                     1/1     Running   0             70m
192.168.56.4     kubenode2.ittraining.loc    <none>           <none>
kube-system   coredns-6d4b75cb6d-kxtqk                              1/1     Running   0             80m
192.168.55.194   kubemaster.ittraining.loc   <none>           <none>
kube-system   coredns-6d4b75cb6d-td7cf                              1/1     Running   0             80m
192.168.55.193   kubemaster.ittraining.loc   <none>           <none>
kube-system   etcd-kubemaster.ittraining.loc                        1/1     Running   1 (74m ago)   80m
10.0.2.65        kubemaster.ittraining.loc   <none>           <none>
kube-system   kube-apiserver-kubemaster.ittraining.loc              1/1     Running   2 (73m ago)   80m
10.0.2.65        kubemaster.ittraining.loc   <none>           <none>
kube-system   kube-controller-manager-kubemaster.ittraining.loc     1/1     Running   5 (67m ago)   80m
10.0.2.65        kubemaster.ittraining.loc   <none>           <none>
kube-system   kube-proxy-fpksg                                      1/1     Running   0             80m
10.0.2.65        kubemaster.ittraining.loc   <none>           <none>
kube-system   kube-proxy-sn26v                                      1/1     Running   0             68m
192.168.56.3     kubenode1.ittraining.loc    <none>           <none>
kube-system   kube-proxy-wxm4z                                      1/1     Running   0             70m
192.168.56.4     kubenode2.ittraining.loc    <none>           <none>
kube-system   kube-scheduler-kubemaster.ittraining.loc              1/1     Running   5 (68m ago)   80m
10.0.2.65        kubemaster.ittraining.loc   <none>           <none>
```

Important : Notez que le pod **nginx** a été détruit tandis que le pod **myapp-deployment-57c6cb89d9-q7d4p** a été **expulsé**. Un nouveau pod dénommé **myapp-deployment-57c6cb89d9-l7lkd** a été créé sur **kubenode2.ittraining.loc** afin de maintenir le nombre à **3**. Les deux pods **calico-node-5htrc** et **kube-proxy-sn26v** ont été ignorés.

Constatez maintenant l'état des noeuds :

```
root@kubemaster:~# kubectl get nodes
NAME                        STATUS                     ROLES           AGE   VERSION
kubemaster.ittraining.loc   Ready                      control-plane   91m   v1.24.2
kubenode1.ittraining.loc    Ready,SchedulingDisabled   <none>          80m   v1.24.2
kubenode2.ittraining.loc    Ready                      <none>          82m   v1.24.2
```

Important : Notez que le STATUS de **kubenode1.ittraining.loc** est **SchedulingDisabled** ce qui implique que le noeud n'accepte plus de nouveaux pods. Dans cet état le neoud est dit **cordoned**.

4.2 - La Commande uncordon

Pour permettre le noeud de recevoir de nouveau des pods, il convient d'utiliser la commande suivante :

```
root@kubemaster:~# kubectl uncordon kubenode1.ittraining.loc
node/kubenode1.ittraining.loc uncordoned
```

Constatez de nouveau l'état des noeuds :

```
root@kubemaster:~# kubectl get nodes
NAME                        STATUS   ROLES           AGE    VERSION
kubemaster.ittraining.loc   Ready    control-plane   124m   v1.24.2
kubenode1.ittraining.loc    Ready    <none>          113m   v1.24.2
kubenode2.ittraining.loc    Ready    <none>          115m   v1.24.2
```

Dernièrement consultez de nouveau l'état des pods :

```
root@kubemaster:~# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE   READINESS GATES
myapp-deployment-57c6cb89d9-dh4cb   1/1     Running   0          91m   192.168.150.2   kubenode2.ittraining.loc
<none>           <none>
myapp-deployment-57c6cb89d9-f69nk   1/1     Running   0          91m   192.168.150.3   kubenode2.ittraining.loc
<none>           <none>
myapp-deployment-57c6cb89d9-l7lkd   1/1     Running   0          52m   192.168.150.4   kubenode2.ittraining.loc
<none>           <none>
```

Important : Notez que l'utilisation de la commande **uncordon** n'implique pas le basculement du pod **l7lkd** vers le noeud **kubenode2.ittraining.loc**.

LAB #5 - Gestion des Mises-à-jour

5.1 - Mise-à-jour de kubeadm

Commencez par modifier les sources de paquets :

```
root@kubemaster:~# mkdir /etc/apt/keyrings
```

```
root@kubemaster:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg

root@kubemaster:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.25/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /

root@kubemaster:~# vi /etc/apt/sources.list

root@kubemaster:~# cat /etc/apt/sources.list
deb http://archive.debian.org/debian/ stretch main
deb-src http://archive.debian.org/debian/ stretch main
deb [arch=amd64] https://download.docker.com/linux/debian stretch stable

root@kubemaster:~# apt update
Ign:1 http://archive.debian.org/debian stretch InRelease
Atteint:2 http://archive.debian.org/debian stretch Release
Réception de:3 https://download.docker.com/linux/debian stretch InRelease [44,8 kB]
Réception de:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  InRelease
[1 192 B]
Réception de:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.28/deb  Packages
[21,3 kB]
67,3 ko réceptionnés en 0s (190 ko/s)
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
8 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Afin de mettre à jour kubeadm, il convient de faire un drain du **Contrôleur** :

```
root@kubemaster:~# kubectl drain kubemaster.ittraining.loc --ignore-daemonsets
node/kubemaster.ittraining.loc cordoned
WARNING: ignoring DaemonSet-managed Pods: kube-system/calico-node-mp24s, kube-system/kube-proxy-btng8
evicting pod kube-system/coredns-6d4b75cb6d-t5rqf
```

```
evicting pod kube-system/calico-kube-controllers-bc5cbc89f-slc7s
evicting pod kube-system/coredns-6d4b75cb6d-hncvw
pod/calico-kube-controllers-bc5cbc89f-slc7s evicted
pod/coredns-6d4b75cb6d-hncvw evicted
pod/coredns-6d4b75cb6d-t5rqf evicted
node/kubemaster.ittraining.loc drained
```

Afin de connaître la ou les version(s) supérieure(s) à celle installée, utilisez la commande suivante :

```
root@kubemaster:~# apt-cache madison kubeadm
   kubeadm | 1.25.16-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.15-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.14-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.13-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.12-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.11-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.10-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.9-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.8-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.7-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.6-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.5-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.4-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.3-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.2-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.1-1.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
   kubeadm | 1.25.0-2.1 | https://pkgs.k8s.io/core:/stable:/v1.25/deb  Packages
```

Procédez maintenant à la mise-à-jour de kubeadm :

```
root@kubemaster:~# apt-get update && apt-get install -y --allow-change-held-packages kubeadm=1.25.0-2.1
Ign:1 http://archive.debian.org/debian stretch InRelease
Atteint:2 http://archive.debian.org/debian stretch Release
Réception de:3 https://download.docker.com/linux/debian stretch InRelease [44,8 kB]
```

```
Atteint:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.25/deb  InRelease
44,8 ko réceptionnés en 0s (81,7 ko/s)
Lecture des listes de paquets... Fait
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  libjsoncpp1 linux-image-4.9.0-8-amd64
Veuillez utiliser « apt autoremove » pour les supprimer.
Les paquets retenus suivants seront changés :
  kubeadm
Les paquets suivants seront mis à jour :
  kubeadm
1 mis à jour, 0 nouvellement installés, 0 à enlever et 7 non mis à jour.
Il est nécessaire de prendre 9 219 ko dans les archives.
Après cette opération, 537 ko d'espace disque seront libérés.
Réception de:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.25/deb  kubeadm
1.25.0-2.1 [9 219 kB]
9 219 ko réceptionnés en 0s (14,6 Mo/s)
apt-listchanges : Lecture des fichiers de modifications (« changelog »)...
(Lecture de la base de données... 137041 fichiers et répertoires déjà installés.)
Préparation du dépaquetage de .../kubeadm_1.25.0-2.1_amd64.deb ...
Dépaquetage de kubeadm (1.25.0-2.1) sur (1.24.2-00) ...
dpkg: avertissement: impossible de supprimer l'ancien répertoire  « /etc/systemd/system/kubelet.service.d » : Le
dossier n'est pas vide
Paramétrage de kubeadm (1.25.0-2.1) ...
```

Important : Notez que l'utilisation de l'option **–allow-change-held-packages**.

Vérifiez que la version désirée a été installée :

```
root@kubemaster:~# kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"25", GitVersion:"v1.25.0",
GitCommit:"a866cbe2e5bbaa01cfd5e969aa3e033f3282a8a2", GitTreeState:"clean", BuildDate:"2022-08-23T17:43:25Z",
GoVersion:"go1.19", Compiler:"gc", Platform:"linux/amd64"}
```

Afin de connaître les version des composants du Control Plane compatibles avec la version 1.25.0 de kubeadm, utilisez la commande **kubeadm upgrade plan** :

```
root@kubemaster:~# kubeadm upgrade plan
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o
yaml'
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.24.2
[upgrade/versions] kubeadm version: v1.25.0
I0314 07:25:04.222393    8210 version.go:256] remote version is much newer: v1.29.2; falling back to: stable-1.25
[upgrade/versions] Target version: v1.25.16
[upgrade/versions] Latest version in the v1.24 series: v1.24.17

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT   CURRENT       TARGET
kubelet     3 x v1.24.2   v1.24.17

Upgrade to the latest version in the v1.24 series:

COMPONENT                 CURRENT   TARGET
kube-apiserver            v1.24.2   v1.24.17
kube-controller-manager   v1.24.2   v1.24.17
kube-scheduler            v1.24.2   v1.24.17
kube-proxy                v1.24.2   v1.24.17
```

```
CoreDNS                     v1.8.6    v1.9.3
etcd                        3.5.3-0   3.5.4-0

You can now apply the upgrade by executing the following command:

        kubeadm upgrade apply v1.24.17


_____________________________________________________________________

Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT   CURRENT       TARGET
kubelet     3 x v1.24.2   v1.25.16

Upgrade to the latest stable version:

COMPONENT                 CURRENT   TARGET
kube-apiserver            v1.24.2   v1.25.16
kube-controller-manager   v1.24.2   v1.25.16
kube-scheduler            v1.24.2   v1.25.16
kube-proxy                v1.24.2   v1.25.16
CoreDNS                   v1.8.6    v1.9.3
etcd                      3.5.3-0   3.5.4-0

You can now apply the upgrade by executing the following command:

        kubeadm upgrade apply v1.25.16

Note: Before you can perform this upgrade, you have to update kubeadm to v1.25.16.


_____________________________________________________________________


The table below shows the current state of component configs as understood by this version of kubeadm.
Configs that have a "yes" mark in the "MANUAL UPGRADE REQUIRED" column require manual config upgrade or
```

```
resetting to kubeadm defaults before a successful upgrade can be performed. The version to manually
upgrade to is denoted in the "PREFERRED VERSION" column.

API GROUP                 CURRENT VERSION   PREFERRED VERSION   MANUAL UPGRADE REQUIRED
kubeproxy.config.k8s.io   v1alpha1          v1alpha1            no
kubelet.config.k8s.io     v1beta1           v1beta1             no
_____________________________________________________________________
```

Procédez donc à la mise-à-jour de kubeadm vers la version **1.25.0** :

```
root@kubemaster:~# kubeadm upgrade apply v1.25.0
[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o
yaml'
[preflight] Running pre-flight checks.
[upgrade] Running cluster health checks
[upgrade/version] You have chosen to change the cluster version to "v1.25.0"
[upgrade/versions] Cluster version: v1.24.2
[upgrade/versions] kubeadm version: v1.25.0
[upgrade] Are you sure you want to proceed? [y/N]: y
```

A l'issu de processus, vous verrez les deux lignes suivantes :

```
...
[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.25.0". Enjoy!

[upgrade/kubelet] Now that your control plane is upgraded, please proceed with upgrading your kubelets if you
haven't already done so.
root@kubemaster:~#
```

Mettez-à-jour maintenant **kubelet** et **kubectl** :

```
root@kubemaster:~# apt-get update && apt-get install -y --allow-change-held-packages kubelet=1.25.0-2.1
```

```
kubectl=1.25.0-2.1
...
```

Au cas où le fichier du service de kubelet a subi des modifications, re-démarrez le daemon systemctl ainsi que le service kubelet :

```
root@kubemaster:~# systemctl daemon-reload

root@kubemaster:~# systemctl restart kubelet
```

Annulez le drain de kubemaster :

```
root@kubemaster:~# export KUBECONFIG=/etc/kubernetes/admin.conf

root@kubemaster:~# kubectl uncordon kubemaster.ittraining.loc
node/kubemaster.ittraining.loc uncordoned
```

Constatez maintenant l'état des noeuds :

```
root@kubemaster:~# kubectl get nodes
NAME                        STATUS   ROLES           AGE     VERSION
kubemaster.ittraining.loc   Ready    control-plane   3h15m   v1.25.0
kubenode1.ittraining.loc    Ready    <none>          3h4m    v1.24.2
kubenode2.ittraining.loc    Ready    <none>          3h6m    v1.24.2
```

Important : Notez que le Control Plane est à la version 1.25.0 tandis que les Travailleurs sont à la version 1.24.2.

5.2 - Mise-à-jour des Travailleurs

Afin de mettre à jour un Travailleur, il convient de faire un drain du Travailleur concerné :

```
root@kubemaster:~# kubectl drain kubenode1.ittraining.loc --ignore-daemonsets --force
node/kubenode1.ittraining.loc cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/calico-node-hgrt9, kube-system/kube-proxy-czrqt
evicting pod kube-system/calico-kube-controllers-bc5cbc89f-q2zkl
pod/calico-kube-controllers-bc5cbc89f-q2zkl evicted
node/kubenode1.ittraining.loc drained
```

Connectez-vous à kubenode1 :

```
root@kubemaster:~# ssh -l trainee kubenode1
trainee@kubenode1's password:
Linux kubenode1.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Mar 14 06:43:48 2024 from 192.168.56.2
trainee@kubenode1:~$ su -
Mot de passe :
root@kubenode1:~#
```

Commencez par modifier les sources de paquets :

```
root@kubemaster:~# mkdir /etc/apt/keyrings

root@kubemaster:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg

root@kubemaster:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
```

```
https://pkgs.k8s.io/core:/stable:/v1.25/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /

root@kubenode1:~# vi /etc/apt/sources.list

root@kubenode1:~# cat /etc/apt/sources.list
deb http://archive.debian.org/debian/ stretch main
deb-src http://archive.debian.org/debian/ stretch main
deb [arch=amd64] https://download.docker.com/linux/debian stretch stable

root@kubenode1:~# apt update
```

Mettez-à-jour le paquet **kubeadm** :

```
root@kubenode1:~# apt-get update && apt-get install -y --allow-change-held-packages kubeadm=1.25.0-2.1
...
```

Mettez-à-jour la configuration de kubeadm :

```
root@kubenode1:~# kubeadm upgrade node
[upgrade] Reading configuration from the cluster...
[upgrade] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[preflight] Running pre-flight checks
[preflight] Skipping prepull. Not a control plane node.
[upgrade] Skipping phase. Not a control plane node.
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[upgrade] The configuration for this node was successfully updated!
[upgrade] Now you should go ahead and upgrade the kubelet package using your package manager.
```

Mettez-à-jour maintenant **kubelet** et **kubectl** :

```
root@kubenode1:~# apt-get update && apt-get install -y --allow-change-held-packages kubelet=1.25.0-2.1
kubectl=1.25.0-2.1
```

```
...
```

Au cas où le fichier du service de kubelet a subi des modifications, re-démarrez le daemon systemctl ainsi que le service kubelet :

```
root@kubenode1:~# systemctl daemon-reload

root@kubenode1:~# systemctl restart kubelet
```

Retournez à la machine **kubemaster** :

```
root@kubenode1:~# exit
déconnexion
trainee@kubenode1:~$ exit
déconnexion
Connection to kubenode1 closed.
root@kubemaster:~#
```

Annulez le drain de kubenode1 :

```
root@kubemaster:~# kubectl uncordon kubenode1.ittraining.loc
node/kubenode1.ittraining.loc uncordoned
```

Constatez maintenant l'état des noeuds :

```
root@kubemaster:~# kubectl get nodes
NAME                        STATUS   ROLES           AGE     VERSION
kubemaster.ittraining.loc   Ready    control-plane   3h43m   v1.25.0
kubenode1.ittraining.loc    Ready    <none>          3h32m   v1.25.0
kubenode2.ittraining.loc    Ready    <none>          3h34m   v1.24.2
```

Important : Notez que le Control Plane et kubenode1 sont à la version 1.25.0 tandis que kubenode2 est à la version 1.24.2.

Faites un drain du kubenode2 :

```
root@kubemaster:~# kubectl drain kubenode2.ittraining.loc --ignore-daemonsets --force
node/kubenode2.ittraining.loc cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/calico-node-7q8nc, kube-system/kube-proxy-xmqkj
evicting pod kube-system/coredns-565d847f94-b6j2v
evicting pod kube-system/calico-kube-controllers-bc5cbc89f-zfdlb
pod/calico-kube-controllers-bc5cbc89f-zfdlb evicted
pod/coredns-565d847f94-b6j2v evicted
node/kubenode2.ittraining.loc drained
```

Connectez-vous à kubenode2 :

```
root@kubemaster:~# ssh -l trainee kubenode2
trainee@kubenode2's password:
Linux kubenode2.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Mar 14 06:45:08 2024 from 192.168.56.2
trainee@kubenode2:~$ su -
Mot de passe :
root@kubenode2:~#
```

Commencez par modifier les sources de paquets :

```
root@kubemaster:~# mkdir /etc/apt/keyrings

root@kubemaster:~# curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
root@kubemaster:~# echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.25/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /

root@kubenode1:~# vi /etc/apt/sources.list

root@kubenode1:~# cat /etc/apt/sources.list
deb http://archive.debian.org/debian/ stretch main
deb-src http://archive.debian.org/debian/ stretch main
deb [arch=amd64] https://download.docker.com/linux/debian stretch stable

root@kubenode1:~# apt update
```

Mettez-à-jour le paquet **kubeadm** :

```
root@kubenode2:~# apt-get update && apt-get install -y --allow-change-held-packages kubeadm=1.25.0-2.1
...
```

Mettez-à-jour la configuration de kubeadm :

```
root@kubenode2:~# kubeadm upgrade node
[upgrade] Reading configuration from the cluster...
[upgrade] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[preflight] Running pre-flight checks
[preflight] Skipping prepull. Not a control plane node.
[upgrade] Skipping phase. Not a control plane node.
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[upgrade] The configuration for this node was successfully updated!
[upgrade] Now you should go ahead and upgrade the kubelet package using your package manager.
```

Mettez-à-jour maintenant **kubelet** et **kubectl** :

```
root@kubenode2:~# apt-get update && apt-get install -y --allow-change-held-packages kubelet=1.25.0-2.1
kubectl=1.25.0-2.1
```

```
...
```

Au cas où le fichier du service de kubelet a subi des modifications, re-démarrez le daemon systemctl ainsi que le service kubelet :

```
root@kubenode2:~# systemctl daemon-reload

root@kubenode2:~# systemctl restart kubelet
```

Retournez à la machine **kubemaster** :

```
root@kubenode2:~# exit
déconnexion
trainee@kubenode2:~$ exit
déconnexion
Connection to kubenode2 closed.
root@kubemaster:~#
```

Annulez le drain de kubenode1 :

```
root@kubemaster:~# kubectl uncordon kubenode2.ittraining.loc
node/kubenode2.ittraining.loc uncordoned
```

Constatez maintenant l'état des noeuds :

```
root@kubemaster:~# kubectl get nodes
NAME                        STATUS   ROLES           AGE     VERSION
kubemaster.ittraining.loc   Ready    control-plane   3h56m   v1.25.0
kubenode1.ittraining.loc    Ready    <none>          3h45m   v1.25.0
kubenode2.ittraining.loc    Ready    <none>          3h47m   v1.25.0
```

Important : Notez que tout a été mis-à-jour.

Copyright © 2024 Hugh Norris