Version - **2025.01**

Last update : 2025/01/17 16:38

DOE302 - Managing Pods, Replication Controllers, ReplicaSets, Deployments, Maintenance and Cluster Updates

Curriculum

LAB #1 - Creating a pod

1.1 - Introducing a pod

A pod is an object that encapsulates a container. The container is an instance of an application. The relationship between a pod and an application container is generally 1:1, i.e. in the case of an increase in load, additional pods are created, each containing an application container, rather than creating several containers in the same pod.

Conversely, when load is reduced, pods are destroyed. With Kubernetes, you can't create multiple containers of the same type in the same pod. However, it is possible to have containers of different types in the same pod.

In this case, we speak of an application container and one or more **Helper** containers. The application container and the Helper container can communicate directly because they share the same **network space**. They also have access to the same **storage space**.

A pod therefore frees the administrator from having to manage Docker **links** and **volumes**.

When a pod is created with the **kubectl** command, it downloads the Docker image needed to create the container from the Docker Hub.

1.2 - Manual pod creation

Start by creating a pod called **nginx** from the nginx image:

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

View the pod with the **kubectl** command:

```
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
```

Consult the information concerning this 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: Note that the first line of the **Events** section clearly indicates that in this example, the kubemaster has scheduled the pod on kubenode1.

Now use the kubectl command with the **-o wide** option:

```
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: Note that the pod's IP address is **192.168.239.1**. This address is **dynamic**. If the pod stops and another one starts, the IP address of the new pod will be different.

Important: Note that the **NOMINATED NODE** column is marked **<none>**. This is because it's possible to assign a pod to a specific node by using a label defined for the nominated node(s). For more information, go to **https://kubernetes.io/docs/tasks/configure-pod-container/assign-pods-nodes/**.

Important: Note that in the **READINESS GATES** column it says **<none>**. In fact, it is possible to assign specific conditions to a pod so that Kubenetes considers the pod to be in a **ready** state. For more information, visit the **https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#pod-readines**

s-gate link.

1.3 - Creating a pod using a YAML file

Kubernetes uses YAML files to create objects. Consequently, the definition of the pod to be created is described in a YAML file. Create the file **pod-definition.yaml** :

To do: Copy the content from **here** **and paste it into your file.**

```
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
```

This file contains the following fields:

apiVersion

- This field is **required**,
- The API version differs according to the type of object created,
- The field value is a string.

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

kind

- This field is **required**,
- The value of the apiServer in relation to the object type is :

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

metadata

- This field is **mandatory**,
- It contains information such as name and labels,
- The information is in the form of a YAML **dictionary**:

```
metadata:
  name: myapp-pod
  labels:
    app: myapp
```

```
    type: front-end
```

spec

* This field is **required**, * It contains information for Kubernetes specific to the type of object to be created, * The information is in the form of a YAML **list**:

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

Using the YAML file

Now use the YAML file to create a pod:

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

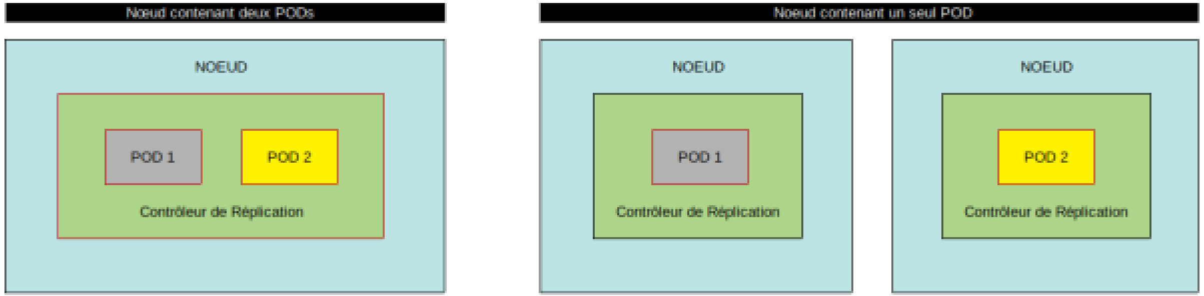
```
root@kubemaster:~# kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
myapp-pod   1/1     Running   0          23s
nginx       1/1     Running   0          29m
```

LAB #2 - Using Replication Controllers and ReplicaSets

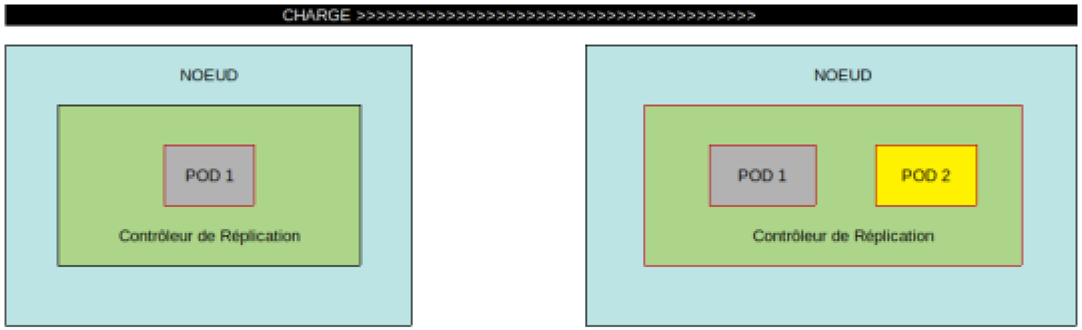
2.1 - Replication Controllers

Overview

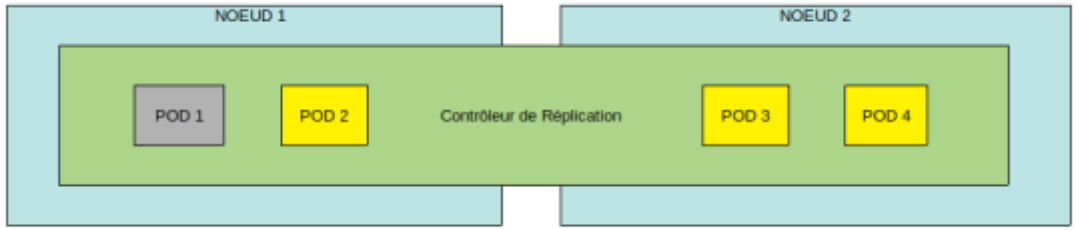
A Replication Controller allows multiple instances of the same pod to be run, providing **high availability** in the event of an application crash and pod failure. Even if there is only one pod, the Replication Controller can automatically start another pod containing the application:



A Replication Controller can also be used to start new pods in the event of an increase in the load, as well as to ensure load balancing between pods:



In the event that the first node runs out of resources, a Replication Controller is able to start new pods on a second node:



Implementation

To create a Replication Controller, you need to create a YAML file. Create the file **cr-definition.yaml** :

To do: Copy the content from **here** **and paste it into your file.**

```
---
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
```

This file contains a section called **template**. This section is a template for the creation of additional pods and is identical to the contents of the **pod-definition.yaml** file without the apiVersion and kind fields:

```
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
```

The **replicas** field indicates the number of pods to be created.

Use the rc-definition.yaml file to create the Replication Controller:

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

To view the Replication Controller, use the following command:

```
root@kubemaster:~# kubectl get replicationcontroller
```

```
NAME       DESIRED   CURRENT   READY   AGE
myapp-cr   3         3         3       71s
```

To view the pods created by the Replication Controller, use the following command:

```
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: Note that the Replication Controller has created two replicas **myapp-cr-6gxg6** and **myapp-cr-78frz** because the first already existed: **myapp-pod**. To identify an existing pod of the same type, the Replication Controller relies on the **labels** field in the **template** section.

Now delete the pod **myapp-pod**:

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

Then note the Replication Controller's reaction:

```
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: Note that the Replication Controller has created the **myapp-cr-pt4zt** pod.

To view the status of a Replication Controller, use the following command:

```
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
```

To delete a Replication Controller, use the following command:

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

2.2 - ReplicaSets

Overview

A ReplicaSet performs the same function as a Replication Controller. ReplicaSets are the latest way to manage replication.

Implementation

To create a ReplicaSet, create the file **replicaset-definition.yaml** :

To do: Copy the content from **here** **and paste it into your file.**

```
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: Note that in the case of a ReplicaSet, it identifies the pods under its control by the value of the **matchLabels** field.

Use the replicaset-definition.yaml file to create the ReplicaSet:

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

To view the ReplicaSet, use the following command:

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

To view the pods created by ReplicaSet, use the following command:

```
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
```

Now modify the **replicaset-definition.yaml** file, increasing the number of replicas from 3 to **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
```

Then run the **kubectl replace** command:

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

View the 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
```

View the pods created by the 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
```

Then run the following command:

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

View the ReplicaSet :

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

View the pods created by 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
```

Note that in this case, the **replicaset** value in the **replicaset-definition.yaml** file has not been modified:

```
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
```

Finally, run the following command :

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

View the ReplicaSet :

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

View the pods created by the 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
```

Now create a pod outside of the ReplicaSet :

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

View the list of 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: Note that **myapp-pod** is in a **Terminating** state. ReplicaSet does not allow the creation of a pod with the same label as specified by the **matchLabels** field in **replicaset-definition.yaml**.

To delete the ReplicaSet, use the following command:

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

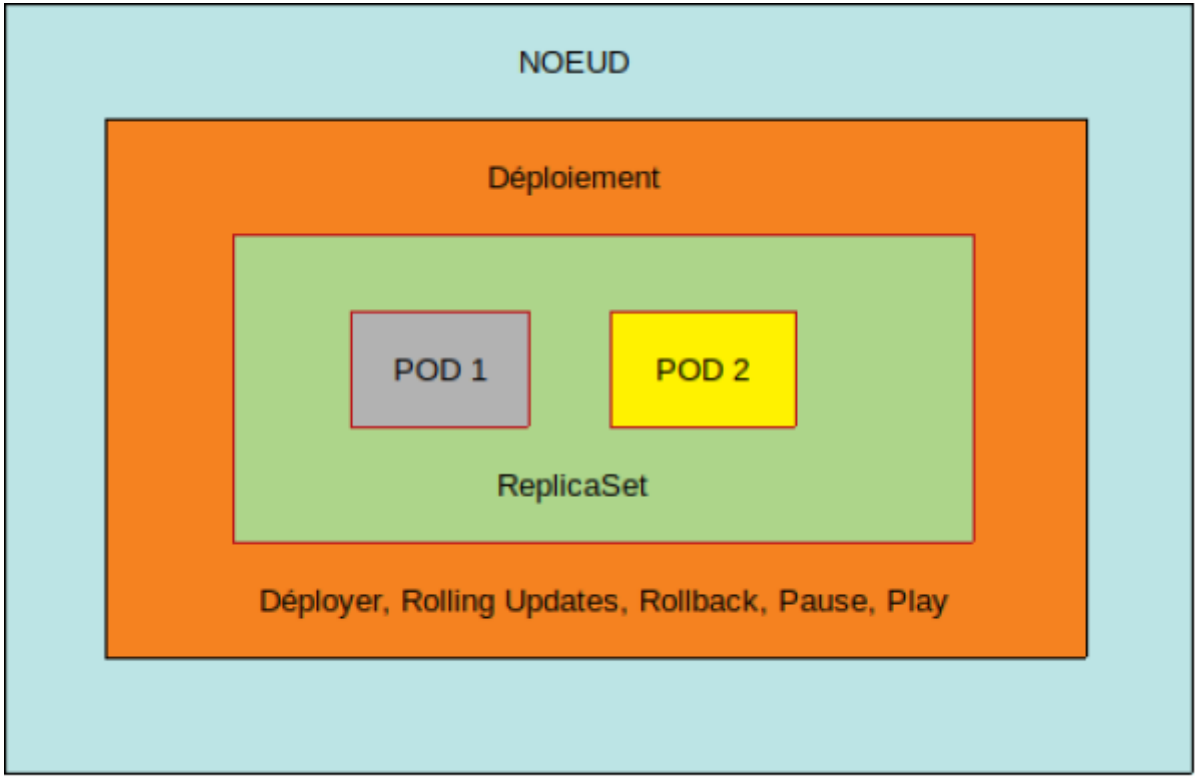
Now view all objects in the cluster:

```
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 - Managing Deployments

3.1 - Overview

A **Deployment** in Kubernetes is an object hierarchically superior to a ReplicaSet :



A Deployment is used to manage :

- Rollouts,
- Rolling Updates,
- Rollbacks.

3.2 - Implementation

Rollouts

To create a Deployment, you need to create a YAML file. Create the file **deployment-definition.yaml** :

To do: Copy the content from **here** **and paste it into your file.**

```
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
```

Use the following command to create the Deployment :

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

View the Deployment:

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

Note that creating the Deployment also created a ReplicaSet :

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

Important: Note that the value **689f9d59** is randomly generated internally by Kubernetes.

The creation of the Deployment created the number of pods indicated in the YAML file:

```
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
```

To see all these objects at once, use the **kubectl get all** command:

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

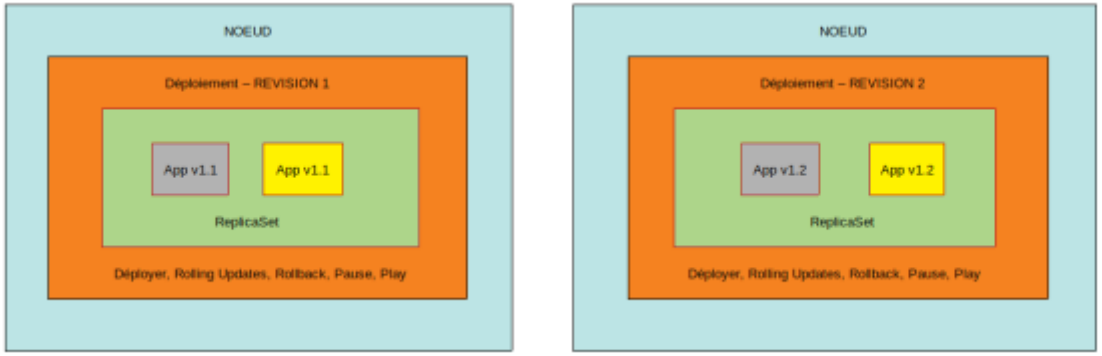
```
pod/myapp-deployment-689f9d59-kt88s   1/1     Running   0          2m10s
pod/myapp-deployment-689f9d59-zlwp4   1/1     Running   0          2m10s

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
```

For more information about the Deployment, use the **kubectl describe** command:

```
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
```

During the Deployment Rollout, a **Revision** is created. This Revision is incremented with each update:



To view the Rollout status, use the following command:

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

To view the list of Revisions, use the following command:

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

Important: Note that the value of **CHANGE-CAUSE** is **<none>** because the **–record** option has not been specified on the command line. It is possible to modify the **CHANGE-CAUSE** value with the **kubectl annotate deployment <deployment> kubernetes.io/change-cause="<Message>" –record=false –overwrite=true** command.

Delete the Deployment with the following command:

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

Check that Deployment has been deleted:

```
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
```

Create the Deployment again, adding the **–record** option:

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

Check the Rollout status:

```
root@kubemaster:~# kubectl rollout status deployment/myapp-deployment
```

```
deployment “myapp-deployment” successfully rolled out
```

Important: Note that a Deployment can be paused with the **kubectl rollout pause deployment <deployment>** command and can be resumed with the **kubectl rollout resume deployment <deployment>** command.

View the list of Revisions :

```
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: Note that the value of **CHANGE-CAUSE** is the command that was executed.

Rolling Updates

There are two Deployment methods for rolling updates:

- **Recreate**,
  - In this case, all existing pods are destroyed at the same time, and pods containing the update are created at a later stage. The disadvantage of this method is obvious - between the destruction of the pods and the re-creation of the new pods, the application is not available,
- **Rolling Update**
  - In this case, pods are destroyed one by one. After each deletion, a new pod is created containing the update. In this way, the application remains available.

Important: Note that **Rolling Update** is the default method.

Now modify the **deployment-description.yaml** file, specifying version **1.12** of **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
```

Apply this change :

```
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
```

Check the status of the 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
```

Note that there is now an additional **Revision**:

```
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
```

View the details of **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: Note that the image used is indeed **nginx:1.12**. Then note that in the **Events** section, the pods have been **Scaled down** one-by-one and **Scaled up** one-by-one. Also note that the value of **StrategyType** can be either **Recreate** or **RollingUpdate**. Finally, note the value of **RollingUpdateStrategy**. **25% max unavailable** indicates that at time "t" 75% of all pods must be available, while **25% max surge** indicates that the total number of pods cannot exceed 1.25 times the value of the **Replicas** field. These values can be modified. See page **https://kubernetes.io/docs/concepts/workloads/controllers/deployment/**.

When updating, Deployment creates another ReplicaSet containing the updated pods, using the Rolling Update method. This can be seen by looking at the output of the **kubectl get replicasets** command:

```
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: Note that the number of old ReplicaSets retained is **10** by default. This value can be modified. See page **https://kubernetes.io/docs/concepts/workloads/controllers/deployment/**.

The image version can also be modified via the command line:

```
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
```

The **nginx-container** container name is defined in the pod definition file:

```
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
```

Check the status of the Deployment:

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

Note that there is now an additional **Revision**:

```
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
```

On update, Deployment creates another ReplicaSet containing the updated pods using the Rolling Update method. This can be seen by looking at the output of the **kubectl get replicasets** command:

```
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
```

See the details of **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: Note that the image used was **nginx:1.14**.

Rollbacks

Thanks to the **Revisions** system, you can go back to the previous **N-1** version of the application. Enter the following command:

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

Important: Note that it is possible to roll back to a specific previous version with the **kubectl rollout undo deployment <deployment> –to-revision=<revision>** command.

Enter the **kubectl get replicasets** command:

```
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: Note that the application has reverted to the previous ReplicaSet.

Use the **kubectl rollout history** command:

```
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: Note that Revision 2 has become Revision 4, demonstrating a rollback.

Now create a Rollout error:

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

Important: Note that the error is **nginx1.14** which should be **nginx:1.14**.

Check the Deployment status:

```
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: Note that the rollout is blocked. An error **error: deployment “myapp-deployment” exceeded its progress deadline** will be returned after about ten minutes!

To see what's happening, use the **kubectl get deployments** command:

```
root@kubemaster:~# kubectl get deployments
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
```

```
myapp-deployment   3/3     1            3           15m
```

The **kubectl get pods** command shows a status of **ImagePullBackOff** for the first pod in the new ReplicaSet which indicates that Kubernetes cannot perform the **pull** of the image from 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
```

When consulting the Rollout history, an additional Revision was added:

```
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
```

To rectify this error, you need to run a Rollback :

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

Note the success of the command :

```
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 - Maintenance

To perform maintenance on a node, it is often necessary to remove it from the cluster. This operation is called a **drain**.

4.1 - The drain command

Check the status of all the 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: Note that on **kubenode1.ittraining.loc**, there are 4 pods, namely **myapp-deployment-57c6cb89d9-q7d4p**, **nginx**, **calico-node-5htrc** and **kube-proxy-sn26v**.

Now drain 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
```

Note that the command returns two errors:

- 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

The first error is due to the fact that the operation cannot move an isolated pod, i.e. a pod not managed by a Controller, from one node to another. In this case, the drain can only delete the **nginx** pod and therefore refuses to do so without using the **–force** option.

Important: The word Controller implies a ReplicationController, a ReplicaSet, a Job, a DaemonSet or a StatefulSet.

The second error is due to the fact that the operation cannot process DaemonSets.

Important: A DaemonSet contains pods that are **linked** to **specific** nodes.

Run the command again, adding the two options **–ignore-daemonsets** and **–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: Note that the command returned no errors.

Check the pod status again:

```
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: Note that the **nginx** pod has been destroyed, while the **myapp-deployment-57c6cb89d9-q7d4p** pod has been **evicted**. A new pod called **myapp-deployment-57c6cb89d9-l7lkd** has been created on **kubenode2.ittraining.loc** to keep the number at **3**. The two pods **calico-node-5htrc** and **kube-proxy-sn26v** have been ignored.

Now look at the status of the nodes:

```
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: Note that the STATUS of **kubenode1.ittraining.loc** is **SchedulingDisabled**,

which means that the node no longer accepts new pods. In this state, the node is **cordoned**.

4.2 - The uncordon command

To enable the node to receive pods again, use the following command:

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

Check the status of the nodes again:

```
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
```

Lastly, check pod status again:

```
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: Note that using the **uncordon** command does not move the **l7lkd** pod to the **kubenode1.ittraining.loc** node.

LAB #5 - Managing Cluster Updates

5.1 - Updating kubeadm

Start by modifying the package sources :

```
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.
```

In order to upgrade kubeadm, a drain of the **Controller** is required:

```
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
```

To find out which version(s) is (are) higher than the installed one, use the following command:

```
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
```

Now update 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
Reached:2 http://archive.debian.org/debian stretch Release
Received:3 https://download.docker.com/linux/debian stretch InRelease [44,8 kB]
Reached:5 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.25/deb InRelease
44.8 kB received in 0s (81.7 kB/s)
Reading packet lists... Done
Read packet lists... Done
Build dependency tree
Read status information... Done
The following packages have been installed automatically and are no longer required:
libjsoncpp1 linux-image-4.9.0-8-amd64
Please use “apt autoremove” to remove them.
The following selected packages will be changed:
kubeadm
The following packages will be updated:
kubeadm
1 updated, 0 newly installed, 0 to be removed and 7 not updated.
It is necessary to take 9,219 kB from the archives.
After this operation, 537 kB of disk space will be freed up.
Receipt of: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 kB received in 0s (14.6 Mb/s)
apt-listchanges : Reads changelog files...
(Reading the database... 137041 files and directories already installed).
Preparing to unpack .../kubeadm_1.25.0-2.1_amd64.deb ...
Unpacking kubeadm (1.25.0-2.1) on (1.24.2-00) ...
dpkg: warning: unable to delete old directory “/etc/systemd/system/kubelet.service.d” : Folder not empty
kubeadm settings (1.25.0-2.1) ...
```

Important : Note the use of the **–allow-change-held-packages** option.

Check that the desired version has been installed:

```
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’}
```

To find out which versions of Control Plane components are compatible with kubeadm version 1.25.0, use the **kubeadm upgrade plan** command:

```
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
_____________________________________________________________________
```

Update kubeadm to 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
```

At the end of the process, you'll see the following two lines:

```
...
[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:~#
```

Upgrade now **kubelet** and **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
...
```

If the kubelet service file has changed, restart the systemctl daemon and the kubelet service:

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

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

Cancel the kubemaster drain:

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

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

Now check the status of the nodes:

```
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: Note that Control Plane is at version 1.25.0 while the Workers are at version 1.24.2.

5.2 - Updating Workers

To update a Worker, you need to drain the Worker concerned:

```
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
```

Connect to kubenode1 :

```
root@kubemaster:~# ssh -l trainee kubenode1
trainee@kubenode1's password: trainee
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 -
Password : fenestros
root@kubenode1:~#
```

Start by modifying the package sources:

```
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
```

Update the **kubeadm** package:

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

Update the node:

```
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.
```

Update **kubelet** and **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
...
```

If the kubelet service file has changed, restart the systemctl daemon and the kubelet service:

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

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

Return to the **kubemaster** machine:

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

Cancel the kubenode1 drain:

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

Now check the status of the nodes:

```
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: Note that Control Plane and kubenode1 are at version 1.25.0 while kubenode2 is at version 1.24.2.

Drain 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-zfdlbb
pod/calico-kube-controllers-bc5cbc89f-zfdlb evicted
pod/coredns-565d847f94-b6j2v evicted
node/kubenode2.ittraining.loc drained
```

Connect to kubenode2 :

```
root@kubemaster:~# ssh -l trainee kubenode2
trainee@kubenode2's password: trainee
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 -
Password : fenestros
root@kubenode2:~#
```

Start by modifying the package sources:

```
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
```

Update the **kubeadm** package:

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

```
...
```

Update the node:

```
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.
```

Update **kubelet** and **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
...
```

If the kubelet service file has changed, restart the systemctl daemon and the kubelet service:

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

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

Return to the **kubemaster** machine:

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

```
root@kubemaster:~#
```

Cancel kubenode1 drain:

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

Now check the status of the nodes:

```
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: Note that everything has been updated.
