The kubectl tool can be used to manage the Kubernetes cluster itself. A good example use case of these commands for this purpose would be removing a physical machine for repairs or upgrades. In this lab, you will use the commands cordon, drain, and uncordon.

Prevent Pod Scheduling
This lab has already created the Pod named bar in the default namespace. List the existing Pods in the default namespace with the following command. The -o wide command-line option will tell you which node the Pod runs on:

kubectl get pods -o wide

As you can see in the output, the Pod named bar runs on the node node01. Say you want to perform some maintenance on the node. Before you can begin your work, you have to ensure that no new workload is scheduled during the maintenance window. When you cordon a node, you prevent future Pods from being scheduled onto that machine.:

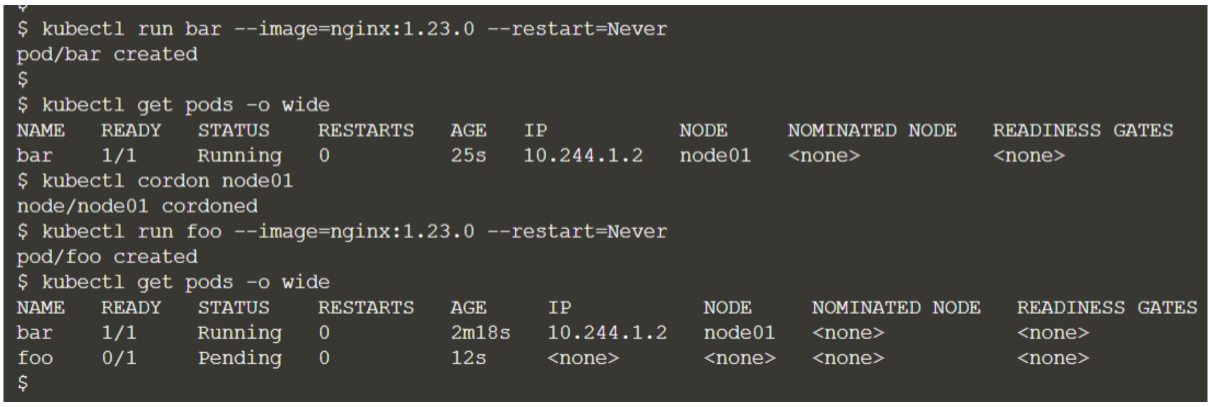
kubectl cordon node01

Try to schedule a new Pod named foo on the node with the following command:

kubectl run foo --image=nginx:1.23.0 --restart=Never

You will see that the scheduler puts the Pod into a "Pending" status, meaning it is awaiting for approval from the node to transition into the "Running" status:

kubectl get pods -o wide



```
$ kubectl run bar --image=nginx:1.23.0 --restart=Never
pod/bar created
$
$ kubectl get pods -o wide
NAME   READY   STATUS    RESTARTS   AGE   IP           NODE     NOMINATED NODE   READINESS GATES
bar    1/1     Running   0          25s   10.244.1.2   node01   <none>           <none>
$ kubectl cordon node01
node/node01 cordoned
$ kubectl run foo --image=nginx:1.23.0 --restart=Never
pod/foo created
$ kubectl get pods -o wide
NAME   READY   STATUS    RESTARTS   AGE     IP           NODE     NOMINATED NODE   READINESS GATES
bar    1/1     Running   0          2m18s   10.244.1.2   node01   <none>           <none>
foo    0/1     Pending   0          12s     <none>       <none>   <none>           <none>
$
```

Terminate Running Pods
When you drain a node, you remove any Pods that are currently running on that machine. The drain command achieves exactly that. After running the command, you can safely remove the machine from the cluster. You may have to force the operation using the --force option. Additionally, provide the option --ignore-daemonsets so a workload can be drained if it has been created by a ReplicationController, ReplicaSet, Job, DaemonSet, or StatefulSet.

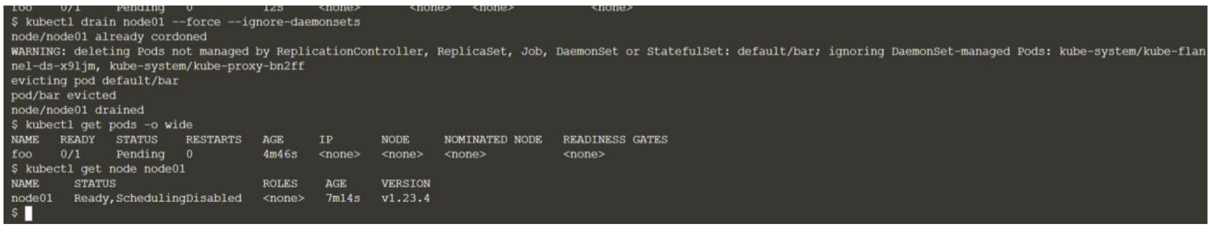
kubectl drain node01 --force --ignore-daemonsets

After draining the node, you will see no more Pods in the "Running" status. Pods in any other state (e.g., "Pending") will still be listed:

kubectl get pods -o wide

You can begin node maintenance at this point. The status of the node will indicate that scheduling has been disabled via the flag SchedulingDisabled. Run the following command to verify:

kubectl get node node01



```
$ kubectl drain node01 --force --ignore-daemonsets
node/node01 already cordoned
WARNING: deleting Pods not managed by ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet: default/bar; ignoring DaemonSet-managed Pods: kube-system/kube-flan
nel-ds-x9ljm, kube-system/kube-proxy-bn2ff
evicting pod default/bar
pod/bar evicted
node/node01 drained
$ kubectl get pods -o wide
NAME   READY   STATUS    RESTARTS   AGE     IP       NODE     NOMINATED NODE   READINESS GATES
foo    0/1     Pending   0          4m46s   <none>   <none>   <none>           <none>
$ kubectl get node node01
NAME     STATUS                     ROLES    AGE     VERSION
node01   Ready,SchedulingDisabled   <none>   7m14s   v1.23.4
$
```

Re-enable Pod Scheduling
Once maintenance has been completed, you can use the uncordon command to re-enable Pod scheduling onto the node:

kubectl uncordon node01

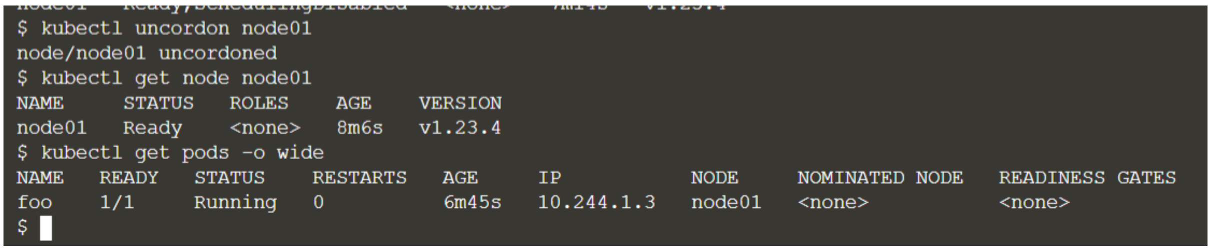
The "Ready" status of the node indicates that scheduling has been enabled again. Run the following command to verify:

kubectl get node node01

There is no undrain command; Pods will naturally get scheduled onto the empty node as they are created. For something quick that affects a node (e.g., a machine reboot), it is generally unnecessary to cordon or drain. It's only necessary if the machine will be out of service long enough that you want the Pods to move to a different machine.

The Pod named foo that was waiting in the "Pending" status is now scheduled and should transition into the "Running" status after a couple of seconds. You can check on the status of the Pod with the following command:

kubectl get pods -o wide



```
$ kubectl uncordon node01
node/node01 uncordoned
$ kubectl get node node01
NAME     STATUS   ROLES    AGE    VERSION
node01   Ready    <none>   8m6s   v1.23.4
$ kubectl get pods -o wide
NAME   READY   STATUS    RESTARTS   AGE     IP           NODE     NOMINATED NODE   READINESS GATES
foo    1/1     Running   0          6m45s   10.244.1.3   node01   <none>           <none>
$
```

The kubectl tool can be used to perform maintenance tasks on the node. You learned that the cordon command will prevent scheduling of new workload on a node. The drain command removes existing, running workloads on a node. Executing the cordon and drain commands will bring the node into a status safe for maintenance. Once maintenance tasks have been performed, you can re-enable workload scheduling on the node with the uncordon command.