Section 6:- Cluster Maintenance.

137 Cluster maintenance - Section Introduction

138 Download presentation Deck

139 OS upgrades

what does "OS Upgrade" mean in kubernetes

⇒ Kubernetes nodes (the machines that run your containers) are just linux servers underneath.

⇒ An OS upgrade means upgrading the OS version on these nodes

Think :-

* Ubuntu 20.04 → 22.04
* CentOS 7 → 8 (or)
* upgrading the linux kernel for security patches.

why do we need OS upgrade

1) security fix vulnerabilities (CVE patches)
2) Stability Newer OS version = fewer bugs
3) Compatibility support for newer K8S versions or container runtimes.

what part of K8S is affected?

component	Impact of OS upgrade
Worker Nodes	Runs your pods - upgrades must be done carefully
Control Plane	Hosts kube-apiserver, etcd, scheduler - critical!
Pod	Evicted during upgrade (unless handled gracefully)

Safely Upgrading OS in K8S

Let's say you are upgrading a worker node

1. Cordoning the Node

prevent new pods from being scheduled there:

```
kubectl cordon <node-name>
```

now the node is marked as unschedulable

2. Drain the node

Move all the running pods to other nodes

```
kubectl drain <node-name> --ignore-daemonsets
                          --delete-emptydir-data
```

this safely evicts non-critical pods from the node

3. Perform the OS upgrade

SSH into the node, and perform your upgrade (eg for ubuntu)

```
sudo apt update && sudo apt upgrade -y
sudo reboot
```

or for kernel upgrade

```
sudo apt install linux-generic
sudo reboot
```

let the node restart & come back online

4) uncordon the node (make it schedulable again)

```
kubectl uncordon <node-name>
```

Control plane Node upgrade

=> upgrade one node at a time

=> Always backup etcd first

=> Same steps cordon → drain → upgrade → uncordon

Common Pitfalls

Mistake	Risks
skipping Cordon	new pods might get scheduled mid-upgrade
skipping drain	Risk of data loss or crashed pods
upgrading all nodes at once	Leads to cluster downtime

Summary

⇒ OS upgrades are essential but must be done Carefully

⇒ Always Cordon and drain the node first

⇒ Require (reboot) your kernel updates to take effect

⇒ Be extra cautious with Control plane nodes

When do you take a Node down

For tasks like:

* Software upgrades * Security patches * Kernel or OS updates

What happens when a Node goes down

⇒ Pod running on the node become inaccessible

⇒ users may or may not be affected depending on how the pods were deployed.

Pod Type	Impact when node fails
Part of Replicaset	other replicas keep the app available
Standalone Pod	App is Completely unavailable

Pod Eviction Timeout

⇒ k8s waits 5mins by default bepore considering the node dead

⇒ This timeout is controlled by
--pod-eviction-timeout
(set in the kube-controller-manager)

Node Recovers In....	Result
<5 mins	pods come back as-is
less than 5mins	
>5mins	old pods are deleted, new ones created (if part of a ReplicaSet)

Safer way to perform Node Maintenance

1. Drain the node 2. Cordon the node 3. perform maintenance (OS update, reboot) 4. uncordon the node (schedulable again)

what happens after reboot

⇒ The node comes up **empty**

⇒ It says unschedulable until you run uncordon

⇒ old pods are not moved back - they stay where they were recreated