Cluster upgrade

It's important to keep k8s components up-to-date with the latest stable version to ensure that the cluster is secure and stable. Here are the several methods for upgrading a k8s cluster:

Kubeadm: Kubeadm is a popular tool for bootstrapping and managing Kubernetes clusters, particularly for self-provisioned clusters. Kubeadm provides commands like `kubeadm upgrade plan` and `kubeadm upgrade apply` to systematically upgrade the control plane and worker nodes. It simplifies the process of upgrading kubeadm-provisioned clusters.

Kubernetes Tools: Various Kubernetes deployment tools such as Kops, Kubespray, Rancher, and others provide their own mechanisms for cluster upgrades. These tools typically offer automation and specific commands for upgrading the cluster. For example, Kops provides the `kops upgrade cluster` and `kops rolling-update` commands to handle the upgrade process.

Cloud Provider Upgrades: Managed Kubernetes services offered by cloud providers, such as Amazon EKS, Azure AKS, and Google GKE, often handle control plane upgrades transparently. The cloud provider automatically manages the upgrade process, including the control plane components. As a user, you only need to update the node machine images to the desired version.

Blue-Green Deployment: The blue-green deployment approach involves creating a parallel "green" cluster with the desired version while the existing "blue" cluster is still running. Once the green cluster is ready, you switch traffic over to it, ensuring minimal downtime. After verifying the green cluster's stability, you can delete the old blue cluster. This method allows for a smooth transition and rollback option if any issues arise.

Kubernetes does support the last three minor versions for 9 months and provides patches for security and bug fixes during that time

V 1.22 — V 1.23 — V 1.24 — V 1.25 — V 1.26

un-Supported — Supported — Latest

when updating Kubernetes it is generally recommended to update only one minor version at a time, Minor version updates are meant to be backwards compatible. So going from 1.x to 1.x+1 should work smoothly

For production Kubernetes clusters, the general recommendation is to stay within 1 minor version of the latest stable Kubernetes release.

The maximum amount of difference that can exist between k8s components

- Control plane components: 0 versions (identical)
- kubelet/kubectl: Up to 2 minor versions behind
- etcd: Up to 1 minor version behind API server

Kubernetes releases its versions based on semantic versioning

v 1.25.3

MAJOR MINOR PATCH

Features Functionalities Bug fixes

How to Upgrade Kubernetes Cluster Using Kubeadm?

Step 1: Prepare for the Upgrade

Before upgrading, it is important to review the release notes and documentation for the target Kubernetes version. Check for any specific requirements or considerations.

https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/

recommended to perform upgrades on a test cluster before upgrading a production cluster to ensure that the process goes smoothly and without any issues

Back up any critical data and configurations, including etcd data, you maybe need to roll back the upgrade.

```
ETCDCTL_API=3 etcdctl snapshot save snapshot.db \
  --endpoints=$ENDPOINTS \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key
```

Step1: Determine which version to upgrade to

My current version is 1.25.3 and we will be upgrading it to one higher version, ie, 1.26.7

```
# Find the latest 1.26 version in the list.
# It should look like 1.26.x-00, where x is the latest patch
```

Step 2: Upgrade Control Plane Nodes

Upgrade the control plane components (API server, controller manager, and scheduler) and etcd (if applicable) on each control plane node one by on

Typically, this involves running a series of commands with `kubeadm` to upgrade the control plane components.

C:Analyzes the current state of the cluster and generates a plan for upgrading the control plane components to a newer version of Kubernetes.

A. Drain the control plane node

```
kubectl drain <control-plane-node-name> --ignore-daemonsets
```

B.Upgrade kubeadm "Skip this step if you want to update to the latest patch in this minor version."

```
sudo apt-mark unhold kubeadm
sudo apt-get upgrade -y kubeadm=1.26.7-00
sudo apt-mark hold kubeadm
```

The `apt-mark` command in the operating system can be used to label packages and update only the operating system without changing their version.

C. Plan the upgrade

```
sudo kubeadm upgrade plan
```

Upgrade to the latest version in the v1. series:

COMPONENT	CURRENT	TARGET
kube-apiserver	v1.25.3	v1.26.7
kube-controller-manager	v1.25.3	v1.26.7
kube-scheduler	v1.25.3	v1.26.7
kube-proxy	v1.25.3	v1.26.7
CoreDNS	v1.9.1	v1.9.3
etcd	3.5.4-0	3.5.6-0

You can now apply the upgrade by executing the following command:
kubeadm upgrade apply v1.26.7

D. Perform the upgrade

```
sudo kubeadm upgrade apply v1.26.7
```

E.Upgrade kubelet and kubectl

```
apt-mark unhold kubelet kubectl
sudo apt-get upgrade -y kubelet=1.26.7-00
sudo apt-get upgrade -y kubectl=1.26.7-00
sudo apt-mark hold kubelet kubectl
```

F. Restart kubelet and Uncordon the node

```
sudo systemctl daemon-reload
sudo systemctl restart kubelet
kubectl uncordon <control-plane-node-name>
```

Step3: Upgrade Worker Nodes

Upgrade the worker nodes one by one. This can be done by draining and cordoning each node, upgrading the necessary components, and then uncordoning the node.

The upgrade process for worker nodes typically involves upgrading the kubelet, kube-proxy, and any other relevant components.

A. Drain the control plane node

```
kubectl drain <worker-node-name> --ignore-daemonsets
```

B. Upgrade kubeadm, kubelet

```
apt-mark unhold kubeadm && \
apt-get update && apt-get install -y kubeadm=1.26.7-00 && \
apt-mark hold kubeadm
```

C.Upgrade the k8s configuration

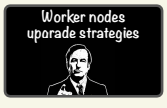
```
sudo kubeadm upgrade node
```

D.Upgrade kubelet and kubectl

```
apt-mark unhold kubelet kubectl && \
apt-get update && \
apt-get install -y kubelet=1.27.x-00 && \
apt-get install -y kubectl=1.27.x-00 && \
apt-mark hold kubelet kubectl
```

E. Restart kubelet and Uncordon the node

```
sudo systemctl daemon-reload
sudo systemctl restart kubelet
kubectl uncordon <worker-node-name>
```



- "All at once": In this strategy, all worker nodes are upgraded at the same time. This approach can be faster than other strategies, but it also carries the highest risk of causing downtime if something goes wrong during the upgrade
- "+1/-1": This strategy involves upgrading one worker node at a time, starting with adding a new node with the updated Kubernetes version, followed by removing an old node with the old version. This process is repeated until all worker nodes have been upgraded. This strategy minimizes the risk of downtime while still allowing for a relatively quick upgrade.
- "-1/+1": This strategy is similar to "+1/-1", but it involves removing an old node first and then adding a new node with the updated Kubernetes version. This strategy carries a slightly higher risk of downtime because there may be fewer worker nodes available during the upgrade process, which could result in an overload on the remaining nodes and potentially cause them to become not ready

Step 4: Verify Cluster Health

After upgrading all the control plane and worker nodes, you should verify the health of the cluster.

Check the status of the control plane components using commands like `kubectl get nodes` and `kubectl get pods -n kube-system`.

Step 5: Update Kubernetes Objects

Some Kubernetes objects, such as Deployments or StatefulSets, may need to be updated to take advantage of new features or changes in the upgraded version