Cluster upgrade process

⇒ We will keep dependency on external components, like ETCD and CoreDNS, aside for now & focus more on the control plane components

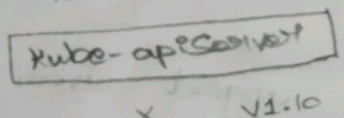
⇒ The components can be at different release versions.

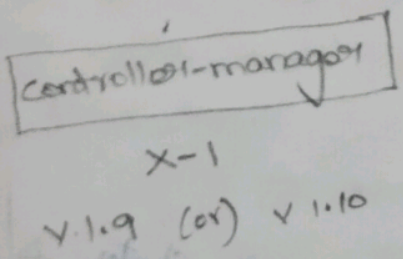
Since the kube-apiserver is the primary component in the control plane, & that is the component that all other components talk to

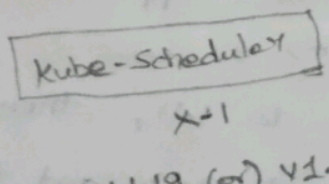
None of the component should ever be at a higher version than the kube-apiversion.

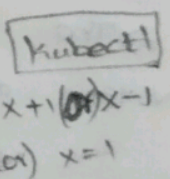
⇒ The control-manager and scheduler should be equal or less than one version lower than kube-apiserver.

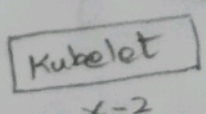
⇒ If kube-apiserver was at 1.10, then the controller-manager and scheduler should be at 1.10 (or) 1.9 then kubelet and kubeproxy should be equal or less than controller-manager and kube-scheduler

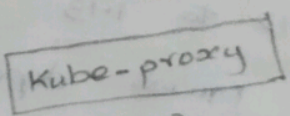












=> None of them could be at a version higher than the kube-apiserver like 1.11

=> This is not the case with kubectl

The kubectl utility could be at

* A version higher than the API server
* Same version as the API server
* Version lower than the API server

=> This permissible skew in versions allows us to carry out live upgrades. We can upgrade component by component if required

When we should upgrade

=> K8S supports only upto the recent 3 minor versions

=> So with 1.12 being the latest release, K8S supports version 1.12, 1.11, & 1.10. So, when 1.13 is released only versions 1.13, 1.12, 1.11 are supported

=> Before the release of 1.13 would be a good time to upgrade your cluster to the next release.

How do we upgrade

=> we cannot directly upgrade from 1.10 to 1.13

=> The recommended approach is to upgrade one minor version at a time,

version 1.10 to 1.11, then

version 1.11 to 1.12 and then

1.12 to 1.13

The upgrade process depends on how your cluster is set up.

Fast apt's

⇒ GKE → Simple clicks

⇒ If you deployed your cluster using kubeadm, then the tool can help you plan & upgrade the cluster

⇒ If you deployed your cluster from scratch, then you manually upgrade the different components of the cluster yourself.

We are approaching "kubeadm" method

So you have a cluster with master and worker nodes, running in production, hosting pods, serving users

The nodes and components are at version v1.10

Upgrading a cluster involves two major steps

First
* Upgrade your master nodes & then
* upgrade the worker nodes

⇒ While the master is being upgraded, the control plane components such as API server, scheduler and control manager go down briefly.

⇒ The master going down doesn't mean your worker nodes & applications on the cluster are impacted.

⇒ All workloads hosted on the worker nodes continue to serve users as normal.

⇒ Since the master is down, all management functions are down

⇒ You cannot access the cluster using kubectl or the k8s API.

⇒ You cannot deploy new applications (or) delete (or) modify existing ones.

The control manager don't function either, If a pod was to fail, a new pod won't be automatically created

But as long as the nodes and the pods are up, your applications should be up & the users are not impacted. once upgrade is complete & the cluster is will be up & running

now the master & master component is at v1.11 & the worker node is at v1.10

To upgrade worker nodes

There are different strategies.

Strategy 1:

To upgrade all of the worker node at once

But then your pods will be down, & users are no longer able to access the application

once upgrade is complete, the nodes are back up & running, new pods are scheduled & user can resume access

This strategy requires down time

Strategy 2:

Upgrade one node at a time

we first upgrade our first node, where the workloads move to the second and thead node & users are served from these

Once the first node is upgraded, back, up & running, we then upgrade the 2nd node, where the workload moves to the first and third node & finally third node until we have all nodes upgraded to a newer version. we follow the same procedure to upgrade the nodes

Strategy 3:-

"To add new nodes to the cluster [nodes with newer k8s version]".

This is convenient if you're in cloud env. where you can easily provision new nodes & decommission old ones.

Nodes with the newer k8s version can be added to the cluster, move the workload over to the new, & remove the old node until you finally have all new nodes with new k8s version.

To upgrade:-

To upgrade the cluster, kubeadm has an upgrade command which gives alot of information

Remember's

⇒ Kubeadm does not install or upgrade kubelets

⇒ It gives you the command to upgrade the cluster

⇒ we have to upgrade the kubeadm tool itself before you can upgrade the cluster

⇒ The kubeadm tool also follows the same k8s versioning as k8s

K8s cluster upgrade (kubeadm-based) step by step nodes.

General Facts

⇒ kubeadm is responsible for planning & upgrading the control plane.

⇒ kubeadm does NOT upgrade the kubelet - you must do that manually.

⇒ you can only upgrade one minor version at a time

Eg:- To go from 1.11 to 1.13

first upgrade to 1.12 → then to 1.13

Upgrade Sequence

Control Plane Node Upgrade

1. Upgrade kubeadm tool to next version

```
sudo apt-get install -y kubeadm=1.12.0-00
```

2. check upgrade plan:

```
kubeadm upgrade plan
```

3. Apply upgrade

```
kubeadm upgrade apply v1.12.0
```

* This pulls images and upgrades:
  * kube-apiserver
  * controller-manager
  * scheduler
  * etcd (if managed by kubeadm)

4) check node version

```
kubectl get nodes
```

This still shows the kubelet version, not the new API server version

5) Manually upgrade kubelet and restart service

```
sudo apt-get install -y kubelet=1.12.0-00
sudo systemctl restart kubelet
```

6) re-check node version

```
kubectl get nodes
```

should now show 1.12 for control plane node.

Worker Node Upgrade (one at a Time)

1. Drain the node

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

safely evicts pods & make node unschedulable

2) upgrade kubeadm on the worker

```
sudo apt-get install -y kubeadm=1.12.0-00
```

3) upgrade the node configuration

```
sudo kubeadm upgrade node
```

4) upgrade kubelet and restart

```
sudo apt-get install -y kubelet=1.12.0-00
sudo systemctl restart kubelet
```

5) Uncordon the node

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

makes the node schedulable again

Summary:-

⇒ To check the current version of the cluster, do

kubectl get nodes [look for version now]

2) How many nodes can host workloads in this cluster?

currently only control plane & nodes is in cluster

kubectl describe node ~~control-plane~~ ~~node01~~

look for taints

kubectl describe node | grep Taints

Taints: <none>

Taints: <none>

which means both the nodes can host applications

3) How many applications are hosted on the cluster

`kubectl get deploy`

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
blue	5/5	5	5	95s

only one application (blue)

4) what nodes the pods hosted on

`kubectl get pods -o wide` look for "node" section

5) what is the latest stable version available for upgrade use the `kubeadm` tool

The question is not to find out this latest stable version of k8s, but what the latest stable version that `kubeadm` tool knows as eg now.

`kubeadm upgrade plan`

around 9th line we see the kubeadm version.

[Upgrade/versions] Kubeadm version: v1.19.0

This is the kubeadm version

This kubeadm version v1.19.0 can upgrade the cluster to v1.19.16 which you can see here

components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':

COMPONENT	CURRENT	AVAILABLE
Kubelet	2x v1.19.0	v1.19.16

If you want to upgrade your cluster to v1.20, then you have to upgrade the kubeadm to the same version which is v1.20 first