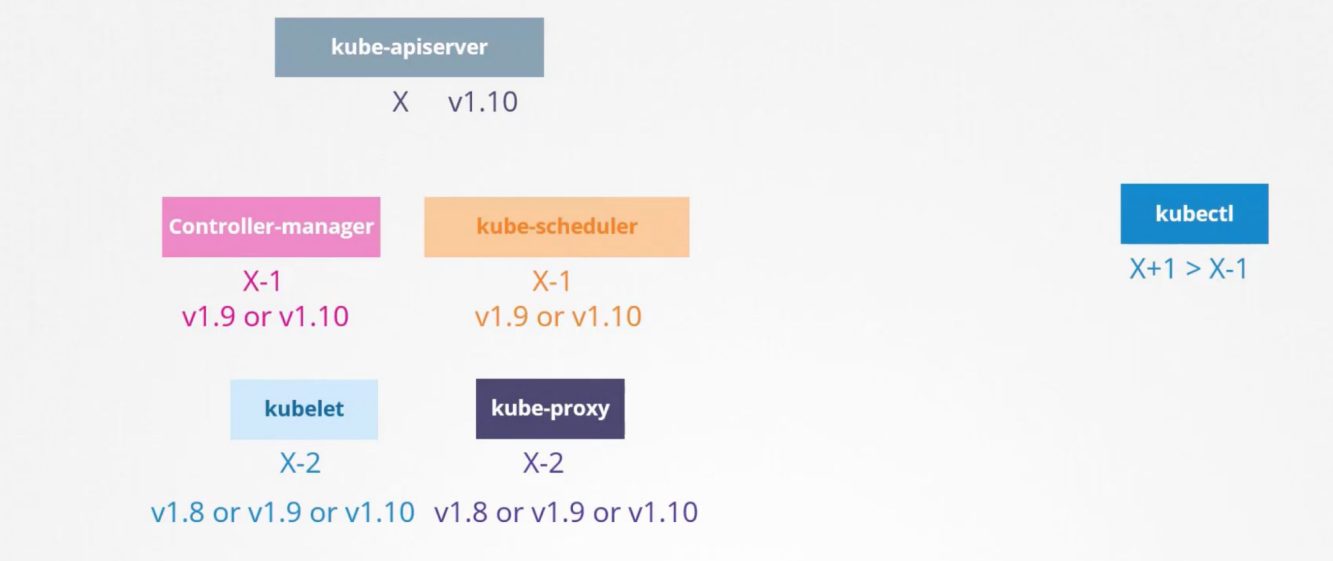
Upgrading k8s cluster

Table of Contents

- Kubernetes components, skew versioning and cluster version
- General rules about k8s cluster upgrading
- Why we should upgrade k8s cluster?
- How to upgrade k8s cluster?

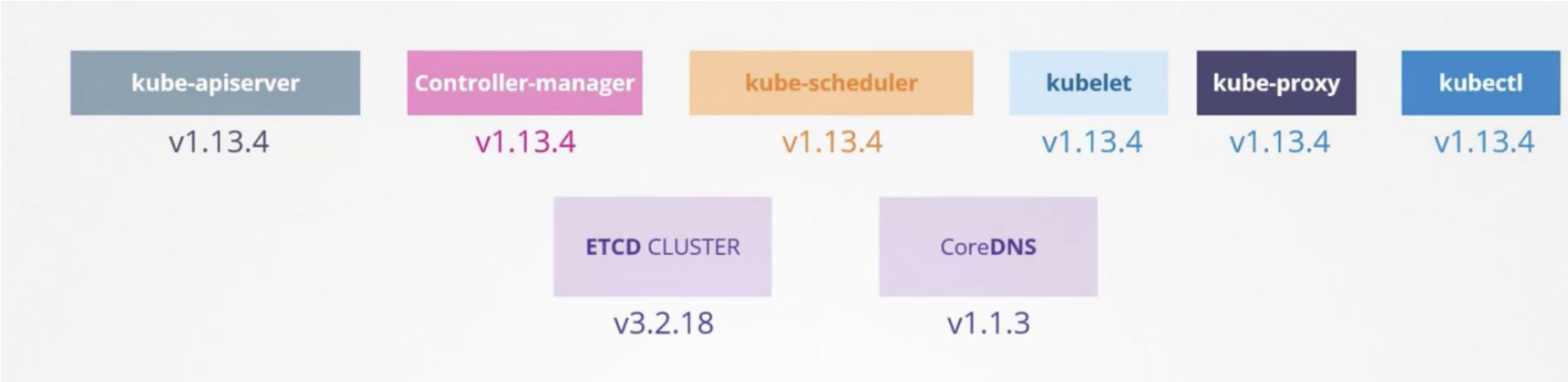
Kubernetes components, skew versioning and cluster version

- First of all we should know the rules among the Kubernetes components and skew versioning policy.



Kubernetes components, skew versioning and cluster version

- Second, ETCD and CoreDNS versions might be different from the other k8s components. It is not mandatory to compatible with the other k8s components.



Kubernetes components, skew versioning and cluster version

- When we enter the “kubectl get nodes” command in a Kubernetes cluster, we can see the versions of the master node and worker nodes.

```
ubuntu@kube-master:~$ kubectl get nodes
NAME          STATUS   ROLES           AGE   VERSION
kube-master   Ready    control-plane   90m   v1.31.0
kube-worker   Ready    <none>          90m   v1.31.0
```

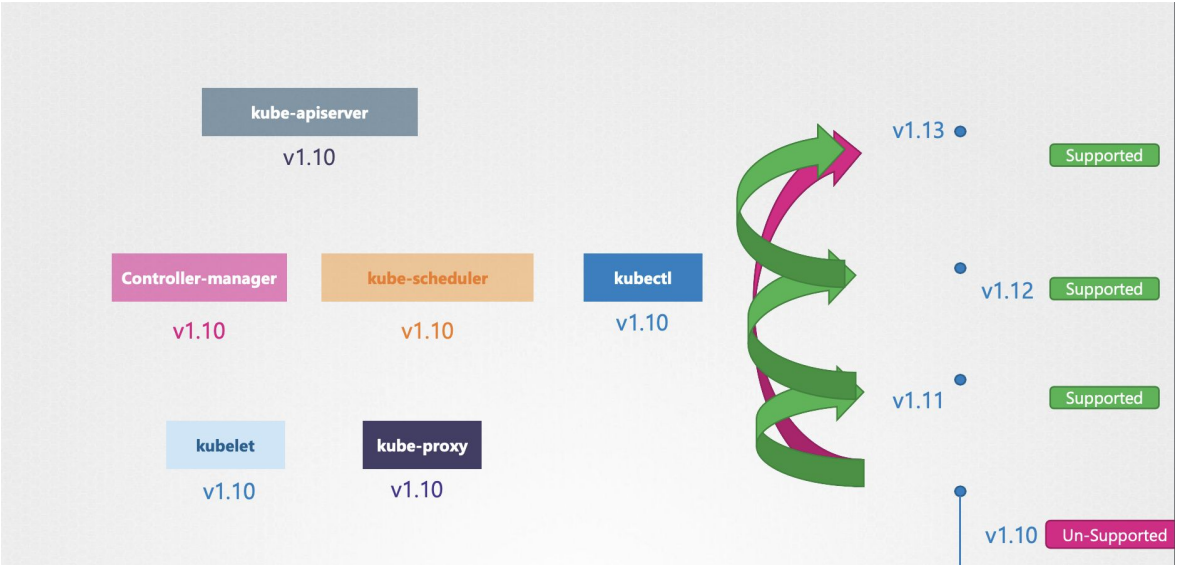
- These versions we see are the versions of our k8s cluster.

General rules about k8s cluster upgrading

- In a k8s cluster upgrade process, the **first rule** is to **upgrade the master node.**
- While the master node is being upgraded, the **master node** is **disabled for a while**. This does not mean that the pods on the worker nodes, namely your application, will stop, only operations directed at the master node cannot be performed.
- In other words, **api-server, kube-scheduler and controller manager** operations cannot be performed. In other words, the commands you make with kubectl will not work because the **api-server** is disabled during this time.

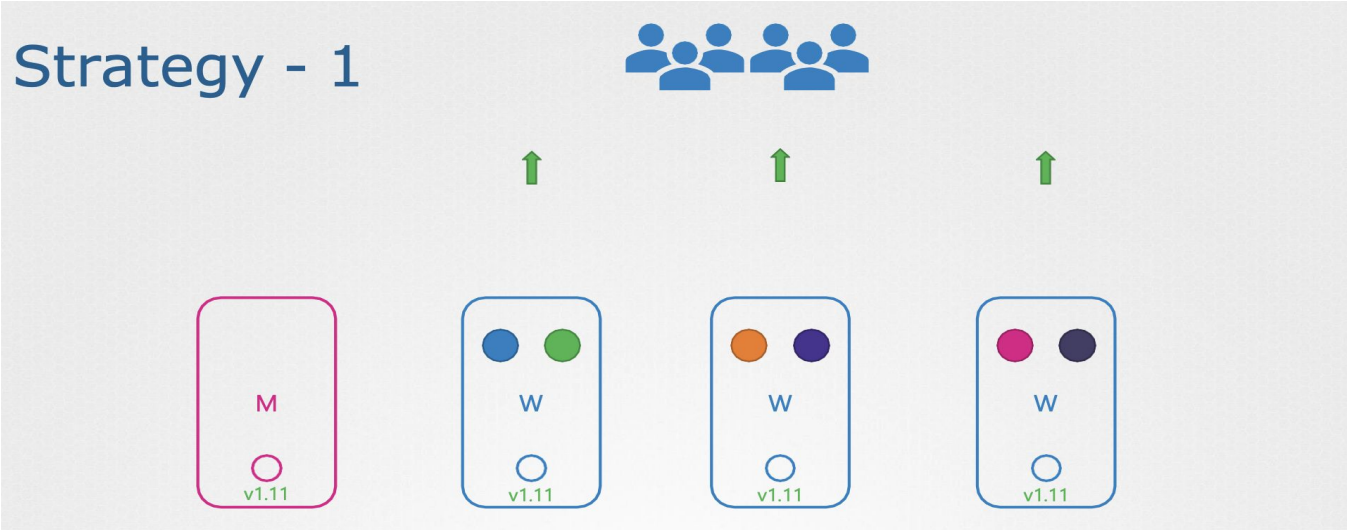
General rules about k8s cluster upgrading

- After the upgrade of the **master node** is completed, the **worker nodes** are upgraded.
- When upgrading versions, for example if we want to upgrade from 1.10 to 1.13, it is recommended to first upgrade to version 1.11, then to 1.12 and finally to 1.13.



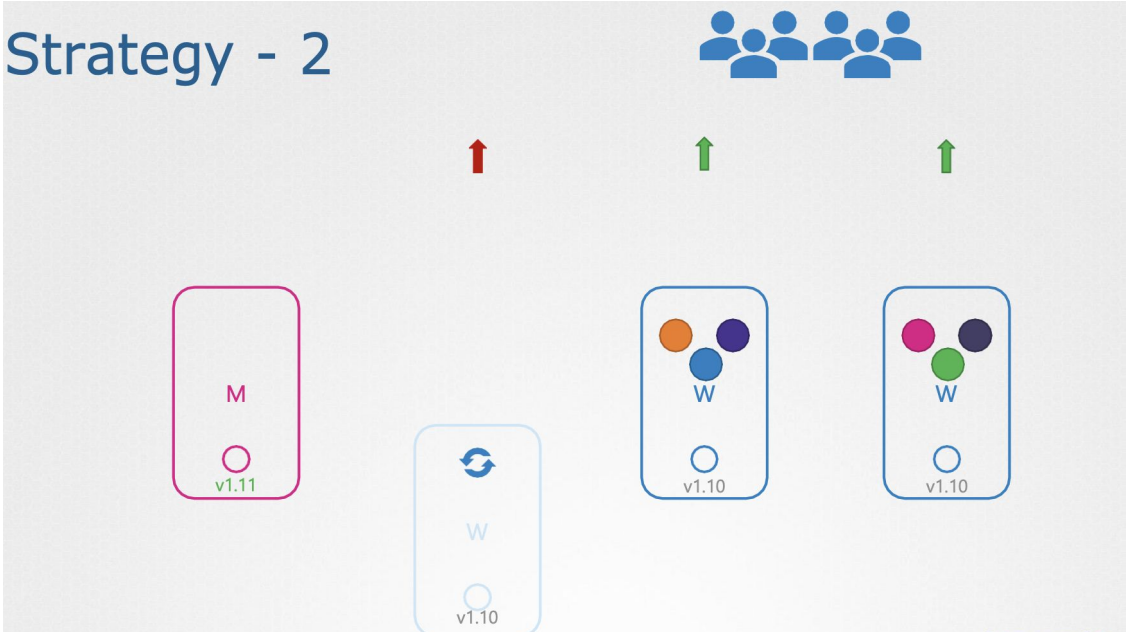
General rules about k8s cluster upgrading

- There are 3 strategy that can be used when upgrading worker nodes.
- Upgrading all three nodes at the same time is not preferred, as when this is done, the pods running on the worker nodes will stop during the upgrade and the application will not be accessible for a certain period of time.



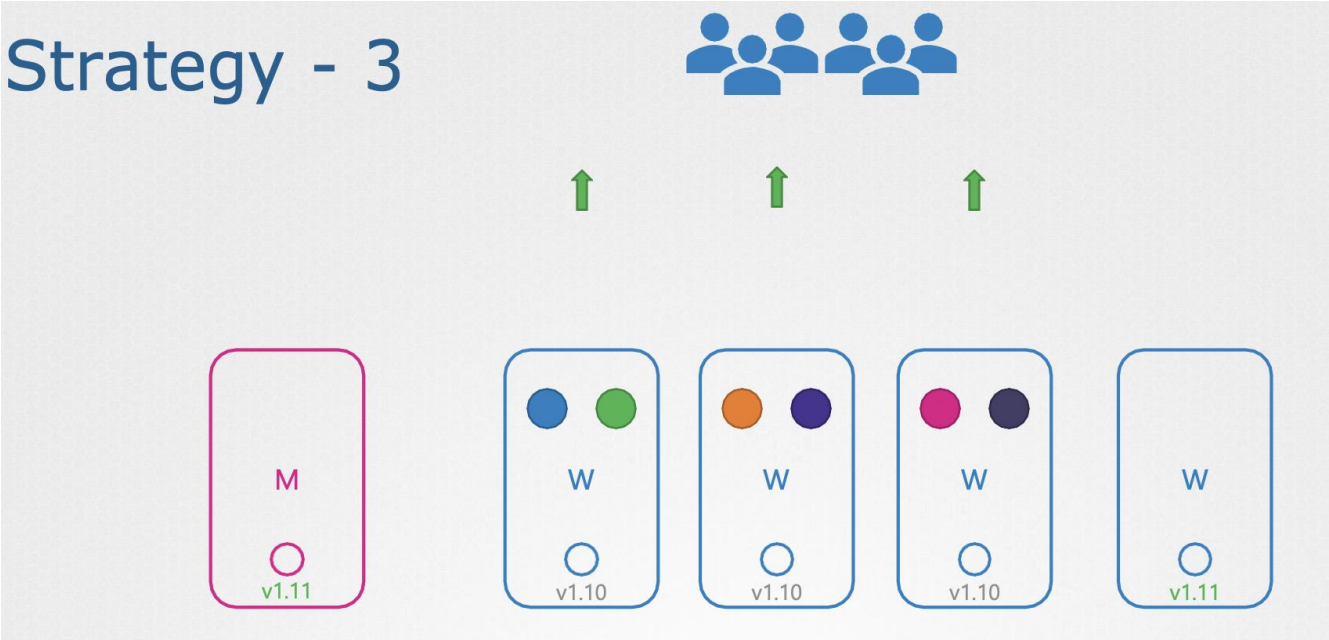
General rules about k8s cluster upgrading

- The second strategy is to upgrade worker nodes one by one, which is the logical thing to do, so the application always stays up.



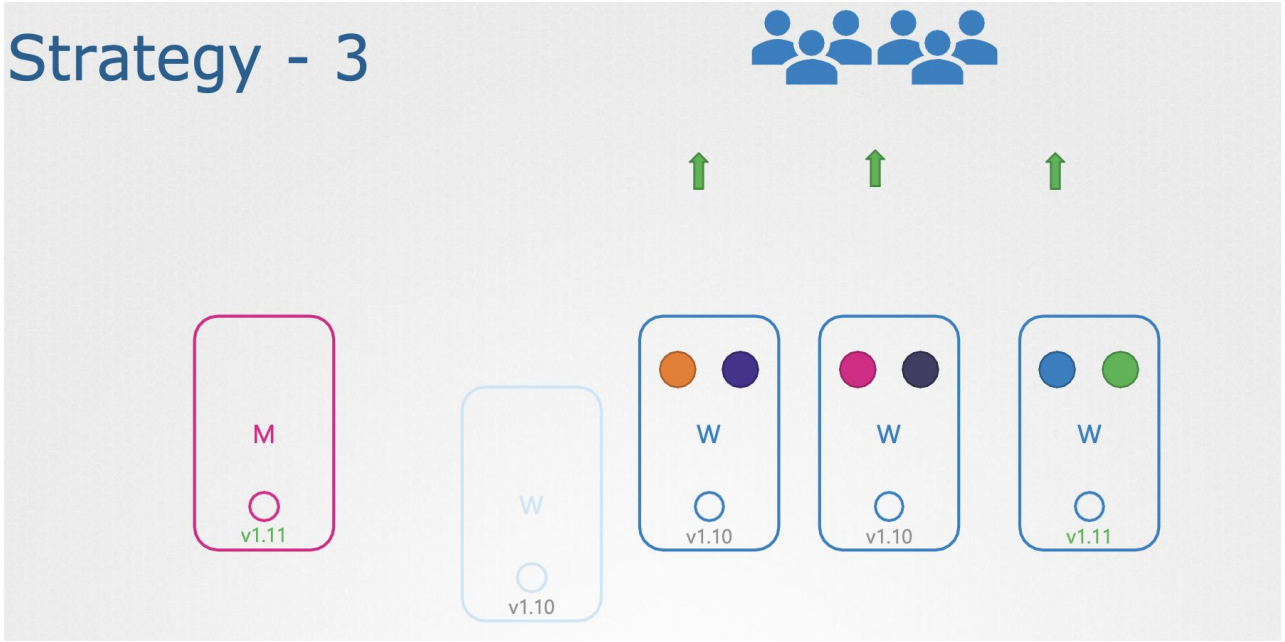
General rules about k8s cluster upgrading

- The third strategy is to add a new worker node to the cluster in the version to be upgraded and then disable the worker nodes of older versions.



General rules about k8s cluster upgrading

- The third strategy is to add a new worker node to the cluster in the version to be upgraded and then disable the worker nodes of older versions.



General rules about k8s cluster upgrading

- The first thing we **need to do is to upgrade kubeadm**.

- First of all, we can see the k8s cluster version and the k8s cluster versions that can be installed by running the command "**sudo kubeadm upgrade plan**". This command also provides useful information about the components of our k8s cluster and their versions.

- After upgrading our k8s cluster, we finally **need to manually set the "kubelet"** version on each node.

General rules about k8s cluster upgrading

- When performing the upgrade process, we need to **know the operating system** on which the k8s cluster is running and use the appropriate commands. (apt-get for an Ubuntu operating system, yum for a Centos operating system, etc.)

- While upgrading the master node and worker node, we **need to drain the nodes**. After the upgrade process is completed, we need to perform the "**uncordon**" process and make the nodes "**schedulable**" again.