RX-M Cloud Native Consulting

# Kubernetes

## Lab – Upgrading a Kubernetes cluster with kubeadm upgrade

The Kubernetes maintainers keep to a one-per-quarter (every three months) release cycle for minor versions, with each branch receiving maintenance for 9 months after release.

`kubeadm upgrade` is a user-friendly command that performs all of the necessary tasks needed to upgrade or downgrade a Kubernetes cluster to another minor release version in one command. It has subcommands for both planning an upgrade and actually performing it.

### Objectives

- Understand the upgrade planning process
- Learn about the version compatibility between kubernetes control plane components
- Upgrade a kubeadm cluster to a newer a minor version

### 1. Prepare a new Kubernetes Cluster with Kubeadm

Get the version of your Kubernetes cluster with `kubectl version`:

```
ubuntu@ip-172-31-12-52:~$ kubectl version

Client Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.4",
GitCommit:"224be7bdce5a9dd0c2fd0d46b83865648e2fe0ba", GitTreeState:"clean", BuildDate:"2019-12-
11T12:47:40Z", GoVersion:"go1.12.12", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.4",
GitCommit:"224be7bdce5a9dd0c2fd0d46b83865648e2fe0ba", GitTreeState:"clean", BuildDate:"2019-12-
11T12:37:43Z", GoVersion:"go1.12.12", Compiler:"gc", Platform:"linux/amd64"}

ubuntu@ip-172-31-12-52:~$
```

The Kubernetes client version indicates the Kubeadm version being used to invoke the command, while the Server version reports the current control plane's version as reported by the kubelet.

Check what version is being reported to the Kubernetes api server by your node's kubelet:

```
ubuntu@ip-172-31-12-52:~$ kubectl get nodes

NAME              STATUS   ROLES    AGE     VERSION
ip-172-31-12-52   Ready    master   27h     v1.16.4

ubuntu@ip-172-31-12-52:~$
```

You can check the version of a nodes kubelet by running the `kubelet` command. Since the kubelet was installed as a systemd daemon, you will need to use sudo:

```
ubuntu@ip-172-31-12-52:~$ sudo kubelet --version

Kubernetes v1.16.4

ubuntu@ip-172-31-12-52:~$
```

### 2. Selecting the version to upgrade to

Before executing a cluster upgrade, there are several things to take into account.

The biggest consideration for picking a new version is whether the control plane components are compatible with the kube-apiserver version you intend to upgrade to. This policy is known as the version skew policy:

| Components | Can be older than kube-apiserver | Can be newer than kube-apiserver |
|---|---|---|
| Other kube-apiservers in a HA cluster | Up to 1 minor version | No |
| kubelet | Up to 2 minor versions | No |
| kube-controller-manager, kube-scheduler, cloud-controller-manager | Up to 1 minor version | No |
| kubectl | Up to 1 minor version | Up to 1 minor version |

So if you are running a Kubernetes cluster at version 1.17, the compatibility matrix would look like this:

| Components | Compatible versions |
|---|---|
| Other kube-apiservers | 1.16, 1.17 |
| kubelet | 1.15, 1.16, 1.17 |
| kube-controller-manager, kube-scheduler, cloud-controller-manager | 1.16, 1.17 |

If there are multiple instances of kube-apiserver in a highly available and load-balanced cluster, then the compatibility should be restricted to the oldest instance of kube-apiserver that is running in that HA cluster.

## 3. Acquire the new version binaries

Before you can upgrade your Kubernetes cluster, you must retrieve the binaries for the version of Kubernetes you intend to upgrade to.

If you followed the Ubuntu installation instructions from Kubernetes (either from a previous lab in this course or the Kubernetes documentation), you will simply need to update the Kubernetes apt repository and use `apt install` to apply the desired versions of the Kubernetes components for the upgrade.

Start by updating the apt repository on your VM:

```
ubuntu@ip-172-31-12-52:~$ sudo apt update

Hit:1 http://us-west-1.ec2.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://us-west-1.ec2.archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:3 http://us-west-1.ec2.archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Hit:5 https://download.docker.com/linux/ubuntu xenial InRelease
Get:6 http://us-west-1.ec2.archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [1,098
kB]
Get:7 http://us-west-1.ec2.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [781
kB]
Hit:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Get:8 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
Fetched 2,204 kB in 0s (2,752 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
41 packages can be upgraded. Run 'apt list --upgradable' to see them.

ubuntu@ip-172-31-12-52:~$
```

Now check the available packages:

```
ubuntu@ip-172-31-12-52:~$ apt-cache policy kubeadm | head -20
```

```
kubeadm:
  Installed: 1.17.2-00
  Candidate: 1.17.2-00
  Version table:
 *** 1.17.2-00 500
        500 http://apt.kubernetes.io kubernetes-xenial/main amd64 Packages
        100 /var/lib/dpkg/status
     1.17.1-00 500
        500 http://apt.kubernetes.io kubernetes-xenial/main amd64 Packages
     1.17.0-00 500
        500 http://apt.kubernetes.io kubernetes-xenial/main amd64 Packages
     1.16.6-00 500
        500 http://apt.kubernetes.io kubernetes-xenial/main amd64 Packages
     1.16.5-00 500
        500 http://apt.kubernetes.io kubernetes-xenial/main amd64 Packages
     1.16.4-00 500
        500 http://apt.kubernetes.io kubernetes-xenial/main amd64 Packages
     1.16.3-00 500
        500 http://apt.kubernetes.io kubernetes-xenial/main amd64 Packages
     1.16.2-00 500

ubuntu@ip-172-31-12-52:~$
```

You should presented with the opportunity to update to the latest version of kubeadm. The only condition to selecting an upgrade version is that it is within one minor release of the version you are on.

This means you can go from version 1.16 to 1.17, but you will not be able to go to 1.15 directly to 1.17. In order to upgrade to 1.17 from 1.15, you will need to upgrade to 1.16 first.

Pick the kubeadm version at the top of the apt-cache list and install it:

```
ubuntu@ip-172-31-12-52:~$ sudo apt install kubeadm=1.17.2-00

Reading package lists... Done
Building dependency tree
Reading state information... Done
kubeadm is already the newest version (1.17.2-00).
0 upgraded, 0 newly installed, 0 to remove and 41 not upgraded.

ubuntu@ip-172-31-12-52:~$
```

Check the version of kubeadm:

```
ubuntu@ip-172-31-12-52:~$ kubeadm version

kubeadm version: &version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2",
GitCommit:"59603c6e503c87169aea6106f57b9f242f64df89", GitTreeState:"clean", BuildDate:"2020-01-
18T23:27:49Z", GoVersion:"go1.13.5", Compiler:"gc", Platform:"linux/amd64"}

ubuntu@ip-172-31-12-52:~$
```

Your kubeadm is now ready to perform the latest update.

## 4. Using kubeadm upgrade

The `upgrade` subcommand for `kubectl` is what you will be using to upgrade your Kubernetes cluster.

In your terminal, show the help for it using the `--help` flag:

```
ubuntu@ip-172-31-12-52:~$ kubeadm upgrade --help

Upgrade your cluster smoothly to a newer version with this command

Usage:
```

```
  kubeadm upgrade [flags]
  kubeadm upgrade [command]

Available Commands:
  apply       Upgrade your Kubernetes cluster to the specified version
  diff        Show what differences would be applied to existing static pod manifests. See also:
kubeadm upgrade apply --dry-run
  node        Upgrade commands for a node in the cluster
  plan        Check which versions are available to upgrade to and validate whether your current
cluster is upgradeable. To skip the internet check, pass in the optional [version] parameter

Flags:
  -h, --help   help for upgrade

Global Flags:
      --add-dir-header           If true, adds the file directory to the header
      --log-file string          If non-empty, use this log file
      --log-file-max-size uint   Defines the maximum size a log file can grow to. Unit is
megabytes. If the value is 0, the maximum file size is unlimited. (default 1800)
      --rootfs string            [EXPERIMENTAL] The path to the 'real' host root filesystem.
      --skip-headers             If true, avoid header prefixes in the log messages
      --skip-log-headers         If true, avoid headers when opening log files
  -v, --v Level                  number for the log level verbosity

Use "kubeadm upgrade [command] --help" for more information about a command.

ubuntu@ip-172-31-12-52:~$
```

Of note here is the `plan` subcommand for `kubectl upgrade`, which will essentially perform a dry-run of the preflight checks and inform you whether the cluster is upgradable or not.

Run `kubeadm upgrade plan` with `sudo` and see if your cluster is ready for an upgrade:

```
ubuntu@ip-172-31-12-52:~$ sudo kubeadm upgrade plan

[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm
kubeadm-config -oyaml'
[preflight] Running pre-flight checks.
[upgrade] Making sure the cluster is healthy:
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: v1.16.4
[upgrade/versions] kubeadm version: v1.17.2
[upgrade/versions] Latest stable version: v1.17.2
[upgrade/versions] Latest version in the v1.16 series: v1.16.6

Components that must be upgraded manually after you have upgraded the control plane with
'kubeadm upgrade apply':
COMPONENT   CURRENT       AVAILABLE
Kubelet     v1.16.4       v1.16.6

Upgrade to the latest version in the v1.16 series:

COMPONENT            CURRENT   AVAILABLE
API Server           v1.16.4   v1.16.6
Controller Manager   v1.16.4   v1.16.6
Scheduler            v1.16.4   v1.16.6
Kube Proxy           v1.16.4   v1.16.6
CoreDNS              1.6.2     1.6.5
Etcd                 3.3.15    3.3.17-0

You can now apply the upgrade by executing the following command:

        kubeadm upgrade apply v1.16.6

_____________________________________________________________________

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

```
COMPONENT   CURRENT       AVAILABLE
Kubelet     v1.16.4       v1.17.2

Upgrade to the latest stable version:

COMPONENT            CURRENT   AVAILABLE
API Server           v1.16.4   v1.17.2
Controller Manager   v1.16.4   v1.17.2
Scheduler            v1.16.4   v1.17.2
Kube Proxy           v1.16.4   v1.17.2
CoreDNS              1.6.2     1.6.5
Etcd                 3.3.15    3.4.3-0

You can now apply the upgrade by executing the following command:

        kubeadm upgrade apply v1.17.2

_____________________________________________________________________

ubuntu@ip-172-31-12-52:~$
```

In this run, `kubeadm upgrade plan` has provided two options:

- Upgrade to the latest patch version available for Kubernetes 1.17
- Upgrade to the next minor version, Kubernetes 1.16.6

These recommendations will differ based on what versions of Kubernetes are available. You are also advised as to which components of the cluster need to be updated manually after `kubeadm upgrade`.

## 5. Upgrade the cluster

After deciding which version to upgrade the cluster to, use one of the suggested `kubeadm upgrade apply` commands given by `kubectl upgrade plan` above.

`kubeadm upgrade apply` will perform the following:

- Checks that the cluster's API server can be reached
- Ensures that all nodes are in the `Ready` state
- The control plane components are reporting `Healthy` status
- Ensures that the new versions of the control plane components abide by the version skew policy
- Makes sure the control plane images can be pulled for the update
- Updates control plane pods and restarts them (or rolls them back if any of them fails to come up).
- Applies the new `kube-dns` and `kube-proxy` manifests and makes sure that all necessary RBAC rules are created.
- Creates new certificate and key files of the API server and backs up old files (if they're about to expire in 180 days).

In this case, we will upgrade to the latest patch version of the next minor version, 1.17.2 (as of this writing):

```
ubuntu@ip-172-31-12-52:~$ sudo kubeadm upgrade apply v1.17.2

[upgrade/config] Making sure the configuration is correct:
[upgrade/config] Reading configuration from the cluster...
[upgrade/config] FYI: You can look at this config file with 'kubectl -n kube-system get cm
kubeadm-config -oyaml'
[preflight] Running pre-flight checks.
[upgrade] Making sure the cluster is healthy:
[upgrade/version] You have chosen to change the cluster version to "v1.17.2"
[upgrade/versions] Cluster version: v1.16.4
[upgrade/versions] kubeadm version: v1.17.2
[upgrade/confirm] Are you sure you want to proceed with the upgrade? [y/N]: y
[upgrade/prepull] Will prepull images for components [kube-apiserver kube-controller-manager
kube-scheduler etcd]
[upgrade/prepull] Prepulling image for component etcd.
[upgrade/prepull] Prepulling image for component kube-apiserver.
[upgrade/prepull] Prepulling image for component kube-controller-manager.
[upgrade/prepull] Prepulling image for component kube-scheduler.
[apiclient] Found 0 Pods for label selector k8s-app=upgrade-prepull-kube-scheduler
[apiclient] Found 0 Pods for label selector k8s-app=upgrade-prepull-kube-apiserver
```

```
[apiclient] Found 0 Pods for label selector k8s-app=upgrade-prepull-kube-controller-manager
[apiclient] Found 0 Pods for label selector k8s-app=upgrade-prepull-etcd
[upgrade/prepull] Prepulled image for component kube-controller-manager.
[upgrade/prepull] Prepulled image for component kube-scheduler.
[upgrade/prepull] Prepulled image for component kube-apiserver.
[upgrade/prepull] Prepulled image for component etcd.
[upgrade/prepull] Successfully prepulled the images for all the control plane components
[upgrade/apply] Upgrading your Static Pod-hosted control plane to version "v1.17.2"...
Static pod: kube-apiserver-ip-172-31-12-52 hash: 90ad6b5d56b73ab30795c42ab840ff43
Static pod: kube-controller-manager-ip-172-31-12-52 hash: 53f43bf67d0189bb12b56ab4fc9688dd
Static pod: kube-scheduler-ip-172-31-12-52 hash: 732be3f14f79b5c85c2b9fc7df90d045
[upgrade/etcd] Upgrading to TLS for etcd
Static pod: etcd-ip-172-31-12-52 hash: 189d44b91d9a1d8a4c6c7c191cce7068
[upgrade/staticpods] Preparing for "etcd" upgrade
[upgrade/staticpods] Renewing etcd-server certificate
[upgrade/staticpods] Renewing etcd-peer certificate
[upgrade/staticpods] Renewing etcd-healthcheck-client certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/etcd.yaml" and backed up
old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2020-01-28-23-50-57/etcd.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap
(timeout 5m0s)
Static pod: etcd-ip-172-31-12-52 hash: 189d44b91d9a1d8a4c6c7c191cce7068
Static pod: etcd-ip-172-31-12-52 hash: 9f96a97c28b22a52682df578f8fa86f7
[apiclient] Found 1 Pods for label selector component=etcd
[upgrade/staticpods] Component "etcd" upgraded successfully!
[upgrade/etcd] Waiting for etcd to become available
[upgrade/staticpods] Writing new Static Pod manifests to "/etc/kubernetes/tmp/kubeadm-upgraded-
manifests379666121"
W0128 23:51:10.616130   14688 manifests.go:214] the default kube-apiserver authorization-mode is
"Node,RBAC"; using "Node,RBAC"
[upgrade/staticpods] Preparing for "kube-apiserver" upgrade
[upgrade/staticpods] Renewing apiserver certificate
[upgrade/staticpods] Renewing apiserver-kubelet-client certificate
[upgrade/staticpods] Renewing front-proxy-client certificate
[upgrade/staticpods] Renewing apiserver-etcd-client certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/kube-apiserver.yaml" and
backed up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2020-01-28-23-50-
57/kube-apiserver.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap
(timeout 5m0s)
Static pod: kube-apiserver-ip-172-31-12-52 hash: 90ad6b5d56b73ab30795c42ab840ff43
Static pod: kube-apiserver-ip-172-31-12-52 hash: e55cbe5c860ab710f0e2e167a5cc2159
[apiclient] Found 1 Pods for label selector component=kube-apiserver
[upgrade/staticpods] Component "kube-apiserver" upgraded successfully!
[upgrade/staticpods] Preparing for "kube-controller-manager" upgrade
[upgrade/staticpods] Renewing controller-manager.conf certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/kube-controller-
manager.yaml" and backed up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2020-
01-28-23-50-57/kube-controller-manager.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap
(timeout 5m0s)
Static pod: kube-controller-manager-ip-172-31-12-52 hash: 53f43bf67d0189bb12b56ab4fc9688dd
Static pod: kube-controller-manager-ip-172-31-12-52 hash: 175bf3c4053a4cb76ff8178e108b65d8
[apiclient] Found 1 Pods for label selector component=kube-controller-manager
[upgrade/staticpods] Component "kube-controller-manager" upgraded successfully!
[upgrade/staticpods] Preparing for "kube-scheduler" upgrade
[upgrade/staticpods] Renewing scheduler.conf certificate
[upgrade/staticpods] Moved new manifest to "/etc/kubernetes/manifests/kube-scheduler.yaml" and
backed up old manifest to "/etc/kubernetes/tmp/kubeadm-backup-manifests-2020-01-28-23-50-
57/kube-scheduler.yaml"
[upgrade/staticpods] Waiting for the kubelet to restart the component
[upgrade/staticpods] This might take a minute or longer depending on the component/version gap
(timeout 5m0s)
Static pod: kube-scheduler-ip-172-31-12-52 hash: 732be3f14f79b5c85c2b9fc7df90d045
Static pod: kube-scheduler-ip-172-31-12-52 hash: 9c994ea62a2d8d6f1bb7498f10aa6fcf
[apiclient] Found 1 Pods for label selector component=kube-scheduler
[upgrade/staticpods] Component "kube-scheduler" upgraded successfully!
[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-
system" Namespace
```

```
[kubelet] Creating a ConfigMap "kubelet-config-1.17" in namespace kube-system with the
configuration for the kubelets in the cluster
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.17"
ConfigMap in the kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for
nodes to get long term certificate credentials
[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically
approve CSRs from a Node Bootstrap Token
[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client
certificates in the cluster
[addons]: Migrating CoreDNS Corefile
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.17.2". Enjoy!

[upgrade/kubelet] Now that your control plane is upgraded, please proceed with upgrading your
kubelets if you haven't already done so.

ubuntu@ip-172-31-12-52:~$
```

After the operation finishes, check the version of the cluster with kubectl:

```
ubuntu@ip-172-31-12-52:~$ kubectl version

Client Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.4",
GitCommit:"224be7bdce5a9dd0c2fd0d46b83865648e2fe0ba", GitTreeState:"clean", BuildDate:"2019-12-
11T12:47:40Z", GoVersion:"go1.12.12", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2",
GitCommit:"59603c6e503c87169aea6106f57b9f242f64df89", GitTreeState:"clean", BuildDate:"2020-01-
18T23:22:30Z", GoVersion:"go1.13.5", Compiler:"gc", Platform:"linux/amd64"}

ubuntu@ip-172-31-12-52:~$
```

Our server version refers to the api-server version. We see that it is now running at version 1.17.2.

Check the control plane pods and see if they have been updated; they should be seconds or minutes old:

```
ubuntu@ip-172-31-12-52:~$ kubectl get pods -n kube-system

NAME                                    READY   STATUS    RESTARTS   AGE
coredns-5644d7b6d9-jp2lb                1/1     Running   0          27h
coredns-5644d7b6d9-q5wzg                1/1     Running   0          27h
etcd-ip-172-31-12-52                    1/1     Running   0          40s
kube-apiserver-ip-172-31-12-52          1/1     Running   0          38s
kube-controller-manager-ip-172-31-12-52   1/1     Running   0          23s
kube-proxy-mzqrp                        1/1     Running   0          27h
kube-scheduler-ip-172-31-12-52          1/1     Running   0          18s
weave-net-9pntv                         2/2     Running   0          27h

ubuntu@ip-172-31-12-52:~$
```

The coredns pods, and many of the primary control plane pods have been recreated and etcd has restarted, which is a good sign. It looks like weave-net, the CNI that was used to power the networking for this cluster, was not restarted.

Now check the node and see if the kubelet has been upgraded:

```
ubuntu@ip-172-31-12-52:~$ kubectl get nodes

NAME              STATUS   ROLES    AGE     VERSION
ip-172-31-12-52   Ready    master   27h     v1.16.4

ubuntu@ip-172-31-12-52:~$ sudo kubelet --version

Kubernetes v1.16.4
```

```
ubuntu@ip-172-31-12-52:~$
```

Looks like the kubelet is still running on the old version. This shouldn't be a problem right now, since it is still within one minor version of the kube-apiserver. Still, you will want to upgrade it to ensure future compatibility.

## 6. Finishing the update

At this point, the server and most of the control plane components have been upgraded, leaving the following components:

- The CNI provider plugin (weave net)
- The node's kubelet
- The Kubernetes Client (aka kubectl)

Start with the kubelet and kubectl. Retrieve the kubelet and kubectl binaries that correspond to our cluster from apt:

```
ubuntu@ip-172-31-12-52:~$ sudo apt install kubelet=1.17.2-00 kubectl=1.17.2-00

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  kubectl kubelet
2 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
Need to get 27.9 MB of archives.
After this operation, 14.8 MB disk space will be freed.
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.17.2-00
[8,738 kB]
Get:2 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet amd64 1.17.2-00
[19.2 MB]
Fetched 27.9 MB in 2s (9,747 kB/s)
(Reading database ... 76879 files and directories currently installed.)
Preparing to unpack .../kubectl_1.17.2-00_amd64.deb ...
Unpacking kubectl (1.17.2-00) over (1.16.4-00) ...
Preparing to unpack .../kubelet_1.17.2-00_amd64.deb ...
Unpacking kubelet (1.17.2-00) over (1.16.4-00) ...
Setting up kubectl (1.17.2-00) ...
Setting up kubelet (1.17.2-00) ...

ubuntu@ip-172-31-12-52:~$
```

When upgrading, it is a good idea to declare what versions of the components you want to update to avoid any surprises during the upgrade process.

When the update completes, check the versions:

```
ubuntu@ip-172-31-12-52:~$ kubectl version

Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2",
GitCommit:"59603c6e503c87169aea6106f57b9f242f64df89", GitTreeState:"clean", BuildDate:"2020-01-
18T23:30:10Z", GoVersion:"go1.13.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2",
GitCommit:"59603c6e503c87169aea6106f57b9f242f64df89", GitTreeState:"clean", BuildDate:"2020-01-
18T23:22:30Z", GoVersion:"go1.13.5", Compiler:"gc", Platform:"linux/amd64"}

ubuntu@ip-172-31-12-52:~$ kubectl get nodes

NAME              STATUS   ROLES    AGE     VERSION
ip-172-31-12-52   Ready    master   27h     v1.17.2

ubuntu@ip-172-31-12-52:~$
```

Great. Our kubectl client version now matches our API server. We also see that the kubelet now matches the API server version.

> Using apt to update the kubelet will restart it, but you may want to run `sudo systemctl restart kubelet` to

> manually ensure that the kubelet has been properly updated

Next, we'll check the CNI plugin. Depending on when the cluster was bootstrapped, there may have been some changes between Kubernetes versions. It will be ideal to update the CNI configuration with the latest versions.

Use `kubectl apply` to update the CNI plugin:

```
ubuntu@ip-172-31-12-52:~$ kubectl apply -f \
"https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"

serviceaccount/weave-net configured
clusterrole.rbac.authorization.k8s.io/weave-net configured
clusterrolebinding.rbac.authorization.k8s.io/weave-net configured
role.rbac.authorization.k8s.io/weave-net configured
rolebinding.rbac.authorization.k8s.io/weave-net configured
daemonset.apps/weave-net configured

ubuntu@ip-172-31-12-52:~$
```

If a new version of the CNI images or resources have been published, they should be updated. If the cluster was bootstrapped recently, there may not be any updates.

Check and see if the network CNI pod has updated:

```
ubuntu@ip-172-31-12-52:~$ kubectl get pods -n kube-system

NAME                                      READY   STATUS    RESTARTS   AGE
coredns-6955765f44-phxcz                  1/1     Running   0          60s
coredns-6955765f44-rpwqd                  1/1     Running   0          60s
etcd-ip-172-31-12-52                      1/1     Running   0          103s
kube-apiserver-ip-172-31-12-52            1/1     Running   0          101s
kube-controller-manager-ip-172-31-12-52   1/1     Running   0          86s
kube-proxy-6gc7q                          1/1     Running   0          50s
kube-scheduler-ip-172-31-12-52            1/1     Running   0          81s
weave-net-9pntv                           2/2     Running   0          27h

ubuntu@ip-172-31-12-52:~$
```

In this example, there don't appear to be any updates to the weave-net pods, but our kube-proxy pod is new.

The control plane node has been successfully upgraded from version 1.16.4 to 1.17.2!

Congratulations, you have completed the lab!