Upgrading the Cluster Manually: Changing the Kubernetes Version

In this lesson, we will upgrade the cluster by changing the version of Kubernetes manually.

WE'LL COVER THE FOLLOWING

Upgrading the Kubernetes Version

The process to upgrade the cluster depends on what we want to do.

If we'd like to upgrade it to a specific Kubernetes version, we can execute a similar process like the one we used to add a new worker node.

```
kops edit cluster $NAME
```

Just as before, we are about to edit the cluster definition. The only difference is that this time we're not editing a specific instance group, but the cluster as a whole.

If you explore the YAML file in front of you, you'll see that it contains the information we specified when we created the cluster, combined with the kops default values that we omitted to set.

For now, we're interested in `kubernetesVersion`. Please find it and change the version from `v1.9.1` to `v1.9.2`. After changing the version save and exit.

Previewing the Update

Now that we modified the desired state of the cluster, we can proceed with `kops update`.

```
kops update cluster $NAME
```

The last line of the output indicates that we `must specify --yes to apply changes`.

Unlike the previous time we executed `kops update`, now we did not specify the argument `--yes`. As a result, we got a preview, or a dry-run, of what would happen if we apply the change.

Previously, we added a new worker node, and that operation did not affect the existing servers. There was no issue in updating the cluster without previewing which resources will be created, updated, or destroyed.

This time we are upgrading the servers in the cluster. Existing nodes will be replaced with new ones, and that is a potentially dangerous operation. Later on, we might trust kops to do what's right and skip the preview altogether. But, for now, we should evaluate what will happen if we proceed.

Please go through the output. You'll see a git-like diff of the changes that will be applied to some of the resources that form the cluster.

Applying the Update

Now that you are confident with the changes, we can apply them.

```
kops update cluster $NAME --yes
```

The last line of the output states that `changes may require instances to restart: kops rolling-update cluster`. We already saw that message before but, this time, the update was not performed. The reason is simple, even though not necessarily very intuitive.

We can update auto-scaling groups since that results in creation or destruction of nodes. But, when we need to replace them, as in this case, it would be disastrous to execute a simple update.

Updating everything at once would produce a downtime. In our case, it's even worse. Destroying all the masters at once would likely result in a loss of

quorum. A new cluster might not be able to recuperate.

Rolling Update

kops requires an extra step when the updating of the cluster might result in undesirable results. So, we need to execute the `kops rolling-update` command. Since we're still insecure, we'll run a preview first.

```
kops rolling-update cluster $NAME
```

The **output** is as follows.

```
NAME                STATUS      NEEDUPDATE READY MIN MAX NODES
master-us-east-2a NeedsUpdate 1          0     1   1   1
master-us-east-2b NeedsUpdate 1          0     1   1   1
master-us-east-2c NeedsUpdate 1          0     1   1   1
nodes               NeedsUpdate 2          0     2   2   2

Must specify --yes to rolling-update.
```

We can see that all the nodes require an update. Since we already evaluated the changes through the output of the `kops update` command, we'll proceed and apply rolling updates.

```
kops rolling-update cluster $NAME --yes
```

The rolling update process started, and it will take approximately 30 minutes to complete.

We'll explore the output as it comes.

In the next lesson, we will explore the output, i.e., the effect of rolling the update.