CKA Curriculum Part 10 - Installation, Configuration & Validation

To really get the grips with the inner workings of Kubernetes clusters, the gold standard is Kelsey Hightower's "Kubernetes the hard way"

https://github.com/kelseyhightower/kubernetes-the-hard-way

Which will cover the following from the curriculum:

- Design a Kubernetes cluster.
- Install Kubernetes masters and nodes.
- Configure secure cluster communications.
- Configure a Highly available Kubernetes cluster.
- Know where to get the Kubernetes release binaries.
- Provision underlying infrastructure to deploy a Kubernetes cluster.
- Choose a network solution.
- Choose your Kubernetes infrastructure configuration.
- Run end-to-end tests on your cluster.
- Analyse end-to-end tests results.
- Run Node end-to-end tests

Install and use kubeadm to install, configure and manage Kubernetes clusters

kubeadm is a utility to bootstrap Kubernetes to a number of existing, vanilla nodes. It takes care of the etcd cluster, Kubernetes master and worker nodes including all the required components to instantiate a viable minimum k8s cluster.

What you get at the end of using kubeadm is a fully working, fully functioning kubernetes cluster.

It's at the opposite end of the spectrum in terms of difficulty compared to doing things "the hard way" as per Kelsey Hightower's guide above.

For the exam, it is recommended that you become familiar with both ways of deploying Kubernetes clusters.

Kubeadm is a command line utility that performs the following functions:

- **kubeadm init** to bootstrap a Kubernetes control-plane node
- **kubeadm join** to bootstrap a Kubernetes worker node and join it to the cluster
- **kubeadm upgrade** to upgrade a Kubernetes cluster to a newer version
- **kubeadm config** if you initialized your cluster using kubeadm v1.7.x or lower, to configure your cluster for kubeadm upgrade
- **kubeadm token** to manage tokens for kubeadm join
- **kubeadm reset** to revert any changes made to this host by kubeadm init or kubeadm join
- **kubeadm version** to print the kubeadm version
- **kubeadm alpha** to preview a set of features made available for gathering feedback from the community

Kubeadm - Master Node Install

In the following examples 3x Ubuntu Server VM's were created

- k8s-cl02-ms01
- k8s-cl02-wk01
- k8s-cl02-wk02

Where appropiate, ensure your nodes have a container runtime installed.

On the master node install the required binaries

```
apt-get update && apt-get install -y apt-transport-https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpghttps://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/https://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

```
apt-get update
apt-get install -y kubelet kubeadm kubectl
apt-mark hold kubelet kubeadm kubectl
```

Initialise a the master node:

```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
```

Note, the requirement to pass --pod-network is dependent on the chosen CNI. For Flannel, this is required. Kubeadm will also let you know if any prerequistes are not made.

Once completed, a message will be displayed:

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.16.10.80:6443 --token j5nqhd.cnfmnjgc68aato60 \
    --discovery-token-ca-cert-hash sha256:cbc91031c1ffa47bbea83aa1cf65e99821a1f582c4363e1a4408715bfd66bb60
```

Some important pieces of information to note:

- Kubeadm has created the admin kubeconfig file for you, and recommends copying this to the logged on users home directory for ease
- Kubeadm has **not** deploed a pod networking solution yet. Therefore this is a post-install activity
- Kubeadm has provided a join command together with a token to add worker nodes. We can regenerate this token if required.

If we issue a kubectl get nodes command we will see the master node is not ready

```
NAME            STATUS     ROLES    AGE     VERSION
k8s-cl02-ms01   NotReady   master   6m20s   v1.14.3
```

As per the output of kubeadm, install a network solution, Ie flannel

```
For flannel to work correctly, you must pass --pod-network-cidr=10.244.0.0/16 to kubeadm init.

Set /proc/sys/net/bridge/bridge-nf-call-iptables to 1 by running sysctl net.bridge.bridge-nf-call-iptables=1
```

And install Flannel

```
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/62e44c867a2846fefb68bd5f178daf4da3095ccb/Documentation/kube-flannel.ymlhttps://raw.githubusercontent.com/coreos/flannel/62e44c867a2846fefb68bd5f178daf4da3095ccb/Documentation/kube-flannel.yml
```

After a few seconds, the master node will now be ready

```
NAME            STATUS   ROLES    AGE   VERSION
k8s-cl02-ms01   Ready    master   10m   v1.14.3
```

Kubeadm - Install worker nodes

The install process is similar to the master node - the only exception is we do **not** execute the "kubeadm init" command, as this is only run on masters. For workers we use "kubeadm join"

- Install a container runtime
- Install the kubeadm binaries (as above)

To join a worker node to a cluster created by kubeadm we need to use the kubeadm join command with a token generated on the master. This is shown after we run kubeadm init on the master node. However, we can easily regenerte this on the master node should it not be noted down or expired:

(on the master node)

```
david@k8s-cl02-ms01:~$ kubeadm token create --print-join-command
kubeadm join 172.16.10.80:6443 --token ht55yv.8lq69q0189xhe2ql     --discovery-token-ca-cert-hash sha256:cbc91031c1ffa47bbea83aa1cf65e99821a1f582c4363e1a4408715bfd66bb60
```

Use this command on the worker (as root)

```
root@k8s-cl02-wk01:~# kubeadm join 172.16.10.80:6443 --token ht55yv.8lq69q0189xhe2ql     --discovery-token-ca-cert-hash sha256:cbc91031c1ffa47bbea83aa1cf65e99821a1f582c4363e1a4408715bfd66bb60
```

After which confirmation will be displayed:

```
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

To validate, run kubectl get nodes on the master node:

```
NAME            STATUS   ROLES    AGE     VERSION
k8s-cl02-ms01   Ready    master   50m     v1.14.3
k8s-cl02-wk01   Ready    <none>   2m10s   v1.14.3
```

Upgrading a cluster using Kubeadm

This is covered in part 4