Linux Academy CKA Notes :

## Upgrading the Kubernetes Cluster :

----------------------------------------------------------------------------

Kubeadm allows us to upgrade our cluster components in the proper order, making sure to include important feature upgrades we might want to take advantage of in the latest stable version of Kubernertes. In this lesson, we will go through upgrading our cluster from version 1.13.5 to 1.14.1.

Get the version of the API server:

kubectl version --short

View the version of kubelet:

kubectl describe nodes

View the version of controller-manager pod:

kubectl get po [controller\_pod\_name] -o yaml -n kube-system

Release the hold on versions of kubeadm and kubelet:

sudo apt-mark unhold kubeadm kubelet

Install version 1.14.1 of kubeadm:

sudo apt install -y kubeadm=1.14.1-00

Hold the version of kubeadm at 1.14.1:

sudo apt-mark hold kubeadm

Verify the version of kubeadm:

kubeadm version

Plan the upgrade of all the controller components:

sudo kubeadm upgrade plan

Upgrade the controller components:

sudo kubeadm upgrade apply v1.14.1

Release the hold on the version of kubectl:

sudo apt-mark unhold kubectl

Upgrade kubectl:

sudo apt install -y kubectl=1.14.1-00

Hold the version of kubectl at 1.14.1:

sudo apt-mark hold kubectl

Upgrade the version of kubelet:

sudo apt install -y kubelet=1.14.1-00

Hold the version of kubelet at 1.14.1:

sudo apt-mark hold kubelet

**Helpful Links**

* [Upgrading Kubernetes](https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm-upgrade/)
* [Changelog for v1.13](https://github.com/kubernetes/kubernetes/blob/master/CHANGELOG-1.13.md)

Simple Steps :

On master:

Unhold kubeadm,kubelet version

Upgrade kubeadm on Master node , hold the version upgrade the control plain components.

Unhold the kubectl version , Upgrade the kubectl , hold the new version again , upgrade the kubelet hold the version.

On Workers:

Unhold kubeadm,kubelet version

Upgrade kubeadm on worker node , hold the version

Unhold the kubectl version , Upgrade the kubectl , hold the new version again , upgrade the kubelet hold the version.

**Operating System Upgrades within a Kubernetes Cluster**

When we need to take a node down for maintenance, Kubernetes makes it easy to evict the pods on that node, take it down, and then continue scheduling pods after the maintenance is complete. Furthermore, if the node needs to be decommissioned, you can just as easily remove the node and replace it with a new one, joining it to the cluster.

See which pods are running on which nodes:

kubectl get pods -o wide

Evict the pods on a node:

kubectl drain [node\_name] --ignore-daemonsets

Watch as the node changes status:

kubectl get nodes -w

Schedule pods to the node after maintenance is complete:

kubectl uncordon [node\_name]

Remove a node from the cluster:

kubectl delete node [node\_name]

Generate a new token:

sudo kubeadm token generate

List the tokens:

sudo kubeadm token list

Print the kubeadm join command to join a node to the cluster:

sudo kubeadm token create [token\_name] --ttl 2h --print-join-command

**Helpful Links**

* [Maintenance on a Node](https://kubernetes.io/docs/tasks/administer-cluster/cluster-management/#maintenance-on-a-node)

## Backing Up and Restoring a Kubernetes Cluster :

Backing up your cluster can be a useful exercise, especially if you have a single etcd cluster, as all the cluster state is stored there. The etcdctl utility allows us to easily create a snapshot of our cluster state (etcd) and save this to an external location. In this lesson, we’ll go through creating the snapshot and talk about restoring in the event of failure.

Get the etcd binaries:

wget https://github.com/etcd-io/etcd/releases/download/v3.3.12/etcd-v3.3.12-linux-amd64.tar.gz

Unzip the compressed binaries:

tar xvf etcd-v3.3.12-linux-amd64.tar.gz

Move the files into /usr/local/bin:

sudo mv etcd-v3.3.12-linux-amd64/etcd\* /usr/local/bin

Take a snapshot of the etcd datastore using etcdctl:

sudo ETCDCTL\_API=3 etcdctl snapshot save snapshot.db --cacert /etc/kubernetes/pki/etcd/server.crt --cert /etc/kubernetes/pki/etcd/ca.crt --key /etc/kubernetes/pki/etcd/ca.key

View the help page for etcdctl:

ETCDCTL\_API=3 etcdctl --help

Browse to the folder that contains the certificate files:

cd /etc/kubernetes/pki/etcd/

View that the snapshot was successful:

ETCDCTL\_API=3 etcdctl --write-out=table snapshot status snapshot.db

Zip up the contents of the etcd directory:

sudo tar -zcvf etcd.tar.gz /etc/kubernetes/pki/etcd

Copy the etcd directory to another server:

scp etcd.tar.gz cloud\_user@18.219.235.42:~/

**Helpful Links**

* [Backing up the etcd Store](https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#backing-up-an-etcd-cluster)
* [etcd Disaster Recovery Examples](https://github.com/etcd-io/etcd/blob/master/Documentation/op-guide/recovery.md)

## Pod and Node Networking

Kubernetes keeps networking simple for effective communication between pods, even if they are located on a different node. In this lesson, we’ll talk about pod communication from within a node, including how to inspect the virtual interfaces, and then get into what happens when a pod wants to talk to another pod on a different node.

See which node our pod is on:

kubectl get pods -o wide

Log in to the node:

ssh [node\_name]

View the node's virtual network interfaces:

ifconfig

View the containers in the pod:

docker ps

Get the process ID for the container:

docker inspect --format '{{ .State.Pid }}' [container\_id]

Use nsenter to run a command in the process's network namespace:

nsenter -t [container\_pid] -n ip addr

**Helpful Links**

* [Cluster Networking](https://kubernetes.io/docs/concepts/cluster-administration/networking/)

## Container Network Interface (CNI)

A Container Network Interface (CNI) is an easy way to ease communication between containers in a cluster. The CNI has many responsibilities, including IP management, encapsulating packets, and mappings in userspace. In this lesson, we will cover the details of the Flannel CNI we used in our Linux Academy cluster and talk about the ways in which we simplified communication in our cluster.

Apply the Flannel CNI plugin:

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681ece4de4c0d86c5cd2643275/Documentation/kube-flannel.yml

**Helpful Links**

* [Flannel Documentation](https://github.com/coreos/flannel/blob/master/Documentation/kubernetes.md)
* [Installing Other CNI Plugins](https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/#pod-network)
* [Installing Addons in Kubernetes](https://kubernetes.io/docs/concepts/cluster-administration/addons/)

## Service Networking

Services allow our pods to move around, get deleted, and replicate, all without having to manually keep track of their IP addresses in the cluster. This is accomplished by creating one gateway to distribute packets evenly across all pods. In this lesson, we will see the differences between a NodePort service and a ClusterIP service and see how the iptables rules take effect when traffic is coming in.

YAML for the nginx NodePort service:

apiVersion: v1

kind: Service

metadata:

name: nginx-nodeport

spec:

type: NodePort

ports:

- protocol: TCP

port: 80

targetPort: 80

nodePort: 30080

selector:

app: nginx

Get the services YAML output for all the services in your cluster:

kubectl get services -o yaml

Try and ping the clusterIP service IP address:

ping 10.96.0.1

View the list of services in your cluster:

kubectl get services

View the list of endpoints in your cluster that get created with a service:

kubectl get endpoints

Look at the iptables rules for your services:

sudo iptables-save | grep KUBE | grep nginx

**Helpful Links**

* [Services in Kubernetes](https://kubernetes.io/docs/concepts/services-networking/service/)

## Ingress Rules and Load Balancers

When handling traffic from outside sources, there are two ways to direct that traffic to your pods: deploying a load balancer, and creating an ingress controller and an Ingress resource. In this lesson, we will talk about the benefits of each and how Kubernetes distributes traffic to the pods on a node to reduce latency and direct traffic to the appropriate services within your cluster.

View the list of services:

kubectl get services

The load balancer YAML spec:

apiVersion: v1

kind: Service

metadata:

name: nginx-loadbalancer

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 80

selector:

app: nginx

Create a new deployment:

kubectl run kubeserve2 --image=chadmcrowell/kubeserve2

View the list of deployments:

kubectl get deployments

Scale the deployments to 2 replicas:

kubectl scale deployment/kubeserve2 --replicas=2

View which pods are on which nodes:

kubectl get pods -o wide

Create a load balancer from a deployment:

kubectl expose deployment kubeserve2 --port 80 --target-port 8080 --type LoadBalancer

View the services in your cluster:

kubectl get services

Watch as an external port is created for a service:

kubectl get services -w

Look at the YAML for a service:

kubectl get services kubeserve2 -o yaml

Curl the external IP of the load balancer:

curl http://[external-ip]

View the annotation associated with a service:

kubectl describe services kubeserve

Set the annotation to route load balancer traffic local to the node:

kubectl annotate service kubeserve2 externalTrafficPolicy=Local

The YAML for an Ingress resource:

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: service-ingress

spec:

rules:

- host: kubeserve.example.com

http:

paths:

- backend:

serviceName: kubeserve2

servicePort: 80

- host: app.example.com

http:

paths:

- backend:

serviceName: nginx

servicePort: 80

- http:

paths:

- backend:

serviceName: httpd

servicePort: 80

Edit the ingress rules:

kubectl edit ingress

View the existing ingress rules:

kubectl describe ingress

Curl the hostname of your Ingress resource:

curl http://kubeserve2.example.com

**Helpful Links**

* [Create an External Load Balancer](https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/)
* [Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/)

## Cluster DNS

CoreDNS is now the new default DNS plugin for Kubernetes. In this lesson, we’ll go over the hostnames for pods and services. We will also discover how you can customize DNS to include your own nameservers.

View the CoreDNS pods in the kube-system namespace:

kubectl get pods -n kube-system

View the CoreDNS deployment in your Kubernetes cluster:

kubectl get deployments -n kube-system

View the service that performs load balancing for the DNS server:

kubectl get services -n kube-system

Spec for the busybox pod:

apiVersion: v1

kind: Pod

metadata:

name: busybox

namespace: default

spec:

containers:

- image: busybox:1.28.4

command:

- sleep

- "3600"

imagePullPolicy: IfNotPresent

name: busybox

restartPolicy: Always

View the resolv.conf file that contains the nameserver and search in DNS:

kubectl exec -it busybox -- cat /etc/resolv.conf

Look up the DNS name for the native Kubernetes service:

kubectl exec -it busybox -- nslookup kubernetes

Look up the DNS names of your pods:

kubectl exec -ti busybox -- nslookup [pod-ip-address].default.pod.cluster.local

Look up a service in your Kubernetes cluster:

kubectl exec -it busybox -- nslookup kube-dns.kube-system.svc.cluster.local

Get the logs of your CoreDNS pods:

kubectl logs [coredns-pod-name]

YAML spec for a headless service:

apiVersion: v1

kind: Service

metadata:

name: kube-headless

spec:

clusterIP: None

ports:

- port: 80

targetPort: 8080

selector:

app: kubserve2

YAML spec for a custom DNS pod:

apiVersion: v1

kind: Pod

metadata:

namespace: default

name: dns-example

spec:

containers:

- name: test

image: nginx

dnsPolicy: "None"

dnsConfig:

nameservers:

- 8.8.8.8

searches:

- ns1.svc.cluster.local

- my.dns.search.suffix

options:

- name: ndots

value: "2"

- name: edns0

**Helpful Links**

* [DNS for Services and Pods](https://kubernetes.io/docs/concepts/services-networking/dns-pod-service/)
* [Debugging DNS Resolution](https://kubernetes.io/docs/tasks/administer-cluster/dns-debugging-resolution/)
* [Customizing DNS](https://kubernetes.io/docs/tasks/administer-cluster/dns-custom-nameservers/)
* [CoreDNS GitHub](https://github.com/coredns/deployment/tree/master/kubernetes)
* [Kubernetes DNS-Based Service Discovery](https://github.com/kubernetes/dns/blob/master/docs/specification.md)
* [Deploying CoreDNS using kubeadm](https://coredns.io/2018/01/29/deploying-kubernetes-with-coredns-using-kubeadm/)