# Learning Kubernetes

Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure.

For more information see: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

This workshop provides a first introduction in Kubernetes. A set of examples are given which will provide hands on experience in using kubernetes.

## Getting information from the cluster

To do this workshop it easy to execute the commands as user root. In a production environment this is not adviced. Logon to your Ubuntu machine and become user root

sudo su -

You can retrieve information on the cluster by querying the kubernetes API.

You can query the API directly with curl but to make life easy we will use the program kubectl to talk to the api. You can for instance collect data about the cluster using:

# kubectl cluster-info  
Kubernetes master is running at https://127.0.0.1:6443  
CoreDNS is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy  
Metrics-server is running at https://127.0.0.1:6443/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy  
  
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

To find out at what version the cluster is running use:

# kubectl version  
Client Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2+k3s1", GitCommit:"cdab19b09a84389ffbf57bebd33871c60b1d6b28", GitTreeState:"clean", BuildDate:"2020-01-27T18:09:26Z", GoVersion:"go1.13.6", Compiler:"gc", Platform:"linux/amd64"}  
Server Version: version.Info{Major:"1", Minor:"17", GitVersion:"v1.17.2+k3s1", GitCommit:"cdab19b09a84389ffbf57bebd33871c60b1d6b28", GitTreeState:"clean", BuildDate:"2020-01-27T18:09:26Z", GoVersion:"go1.13.6", Compiler:"gc", Platform:"linux/amd64"}

To see the nodes the cluster contains use:

# kubectl get nodes  
NAME STATUS ROLES AGE VERSION  
localhost Ready master 3d v1.17.2+k3s1

In above example only one node is showed. This is a one node cluster, in a production cluster there are normaly more nodes showed. There can be a master role and a worker role. In this setup the master is used to schedule workload as well.

## namespaces

On the cluster different namespaces have been deployed by default. A namespace separates resources. You can view the namespaces deployed on this cluster using:

# kubectl get namespaces  
NAME STATUS AGE  
default Active 3d  
kube-system Active 3d  
kube-public Active 3d  
kube-node-lease Active 3d

If you do not specify the namespace, the default namespace is used to schedule a workload.

## creating your first pod

A pod is set of containers running together and sharing a network. The containers are deployed together on the same host within a certain namespace. A pod can be launched using the command line tool kubectl.

In this example you will deploy a pod with a single container using the image: kpnappfactory/testcontainer.

$ kubectl run example --image=nginx --port 80 --restart=Never  
pod "example" created

Once you have deployed the pod you should be able to see it using:

$ kubectl get pods

You can view the details of the pod using:

$ kubectl describe pod <podname>  
Name: example  
Namespace: platform  
Node: k8snode1/10.0.0.14  
Start Time: Tue, 13 Jun 2017 14:37:59 +0200  
Labels: <none>  
Status: Running  
IP: 10.47.0.14  
Controllers: <none>  
...

You should be able to ping the example pod from you shell. Use the IP address you just found:

ping <IP>  
64 bytes from 10.47.0.14: seq=1 ttl=64 time=0.540 ms  
64 bytes from 10.47.0.14: seq=2 ttl=64 time=1.187 ms  
^C  
--- 10.47.0.14 ping statistics ---  
2 packets transmitted, 2 packets received, 0% packet loss  
round-trip min/avg/max = 0.540/1.112/1.825 ms

The example pod is running a webserver you should be able to query. We should be able to query the webserver from the shell. For example by using a command line web browser (replace the IP address with the address you have found yourself):

curl http://<IP>

Note! This webserver is running on the Kubernetes overlay network whithin the cluster and is only accessible from inside the cluster.

## Using yaml

Simple resources can be created using the commandline only but for more complex resources using a yaml specification is a must. When you create a resource via kubectl as done before you can view the yaml by using:

kubectl get pods example -o yaml

Now create a file named myfirstpod.yml with the configuration below. You van use vim or nano as your editor.

---  
apiVersion: v1  
kind: Pod  
metadata:  
 labels:  
 phase: prod  
 role: frontend  
 name: myfirstpod  
 name: myfirstpod  
spec:  
 containers:  
 - name: filepuller  
 image: sliranc/filepuller:latest  
 volumeMounts:  
 - mountPath: /usr/share/nginx/html  
 name: static-vol  
 - name: webserver  
 image: nginx:latest  
 ports:  
 - containerPort: 80  
 volumeMounts:  
 - mountPath: /usr/share/nginx/html  
 name: static-vol  
 volumes:  
 - name: static-vol  
 emptyDir: {}

We can create a new deployment with this yaml file using:

$ kubectl apply -f myfirstpod.yml  
deployment "example" created

You can see the pods created with:

kubectl get pods  
NAME READY STATUS RESTARTS AGE  
myfirstpod 2/2 Running 0 3m