Procedure Document

**Kubernetes Procedure Document**

**Github repository [Read this first]**

* Download all the course material from: <https://github.com/wardviaene/kubernetes-course>
* Kubernetes releases minor version updates of its distribution every 3 months
  + Rather than updating the scripts in the video lectures, **the repository in Github is updated if any script need changes**
  + The changes are often very minor, the API is very stable. Often API versions like v1betaX change to v1betaX+1 or to v1 (stable)
  + All the scripts you can find in the repository **should work with the latest version of Kubernetes**, if you have any issues, contact me through one of the channels listed below

**Kubernetes setup lectures**

There are multiple ways to setup a kubernetes cluster. You only need 1 working cluster to do the demos, but I've added different ways that you can create your cluster.

* A local cluster (on your machine): you can follow the minikube or docker for windows/mac lectures
* A production cluster using Kops on AWS
* A on-prem or cloud-agnistic cluster using kubeadm (lecture is at the end of the course)
* A managed production cluster on AWS using EKS (lecture can also be found at the end of the course)

If you want to test all features, kops is the best choice. If you want to have a local cluster, try out minikube first. If you have issues getting minikube up, docker for windows/mac comes with kubernetes and is a great alternative.

Kops gives you full access to the master nodes. To learn to work with Kubernetes, kops is preferred. If you have issues setting up kops, you can give EKS a try (lectures at the end of the course). EKS is more expensive than kops, so make sure to keep an eye on your billing and destroy the cluster after you're done with testing/demos.

If you want to deploy a cluster on any cloud or on an on-prem machine, then kubeadm will be the way to go. If you want to use a cloud provider like Azure, Google, or DigitalOcean, that will also work. They all provide their own managed kubernetes solutions.

**Slides**

The slides can be downloaded from: <https://d3jb1lt6v0nddd.cloudfront.net/udemy/Learn+DevOps+-+Kubernetes.pdf>

**Questions?**

* Send me a message
* Use Q&A
* Join our facebook group: <https://www.facebook.com/groups/840062592768074/>

**Download Kubectl**

* Linux: <https://storage.googleapis.com/kubernetes-release/release/v1.17.0/bin/linux/amd64/kubectl>
* MacOS: <https://storage.googleapis.com/kubernetes-release/release/v1.17.0/bin/darwin/amd64/kubectl>
* Windows: <https://storage.googleapis.com/kubernetes-release/release/v1.17.0/bin/windows/amd64/kubectl.exe>
* Or use a packaged version for your OS: see <https://kubernetes.io/docs/tasks/tools/install-kubectl/>

**Minikube**

* Project URL: https://github.com/kubernetes/minikube
* Latest Release and download instructions: https://github.com/kubernetes/minikube/releases
* VirtualBox: http://www.virtualbox.org

**Minikube on windows:**

* Download the latest minikube-version.exe
* Rename the file to minikube.exe and put it in C:\minikube
* Open a cmd (search for the app cmd or powershell)
* Run: cd C:\minikube and enter minikube start

**Test your cluster commands**

Make sure your cluster is running, you can check with **minikube status.**

If your cluster is not running, enter **minikube start** first.

kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.4 --port=8080  
kubectl expose deployment hello-minikube --type=NodePort

minikube service hello-minikube --url

<open a browser and go to that url>

**Kops**

**Project URL**

* <https://github.com/kubernetes/kops>

**Free DNS Service**

* Sign up at <http://freedns.afraid.org/>
  + Choose for subdomain hosting
  + Enter the AWS nameservers given to you in route53 as nameservers for the subdomain
* <http://www.dot.tk/> provides a free .tk domain name you can use and you can point it to the amazon AWS nameservers
* Namecheap.com often has promotions for tld’s like .co for just a couple of bucks

**Cluster Commands**

* kops create cluster --name=kubernetes.newtech.academy --state=s3://kops-state-b429b --zones=eu-west-1a --node-count=2 --node-size=t2.micro --master-size=t2.micro --dns-zone=kubernetes.newtech.academy
* kops update cluster kubernetes.newtech.academy --yes --state=s3://kops-state-b429b
* kops delete cluster --name kubernetes.newtech.academy --state=s3://kops-state-b429b
* kops delete cluster --name kubernetes.newtech.academy --state=s3://kops-state-b429b --yes

**Kubernetes from scratch**

* You can setup your cluster manually from scratch
* If you’re planning to deploy on AWS / Google / Azure, use the tools that are fit for these platforms
* If you have an unsupported cloud platform, and you still want Kubernetes, you can install it manually
* CoreOS + Kubernetes: ###a href="https://coreos.com/kubernetes/docs/latest/getting-started.html">https://coreos.com/kubernetes/docs/latest/getting-started.html

**Docker**

* You can download Docker Engine for:
  + Windows:<https://docs.docker.com/engine/installation/windows/>
  + MacOS:<https://docs.docker.com/engine/installation/mac/>
  + Linux:<https://docs.docker.com/engine/installation/linux/>

**DevOps box**

* Virtualbox: <http://www.virtualbox.org>
* Vagrant: <http://www.vagrantup.com>
* DevOps box: <https://github.com/wardviaene/devops-box>
* Launch commands (in terminal / cmd / powershell):
  + cd devops-box/
  + vagrant up
* Launch commands for a plain ubuntu box:
  + mkdir ubuntu
  + vagrant init ubuntu/xenial64
  + vagrant up

**Cheatsheet: Docker commands**

Build image: docker build .

Build & Tag: docker build -t wardviaene/k8s-demo:latest .

Tag image: docker tag imageid wardviaene/k8s-demo

Push image: docker push wardviaene/k8s-demo

List images: docker images

List all containers: docker ps -a

**Cheatsheet: Kubernetes commands**

kubectl get pod: Get information about all running pods

kubectl describe pod <pod>: Describe one pod

kubectl expose pod <pod> --port=444 --name=frontend: Expose the port of a pod (creates a new service)

kubectl port-forward <pod> 8080: Port forward the exposed pod port to your local machine

kubectl attach <podname> -i: Attach to the pod

kubectl exec <pod> -- command: Execute a command on the pod

kubectl label pods <pod> mylabel=awesome: Add a new label to a pod

kubectl run -i --tty busybox --image=busybox --restart=Never -- sh: Run a shell in a pod - very useful for debugging

kubectl get deployments: Get information on current deployments

kubectl get rs: Get information about the replica sets

kubectl get pods --show-labels: get pods, and also show labels attached to those pods

kubectl rollout status deployment/helloworld-deployment: Get deployment status

kubectl set image deployment/helloworld-deployment k8s-demo=k8s-demo:2: Run k8s-demo with the image label version 2

kubectl edit deployment/helloworld-deployment: Edit the deployment object

kubectl rollout status deployment/helloworld-deployment: Get the status of the rollout

kubectl rollout history deployment/helloworld-deployment: Get the rollout history

kubectl rollout undo deployment/helloworld-deployment: Rollback to previous version

kubectl rollout undo deployment/helloworld-deployment --to-revision=n: Rollback to any version version

**AWS Commands**

* aws ec2 create-volume --size 10 --region us-east-1 --availability-zone us-east-1a --volume-type gp2

**Certificates**

* Creating a new key for a new user: openssl genrsa -out myuser.pem 2048
* Creating a certificate request: openssl req -new -key myuser.pem -out myuser-csr.pem -subj "/CN=myuser/O=myteam/"
* Creating a certificate: openssl x509 -req -in myuser-csr.pem -CA /path/to/kubernetes/ca.crt -CAkey /path/to/kubernetes/ca.key -CAcreateserial -out myuser.crt -days 10000

**Abbreviations used**

Resource type: Abbreviated alias

configmaps: cm

customresourcedefinition: crd

daemonsets: ds

deployments deploy

horizontalpodautoscalers: hpa

ingresses ing

limitranges limits

namespaces: ns

nodes: no

persistentvolumeclaims: pvc

persistentvolumes: pv

pods: po

replicasets: rs

replicationcontrollers: rc

resourcequotas: quota

serviceaccounts: sa

services: svc