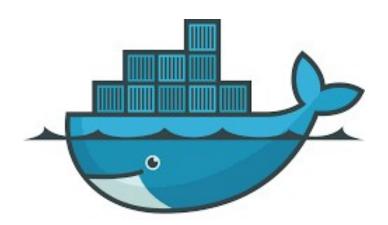


# T6 - DevOps

T-DOP-600

# DevOps - Kubernetes

Bootstrap







# DevOps - Kubernetes



- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (O if there is no error).

#### + STEP O: INSTALL

- Virtualbox or VMware
- Docker:
  - Docker: https://docs.docker.com/install/linux/docker-ce/fedora/
  - Docker Machine: https://docs.docker.com/machine/install-machine/, Minikube dependency
- Kubectl:
  - CLI for sending commands to k8s
  - https://kubernetes.io/docs/tasks/tools/install-kubectl/
  - awesome bash autocompletion: \$ kubectl completion bash
- Minikube:
  - a tool for starting a 1 node k8s cluster, locally
  - https://kubernetes.io/docs/tasks/tools/install-minikube/





# + STEP O: ABOUT MINIKUBE

Minikube start a VM into VirtualBox (by default) and installs a Kubernetes cluster inside. Start cluster:

```
minikube start

minikube status
minikube ip

# only the first time:
kubectl config current-context
kubectl config use-context minikube

Delete cluster:
minikube stop
```

minikube delete



# + STEP O: K8S VOCABULARY

- kubernetes nodes:
  - host, machine, vm, bare-metal...
  - \$ kubectl get nodes
- kubernetes namespaces:
  - separated environments
  - \$ kubectl get namespaces
  - at least 2 namespace on a fresh new cluster: default, kube-system
- kubernetes pods:
  - container or group of container
  - \$ kubectl get pods
  - \$ kubectl get pods --namespace=kube-system



## + STEP 1: CREATE A POD

Create a pod with the following Docker image: samber/hello-world-nodejs (https://hub.docker.com/r/samber/hello-world-nodejs).

```
# hello-world.pod.yaml

apiVersion: v1
kind: Pod
metadata:
   name: hello-world
spec:
   containers:
   - name: hello-world
   image: samber/hello-world-nodejs
```

```
Terminal

- + x

~/T-DOP-600> kubectl apply -f hello-world.pod.yaml

Terminal

- + x
```

Then wait few seconds for container deployment.

#### Then:

- Can you list existing pods in your k8s cluster?
- Can you print more details about this pod?
- Can you fetch and follow the container logs?
- Can you execute the 'date' command inside the running container?
- Can you delete this pod?

# + STEP 2: ENVIRONMENT VARIABLES

- Can you update hello-world.pod.yaml and add to the container the following environment variable: PORT=8080?
- Delete pod and apply again the new configuration.
- Check if it worked:
  - execute env inside container
  - Or kubectl describe pod <pod-id>

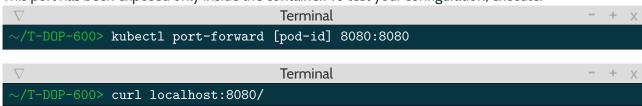




# + STEP 3: EXPOSE PORTS

This small application is a web server. Can you ask Kubernetes to expose the port 8080? Delete and apply hello-world.pod.yaml again;)

This port has been exposed only inside the container. To test your configuration, execute:



Port-forward command built a tunnel between your computer and Kubernetes cluster.

- Can you print full configuration of the container with kubectl describe?
- Do you see the exposed port?
- Do you see the container IP?

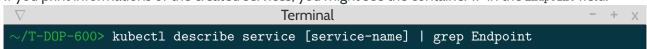
## + STEP 4: CREATE AN INTERNAL DNS

Thanks to the previous step, other services can request our HTTP server on :8080. But if your container dies, IP address will change. DNS are much better!

Create a k8s "Service" of type ClusterIP, linked to the hello-world app.



If you print informations of the created services, you might see the container IP in the Endpoint field.



Now, you should be able to ping your service from everywhere in the Kubernetes cluster:

```
Terminal

- + x

~/T-DOP-600> ping [service-name].[namespace].svc.cluster.local
```





# + STEP 5: VOLUMES

Can you attach a 512MB disk volume to the hello-world container?

# + STEP 6: FROM PODS TO DEPLOYMENTS

In the future, you will use k8s deployments instead of pods.

Deployments offer more features, such as:

- Replication policies
- Deployment history (and rollback!)
- Auto-scaling rules
- ...

Can you convert your hello-world.pod.yaml into a deployment?





## + TIPS AND TRICKS

You can get a full web UI for kubernetes with:



Most used kubectl commands:



Kubectl auto-completion is awesome:

- kubectl [tab]
- kubectl describe [tab]
- kubectl describe pod [tab]
- kubectl describe pod hello-world
- kubectl get pods --namespace [tab]

Useful 3rd-party tools:

- kubectx
- kubens





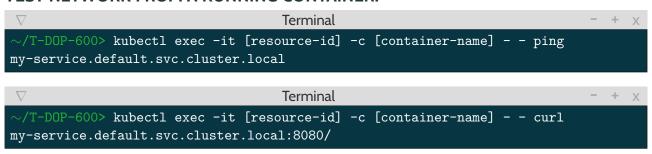
## + TROUBLESHOOTING WITH K8S

#### **GET INFO ABOUT EXISTING RESOURCE:**

 $\nabla$  Terminal - +  $\chi$   $\sim$ /T-DOP-600> kubectl describe [resource-id]

# **OPEN A SHELL INTO A RUNNING CONTAINER:**

#### **TEST NETWORK FROM A RUNNING CONTAINER:**



## **TEST PORT:**

By default, exposed ports are available only inside k8s cluster. So we need to build a tunnel.

#### From your linux:

 $\nabla$  Terminal - +  $\chi$   $\sim$ /T-DOP-600> curl localhost:8080/



Port-forward will never, ever, replace a VPN between 2 datacenters: only for debugging!





# **CHECK LINK BETWEEN K8S SERVICE AND PODS**

# **RTFM**

https://kubernetes.io/docs/concepts/

