



Kubernetes Tutorial

# Kubernetes on OpenSuse Leap 15

Youssef Hbali(c) 2019

Version 1.0, 08.03.2019

# Table of Contents

1. Summary .....	1
2. Docker .....	2
2.1. Docker installation .....	2
2.2. Start & check docker .....	2
2.3. Create an image .....	2
2.4. Build the image .....	2
2.5. Run and test the buit image .....	3
3. Kubernetes .....	4
3.1. Basic Architecture .....	4
3.2. Requirement : Virtual Box installation .....	4
3.3. Minikube .....	4
3.4. Kubectl .....	5
3.5. Starting Minikube .....	5
3.6. Configuration Sample .....	5
3.7. Pod .....	5
3.7.1. Pod creation and logs .....	5
3.7.2. Pods deletion .....	6
3.7.3. Organizing pods with labels .....	6
3.7.4. Display and filter lables .....	7
3.8. Node .....	7
3.8.1. Categorizing nodes .....	7
3.8.2. Sheduling pods to specific nodes. ....	7
3.9. Annotation .....	8
3.10. Namespace .....	8
3.10.1. Creating a namespace .....	8
3.10.1.1. From YAML .....	8
3.10.1.2. From command line .....	9
3.10.2. Managing objects in other namespaces. ....	9
4. Ansible .....	10
4.1. Installation .....	10
4.2. Clients Configuration .....	10
4.2.1. Clients Access by names .....	10
4.3. Ssh key-based access .....	10
4.4. Escalate without password .....	10
4.5. Set the ansible inventory .....	10

4.6. Run ansible module (-m).....	11
-----------------------------------	----

# 1. Summary

## 2. Docker

### 2.1. Docker installation

```
sudo zypper ar -cf \
http://download.opensuse.org/repositories/home:/janhebler:/kubi/openSUSE_Leap_15.0/ Docker

sudo zypper in docker
sudo zypper in docker-bash-completion
```

### 2.2. Start & check docker

```
sudo systemctl start docker
sudo systemctl status docker

# Add the current user to the docker group
sudo usermod -aG docker $USER

# Log out from the session or restart the OS in case you're using a VM

docker run hello-world
```

### 2.3. Create an image

```
touch Dockerfile

# Add
FROM node:7
ADD app.js /app.js
ENTRYPOINT ["node", "app.js"]

touch app.js

# Add
const http = require('http');
const os = require('os');
console.log("Kubia server starting...");
var handler = function(request, response) {
  console.log("Received request from " + request.connection.remoteAddress);
  response.writeHead(200);
  response.end("You've hit " + os.hostname() + "\n");
};
var www = http.createServer(handler);
www.listen(8080);
```

### 2.4. Build the image

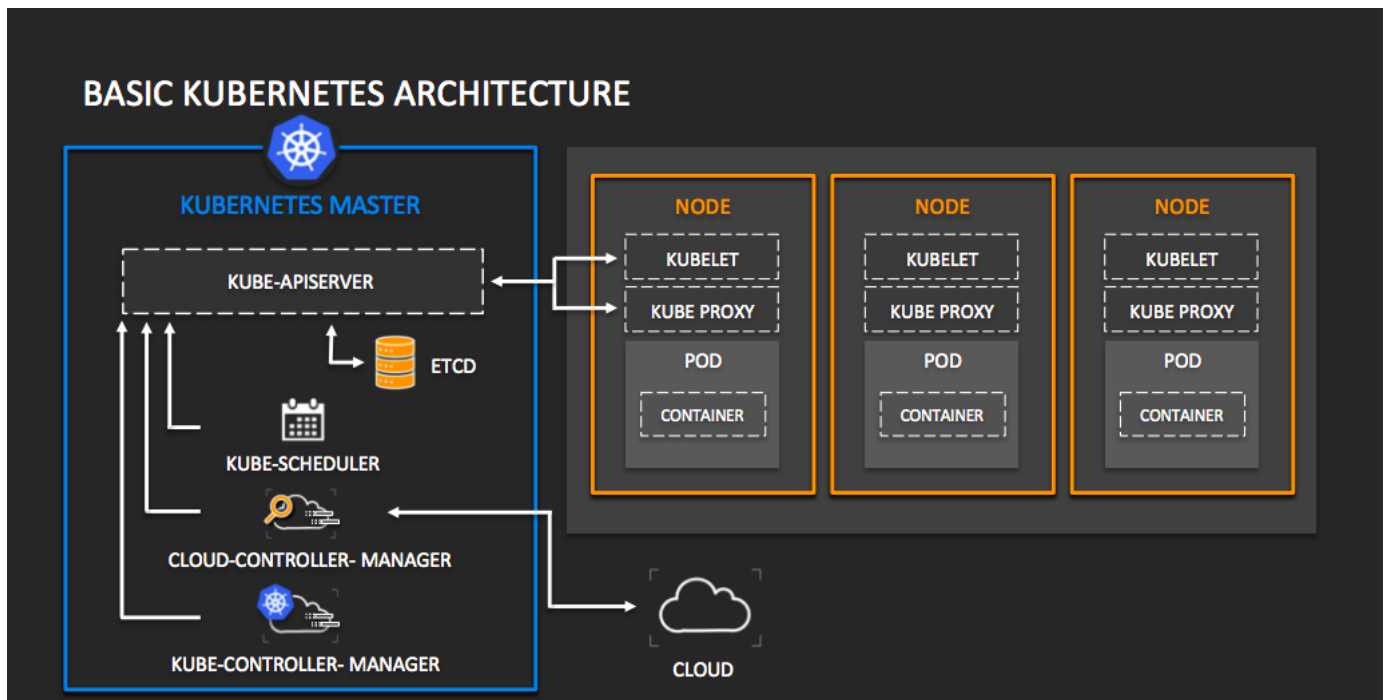
```
docker build -t image-name .
```

## 2.5. Run and test the built image

```
docker run --name container-name -p 8080:8080 -d image-name  
curl localhost:8080
```

# 3. Kubernetes

## 3.1. Basic Architecture



## 3.2. Requirement : Virtual Box installation

```
sudo zypper ar -cf \
http://download.opensuse.org/update/leap/15.0/oss oss

sudo zypper in virtualbox
sudo usermod -aG vboxusers $USER
# relog

sudo zypper install virtualbox-host-source kernel_devel
sudo /sbin/vboxconfig

sudo systemctl enable vboxdrv
sudo systemctl start vboxdrv
sudo systemctl status vboxdrv
```

## 3.3. Minikube

The simplest and quickest path to a fully functioning Kubernetes cluster is by using Minikube. Minikube is a tool that sets up a single-node cluster that's great for both testing Kubernetes and developing apps locally. Although we can't show certain Kubernetes features related to managing apps on multiple nodes, the single-node cluster should be enough for exploring most topics discussed in this book.

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/v0.35.0/minikube-linux-amd64 && chmod +x minikube
&& sudo mv minikube /usr/local/bin/
```

## 3.4. Kubectl

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s
https://storage.googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl && chmod +x kubectl &&
sudo mv kubectl /usr/local/bin/
```

## 3.5. Starting Minikube

```
youssef@tokyo:Kubernetes-training$ minikube start
minikube v0.35.0 on linux (amd64)
Tip: Use 'minikube start -p <name>' to create a new cluster, or 'minikube delete' to delete this one.
Restarting existing virtualbox VM for "minikube" ...
Waiting for SSH access ...
"minikube" IP address is 192.168.99.100
Configuring Docker as the container runtime ...
Preparing Kubernetes environment ...
Pulling images required by Kubernetes v1.13.4 ...
Relaunching Kubernetes v1.13.4 using kubeadm ...
Waiting for pods: apiserver proxy etcd scheduler controller addon-manager dns
Updating kube-proxy configuration ...
Verifying component health .....
kubectl is now configured to use "minikube"
Done! Thank you for using minikube!
```

## 3.6. Configuration Sample

```
### kuba-manual.yaml
apiVersion : v1 # Version of the Kubernetes API
kind: Pod
metadata:
  name : kuba-manual # The name of the Pod
spec:
  containers:
  - image : luksa/kuba
    name : kuba
    ports :
    - containerPort : 8080 # The port the app is listening on
      protocol : TCP

# Specifying ports in the pod definition is purely informational.
```

## 3.7. Pod

### 3.7.1. Pod creation and logs



```

kubectrl explain pods
kubectrl explain pod.spec
kubectrl create -f kubia-manual.yaml

# Retrieving the definition of a running pod
kubectrl get po kubia-manual -o yaml
kubectrl get po kubia-manual -o json

# List all pods
kubectrl get pods

# Viewing the logs of pod
kubectrl logs kubia-manual

# Viewing logs of a multi-container pod.
kubectrl logs kubia-manual -c kubia

# Port forwarding
kubectrl port-forward kubia-manual 8888:8080
curl localhost:8888

```

### 3.7.2. Pods deletion

```

kubectrl delete po kubia-gpu

# Delete Pods using labels.
kubectrl delete po -l creation_method=manual

# Deleting pods by namespace deletion.
kubectrl delete ns custom-namespace

# Deleting all namespaces.
kubectrl delete po --all

# Deleting all resources, ReplicationController, Pods and Services.
kubectrl delete all --all

```

### 3.7.3. Organizing pods with labels

```

### kubia-manual-v2.yaml
apiVersion : v1
kind: Pod
metadata:
  name: kubia-manual
  labels:      # Adding two labels to the Pod
    creation_method: kubia-manual
    env: prod
spec:
  containers:
  - image : luksa/kubia
    name : kubia
    ports :
    - containerPort : 8080
      protocol : TCP

```

### 3.7.4. Display and filter lables

```
kubectl get po --show-labels

# Display the two lables as columns
kubectl get po -L creation_method,env

# Add the label creation_method with value manual to the Pod kubia-manual
kubectl lable po kubia-manual creation_method=manual

# Change existing Pod value
kubectl label po kubia-manual-v2 env=debug --overwrite

# Filter pods with label creation_method=manual
kubectl get po -l creation_method=manual

# Pods with label named env
kubectl get po -l env

# Pods that does not have env label
kubectl get po -l '!env'

kubectl get po -l creation_method!=manual

# Pod with label env and value in (prod,devel)
kubectl get po -l env in (prod,devel)

kubectl get po -l env notin (prod,devel)

# Selection with two lables.
kubectl get po -l env=prod,creation_method=manual
```

## 3.8. Node

### 3.8.1. Categorizing nodes

```
# Adding label gpu to the node gke-kubia-85f6-node-0rrx
kubectl label node gke-kubia-85f6-node-0rrx gpu=true

# List all nodes with label gpu = true
kubectl get node -l gpu=true
```

### 3.8.2. Sheduling pods to specific nodes.

If you need to deploy a new pod that needs GPU, you can use nodeSelector.

```

apiVersion: v1
kind:Pod
metadata:
  name: kubia
  labels:
    env: devel
spec:
  nodeSelector:
    gpu : "true"
  containers:
  - image: luksa/kubia
    name: kubia
    ports:
    - containerPort: 8080
      protocol: TCP

```

## 3.9. Annotation

annotations can hold much larger pieces of information and are primarily meant to be used by tools. Certain annotations are automatically added to objects by Kubernetes, but others are added by users manually.

```

kubectl annotation pod kubia-label mycompany.com/someannotation="foo bar"
kubectl describe pod kubia-manual

```

## 3.10. Namespace

The need to split objects into two groupings.

```

# Lists all the namespaces
kubectl get ns

# Gets the pods belonging to the namespace kube-system.
kubectl get po --namespace kube-system

```

### 3.10.1. Creating a namespace

#### 3.10.1.1. From YAML

```

apiVersion: v1
kind: Namespace
metadata:
  name: custom-namespace

```

```

kubectl create -f custom-namespace.yaml

```

### 3.10.1.2. From command line

```
kubectl create namespace custom-namespace
```

### 3.10.2. Managing objects in other namespaces.

```
kubectl create -f kubia-manual.yaml -n custom-namespace
```

#### *Example 1. TIP*

```
kcd='kubectl config set-context $(kubectl config currentcontext) --namespace '
```

```
# use the new alias to switch to another namespace kcd some-namespace.
```

## 4. Ansible

In computing, Ansible is an open-source software provisioning, configuration management, and application deployment tool. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows.

### 4.1. Installation

```
sudo zypper in ansible
```

### 4.2. Clients Configuration

#### 4.2.1. Clients Access by names

```
# You have three VMs
# 127.0.10.1      ouarzazate
# 127.0.10.2      marrakech
# 127.0.10.3      casa

sudo vi /etc/hosts

# Add the following entries
127.0.10.1      ouarzazate
127.0.10.2      marrakech
127.0.10.3      casablanca
```

### 4.3. Ssh key-based access

```
# Generate RSA key pair
ssh-keygen
ssh-copy-id -i ~/.ssh/id_rsa.pub user1@ouarzazate # user1 is optionnal if is the same user
ssh-copy-id -i ~/.ssh/id_rsa.pub user1@marrakech
ssh-copy-id -i ~/.ssh/id_rsa.pub user1@casablanca
```

### 4.4. Escalate without password

```
sudo visudo
# Add
youssef ALL=(ALL) NOPASSWD: ALL
```

### 4.5. Set the ansible inventory

```
sudo vi /etc/ansible/hosts
# Two groups grp1 and grp2
[gui]
ouarzazate
marrakech

[nogui]
casablanca ansible_user=admin

# Group lab using range
[lab]
127.10.0.[1:6]
```

## 4.6. Run ansible module (-m)

```
ansible -m ping all
```