|  |  |
| --- | --- |
| **Kubernetes** | |
|  | **EQUIPE :BEJGANA ABDERAZZAK** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | |

SOMMAIRE

[INTRODUCTION 3](#_Toc206420653)

[MISE EN PLACE 3](#_Toc206420654)

[Démmarer le premier container 3](#_Toc206420655)

[Démmarer un container via fichier YAML (*whoami*) 4](#_Toc206420656)

[Démmarer un container via fichier YAML (*probes*) 5](#_Toc206420657)

[Démmarer deux containers 6](#_Toc206420658)

INTRODUCTION

L’objectif de ce document est de préparer le certificat CKA

MISE EN PLACE

Il faut lancer au premier Minikube

Démmarer le premier container

|  |
| --- |
| $ kubectl run pingpong --image alpine ping 8.8.8.8 |

|  |
| --- |
| $ kubectl get all  NAME READY STATUS RESTARTS AGE  pod/pingpong 0/1 ContainerCreating 0 5s  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  service/kubernetes ClusterIP 10.96.0.1 443/TCP 4d3h |

|  |
| --- |
| $ kubectl logs pod/pingpong  PING 8.8.8.8 (8.8.8.8): 56 data bytes  64 bytes from 8.8.8.8: seq=0 ttl=114 time=23.047 ms  64 bytes from 8.8.8.8: seq=1 ttl=114 time=22.819 ms  64 bytes from 8.8.8.8: seq=2 ttl=114 time=22.509 ms  64 bytes from 8.8.8.8: seq=3 ttl=114 time=22.117 ms |

**-**f/--follow : pour afficher les logs en temps réel

--tail : Afficher le nombre des lignes

--since : afficher les logs depuis une durée spécifique

Démmarer un container via fichier YAML (*whoami*)

Nous allons créer un Pod/container avec les paramètres suivants

* nom du Pod: *whoami*
* image du container: *containous/whoami*
* nom du container: *whoami*

**podinfo.yaml**

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: whoami  spec:  containers:  - name: whoami  image: containous/whoami |

Via invite de commande:

|  |
| --- |
| $ kubectl apply -f podinfo.yaml |

Verifier le status de pods:

|  |
| --- |
| kubectl get pods |

Accès à l’application via un port-forward

|  |
| --- |
| $ kubectl port-forward whoami 8888:80 |

Via un autre console , tester l’application

|  |
| --- |
| $ curl localhost:8888  Hostname: whoami  IP: 127.0.0.1  IP: ::1  IP: 10.244.3.5  IP: fe80::d879:caff:fe6d:2f83  RemoteAddr: 127.0.0.1:37628  GET / HTTP/1.1  Host: localhost:8888  User-Agent: curl/7.76.1  Accept: \*/\* |

Le Pod peut être supprimé avec la commande suivante :

|  |
| --- |
| $ kubectl delete po/whoami |

Démmarer un container via fichier YAML (*probes*)

Crée un pod/container nginx via un fichier yaml

www.yaml

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: www  spec:  containers:  - name: nginx  image: nginx:1.12.2 |

Démarrer le pod:

|  |
| --- |
| kubectl create -f www.yaml |

Afficher le pod:

|  |
| --- |
| kubectl port-forward www 8080:80  NAME READY STATUS RESTARTS AGE  wwww 1/1 Running 0 16m |

Activer le port forwarding:

|  |
| --- |
| kubectl port-forward wwww 8080:80 |

NB : la connexion entre deux containers sur le même pods peux être fait sans NAT ou autre

(exemple : nginx + alpine sur le meme pod)

Démmarer deux containers pour wordpress (Wordpress + DB)

Nous allons maintenant démarrer deux containers sur le même pod :

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: wp  spec:  containers:  - image: wordpress:4.9-apache  name: wordpress  env:  - name: WORDPRESS\_DB\_PASSWORD  value: mysqlpwd  - name: WORDPRESS\_DB\_HOST  value: 127.0.0.1  - image: mysql:5.7  name: mysql  env:  - name: MYSQL\_ROOT\_PASSWORD  value: mysqlpwd  volumeMounts:  - name: data  mountPath: /var/lib/mysql  volumes:  - name: data  emptyDir: {} |

Pour afficher les informations de pod.

|  |
| --- |
| kubectl get pod/wp  NAME READY STATUS RESTARTS AGE  wp 2/2 Running 0 2m |

Accès l’application :

kubectl port-forward wp 8080:80

ou

|  |
| --- |
| kubectl port-forward --address 0.0.0.0 wp 8080:80 |

Via l’interface web

[**http://192.168.1.34:8080/**](http://192.168.1.34:8080/)

Les objets: Service

* Expose les pods d’une application via des règles réseau.
* Utilise des labels pour grouper les Pods
* Adresse IP presistance (VIP : virtual IP adress)
* Kube-proxy en charge du load balancing sur les Pods

Les types de services :

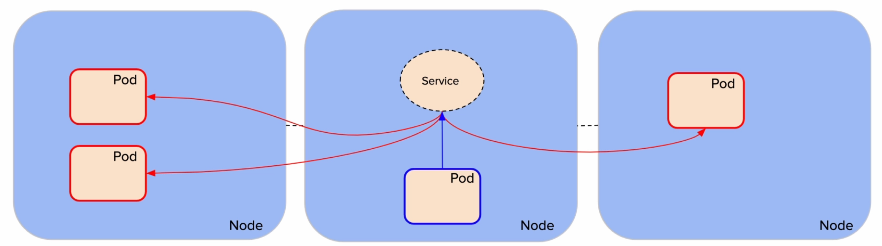
ClusterIP (défaut): exposition à l’intérieur du cluster

NodePort : exposition vers l’extérieur

LoadBalancer : intégration avec un cloud Provider

ExternalName : associe le service à un nom DNS

ClusterIP



Exemple:

Pod.yaml

|  |
| --- |
| apiVersion: v1  kind: Pod  metadata:  name: vote  labels:  app: vote  spec:  containers:  - name: vote  image: instavote/vote |

svc.yaml

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name : vote  spec:  selector:  app: vote  type: ClusterIP  ports:  - port: 80  targetPort: 80 |

L’accès via un autre container :

|  |
| --- |
| kubectl run alpine -ti --image=alpine  /apk add curl  /curl <http://vote>  [Resultat]  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  <title>Cats vs Dogs!</title>  <base href="/index.html">  <meta name = "viewport" content = "width=device-width, initial-scale = 1.0">  <meta name="keywords" content="docker-compose, docker, stack">  <meta name="author" content="Tutum dev team">  <style type="text/css"> |

Vous pouvez acceder au service aussi via :

Port-forward

Ou Proxy du kubernetes

NodePort

LoadBalancer :

ExternalName

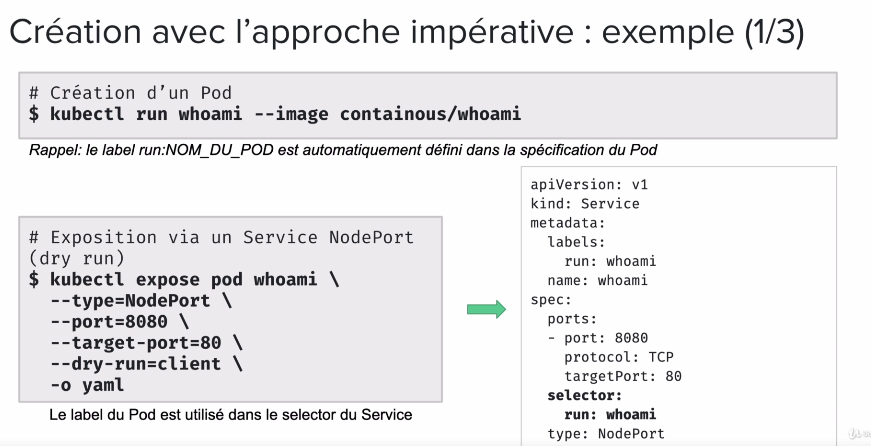
via les commandes impérative

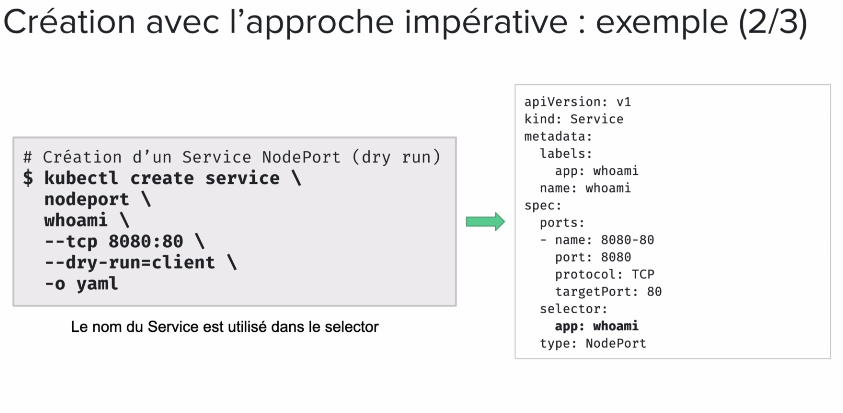
Creation d’un Pod

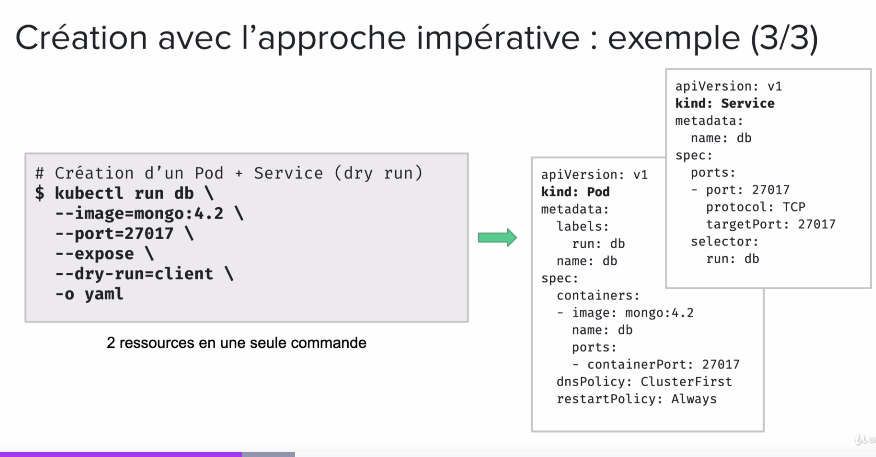
|  |
| --- |
| kubectl run whoami --image containous/whoami |

Exposition via service Nodeport

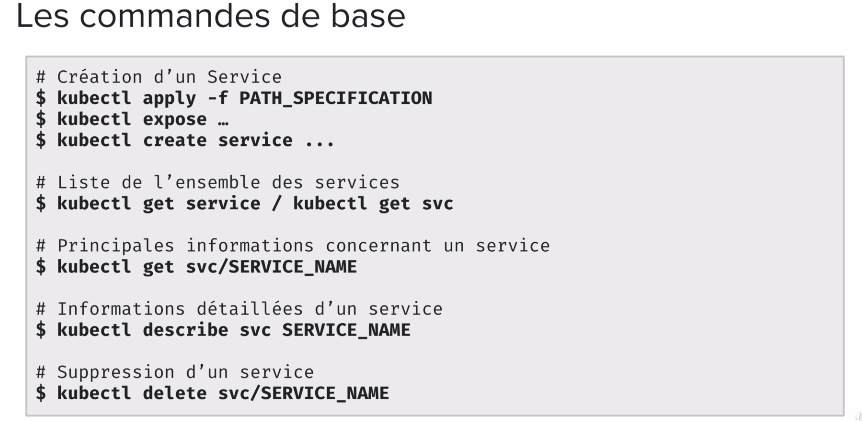
|  |
| --- |
| kubectl expose pod whoami --type=NodePort --port=8080 --target-port=80 --dry-run=client -o yaml |





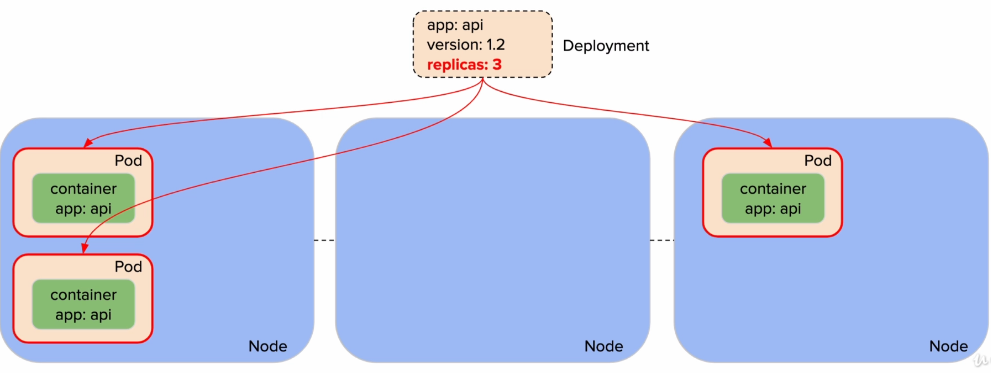


Les commandes de base:



Les objets: Deployment

Un deployment permet de gerer un ensemble de Pods identiques (mise à jours / Rollback).



**Utilisation:**

Gestion du cycle de vie de Pods

Création/suppression

Scaling

Rollout /Rollback

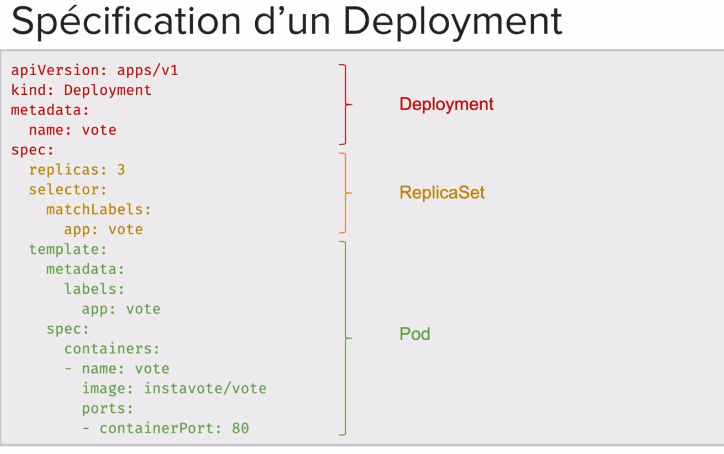
Different niveaux d’abstraction.

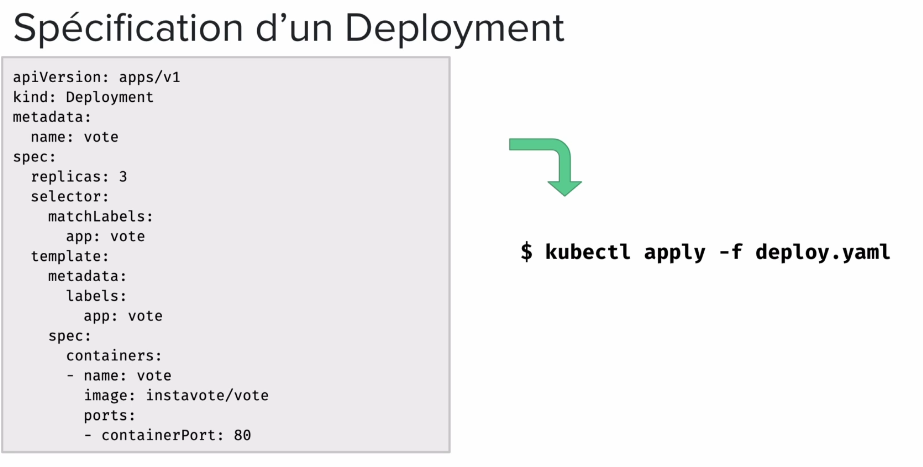
Deployment

ReplicatSet

Pod

**Mise à jour d'un Deployment**





Creation d’un Pod

|  |
| --- |
| kubectl run whoami --image containous/whoami |

Mise à jour d'un Deployment

Creation d’un deployment

|  |
| --- |
| kubectl create deploy www --image nginx:1.16 |

Scaling:

|  |
| --- |
| kubectl scale deploy/www –replicas 3 |

|  |
| --- |
| kubectl scale deploy/www –replicas 3 |

Liste des resources:

|  |
| --- |
| kubectl get deploy,rs,pod  NAME DESIRED CURRENT READY AGE  replicaset.apps/www-785f696554 3 3 3 24s  replicaset.apps/www-89758c68d 0 0 0 3m17s  NAME READY STATUS RESTARTS AGE  pod/www-785f696554-8lmc8 1/1 Running 0 21s  pod/www-785f696554-cgxkz 1/1 Running 0 24s  pod/www-785f696554-kb87j 1/1 Running 0 23s |

* un pour la gestion des Pods utilisant l’image *nginx:1.16*. Celui-ci n’est plus actif, comme le montre la valeur *0* des champs *DESIRED*, *CURRENT* et *READY* relatifs aux Pods gérés par le ReplicaSet
* un second qui a été créé lors de la mise à jour de l’image, il gère 3 Pods, chacun ayant un container basé sur l’image *nginx:1.16-alpine*

Historique des mises à jour

|  |
| --- |
| kubectl rollout history deploy/www |

Rollback

|  |
| --- |
| kubectl rollout undo deploy/www |

HorizontalPodAutoscaler

Vous devez installer Metrics server, pour utiliser ce service

|  |
| --- |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: w3  spec:  selector:  matchLabels:  app: w3  replicas: 1  template:  metadata:  labels:  app: w3  spec:  containers:  - image: nginx:1.20-alpine  name: w3  ports:  - containerPort: 80  resources:  requests:  cpu: 200m |

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name: w3  spec:  ports:  - port: 80  protocol: TCP  targetPort: 80  selector:  app: w3 |

|  |
| --- |
| apiVersion: v1  kind: Service  metadata:  name: w3  spec:  ports:  - port: 80  protocol: TCP  targetPort: 80  selector:  app: w3 |

|  |
| --- |
| apiVersion: autoscaling/v2  kind: HorizontalPodAutoscaler  metadata:  name: hpa-v2  spec:  scaleTargetRef:  apiVersion: apps/v1  kind: Deployment  name: w3  minReplicas: 1  maxReplicas: 10  metrics:  - type: Resource  resource:  name: cpu  target:  type: Utilization  averageUtilization: 10 |

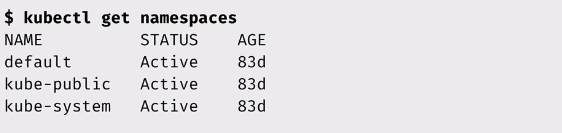
Les objects : NameSpace

Scope pour les Pods, Services, Deployment

Partage d’un cluster

Équipe / Projet /client

3 namespaces par défaut:



**La creation:**

|  |
| --- |
| kubectl create namespace development |

**Suppression:**

|  |
| --- |
| kubectl delete namespace/development |

|  |
| --- |
| vi development.yaml  {  } |

|  |
| --- |
| Kubectl create -f development.yaml |