



Kubernetes

Mise en oeuvre



Table des Matières

1. Introduction
2. Installation
3. Configuration
4. Les Pods
5. Le Scheduler
6. Service
7. Deployment
8. Namespace

1.

Introduction

et historique



Historique

Kubernetes ou k8s

Borg de Google

Version 1.0 07/2015

Version 1.23 12/2021

Cloud Native Computing Foundation (CNCF)

Qu'est ce que Kubernetes

Orchestrateur

Gestion d'applications conteneurisées

- Déploiement
- Self-healing
- Montée en charge
- Mise à jour

Gestion des configurations

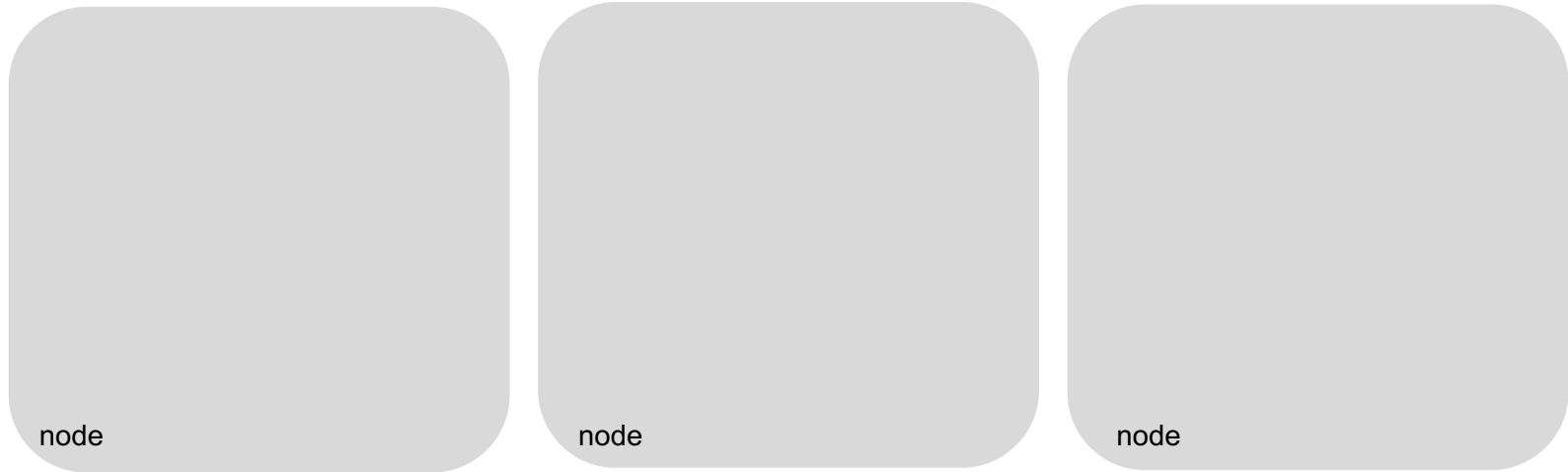
Gestion des mots de passes (Secrets)

RBAC

Concepts de base / Cluster

Cluster Kubernetes

- Ensemble de Nodes
- Linux ou Windows(1.14)

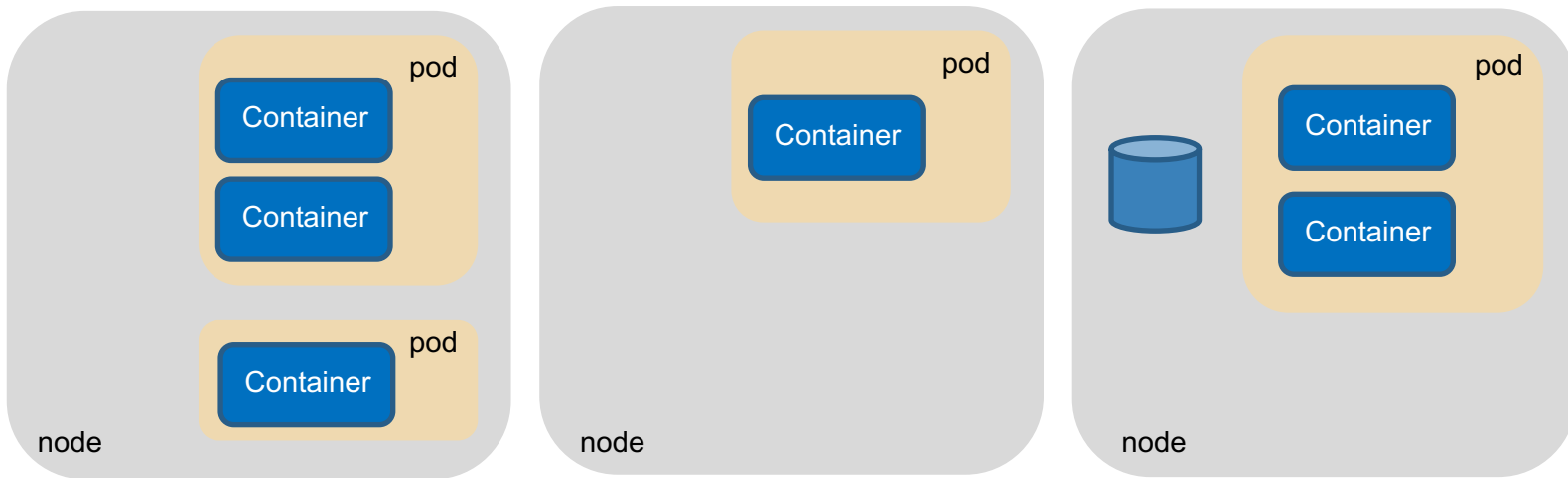


Concepts de base / Pods

Pods

- Ensemble de conteneurs
- Partage Réseau
- Partage Stockage

```
kind: Pod
spec:
  containers:
    - name: web
```

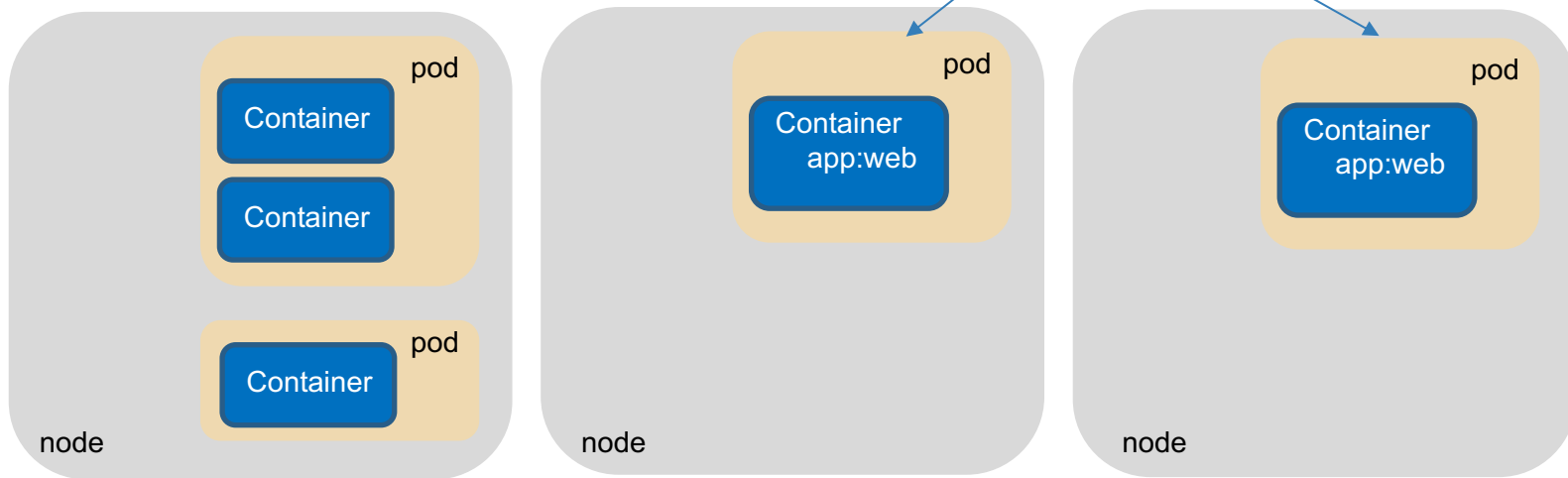


Concepts de base / Deployment

Deployment

- Définit la gestion d'un Pod
- Replicas
- Mise à Jour

```
kind: Deployment
spec:
  replicas: 2
```

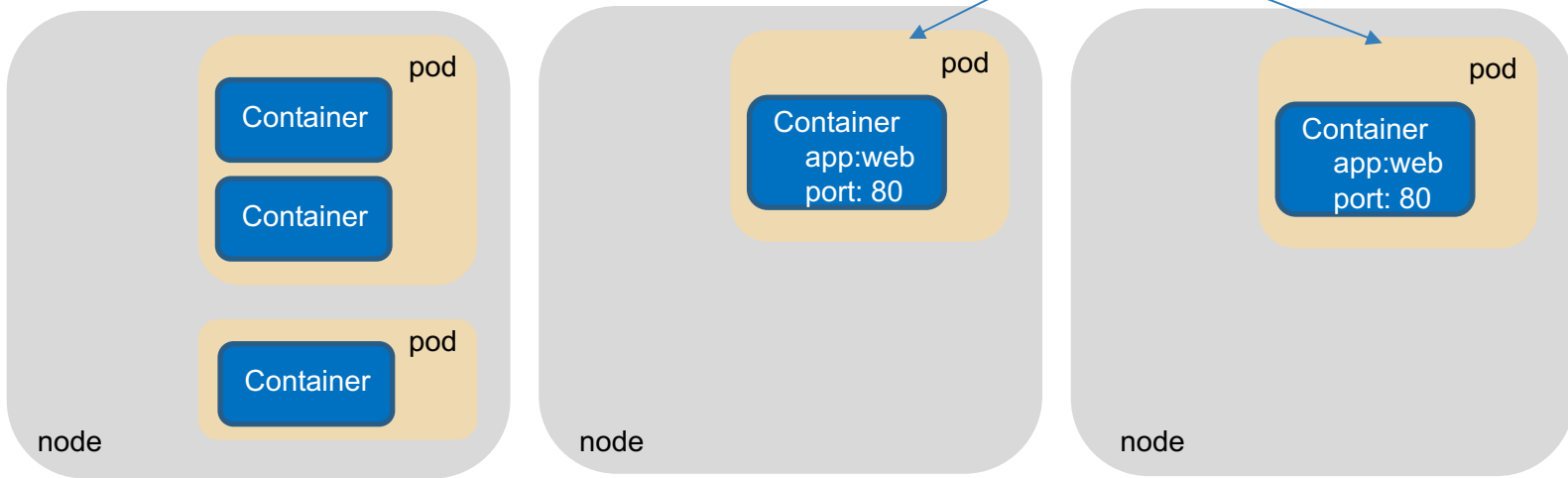


Concepts de base / Service

Service

- Permet d'exposer les ports d'un Pod
- A l'intérieur du Cluster
- A l'extérieur du Cluster

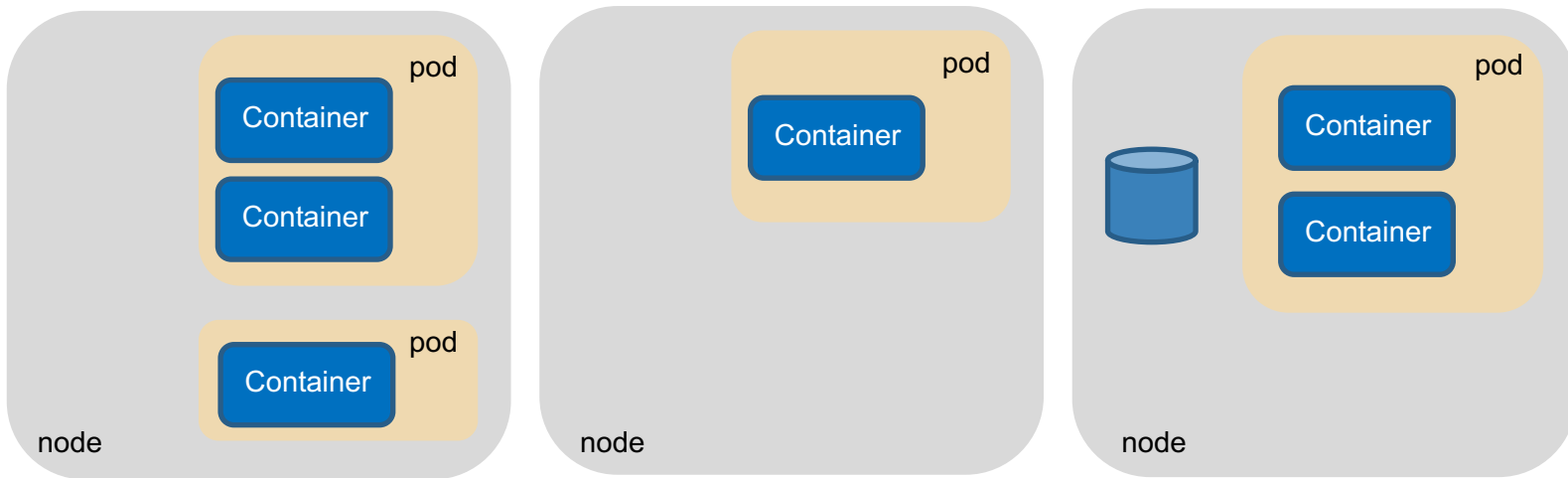
```
kind: Service
spec:
  ports:
    - port: 8080
      targetPort: 80
```



Concepts de base / Clients

Clients

- CLI: `kubectl`
- BUI
- Fichier `config`
- Gestion des Accès `Users/Access`



Concepts de base / Nodes

Master

- Gestion du cluster
- Orchestre l'exécution des Pods sur les Nodes
- Expose l'API server

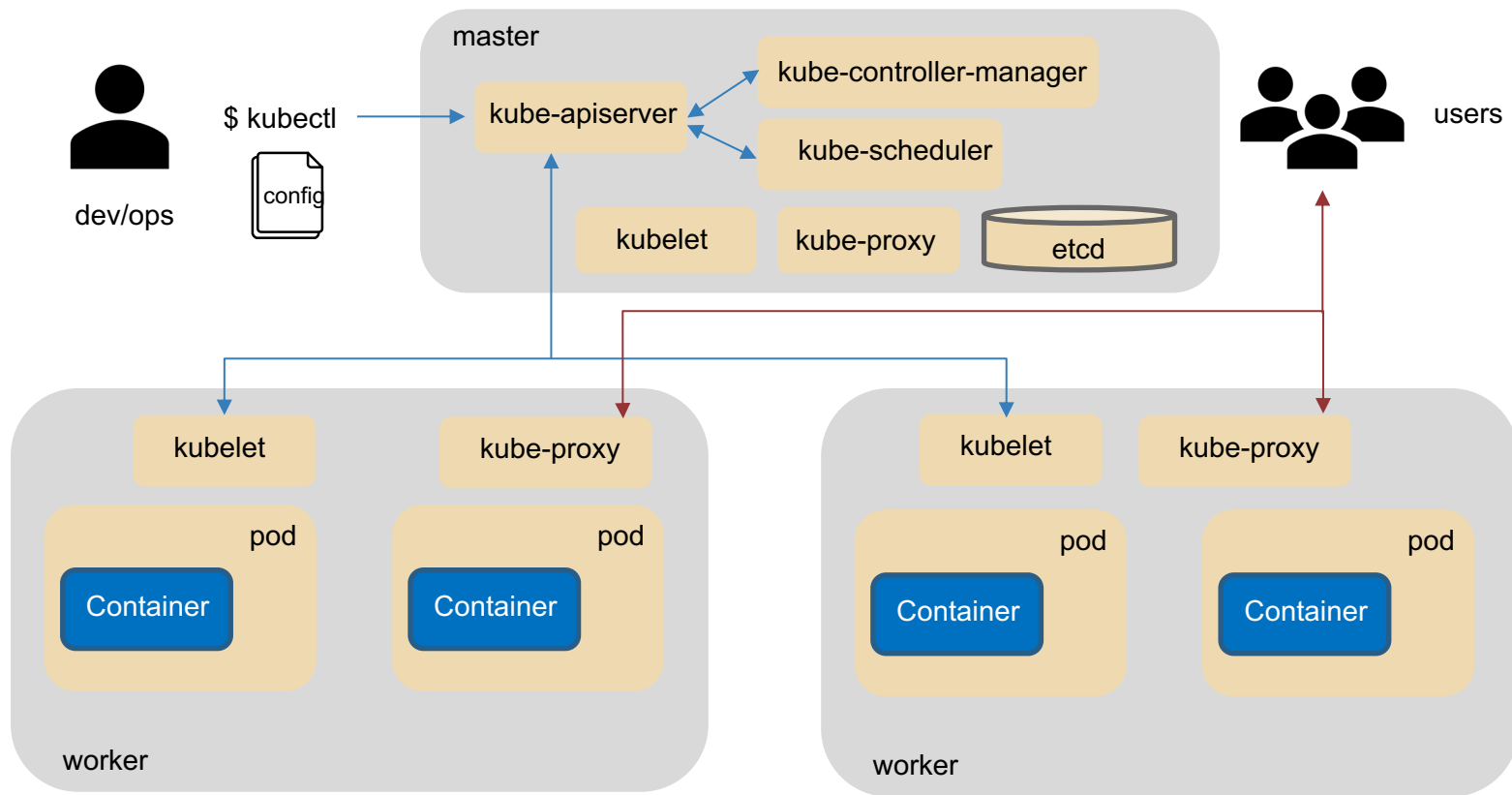
Worker

- Exécute les Pods du cluster
- Fournit les ressources aux Pods
- Communique avec le Master

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	2d20h	v1.15.3
worker1	Ready	<none>	2d20h	v1.15.3
worker2	Ready	<none>	2d15h	v1.15.3

Concepts de base / Process



Process & Composants du Master

`kube-apiserver`

- Expose l' API Kubernetes
- Front-end du Master ou Kubernetes Control Plane

`kube-scheduler`

- Selectionne le nœud d'exécution du Pod
- Respecte les spécifications du Pod

`kube-controller-manager`

- Node Controller
- Replication Controller
- End-Point Controller
- Account Controller

`etcd`

- Store key=value

Process & Composants du Worker

kubelet

- Communique avec l'API server
- Surveille le Pod
- S'assure qu'il tourne conformément à sa spécification

kube-proxy

- Gère le réseau
- Expose les services

Clients Kubernetes

kubectl

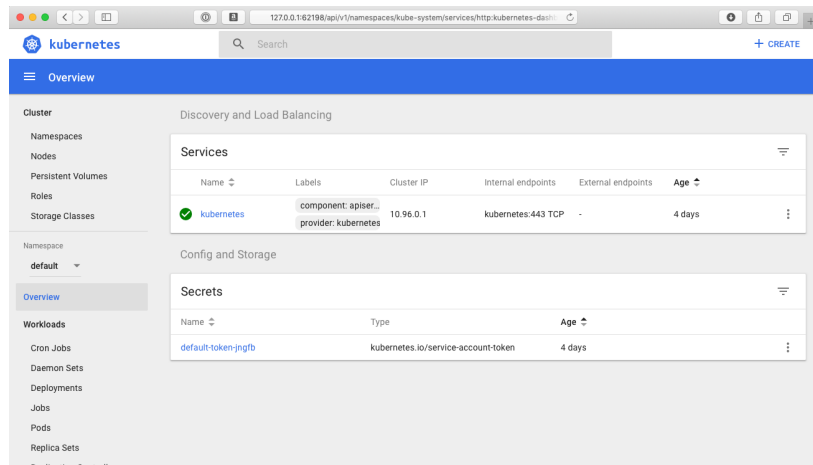
- CLI: ligne de commande
- kubeconfig:
 - fichier de configuration du client kubectl

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master	Ready	master	2d20h	v1.15.3
worker1	Ready	<none>	2d20h	v1.15.3
worker2	Ready	<none>	2d15h	v1.15.3

dashboard

- BUI: interface web d'administration



2.

Installation

et solutions



Types d'installation

Solutions locales

- Minikube
- Docker Desktop (Mac & Windows)

Solutions hébergées

- EKS : Amazon Elastic Kubernetes Service
- GKE : Google Kubernetes Engine
- AKS : Azure Kubernetes Service

Solutions On-Premise

- kubeadm
- kubespray
- OpenShift (RedHat)
- Rancher (SUSE)
- Docker EE (Mirantis)
- Tanzu (Vmware)

Environnement de tests

- play-with-k8s.com

Solution tout en un

Locale

Orientée développeur

Basée sur une VM d'un hyperviseur Host-Based

- VirtualBox
- Vmware
- Fusion

Sur poste de travail

- Linux
- Windows
- Mac

Minikube pré-requis

Installer un hyperviseur en local

Installer le binaire minikube

- Linux : <https://github.com/kubernetes/minikube>
- Windows : <https://github.com/kubernetes/minikube> **minikube-installer.exe**
- MacOS : [https://storage.googleapis.com/minikube minikube-darwin-amd64](https://storage.googleapis.com/minikube/minikube-darwin-amd64)

Installer le binaire kubectl

- <https://kubernetes.io/fr/docs/tasks/tools/install-kubectl/>

Docker Desktop

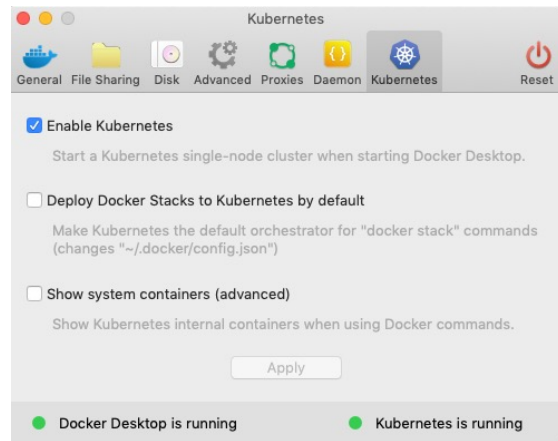
Basée sur un VM linux avec l'hyperviseur Natif

- Hyper-V à partir de Windows 10 Pro et Entreprise
- Hyperkit (xhyve) à partir de macOS Sierra 10.12

Activer Kubernetes comme solution Cluster

```
MacBook$ kubectl config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO
	docker-desktop	docker-desktop	docker-desktop
	docker-for-desktop	docker-desktop	docker-desktop
*	minikube	minikube	minikube



kubeadm

Binaire kubeadm installé sur tous les nœuds du Cluster

Docker ou autre Container Run Time installé

Master

- Lancer `kubeadm init`
- Mettre en place le réseau

```
master# kubeadm init
master$ kubectl apply -f https://raw.githubusercontent.com/coreos/kube-flannel.yml
```

Worker

- Lancer `kubeadm join`

```
worker# kubeadm join 192.168.116.160:6443 --token mm20xq.goxx7plwzrx75tv3
```

minikube

- Installer sur le machine locale le binaire `minikube`
- Installer sur le machine locale le binaire `kubectl`
- Créer une instance minikube par défaut
- Vérifier par la commande `kubectl` les nœuds du cluster

kubeadm

- Installer sur le machine locale le binaire `kubectl`
- Vérifier par la commande `kubectl` les nœuds du cluster

3.

Configuration

et contextes



Context

Context d'utilisation

- Définit le cluster k8s cible
- Définit l'utilisateur

Plusieurs contexts peuvent être disponibles

Un seul context actif indiqué par CURRENT *

```
$ kubectl config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO	NAMESPACE
	docker-desktop	docker-desktop	docker-desktop	
*	minikube	minikube	minikube	

Context de configuration

Context utilisé par `kubectl` dans l'ordre de priorité

1. Option de la commande
`--kubeconfig`
spécifie un fichier de config
2. Variable d'environnement
`$KUBECONFIG`
3. Fichier kubeconfig par défaut
`$HOME/.kube/config`

Context de configuration

```
$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://kubernetes.docker.internal:6443
    name: docker-desktop
- cluster:
    certificate-authority: /Users/samir/.minikube/ca.crt
    server: https://192.168.99.100:8443
    name: minikube
contexts:
- context:
    cluster: docker-desktop
    user: docker-desktop
    name: docker-desktop
- context:
    cluster: minikube
    user: minikube
    name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: docker-desktop
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
- name: minikube
  user:
    client-certificate: /Users/samir/.minikube/client.crt
    client-key: /Users/samir/.minikube/client.key
```

Commandes de Context

Lister les contexts

```
$ kubectl config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO	NAMESPACE
	docker-desktop	docker-desktop	docker-desktop	
*	minikube	minikube	minikube	

Changer de context

```
$ kubectl config use-context docker-desktop  
Switched to context "docker-desktop".
```

Supprimer un context

```
$ kubectl config delete-context minikube  
deleted context minikube from /Users/samir/.kube/config
```

Sur le poste client

- Transférer Installer sur le machine locale le binaire `minikube`
- Installer sur le machine locale le binaire `kubectl`
- Créer une instance minikube par défaut
- Vérifier par la commande `kubectl` les nœuds du cluster

kubeadm

- Installer sur le machine locale le binaire `kubectl`
- Créer une instance minikube par défaut
- Vérifier par la commande `kubectl` les nœuds du cluster

4.

Les Pods

Catégories de ressources

Gestion des Applications

- Deployment
- Pod
- ReplicaSet

Load balancing

- Service

Configuration des Applications

- ConfigMap
- Secret

Stockage

- Persistent Volume
- Persistent Volume Claim

Configuration du Cluster

- Metadata
- Namespace
- Roles

Clés de définition

apiVersion

- v1
- apps/v1

kind

- Pod
- Deployment

metadata

- name
- labels

spec

- containers
- ports

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

Pod

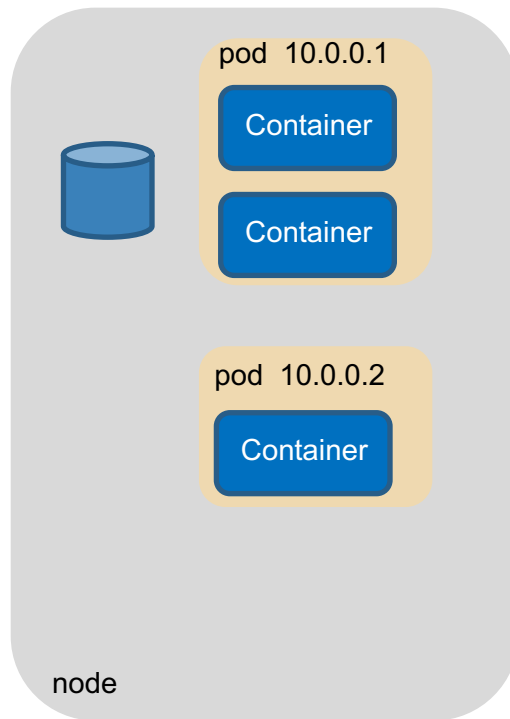
Unité de déploiement applicative

Ensemble de conteneurs

- même contexte d'isolation
- même stack réseau
- mêmes volumes partagés
- même adresse IP dédiée
- des spécification d'exécution

Ephémères

Best Practice : 1 Pod = 1 Container

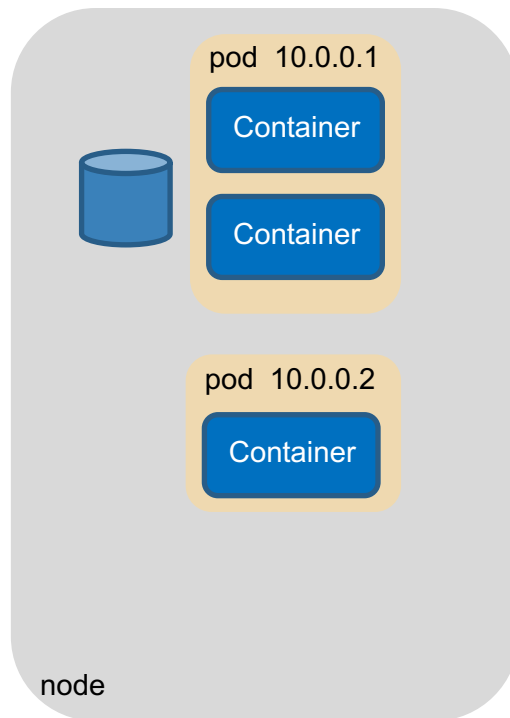


Pod

Une Application est découpée en 1 ou plusieurs Pods

Chaque Pod fournit un service

Scaling horizontal



Exemple de Spécification

Spécification au format YAML ou JSON

```
$ more web.yml
apiVersion: v1  ← version de l' API
kind: Pod      ← type d'objet
metadata:      ← informations sur l'objet
  name: web    ← nom de l'objet
spec:          ← les spécifications de l'objet
  containers:  ← section conteneurs
    - name: www ← nom du conteneur
      image: nginx:1.17 ← nom de l'image
```

Création d'un Pod

Création d'un Pod

```
$ kubectl create -f SPEC.yml
```

```
$ kubectl apply -f SPEC.yml
```

```
$ kubectl apply -f web.yml  
pod/web created
```

Liste des Pod

```
$ kubectl get pod [--namespace=NS ]
```

```
$ kubectl get pods  
NAME      READY   STATUS    RESTARTS   AGE  
web       1/1     Running   0           41s
```

Pod / Etats

Pending

Running

Succeeded

Failed

```
$ kubectl get po
```

NAME	READY	STATUS	RESTARTS	AGE
web	0/1	ContainerCreating	0	2s

```
$ kubectl get po
```

NAME	READY	STATUS	RESTARTS	AGE
web	1/1	Running	0	28s

Suppression d'un Pod

Suppression d'un Pod

```
$ kubectl delete pod NAME
```

```
$ kubectl delete po web  
pod "web" deleted
```

Liste des Pod

```
$ kubectl get pod [--namespace=NS ]
```

```
$ kubectl get pods  
No resources found.
```

Description d'un Pod

Description d'un Pod

```
$ kubectl describe pod/NAME
```

```
$ kubectl describe pod/web
```

```
Name:                web
Namespace:           default
Priority:             0
PriorityClassName:    <none>
Node:                minikube/10.0.2.15
Start Time:          Sat, 21 Sep 2019 14:56:53 +0200
Labels:              <none>
Annotations:         <none>
Status:              Running
IP:                  172.17.0.5
Containers:
  www:
    Container ID:     docker://882923c4e19d690c6efea92bebeb4dd90f46e05cdcb63279b96b99dc1ed0b3fd
    Image:             nginx:1.17
```

Description d'un Pod

Conditions:

Type	Status
Initialized	True
Ready	True
ContainersReady	True
PodScheduled	True

Volumes:

default-token-xk2ns:

Type:	Secret (a volume populated by a Secret)
SecretName:	default-token-xk2ns
Optional:	false

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
node.kubernetes.io/unreachable:NoExecute for 300s

Events:

Type	Reason	Age	From	Message
----	-----	----	----	-----
Normal	Scheduled	11m	default-scheduler	Successfully assigned default/web to minikube
Normal	Pulling	11m	kubelet, minikube	Pulling image "nginx:1.17"
Normal	Pulled	11m	kubelet, minikube	Successfully pulled image "nginx:1.17"
Normal	Created	11m	kubelet, minikube	Created container www
Normal	Started	11m	kubelet, minikube	Started container www

Logs d'un Pod

Logs d'un Pod

```
$ kubectl logs [-f] pod/NAME [-c CONTAINER]
```

```
$ kubectl logs -f po/web -c www
```

```
127.0.0.1 - - [21/Sep/2019:13:33:13 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"  
127.0.0.1 - - [21/Sep/2019:13:33:17 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"  
127.0.0.1 - - [21/Sep/2019:13:33:18 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"  
127.0.0.1 - - [21/Sep/2019:13:33:59 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.29.0" "-"
```

Exécution d'une commande

Exécution d'une commande dans le container d'un Pod

```
$ kubectl exec [-it] pod/NAME [-c CONTAINER] -- COMMAND
```

```
$ kubectl exec -it web -- bash
root@web:/#
```

Exécution d'une commande dans le container par défaut d'un Pod

```
$ kubectl exec -it web -- bash
Defaulted container "www" out of: www, debug
root@web:/#
```

```
$ kubectl exec -it web -c debug -- bash
[root@web /]#
```

Port Forwarding

Publie le port d'un Pod sur le Host (mapping de port)

Pultôt utilisée en mode debug ou en developpement

```
$ kubectl port-forward POD Host_PORT:CONTAINER_PORT
```

```
$ kubectl port-forward web 8000:80
Forwarding from 127.0.0.1:8000 -> 80
Forwarding from [::1]:8000 -> 80
Handling connection for 8000
```

```
$ curl 127.0.0.1:8000
<!DOCTYPE html>
<html>
<title>Welcome to nginx!</title>
```

Impérative vs Déclarative

Approche impérative

- Expliciter les étapes à exécuter pour créer l'objet final

Approche déclarative

- Déclarer que est le résultat final désiré, sans indiquer le processus pour y parvenir

Impérative

Créer un objet en ligne de commande

```
$ kubectl run --image=nginx:1.21-alpine web
```

Créer un objet avec un fichier de spécification

```
$ kubectl create -f nginx.yml
```

Modifier un objet avec un fichier de spécification

```
$ kubectl replace -f nginx.yml
```

Supprimer un objet

```
$ kubectl delete pod web
```

Créer un objet avec un fichier de spécification

```
$ kubectl apply -f nginx.yml
```

Modifier un objet avec un fichier de spécification

```
$ kubectl apply -f nginx.yml
```

Supprimer un objet avec un fichier de spécification

```
$ kubectl apply -f nginx.yml
```

<https://kubernetes.io/docs/tasks/manage-kubernetes-objects/declarative-config/>



Fichier Local

```
apiVersion: v1
kind: Pod
metadata:
  name: web
  labels:
    app: myapp
spec:
  containers:
  - name: www
    image: nginx:1.17
```

dernière configuration appliquée

```
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "annotations": {},
    "labels": {
      "app": "myapp"
    },
    "name": "web",
  },
  "spec": {
    "containers": [
      {
        "image": "nginx:1.17",
        "name": "www"
      }
    ]
  }
}
```

Live Kubernetes

```
apiVersion: v1
kind: Pod
metadata:
  name: web
  labels:
    app: myapp
spec:
  containers:
  - name: www
    image: nginx:1.17
status:
```

5.

Scheduler

Labels

Paires Clé=Valeur

Attachées aux Objets Kubernetes

- Pods
- Services
- Déploiement
- Nœuds

Identifier les attributs des objets

Organiser les applications sur le Cluster

Créés lors du déploiement ou ultérieurement

Modifiés à tout moment

- `"release" : "stable" , "release" : "test"`
- `"environment" : "dev" , "environment" : "production"`

Ajout de Labels

Sur un Noeud en CLI

```
$ kubectl label nodes worker1 disktype=ssd
```

```
node/worker1 labeled
```

```
$ kubectl get nodes --show-labels
```

NAME	STATUS	ROLES	AGE	VERSION	LABELS
master	Ready	master	9h	v1.16.0	
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=master,kubernetes.io/os=linux,node-role.kubernetes.io/master=					
worker1	Ready	<none>	9h	v1.16.0	
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,disktype=ssd,kubernetes.io/arch=amd64,kubernetes.io/hostname=worker1,kubernetes.io/os=linux					
worker2	Ready	<none>	9h	v1.16.0	
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=worker2,kubernetes.io/os=linux					

Sélection d'objets à partir de Labels

Identifie un ensemble spécifique d'objets

Deux types

- Equality-based
 - = égalité stricte
 - != différence
- Set-based
 - IN la valeur est dans un ensemble de valeurs définies
 - NOTIN la valeur ne doit pas figurer dans un ensemble de valeurs définies
 - EXISTS détermine seulement si le Label existe ou pas

nodeName

Permet de réaliser un scheduling manuel

```
$ cat web.yml
apiVersion: v1
kind: Pod
metadata:
  name: web
spec:
  containers:
  - name: www
    image: nginx:1.17
  nodeName: worker1
```

nodeSelector

Permet de lancer un Pod sur un nœud ayant un Label spécifique

```
$ cat web.yml
apiVersion: v1
kind: Pod
metadata:
  name: web
spec:
  containers:
  - name: www
    image: nginx:1.17
  nodeSelector:
    disktype: ssd
```

nodeAffinity

requiredDuringSchedulingIgnoredDuringExecution

preferredDuringSchedulingIgnoredDuringExecution

```
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: kubernetes.io/hostname
                operator: In
                values:
                  - worker1
                  - worker2
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
              - key: disktype
                operator: In
                values:
                  - ssd
```

podAffinity

Schéduer un pod par affinité avec les autres pods

Règles hard ou soft

Clé `topologyKey` : spécifie le domaine d'exécution (hostname,region,az)

```
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: security
                operator: In
                values:
                  - S1
            topologyKey: kubernetes.io/zone
```

podAntiAffinity

Schéduer un pod par affinité négative avec les autres pods

Règles hard ou soft

Clé `topologyKey` : spécifie le domaine d'exécution (hostname,region,az)

```
spec:
  affinity:
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 100
          podAffinityTerm:
            labelSelector:
              matchExpressions:
                - key: security
                  operator: In
                  values:
                    - S2
            topologyKey: kubernetes.io/hostname
```

Taints

Taint vs NodeAffinity

- Taint est l'inverse de l'affinité
- Déclaré sur un noeud
- Permet à un nœud de repousser un pod

```
kubectl taint nodes <node> key=value:<action>
```

Action

- NoSchedule
- PreferNoSchedule
- NoExecute

```
$ kubectl taint nodes host11 app=db:NoSchedule  
node/host11 tainted
```

Tolerations

Tolerations

- Les tolérances sont appliquées aux pods
- Permettent aux pods de tolérer un Taint et d'être scheduler sur ces nœuds

```
kubectl taint nodes <node> key=value:<action>
```

```
apiVersion: v1
kind: Pod
metadata:
  name: web
  labels:
    app: myapp
spec:
  tolerations:
  - key: "app"
    operator: "Equal"
    value: "db"
    effect: "NoSchedule"
```

Completion

- Commandes et sous-commandes
- ressources

```
$ yum install -y bash-completion
Installé :
bash-completion.noarch 1:2.1-6.el7
```

```
$ cat .bashrc
source <(kubectl completion bash)
```

```
$ kubectl delete pods →|
debug web
```

La commande `kubectl` liste les ressources accessibles par l'API

```
$ kubectl api-resources
```

NAME	SHORTNAMES	APIGROUP	NAMESPACED	KIND
bindings			true	Binding
componentstatuses	cs		false	ComponentStatus
configmaps	cm		true	ConfigMap
endpoints	ep		true	Endpoints
events	ev		true	Event
limitranges	limits		true	LimitRange
namespaces	ns		false	Namespace
nodes	no		false	Node
persistentvolumeclaims	pvc		true	
PersistentVolumeClaim				
persistentvolumes	pv		false	PersistentVolume
Pods	po		true	Pod
podtemplates			true	PodTemplate
.../...				
roles		rbac.authorization.k8s.io	true	Role
priorityclasses	pc	scheduling.k8s.io	false	PriorityClass
csidrivers		storage.k8s.io	false	CSIDriver
csinodes		storage.k8s.io	false	CSINode
storageclasses	sc	storage.k8s.io	false	StorageClass
volumeattachments		storage.k8s.io	false	VolumeAttachment

Ressources Documentation

La commande `kubectl explain` permet d'avoir la documentation des ressources

```
$ kubectl explain pod.spec.affinity.podAffinity
```

```
KIND:      Pod
```

```
VERSION:   v1
```

```
RESOURCE: podAffinity <Object>
```

```
DESCRIPTION:
```

```
Describes pod affinity scheduling rules (e.g. co-locate this pod in the
same node, zone, etc. as some other pod(s)).
```

```
Pod affinity is a group of inter pod affinity scheduling rules.
```

```
FIELDS:
```

```
preferredDuringSchedulingIgnoredDuringExecution    <[]Object>
```

```
The scheduler will prefer to schedule pods to nodes that satisfy the
affinity expressions specified by this field, but it may choose a node that
violates one or more of the expressions. The node that is most preferred is
the one with the greatest sum of weights, i.e. for each node that meets all
of the scheduling requirements (resource request, requiredDuringScheduling
affinity expressions, etc.), compute a sum by iterating through ...
```

```
requiredDuringSchedulingIgnoredDuringExecution    <[]Object>
```

```
If the affinity requirements specified by this field are not met at
scheduling time, the pod will not be scheduled onto the node. If the
affinity requirements specified by this field cea
```

Mode Dry Run

La commande `kubectl -dry-run` permet de simuler une commande sans modifications

```
$ kubectl run --image=nginx web1 --restart=Never --dry-run -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: web1
  name: web1
spec:
  containers:
  - image: nginx
    name: web1
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
```

Format JSON

Template JSONPATH <https://kubernetes.io/docs/reference/kubectl/jsonpath>

```
$ kubectl get pods -o json
$ kubectl get pods -o yaml

$ kubectl get pods -o jsonpath='{.items[0]}'
$ kubectl get pods -o=jsonpath='{.items[0].metadata.name}'

$ kubectl get po/web1 -o jsonpath='{.spec}'

$ kubectl get pods -o=jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.status.startTime}{"\n"}{end} '
web1      2019-09-22T13:46:19Z
web2      2019-09-22T13:46:19Z
web3      2019-09-22T13:46:19Z
```

6.

Service

Service

Ensemble de Pods

Regroupés à partir de Labels

Instances

Load-Balancing

Adresse IP

Types de Service

ClusterIP

- Expose à l'intérieur du Cluster
- Accès via le nom du Service

NodePort

- Expose à l'extérieur du cluster
- Accès via des Ports sur les hosts du Cluster

LoadBalancer

- Ingress Controlrer
- Spécifique aux Cloud Provider

ExternalName

- Alias vers un Service externe au Cluster

Types de Service

ClusterIP

- Expose à l'intérieur du Cluster
- Accès via le nom du Service

Algorithme Random

Session Affinity

- `None`: par défaut, pas d'affinité entre le client et le serveur (pod)
- `ClientIP`: affinité entre le client et le serveur

Spécification du Service

```
$ more web-service.yml
apiVersion: v1  ← version de l' API
kind: Service ← type d'objet
metadata:
  name: web-service ← nom de l'objet
spec:
  selector:      ← Selecteur de groupe de Pods
    app: web     ← Clé: Valeur du selecteur
  type: ClusterIP ← type du service
  ports:
    - port: 80   ← port exposé dans le cluster
      targetPort: 80 ← port cible d'un Pod du groupe
```

Spécification du Pod à regrouper

```
$ more web.yml
apiVersion: v1
kind: Pod
metadata:
  name: www
  labels:
    app: web
spec:
  containers:
  - name: www
    image: nginx:1.17
```

← Label de regroupement

NodePort

Spécification du Service

```
$ more web-service.yml
apiVersion: v1
kind: Service
metadata:
  name: web-service
spec:
  selector:
    app: web
  type: NodePort
  ports:
    - port: 80          ← port exposé dans le cluster
      targetPort: 80    ← port cible d'un Pod du groupe
      nodePort: 30000   ← Service accessible à l'extérieur du cluster
```

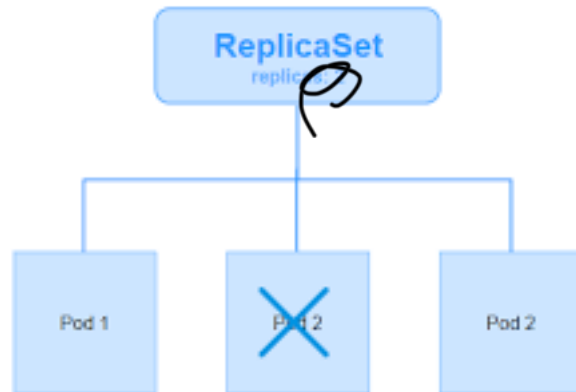
7.

Deployment

ReplicaSet/ReplicationController

ReplicaSet

- gère des Pods de même spécification
- s'assure de l'état et du nombre souhaité de replicas
- Load Balancing
- Scaling



ReplicationController

```
$ cat myapp-rc.yml
apiVersion: v1
kind: ReplicationController
metadata:
  name: myapp
spec:
  replicas: 3
  template:
    metadata:
      name: myapp
      labels:
        app: myapp
    spec:
      containers:
        - name: myapp
          image: nginx:1.21-alpine
```

← 3 pods désirés

← spécification des Pods

← labels des Pods et Selector par défaut

ReplicationController

```
$ kubectl apply -f myapp-rc.yml
replicationcontroller/myapp-rc created
```

```
$ kubectl get all -o wide --show-labels
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS
GATES LABELS									
pod/myapp-66sgm	1/1	Running	0	31s	192.168.0.83	host12	<none>		<none>
app=myapp									
pod/myapp-9fnp7	1/1	Running	0	31s	192.168.0.82	host12	<none>		<none>
app=myapp									
pod/myapp-hmjz7	1/1	Running	0	31s	192.168.0.151	host13	<none>		<none>
app=myapp									

NAME	DESIRED	CURRENT	READY	AGE	CONTAINERS	IMAGES	SELECTOR
LABELS							
replicationcontroller/myapp	3	3	3	31s	myapp	nginx:1.21-alpine	app=myapp
app=myapp							

ReplicaSet

```
$ cat myapp-rs.yml
```

```
apiVersion: apps/v1
```

```
kind: ReplicaSet
```

```
metadata:
```

```
  name: myapp
```

```
spec:
```

```
  replicas: 3
```

```
  selector:
```

```
    app: myapp
```

```
  template:
```

```
    metadata:
```

```
      name: myapp
```

```
      labels:
```

```
        app: myapp
```

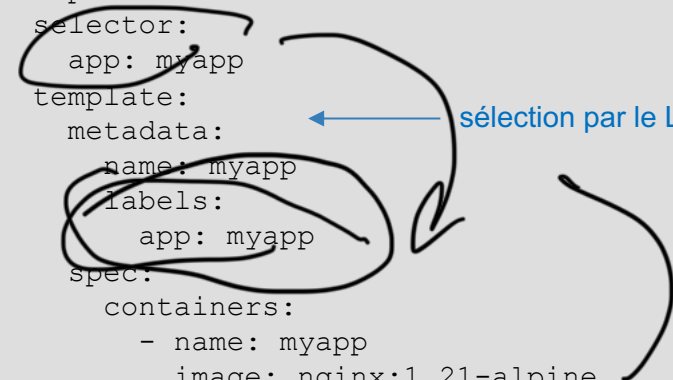
```
    spec:
```

```
      containers:
```

```
        - name: myapp
```

```
          image: nginx:1.21-alpine
```

← sélection par le Label des Pods gérés par le ReplicaSet



ReplicaSet

```
$ kubectl apply -f myapp-rs.yml
replicaset.apps/myapp created
```

```
$ kubectl get all -o wide --show-labels
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS
GATES LABELS									
pod/myapp-hp4cr	1/1	Running	0	3s	192.168.0.157	host13	<none>		<none>
app=myapp									
pod/myapp-mrspn	1/1	Running	0	3s	192.168.0.89	host12	<none>		<none>
app=myapp									
pod/myapp-xjmtt	1/1	Running	0	3s	192.168.0.88	host12	<none>		<none>
app=myapp									

NAME	DESIRED	CURRENT	READY	AGE	CONTAINERS	IMAGES	SELECTOR
LABELS							
replicaset.apps/myapp	3	3	3	3s	myapp	nginx:1.21-alpine	app=myapp
<none>							

ReplicationController vs ReplicaSet

ReplicationController

- Première forme de réplication
- Remplacé par ReplicaSet
- Selector optionnel (identique à Labels template par défaut)

FIELD: selector <map[string]string>

ReplicaSet

- Nouvelle forme de réplication
- Selector obligatoire

FIELDS:

matchExpressions <[]Object>

matchLabels <map[string]string>

ReplicaSet Selector

```
$ cat myapp-rs.yml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp
spec:
  replicas: 3
  selector:
    matchLabels:
      env: prod
      ver: "1.2"
    matchExpressions:
      - { key: tier, operator: In, values: [frontend] }
      - { key: ver, operator: In, values: ["1.2","1.3"] }
  template:
    metadata:
      name: myapp
```

Scale

Modification du fichier de description et mise à jour

```
kubectl replace -f rs-definition.yml
```

Commande scale et fichier yaml

```
kubectl scale --replicas=<new-value> -f rs-definition.yml
```

Commande scale et arguments

```
kubectl scale --replicas=<new-value> replicaset myapp-rs
```

Commande patch

```
kubectl patch rs myapp-rs -p '{"spec":{"replicas":6}}'
```

Deployment

Deployment

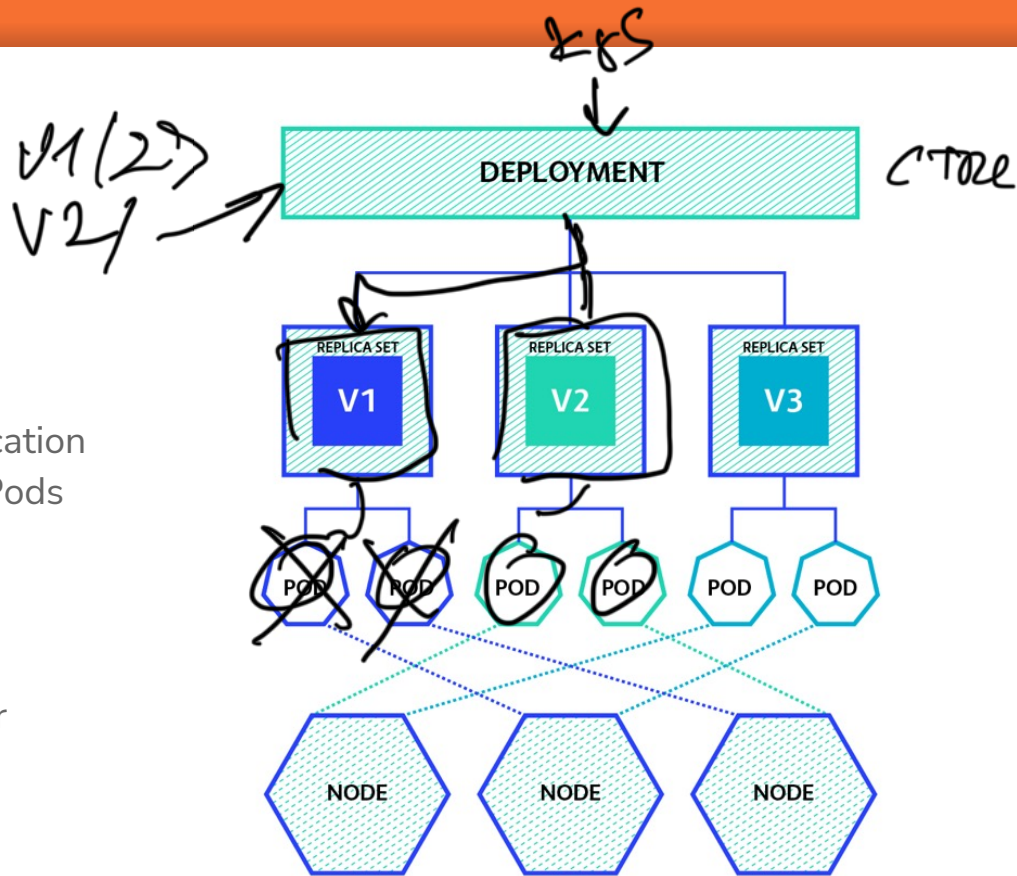
- haut niveau d'abstraction
- gère des ReplicaSet

ReplicaSet

- gère des Pods de même spécification
- s'assure de l'état souhaité des Pods

Pod instance applicative

- géré par un ReplicaSet
- schedulé sur un node du Cluster



RS(V1) 2 ✓
RS(V2) 2 ✓

Deployment

Défini un état souhaité

- suivant la spécification d'un Pod
- du nombre de répliquas désiré

Associe un ReplicatSet ou Contrôleur pour s'assurer de l'état désiré

Gère les mise à jour des applications

- rolling update
- rollback
- scaling

Deployment

```
$ cat web-deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-deployment
  labels:
    app: web
spec:
  replicas: 3      ← 3 pods désirés
  selector:
    matchLabels:  ← Selection par le Label des Pods gérés par le ReplicaSet
      app: web
  template:       ← spécification des Pods
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: nginx
          image: nginx:1.17
          ports:
            - containerPort: 80
```

Deployment

```
$ kubectl apply -f web-deployment.yml
deployment.apps/web-deployment created
```

```
$ kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
web-deployment	3/3	3	3	15s

```
$ kubectl get replicaset
```

NAME	DESIRED	CURRENT	READY	AGE
web-deployment-67c68c959f	3	3	3	21s

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
web-deployment-67c68c959f-bgm6p	1/1	Running	0	29s
web-deployment-67c68c959f-rgfdn	1/1	Running	0	29s
web-deployment-67c68c959f-w9m6r	1/1	Running	0	29s

Deployment

```
$ kubectl describe deploy/web-deployment
```

```
Name:                web-deployment
Namespace:           default
CreationTimestamp:    Sun, 22 Sep 2019 22:44:43 +0200
Labels:              app=web
Annotations:         deployment.kubernetes.io/revision: 1
                    kubectl.kubernetes.io/last-applied-configuration:

{"apiVersion":"apps/v1","kind":"Deployment","metadata":{"annotations":{},"labels":{"app":"web"},"name":"web-deployment","namespace":"defau...
Selector:            app=web
Replicas:            3 desired | 3 updated | 3 total | 3 available | 0 Pod Template:
  Labels:  app=web
  Containers:
    nginx:
      Image:      nginx:1.17
      Port:       80/TCP
NewReplicaSet:      web-deployment-67c68c959f (3/3 replicas created) ← ReplicaSet associé au Deployment
Events:
  Type     Reason             Age   From                    Message
  ----     -
  Normal   ScalingReplicaSet  11m   deployment-controller   Scaled up replica set web-deployment-67c68c959f to 3
```

Scale

Modification du fichier de description et mise à jour

```
kubectl replace -f deploy-definition.yml
```

Commande `scale` et fichier `yaml`

```
kubectl scale --replicas=<new-value> -f deploy-definition.yml
```

Commande `scale` et arguments

```
kubectl scale --replicas=<new-value> deploy myapp-deploy
```

Commande `patch`

```
kubectl patch deploy myapp-deploy -p '{"spec":{"replicas":6}}'
```

Commande `edit`

```
kubectl edit deploy myapp-deploy
```

Rolling Update

Modification du fichier de description et mise à jour

```
kubectl apply -f <deploy-manifest>
```

Modification par la commande edit

```
kubectl edit deployment <deploy>
```

Commande set

```
kubectl set image deployment <deploy> <container>=<image>
```

Commande status, suivre l'évolution de la mise à jour

```
kubectl rollout status -w deployment <deploy>
```

Historique

Historique des mises à jour

```
kubectl rollout history deployment <deploy>
```

Détails d'une version de l'historique

```
kubectl rollout history deployment <deploy> --revision=<number>
```

Rollback

Modification du fichier de description et mise à jour

```
kubectl apply -f <deploy-manifest>
```

Commande undo

```
kubectl rollout undo deployment <deploy>
```

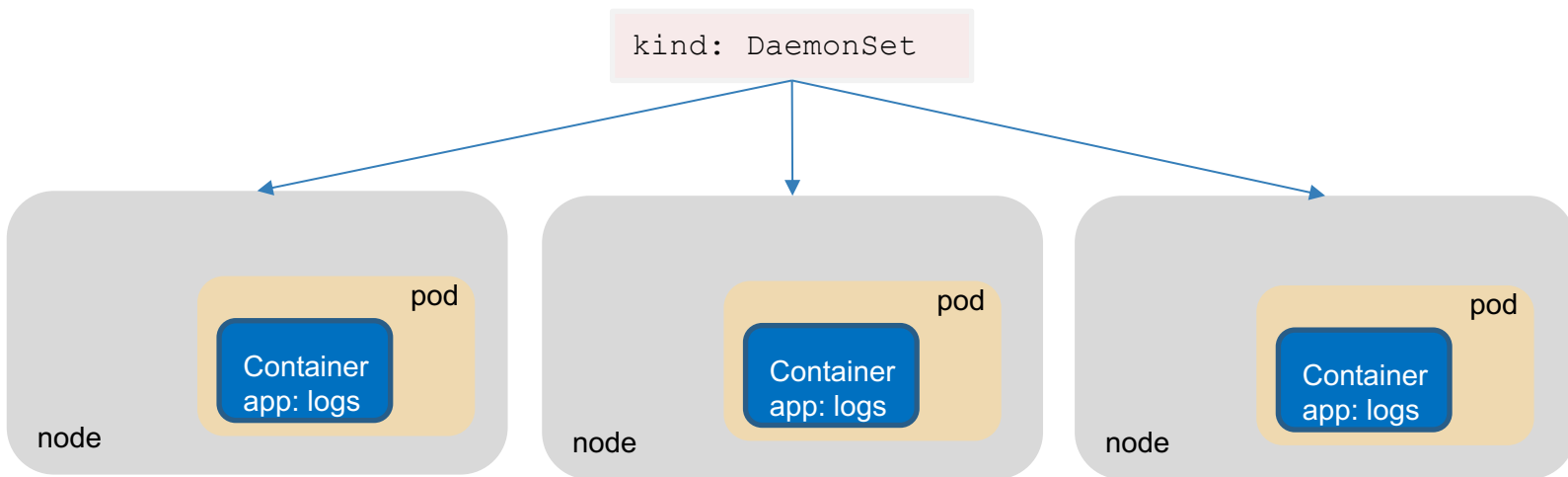
Rollback vers une version spécifique

```
kubectl rollout undo deployment <deploy> --to-revision=2
```

DaemonSet

Un DaemonSet garantit que tous les nœuds exécutent une copie du pod.

Lorsque des nœuds sont ajoutés au cluster, des pods leur sont ajoutés.

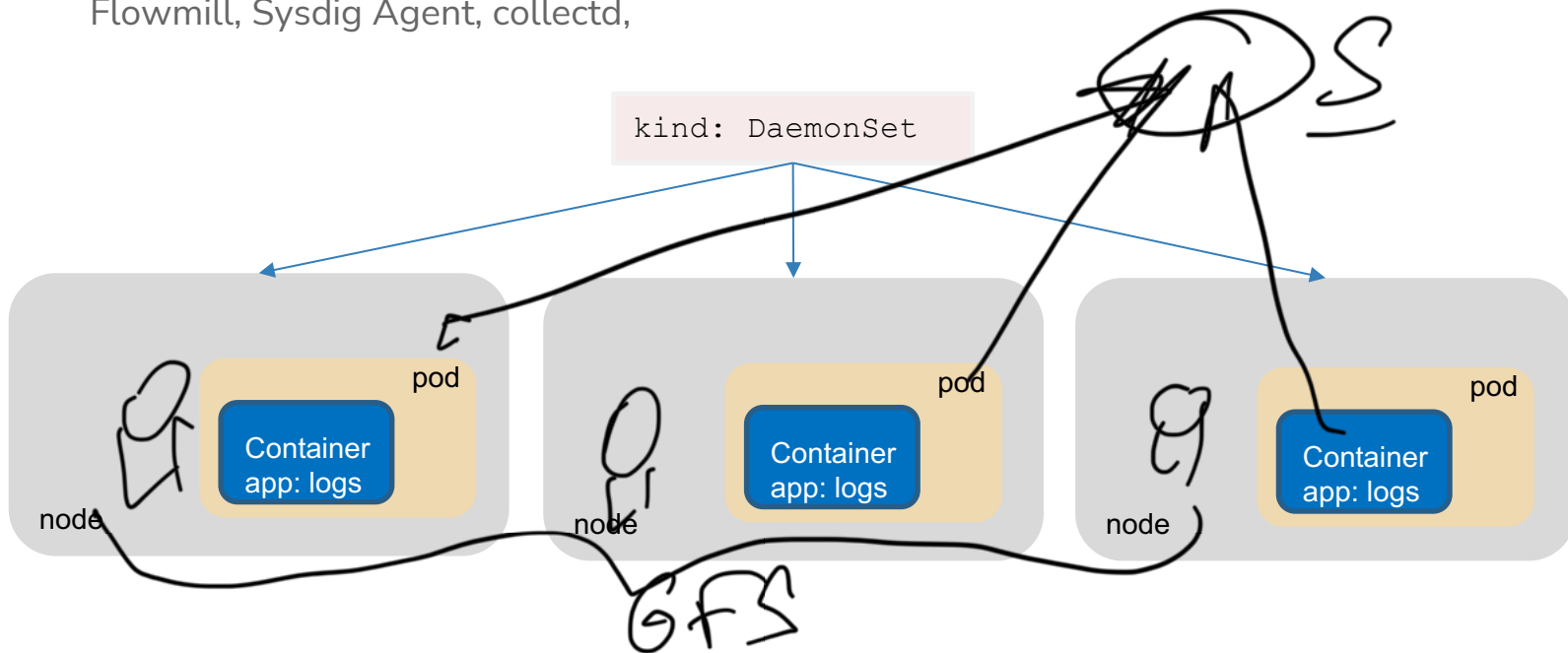


DaemonSet / Use Case

Exécuter un démon de stockage en cluster, tel que glusterd, ceph, sur chaque nœud.

Exécuter un démon de collecte de journaux sur chaque nœud, tel que fluentd ou filebeat.

Exécuter un démon de surveillance sur chaque nœud, tel que Prometheus Node Exporter, Flowmill, Sysdig Agent, collectd,



DaemonSet / Spécification

```
$ cat web-ds.yml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: web
        image: nginx:1.17-alpine
```

DaemonSet

```
$ kubectl apply -f web-daemonset.yml
```

```
daemonset.apps/web created
```

```
$ kubectl get daemonsets
```

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
web	3	3	3	3	3	<none>	2m15s

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
web-4mvs6	1/1	Running	0	3m15s
web-4nj8x	1/1	Running	0	3m15s
web-l6w6p	1/1	Running	0	3m15s

Rolling Update et Rollback

Identique au deployment

Stratégie de mise à jour

spec:

updateStrategy:

type: RollingUpdate (valeur par défaut)

- RollingUpdate : les pods sont mis à jour en même temps que le DaemonSet
- OnDelete : les pods sont mis à jour suite à la commande delete

8.

Namespace



Namespace

Isoler les ressources

- Pods
- Services
- Deployments

Segmenter un Cluster

- Departements
- Projets
- Clients

```
$ kubectl get namespaces
```

NAME	STATUS	AGE
default	Active	46h
kube-public	Active	46h
kube-system	Active	46h

Création

```
kubectl create namespace NAMESPACE
```

```
$ kubectl create namespace production  
namespace/production created
```

A partir d'un fichier de spécification

```
$ cat production.yml  
apiVersion: v1  
kind: Namespace  
metadata:  
  name: production  
  
$ kubectl apply -f production.yml  
namespace/production created
```

Désignation du Namespace 1/3

Le namespace est assigné dans les metadata du fichier de spécifications

```
$ cat web-pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: web
  namespace: production
spec:
  containers:
  - name: www
    image: nginx:1.17
```

← affectation du namespace au Pod

Création du Pod

```
$ kubectl apply -f web-pod.yml
pod/web created
```

Désignation du Namespace 2/3

Le namespace est assigné lors de la création en ligne de commande

```
$ cat web-pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: web
spec:
  containers:
  - name: www
    image: nginx:1.17
```

Création du Pod

```
$ kubectl apply -f web-pod.yml --namespace=production
pod/web created
```

Désignation du Namespace 3/3

Le namespace est assigné dans le context

```
$ kubectl config set-context minikube --namespace=production
```

```
Context "minikube" modified.
```

```
$ kubectl config view
```

```
apiVersion: v1
```

```
contexts:
```

```
- context:
```

```
  cluster: docker-desktop
```

```
  user: docker-desktop
```

```
  name: docker-desktop
```

```
- context:
```

```
  cluster: minikube
```

```
  namespace: production
```

```
  user: minikube
```

```
  name: minikube
```

```
current-context: minikube
```

← le namespace production est utilisé dans le context minikue

Ressource Quota

Permet de limiter la consommation de certaines ressources

- `limits.cpu`: CPU maximum consommée par l'ensemble des pods du namespace
- `limits.memory`: RAM maximum consommée par l'ensemble des pods
- `requests.cpu`: CPU minimum requise par l'ensemble des pods du namespace
- `requests.memory`: RAM minimum requise par l'ensemble des pods du namespace

Ressource Quota

Créer une ressource Quota

```
$ cat rq-dev.yml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

Ressource Quota

Appliquer les quotas à un namespace

```
$ kubectl apply -f rq-dev.yml --namespace development
resourcequota/dev-quota created
```

Appliquer les quotas à un namespace

```
$ kubectl get resourcequota --namespace development
```

NAME	AGE	REQUEST	LIMIT
dev-quota	57s	requests.cpu: 0/1, requests.memory: 0/1Gi	limits.cpu: 0/2, limits.memory: 0/2Gi

9.

Volumes

Volume

Extérieur au container

Cycle de vie indépendant

Persistance des données

Partage de volume entre containers d'un même Pod

Plusieurs types disponibles

- emptyDir
- configMap
- iscsi
- nfs
- cephfs

Cloud
SAN
N/A

EmptyDir

Lié au cycle de vie d'un Pod

Vide à la création

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: mysql
```

```
spec:
```

```
  containers:
```

```
  - name: mysql
```

```
    image: mysql:7.5
```

```
    volumeMounts:
```

```
    - name: data
```

```
      mountPath: /var/lib/mysql
```

```
  volumes:
```

```
  - name: data
```

```
    emptydir:
```

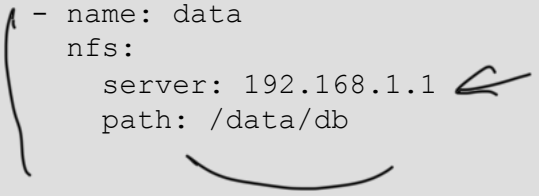
tmpfs

Monter un répertoire ou un fichier du hôte

```
apiVersion: v1
kind: Pod
metadata:
  name: mysql
spec:
  containers:
  - name: mysql
    image: mysql:7.5
    volumeMounts:
    - name: data
      mountPath: /var/lib/mysql
  volumes:
  - name: data
    hostPath:
      path: /data/db
```

Monter un partage NFS

```
apiVersion: v1
kind: Pod
metadata:
  name: mysql
spec:
  containers:
  - name: mysql
    image: mysql:7.5
    volumeMounts:
    - name: data
      mountPath: /var/lib/mysql
  volumes:
  - name: data
    nfs:
      server: 192.168.1.1
      path: /data/db
```



Fibre Channel

Volume de type SAN

apiVersion: v1

kind: Pod

metadata:

name: mysql

spec:

containers:

- name: mysql

image: mysql:7.5

volumeMounts:

- name: data

mountPath: /var/lib/mysql

volumes:

- name: data

fc:

targetWWNs:

- XXXXXXXXXX

lun: 0

fstype: ext4

mount

FS

Volume de type iSCSI

```
apiVersion: v1
kind: Pod
metadata:
  name: mysql
spec:
  containers:
  - name: mysql
    image: mysql:7.5
    volumeMounts:
    - name: data
      mountPath: /var/lib/mysql
  volumes:
  - name: data
    iscsi:
      iqn: iqn.2001-08.com.netapp.xxxx
      lun: 0
      fstype: ext4
```



Persistent Volume

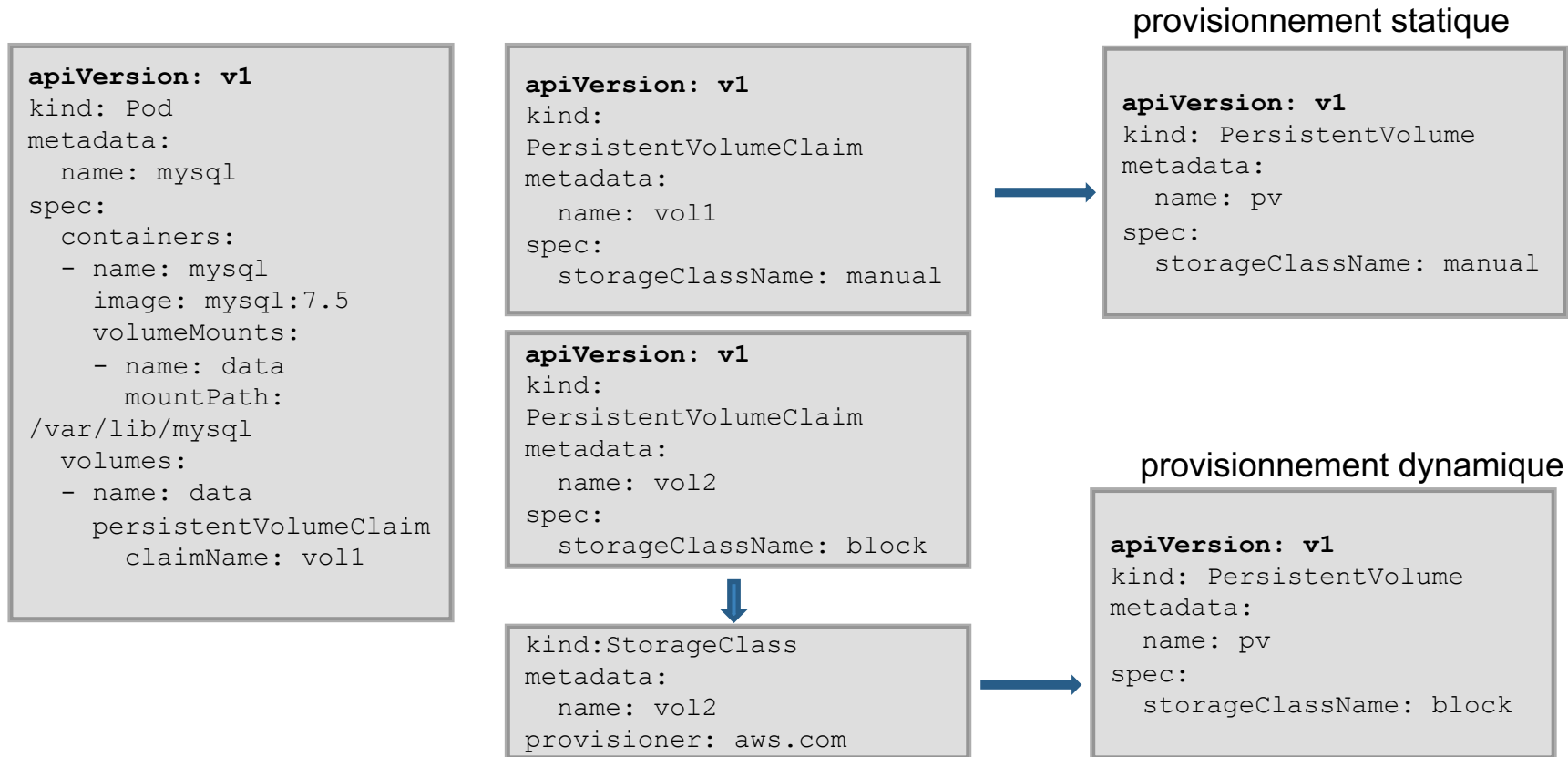
Persistent Volume

- Stockage provisionné
 - statique
 - dynamique

Persistent Volume Claim

- Demande de Stockage
- Spécifie des Contraintes
 - taille, type ...
- Consomme un PV existant ou provisionnement PV par StorageClass
- Référencé dans un Pod

Persistent Volume



Exemple statique 1/3

```
$ cat pv.yml
```

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv
spec:
  storageClassName: manual
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/pv
```

```
$ kubectl apply -f pv.yml
```

```
persistentvolume/pv created
```

```
$ kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv	5Gi	RWO	Retain	Available		manual		8s

RWO 1 RW
+ RO

RWM + RW

ROM + RO

Exemple statique 2/3

```
$ cat pvc.yml
```

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
```

```
metadata:
```

```
  name: claim
```

```
spec:
```

```
  storageClassName: manual
```

```
  accessModes:
```

```
    - ReadWriteOnce
```

```
  resources:
```

```
    requests:
```

```
      storage: 5Gi
```

```
$ kubectl apply -f pvc.yml
```

```
persistentvolumeclaim/claim created
```

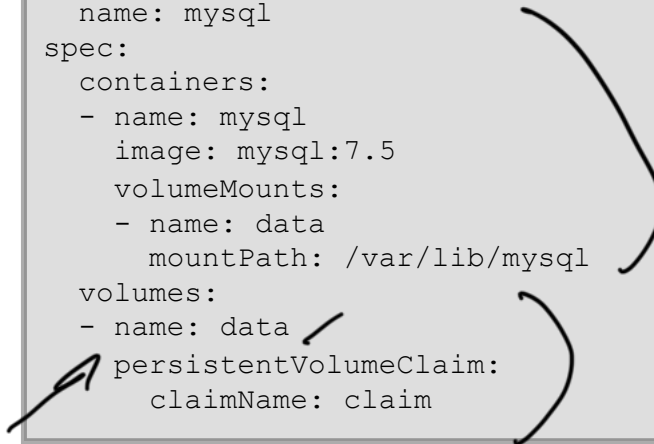
```
$ kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
claim	Bound	pv	5Gi	RWO	manual	6s

PVC → PV

Exemple statique 3/3

```
$ cat pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: mysql
spec:
  containers:
  - name: mysql
    image: mysql:7.5
    volumeMounts:
    - name: data
      mountPath: /var/lib/mysql
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: claim
```



10.

ConfigMap

ConfigMap

Permet de détacher l'application de sa configuration

Portabilité

Créée à partir

- d'un fichier
- d'un répertoire
- de valeurs littérales

Constituée de

- une ou plusieurs clé=valeur

A partir d'un fichier

```
$ cat nginx.conf
server {
    location / {
        root /data/www;
    }
    location /images/ {
        root /data;
    }
}
```

A partir d'un fichier de configuration

```
$ kubectl create configmap nginx-conf --from-file=nginx.conf
configmap/nginx-conf created
```

```
$ kubectl get configmap nginx-conf -o yaml
```

```
apiVersion: v1
data:
  nginx.conf: |
    server {
      location / {
        root /data/www;
      }
      location /images/ {
        root /data;
      }
    }
kind: ConfigMap
metadata:
  creationTimestamp: "2019-10-26T09:48:39Z"
  name: nginx-conf
  namespace: default
  selfLink: /api/v1/namespaces/default/configmaps/nginx-conf
```

A partir d'un fichier de variables d'environnement

```
$ cat conf.env
env=prod
log_level=debug

$ kubectl create cm config-env --from-env-file=./conf.env
configmap/config-env created

$ kubectl get cm config-env -o yaml
apiVersion: v1
data:
  env: prod
  log_level: debug
kind: ConfigMap
metadata:
  creationTimestamp: "2019-09-23T20:43:35Z"
  name: config-env
  namespace: production
  resourceVersion: "226966"
  selfLink: /api/v1/namespaces/production/configmaps/config-env
  uid: 175066c8-ce16-41e5-9486-6e860f31e341
```

A partir d'un fichier de valeurs littérales

```
$ kubectl create cm config-lit --from-literal=env=prod --from-literal=log_level=debug
configmap/config-lit created

$ kubectl get cm config-lit -o yaml
apiVersion: v1
data:
  env: prod
  log_level: debug
kind: ConfigMap
metadata:
  creationTimestamp: "2019-09-23T20:43:35Z"
  name: config-lit
  namespace: production
  resourceVersion: "226966"
  selfLink: /api/v1/namespaces/production/configmaps/config-env
  uid: 175066c8-ce16-41e5-9486-6e860f31e341
```

Utilisation dans un Pod

A partir d'un volume

```
$ cat cm-volume.yml
```

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: web
```

```
spec:
```

```
  containers:
```

```
  - name: proxy
```

```
    image: nginx:1.17
```

```
    ports:
```

```
    - containerPort: 8000
```

```
    volumeMounts:
```

```
    - name: config
```

```
      mountPath: "/etc/nginx/nginx.conf"
```

```
      subPath: "nginx.conf"
```

```
volumes:
```

```
  - name: config
```

```
    configMap:
```

```
      name: nginx-config
```

← Montage du volume dans le container

Définition d'un volume basé sur la ConfigMap nginx-config

Lancement du Pod

```
$ kubectl create -f cm-volume.yml
pod/web created

$ kubectl exec -it web -- bash
root@web:/# cat /etc/nginx/nginx.conf
user nginx;
worker_processes 4;
pid /run/nginx.pid;
events {
    worker_connections 768;
}
http {
    server {
        listen *:8000;
        location = / {
            proxy_pass http://localhost;
        }
    }
}
```

Utilisation dans un Pod

Variable d'environnement

```
$ cat cm-env.yml
apiVersion: v1
kind: Pod
metadata:
  name: web
spec:
  containers:
  - name: www
    image: nginx:1.17
    env:
    - name: LOG_LEVEL
      valueFrom:
        configMapKeyRef:
          name: config-env
          key: log_level
    - name: SITE
      valueFrom:
        configMapKeyRef:
          name: config-lit
          key: env
```

Lancement du Pod

```
$ kubectl create -f cm-env.yml
pod/web created

$ kubectl exec -it web -- bash
root@web:/# env
KUBERNETES_SERVICE_PORT=443
LOG_LEVEL=DEBUG
HOSTNAME=web
SITE=PROD
NJS_VERSION=0.3.5
PROXY_PORT=tcp://10.98.171.93:80
PROXY_PORT_80_TCP=tcp://10.98.171.93:80
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.96.0.1
KUBERNETES_SERVICE_HOST=10.96.0.1
KUBERNETES_PORT=tcp://10.96.0.1:443
KUBERNETES_PORT_443_TCP_PORT=443
NGINX_VERSION=1.17.3
_=/usr/bin/env
```

11.

Secrets

Protection des données

- mots de passe
- clés privées

Sortir les informations sensibles des images ou des fichiers de spécification

Plus de contrôle de leur utilisation

Objet créé par l'utilisateur ou le système

Utilisé dans un Pod

Stocké dans etcd

Types

Generic

fichier

répertoire

valeur littérale

Docker-registry

- Authentification à Registre Docker

TLS

- Gestion de clés

Création à partir de fichier ou littéral

```
$ cat user.txt
```

```
admin
```

```
$ cat pass.txt
```

```
T6f45e3ab
```

```
$ kubectl create secret generic secret1 --from-file=user.txt --from-file=pass.txt
```

```
secret/secret1 created
```

```
$ kubectl create secret generic secret2 --from-literal=username=admin --from-literal=password=T6f45e3a
```

```
secret/secret2 created
```

```
$ kubectl get secrets
```

NAME	TYPE	DATA	AGE
default-token-9jsvl	kubernetes.io/service-account-token	3	23h
secret1	Opaque	2	3m21s
secret2	Opaque	2	8s

Création manuelle

```
$ echo -n 'admin' | base64
```

```
YWRtaW4=
```

```
$ echo -n '1f2d1e2e67df' | base64
```

```
MWYyZDFIMmU2N2Rm
```

```
$ cat secret.yml
```

```
apiVersion: v1
```

```
kind: Secret
```

```
metadata:
```

```
  name: mysecret
```

```
type: Opaque
```

```
data:
```

```
  username: YWRtaW4=
```

```
  password: MWYyZDFIMmU2N2Rm
```

```
$ kubectl apply -f secret.yml
```

```
secret "myscret" created
```

Utilisation comme Variable d'environnement

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-env-pod
spec:
  containers:
  - name: mycontainer
    image: redis
    env:
      - name: SECRET_USERNAME
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: username
      - name: SECRET_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mysecret
            key: password
```

Utilisation comme Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: mysecret
```

12.

Jobs



Ressource utilisée pour lancer des Jobs

Lance un Job

Lance plusieurs jobs en parallèle (.spec.parallelism)

Lance plusieurs jobs en séquentiel (.spec.completions)

Example

```
apiVersion: batch/v1
kind: Job
metadata:
  name: batch
spec:
  completions: 3
  parallelism: 1
  template:
    spec:
      restartPolicy: OnFailure
      containers:
      - name: batch
        image: alpine:3.12
        command: ["echo", "Hello World"]
```

Création

Création du job

```
$ kubectl apply -f job.yaml
job.batch/countdown created
```

Lister les jobs

```
$ kubectl get jobs -w
```

NAME	COMPLETIONS	DURATION	AGE
countdown	1/1	46s	73s

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
countdown-qs2h4	0/1	Completed	0	78s

Lancement de tâches automatiques

Crontab Linux

Use Case

- Backup de database
- Rotation de fichiers logs
- Envoi d'emails

Example

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: cron-example
spec:
  schedule: "* * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
            - name: cron-example
              image: alpine:3.12
              args:
                - /bin/sh
                - -c
                - date;echo Hello there
          restartPolicy: OnFailure
```

Example

```
$ kubectl apply -f cron.yml
cronjob.batch/cron-example created
```

```
$ kubectl get cronjob
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
cron-example	*/1 * * * *	False	0	<none>	5s

```
$ kubectl get cronjob
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST SCHEDULE	AGE
cron-example	*/1 * * * *	False	0	12s	94s

```
$ kubectl get job
```

NAME	COMPLETIONS	DURATION	AGE
cron-example-1645353240	0/1	63s	63s
cron-example-1645353300	0/1	3s	3s

13.

HELM



Gestionnaire de packages

Charts

- Applications packagées
- Chart.yaml: metadatas de l'application
- Templates de spécification des ressources (Deployment, Service ...)
- Fichiers de valeurs pour la configuration de l'application

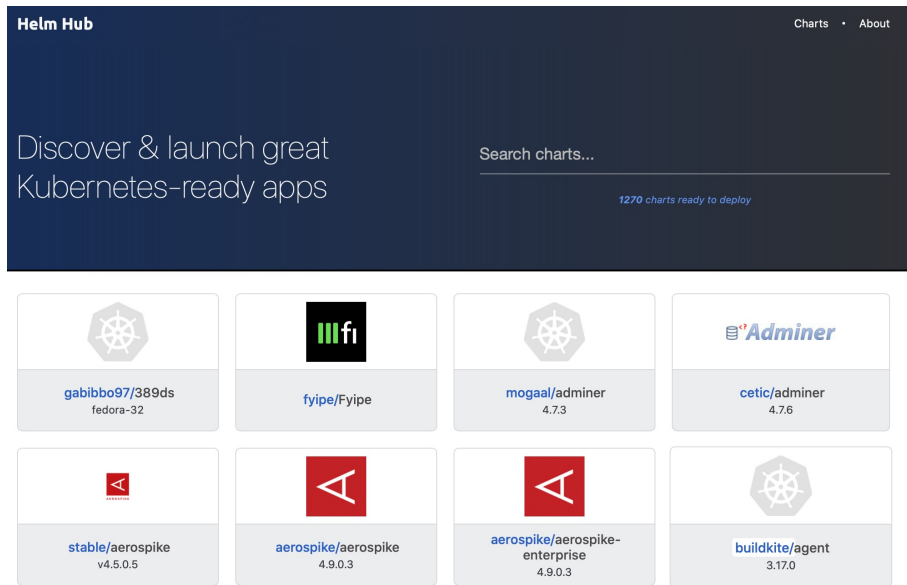
Architecture

- HELM 2 : client local + daemon sur le cluster
- HELM 3 : uniquement client local

hub.helm.sh

Charts

- Stables
- Incubator (en developpement)



Installation

installation

- <https://helm.sh/docs/intro/install/>
- <https://github.com/helm/helm/releases>

```
$ helm version  
version.BuildInfo{Version:"v3.2.1",  
GitCommit:"fe51cd1e31e6a202cba7dead9552a6d418ded79a", GitTreeState:"clean",  
GoVersion:"go1.13.10"}
```

Installer le repository officiel contenant les charts stable

```
$ helm repo add stable https://kubernetes-charts.storage.googleapis.com/  
"stable" has been added to your repositories
```

Liste des repo stable

```
$ helm search repo stable
```

NAME	CHART	VERSION	APP
stable/acs-engine-autoscaler	2.2.2	2.1.1	DEPRECATED Scales
worker nodes within agent pools			
stable/aerospike	0.3.2	v4.5.0.5	A Helm chart for
Aerospike in Kubernetes			
stable/airflow	6.10.4	1.10.4	Airflow is a platform
to programmatically autho...			
stable/ambassador	5.3.1	0.86.1	A Helm chart for
Datawire Ambassador			

Installation d'un Chart

Avec HELM 3

```
helm install <name> <chart>
```

```
$ helm install dashboard stable/kubernetes-dashboard
```

```
NAME: dashboard
```

```
LAST DEPLOYED: Wed May 13 06:19:01 2020
```

```
NAMESPACE: default
```

```
STATUS: deployed
```

```
REVISION: 1
```

```
TEST SUITE: None
```

```
NOTES:
```

```
Get the Kubernetes Dashboard URL by running:
```

```
export POD_NAME=$(kubectl get pods -n default -l "app=kubernetes-  
dashboard,release=dashboard" -o jsonpath="{.items[0].metadata.name}")
```

```
echo https://127.0.0.1:8443/
```

```
kubectl -n default port-forward $POD_NAME 8443:8443
```

Liste des Charts

Lister les Charts installés

```
$ helm list
```

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART	APP VERSION
dashboard	default	1	2020-05-13 06:19:01.46024026 +0200 CEST	deployed	kubernetes-	
dashboard-1.10.1	1.10.1					

Supprimer un Chart

```
$ helm uninstall nginx
```

```
release "nginx" uninstalled
```