



Kubernetes: les bases pour DevOps

Par Dirane TAFEN



Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- Gestion du réseau
- Gestion du stockage
- Introduction à helm
- Mini-projet



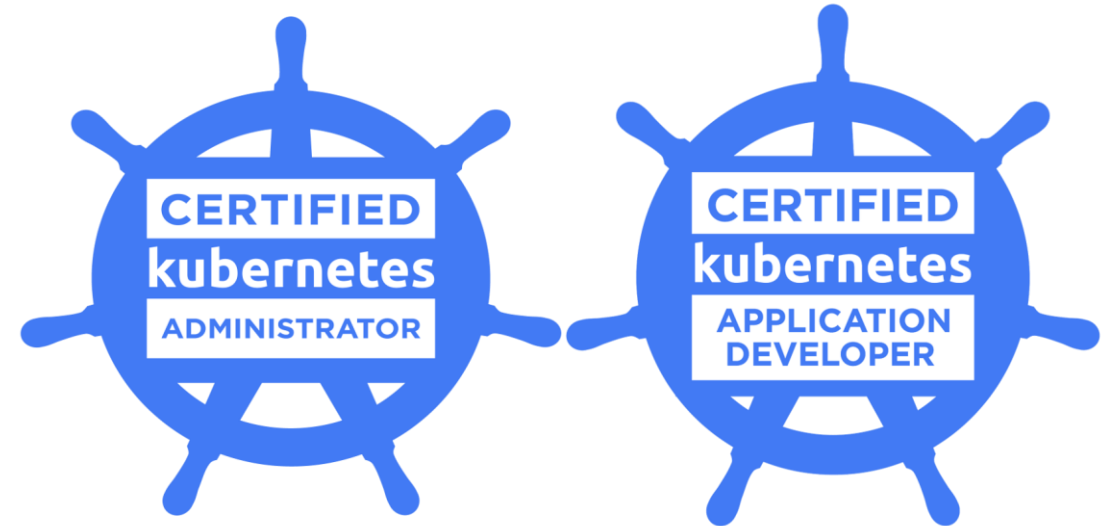
Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- Gestion du réseau
- Gestion du stockage
- Introduction à helm
- Mini-projet



Présentation du formateur

- Dirane TAFEN (formateur et consultant DevOps)
- Capgemini
- Sogeti
- ATOS
- BULL
- AIRBUS
- ENEDIS



Plan

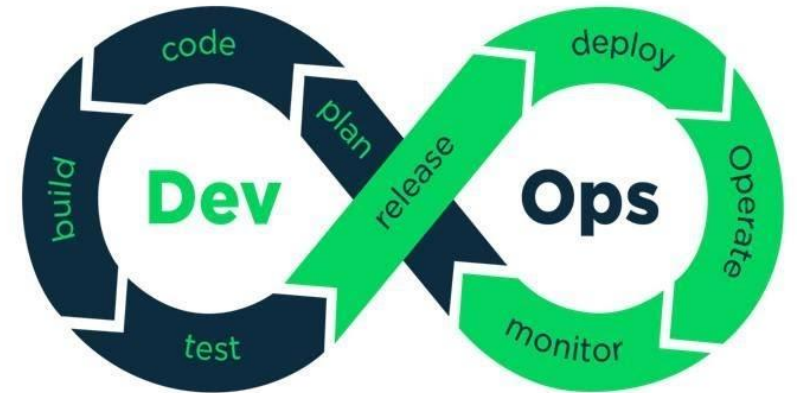
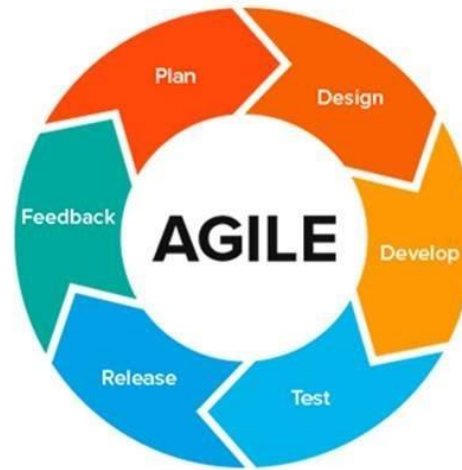
- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- Gestion du réseau
- Gestion du stockage
- Introduction à helm
- Mini-projet



Introduction au DevOps et à l'orchestration (1/5): Le DevOps

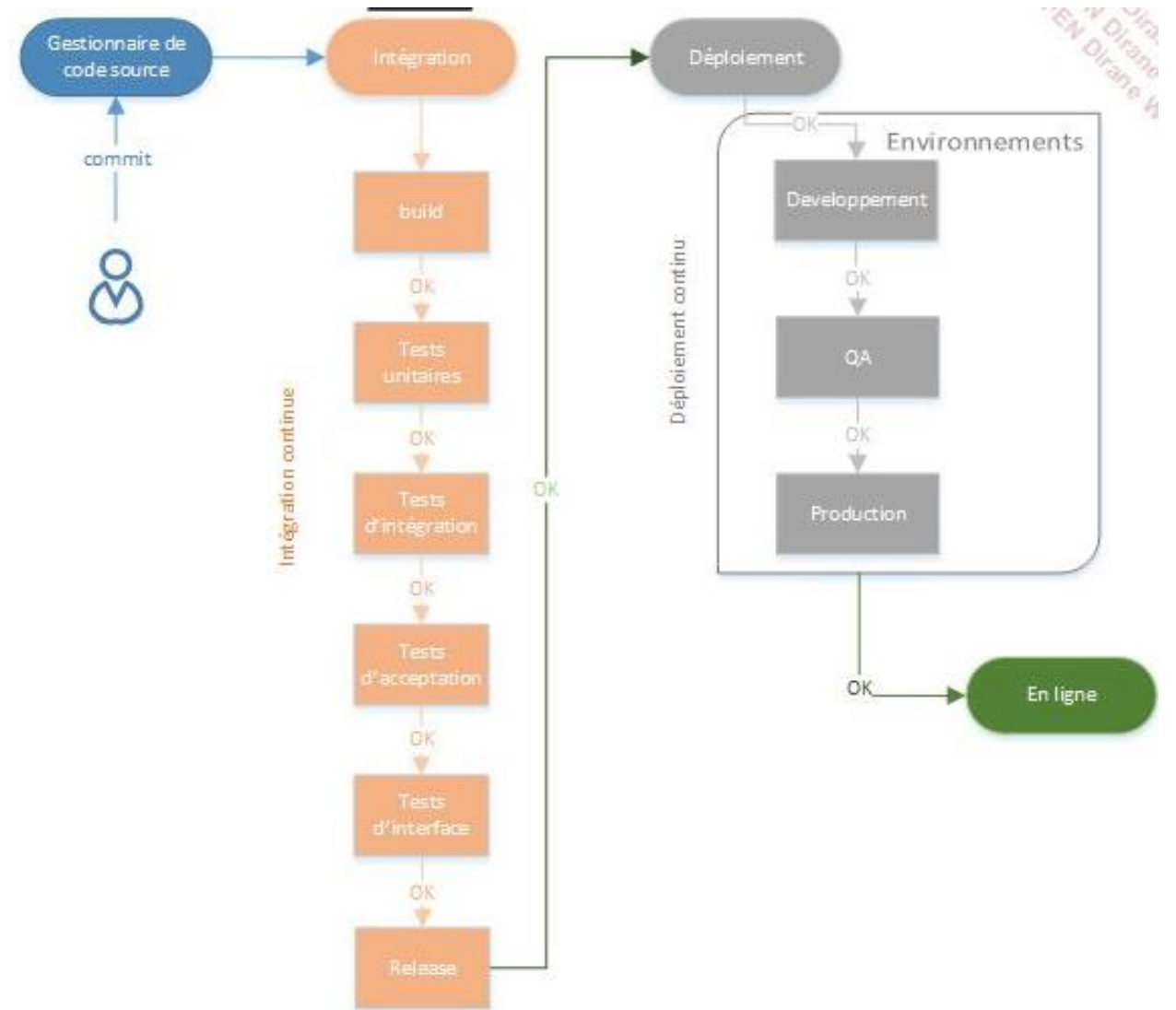
- Agile: méthode de développement
- DevOps: agilité dans le Dev et l'Ops = CI + CD

Agile vs. DevOps



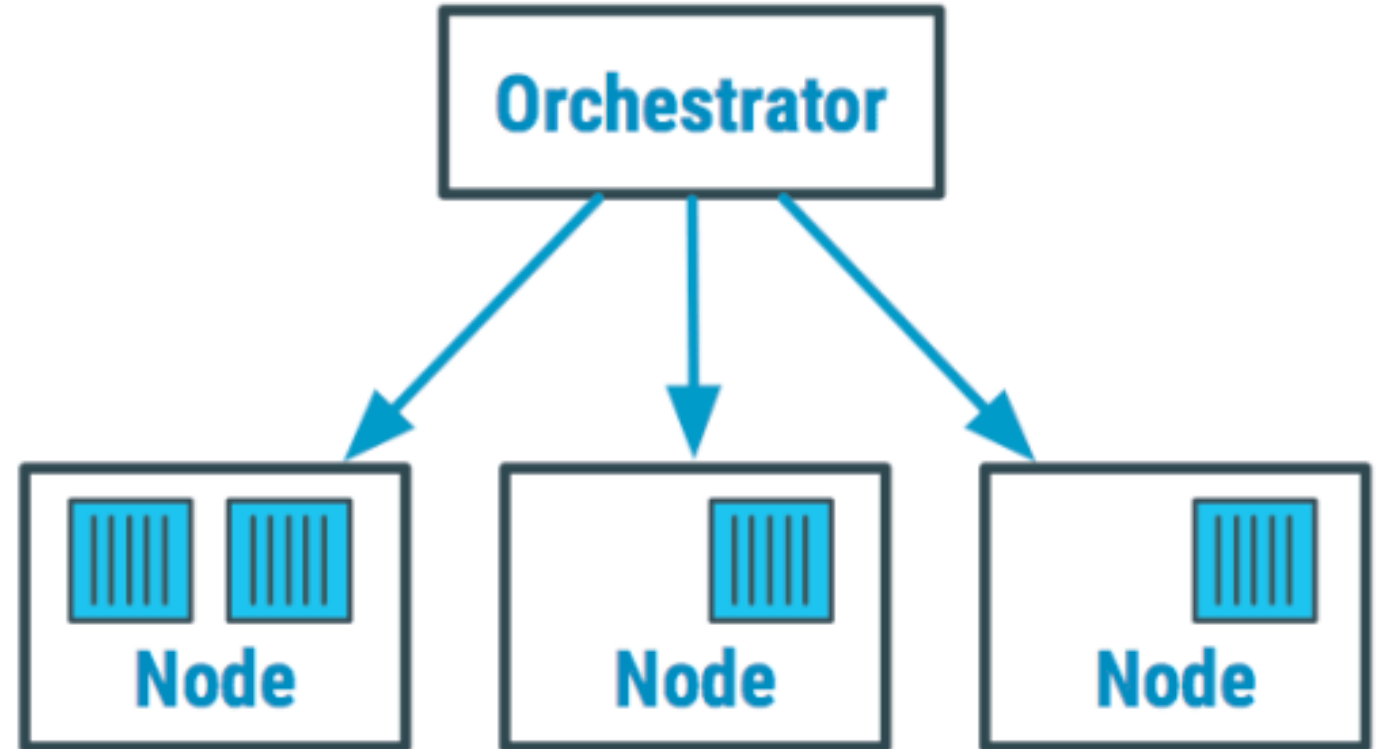
Introduction au DevOps et à l'orchestration (2/5): CI/CD

- Intégration en continu
- Test en continu
- Déploiement en continu
- Orchestration



Introduction au DevOps et à l'orchestration (3/5): Orchestrateur

- Etat des applications /conteneurs
- Auto-discovery
- Rolling Update / Blue-Green deployment / rollbacks
- Scalabilité des services
- Load balancing
- Secret Management
- Fail-over app ...

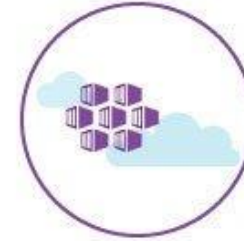


Introduction au DevOps et à l'orchestration (4/5): Solutions d'orchestration

Tools of Container Orchestration



Amazon ECS
FROM AMAZON



Azure Container Services
FROM MICROSOFT



Docker Swarm
DOCKER OPENSOURCE TOOLS



Google Container Engine
FROM GOOGLE CLOUD PLATFORM



Kubernetes
DOCKER OPENSOURCE TOOLS



CoreOS Fleet
FROM COREOS



Mesosphere Marathon
FROM MARATHON

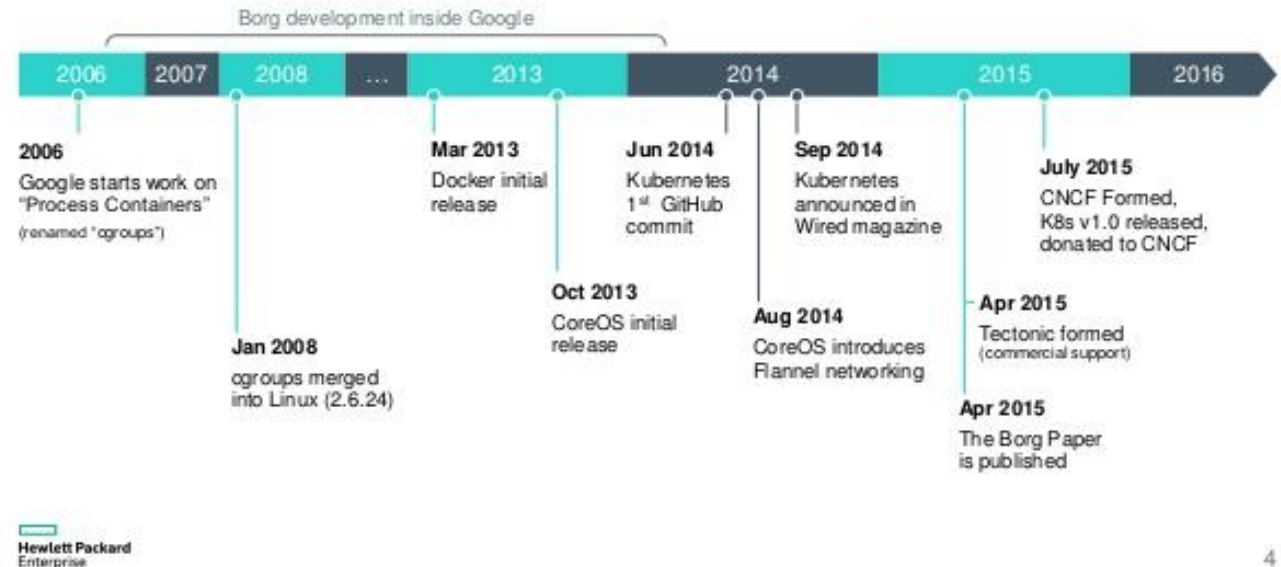


Cloud Foundry's Diego
FROM CLOUD FOUNDRY

Introduction au DevOps et à l'orchestration (5/5): Kubernetes

- Multi-Cloud (portable)
- Extensible
- GitOps
- Open-Source (CNCF)
- Certification : CKA, CKAD
- Latest release (18/06/2020):
1.18.4

Kubernetes History



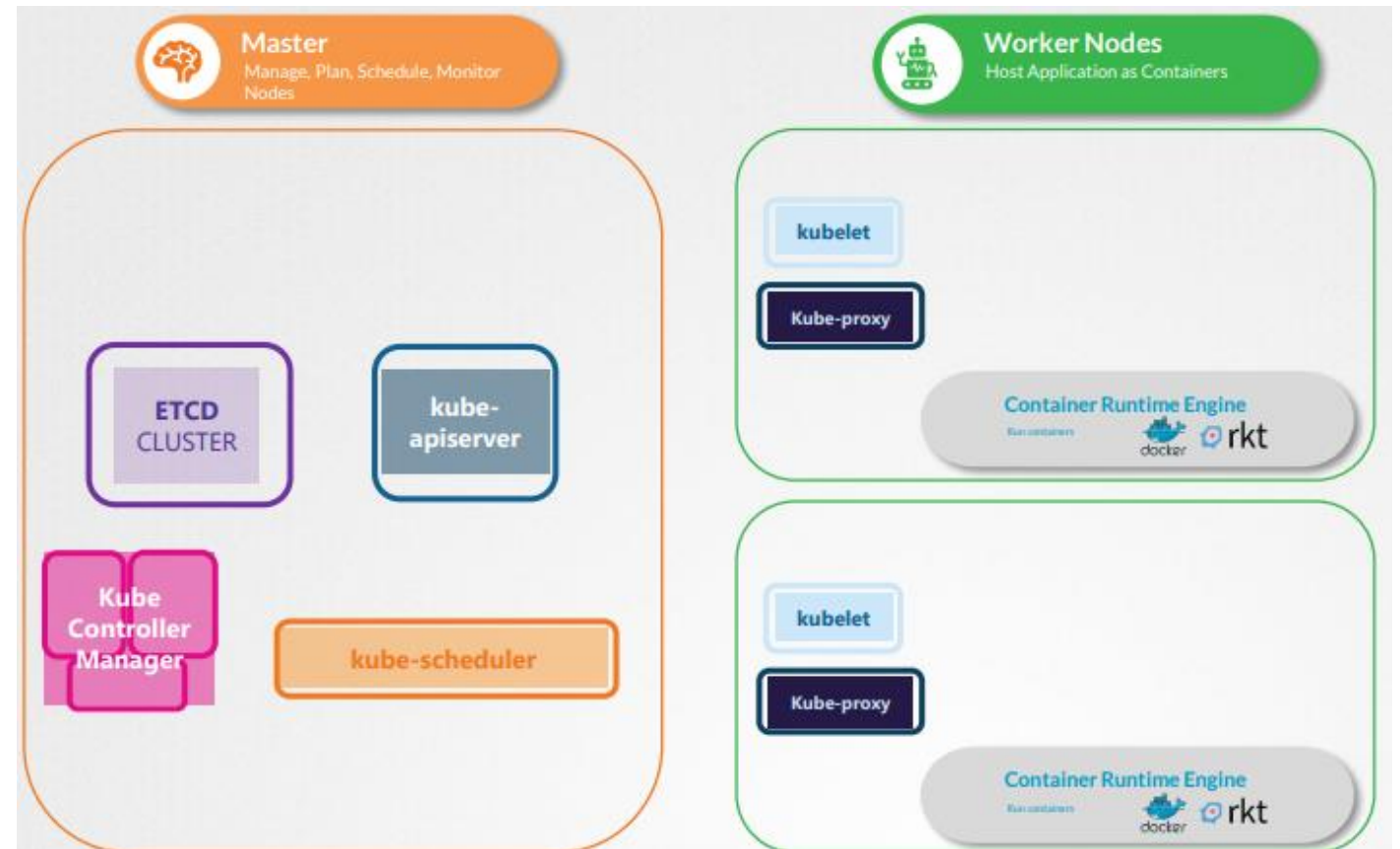
Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- Gestion du réseau
- Gestion du stockage
- Introduction à helm
- Mini-projet



Composants et Installation de Kubernetes (1/2): Composants

- Master/worker
- Docker/rkt
- Sécurité



Composants et Installation de Kubernetes (2/2): Installation

- Cloud: GCP, AKS, EKS, OVH Kubernetes ...
- Baremetal: à la main, kubespray
- Local Dev: Docker pour Windows/Mac, Minikube
- On-premise Production: Kubeadm (multi-node cluster)

TP-0: Découvrir la plateforme de TP

- Accès
- Labs
- Durée d'une session
- Données sensibles
- Agrandir la fenêtre du terminal
- Connexion ssh
- Installation de paquet
- Ouverture de port

TP-1: Installation de kubernetes

- Installation de minikube sur centos 7
- Présentation de l'environnement Kubernetes d'EAZYTraining
- Single-node

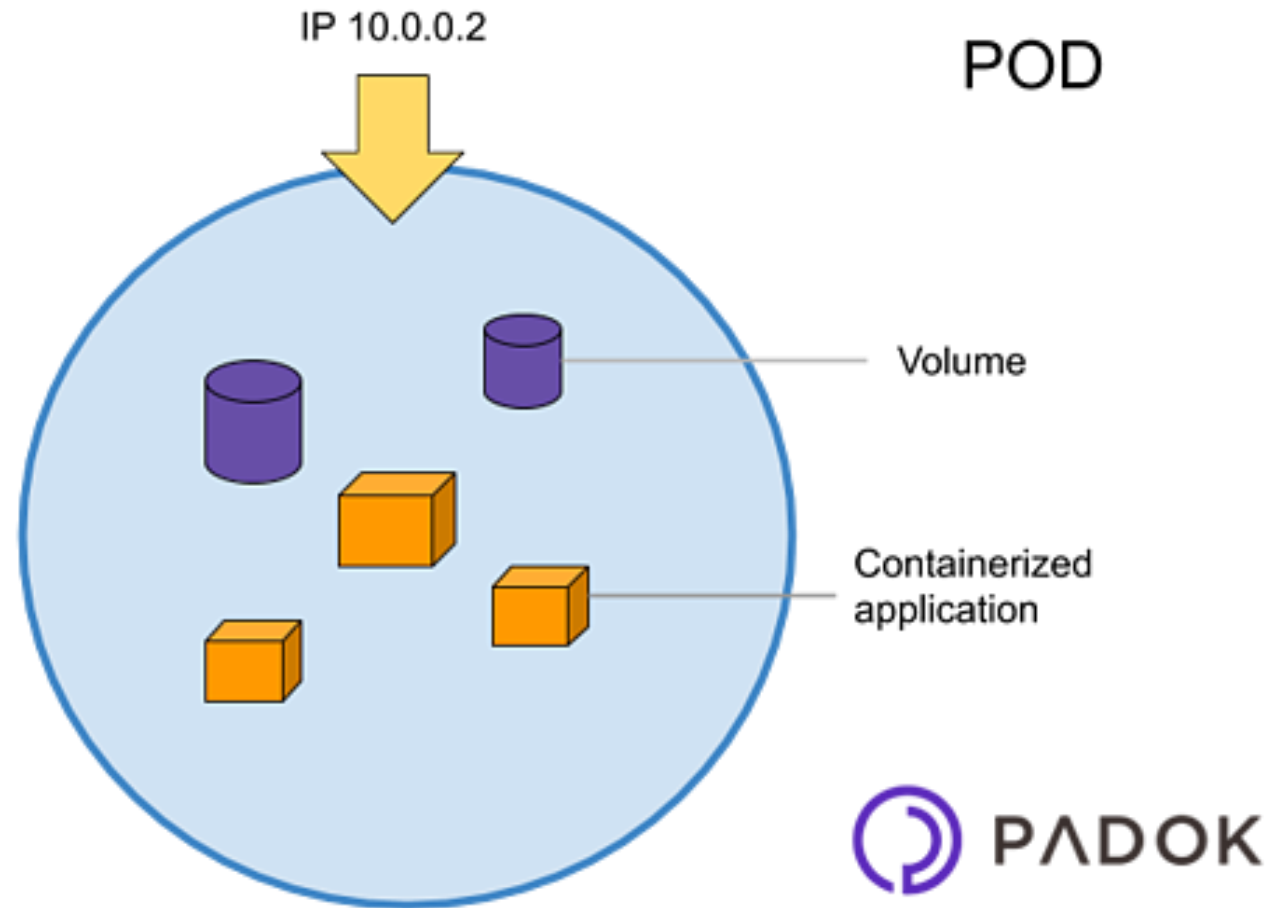
Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- [Déployez vos premières applications](#)
- Gestion du réseau
- Gestion du stockage
- Introduction à helm
- Mini-projet



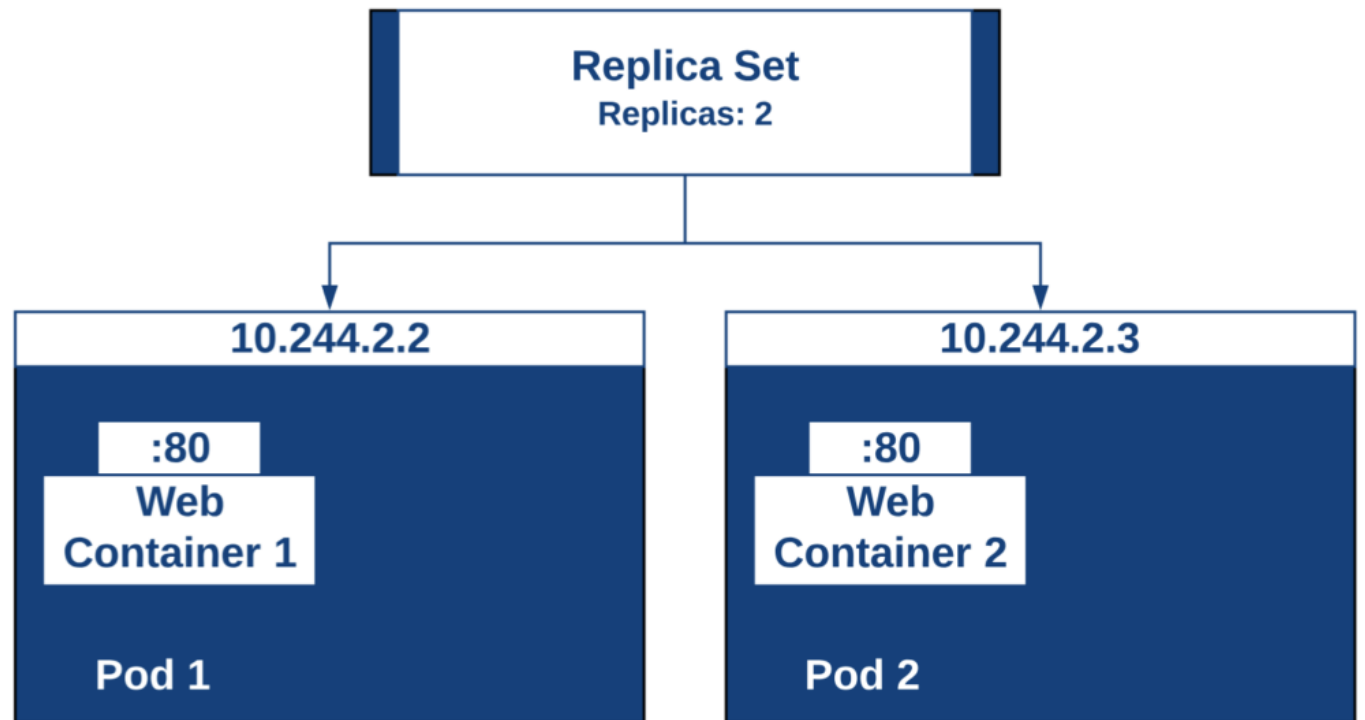
Déployez vos premières applications (1/7): Pod

- Plus petite unité d'exécution
- Espace réseau
- Espace de volume
- Données Ephémères



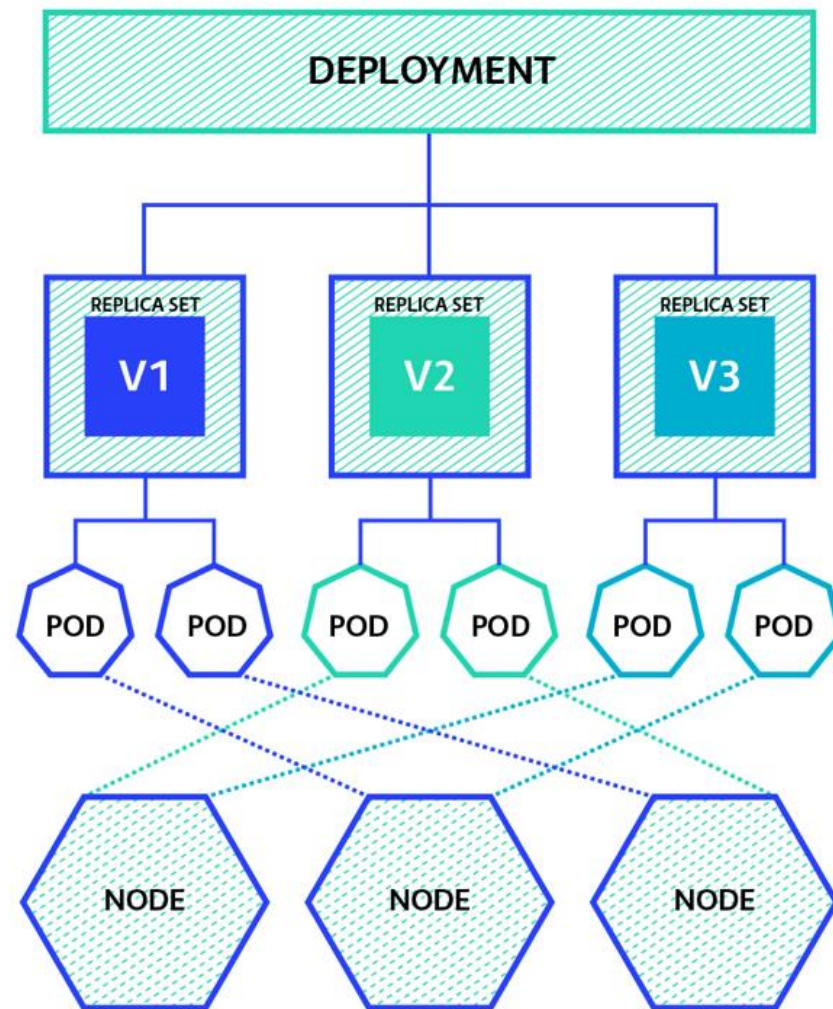
Déployez vos premières applications (2/7): Replicaset

- Scalabilité
- Résilience




Déployez vos premières applications (3/7): Deployment

- Update Software (rolling)
- Update Replicas (scaling)



Déployez vos premières applications (4/7): kubectl


- Impérative
- Déclarative



kubernetes

Cheatsheet

version 2019-03-12 EN



infraBuilder
infrastructure devops

CLI tools	<pre># CLI setup https://ibd.sh/kubectl https://ibd.sh/kubectx https://ibd.sh/kubetail https://ibd.sh/helm</pre>	deploy / expose	<pre># Create a deployment named web, using image nginx into prod namespace kubectl create -n prod deploy web --image=nginx # Expose port 80 of deployment web with an internal service named front kubectl expose deploy/web --port=80 --name=front # Retrieve logs of pods with tag app=web kubetail -l app=web # Open a tunnel listening on 127.0.0.1:8080 to port 80 of deployment web kubectl port-forward deploy/web 8080:80 # Create a Yaml manifest, without sending it to the cluster kubectl create deploy web --image=nginx --dry-run -o yaml > web.yaml # Edit deployment web kubectl edit deploy/web</pre>	
contexts	<pre># List available contexts kubectx # Change context to dev kubectx dev</pre>		help / debug	<pre># Retrieve detailed state of pod test kubectl describe pod test # Get all possible attributes of a resource kubectl explain pod --recursive # Open a bash terminal in pod app kubectl exec -it app -- bash # NB : The flag --help provide help of any command</pre>
namespaces	<pre># List namespaces kubens # Change namespace to prod kubens prod # Create namespace test kubectl create ns test</pre>	configuration		<pre># Use the config file /path/to/config rather than ~/.kube/config export KUBECONFIG=/path/to/config # Merge two configuration files config1 and config2 in one file config KUBECONFIG=config1:config2 kubectl config view --flatten > config # Export only the current context configuration to file config kubectl config view --minify --flatten > config</pre>
ingress	<pre># Ingress manifest example apiVersion: extensions/v1beta1 kind: Ingress metadata: name: test-ingress spec: rules: - host: foo.bar.com http: paths: - path: /testpath backend: serviceName: test servicePort: 80</pre>			

Command cheatsheet
for Kubernetes 1.13
by Alexis DUCASTEL
Web : infraBuilder.com

Déployez vos premières applications (5/7): Manifest Yaml

- Versionning du code
- Partage des manifest
- IaC
- `kubectl apply -f <manifest.yaml>`

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: hostname-101-deployment
spec:
  replicas: 3
  selector:
    # Like saying "Make sure there are three pods running
    # with the label app = hostname and version = v101"
    matchLabels:
      app: hostname
      version: v101
  template:
    metadata:
      labels:
        # The `app` label is used by both the service
        # and the deployment to select the pods they operate on.
        app: hostname
        # The `version` label is used only by the deployment
        # to control replication.
        version: v101
    spec:
      containers:
        - name: nginx-hostname
          image: kubegoldenguide/nginx-hostname:1.0.1
          ports:
            - containerPort: 80
```

Déployez vos premières applications (6/7): Variable d'environnement

```
▶ docker run -e APP_COLOR=pink simple-webapp-color
```

pod-definition.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: simple-webapp-color
spec:
  containers:
  - name: simple-webapp-color
    image: simple-webapp-color
    ports:
      - containerPort: 8080
    env:
      - name: APP_COLOR
        value: pink
```

Déployez vos premières applications (7/7): Type de Variable d'environnement

env:

- **name:** APP_COLOR
value: pink

1 Plain Key Value

env:

- **name:** APP_COLOR
valueFrom:
configMapKeyRef:

2 ConfigMap

env:

- **name:** APP_COLOR
valueFrom:
secretKeyRef:

3 Secrets

TP-2: Déployez votre première application

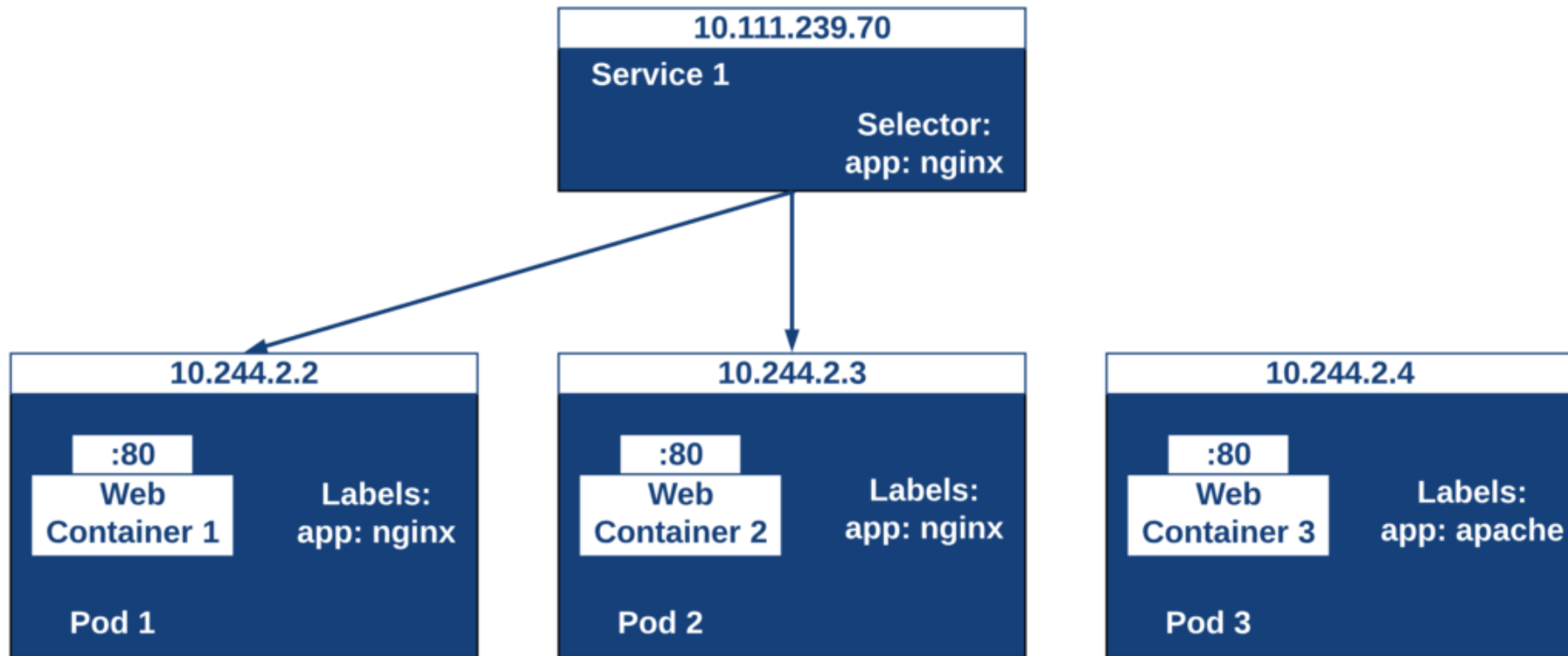
- Ecrivez un manifest `pod.yml` pour déployer un pod avec l'image `mmumshad/simple-webapp-color` en précisant que la color souhaitée est la rouge
- Lancez votre pod et vérifiez qu'il est bien en cours d'exécution
- Exposez votre pod en utilisant la commande `kubectl port-forward <nom de votre pod> 8080:8080 --address 0.0.0.0`
- Vérifiez que l'application est bien joignable en ouvrant le port 8080 de votre node
- Ecrivez un manifest `nginx-deployment.yml` pour déployer 2 replicas d'un pod nginx (en version 1.18.0)
- Lancez le deployment, vérifiez le nombre de pods et vérifiez que le deployment et le replicaset (ainsi que la version de l'image utilisée) ont été créés
- Modifiez le fichier `nginx-deployment.yml` afin d'utiliser l'image nginx en version latest, appliquez la modification (`kubectl apply`)
- Que se passe-t-il ? Combien de replicasets avez-vous ? Quelle est l'image utilisée par le replicaset en cours d'utilisation ?
- Supprimez toutes les ressources créées et recréez les en utilisant les commandes impératives
- Créez un répertoire `Kubernetes-training` et un sous-dossier `tp-2` et copiez vos manifests à l'intérieur
- Enfin, poussez ce dossier sur github afin de conserver tous vos fichiers

Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- [Gestion du réseau](#)
- Gestion du stockage
- Introduction à helm
- Mini-projet



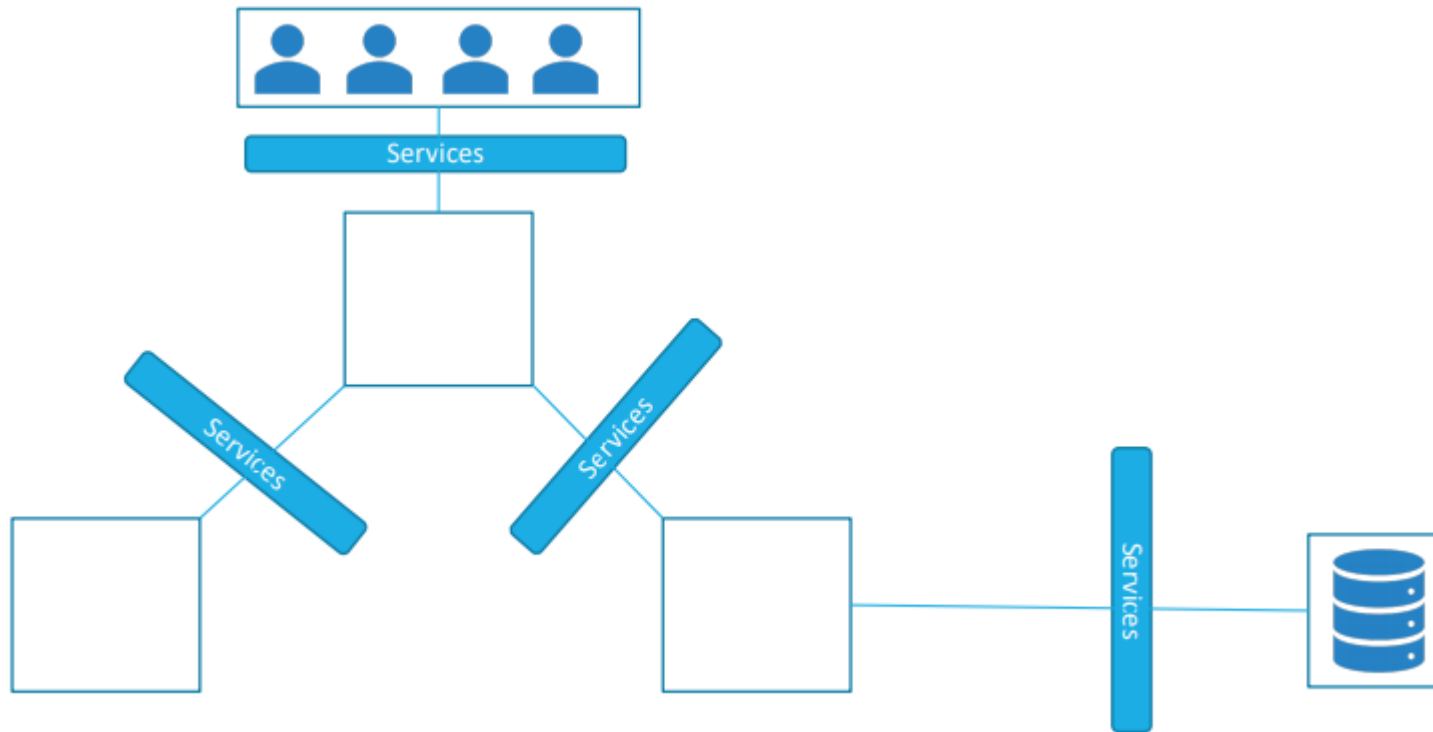
Gestion du réseau (1/7): Label et Selecteur



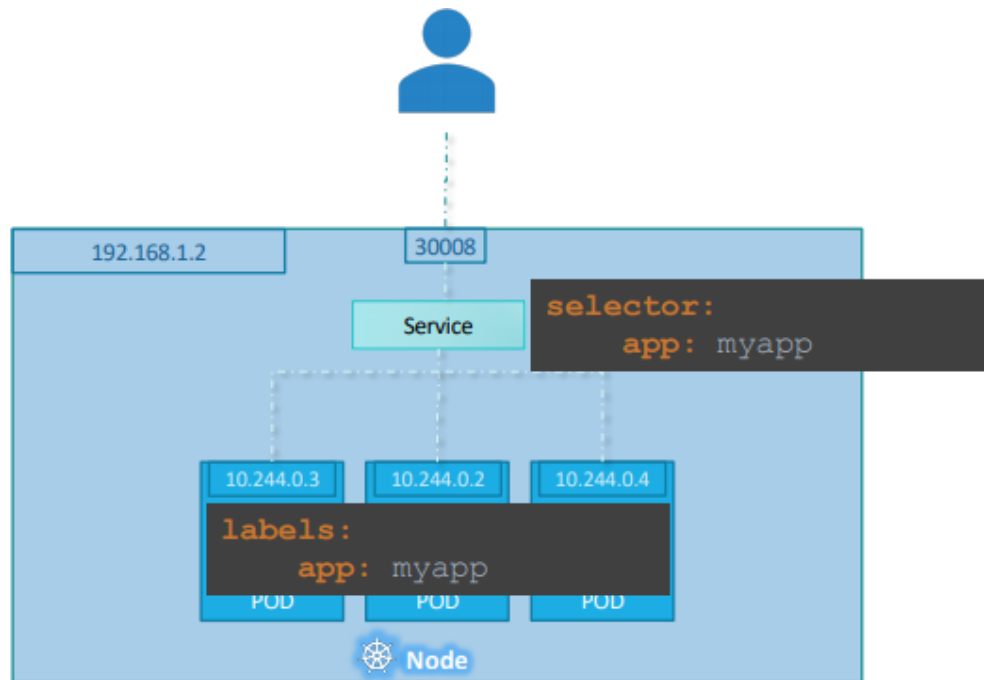
Gestion du réseau (2/7): Smart Search

```
kubectl get po --show-labels
kubectl get pods -l app=nginx --namespace=production
kubectl get pods -l env!=production --namespace=production
kubectl get pods -l 'env in (production)' --namespace=production
kubectl get pods -l 'app notin (nginx)' --namespace=production
```

Gestion du réseau (3/7): Services



Gestion du réseau (4/7): Services NodePort



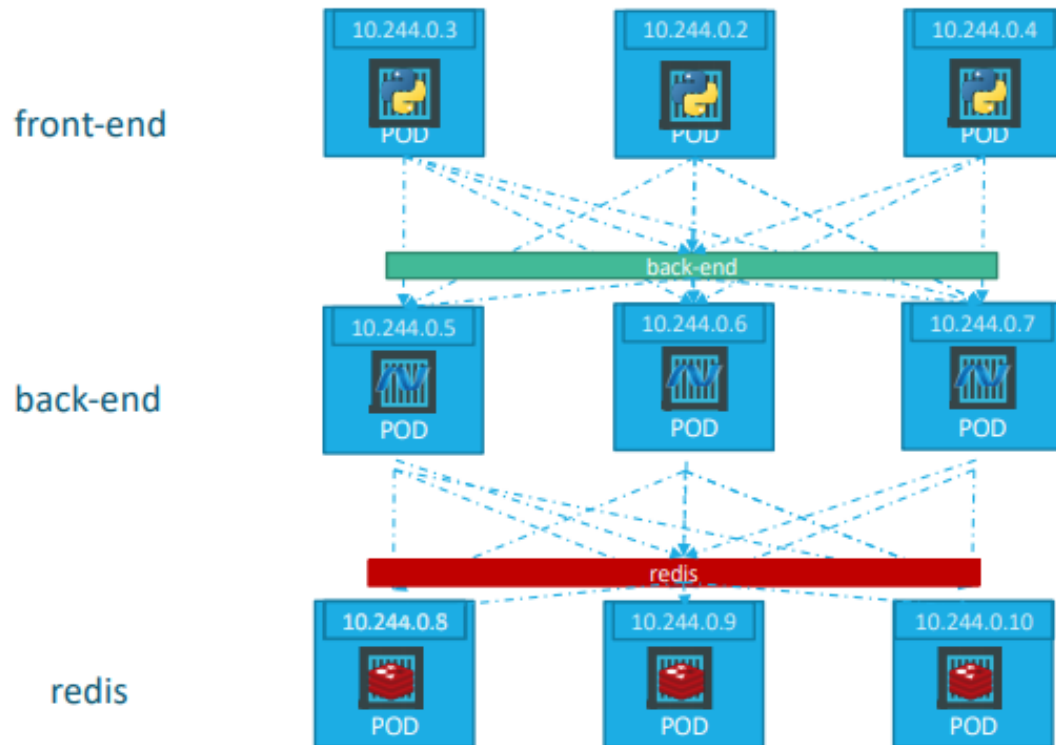
```
service-definition.yml
apiVersion: v1
kind: Service
metadata:
  name: myapp-service
spec:
  type: NodePort
  ports:
    - targetPort: 80
      port: 80
      nodePort: 30008
  selector:
    app: myapp
```

```
pod-definition.yml
> kubectl create -f service-definition.yml
service "myapp-service" created

> kubectl get services
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP          16d
myapp-service NodePort    10.106.127.123 <none>        80:30008/TCP     5m

> curl http://192.168.1.2:30008
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
```

Gestion du réseau (5/7): Service ClusterIP



service-definition.yml

```
apiVersion: v1
kind: Service
metadata:
  name: back-end
spec:
  type: ClusterIP
  ports:
    - targetPort: 80
      port: 80
  selector:
```

pod-definition.yml

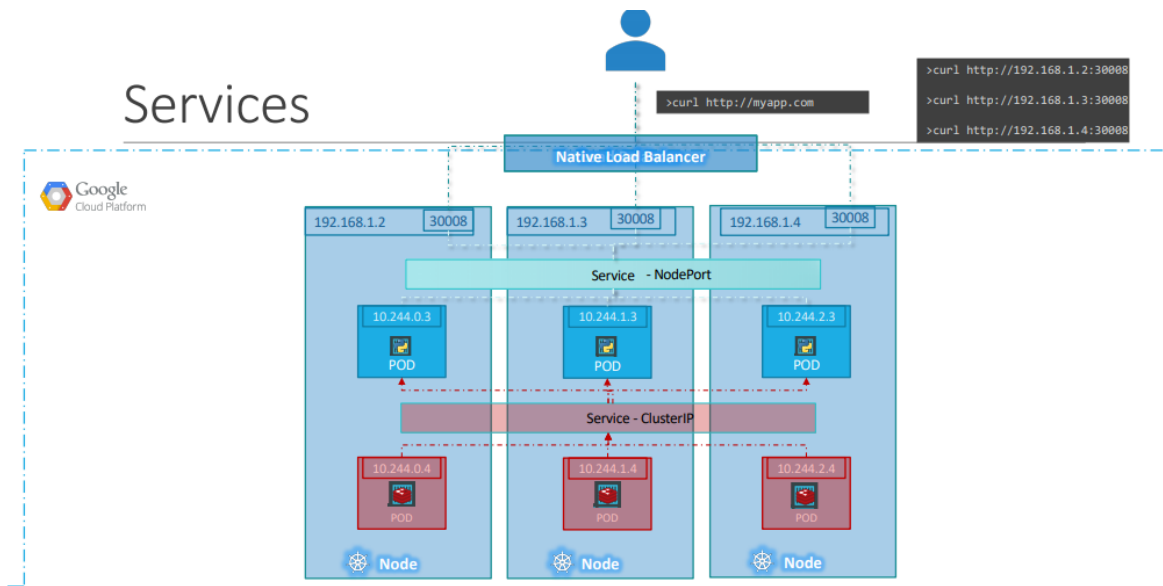
```
> kubectl create -f service-definition.yml
service "back-end" created
```

```
> kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	16d
back-end	ClusterIP	10.106.127.123	<none>	80/TCP	2m

```
app: myapp
type: back-end
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Gestion du réseau (6/7): Service Loadbalancer



```
service-definition.yml
apiVersion: v1
kind: Service
metadata:
  name: front-end
spec:
  type: NodeBalancer
  ports:
    - targetPort: 80
      port: 80
  selector:
    app: myapp
    type: front-end
```

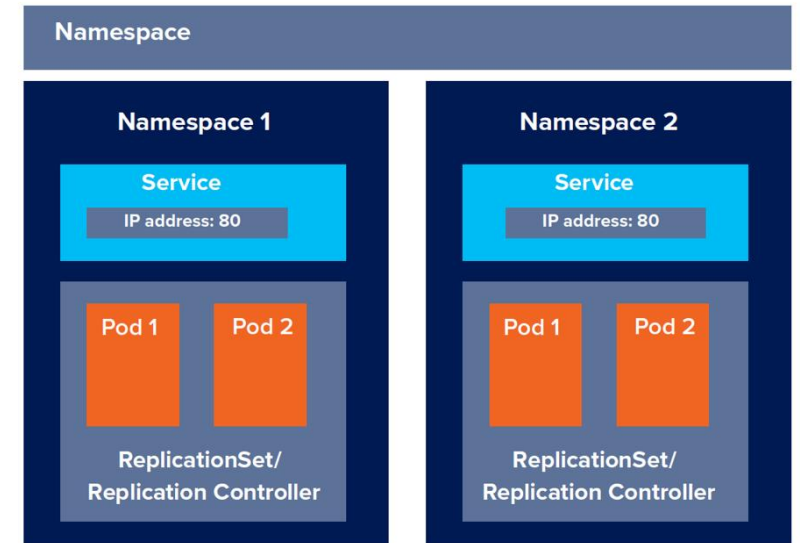
```
> kubectl create -f service-definition.yml
service "front-end" created
```

```
> kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	16d
front-end	LoadBalancer	10.186.127.123	<Pending>	80/TCP	2m

Gestion du réseau (7/7): Namespace

- Isolation / limitation d'environnement
- Sécurité
- Droit d'accès



```
1  apiVersion: v1
2  kind: Namespace
3  metadata:
4    name: production
```

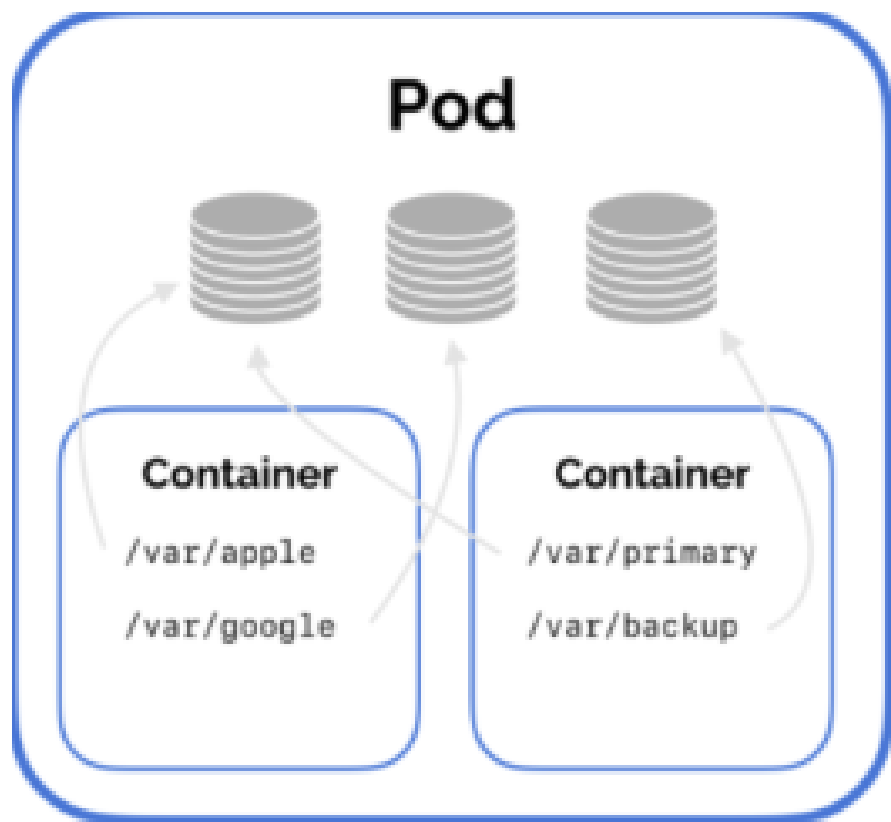

TP-3: Créez un service de type nodeport

- Ecrivez un manifest namespace.yml qui crée un namespace nommé production et lancez la création de ce namespace à partir du manifest
- Toutes vos prochaines ressources doivent être créées dans le namespace production
- Ecrivez un manifest pod-red.yml pour déployer un pod avec l'image mmumshad/simple-webapp-color en précisant que la color souhaitée est la rouge (red), le pod doit posséder le label « app: web »
- Ecrivez un manifest pod-blue.yml pour déployer un pod avec l'image mmumshad/simple-webapp-color en précisant que la color souhaitée est la bleue (blue) le pod doit posséder le label « app: web »
- Lancez la création des deux pods
- Ecrivez un manifest service-nodeport-web.yml qui permettra exposer les pods via un service de type node port, le nodeport devra être le 30008 et les target les ports 8080 de nos pods dont le label est « app: web »
- Lancez la création du service et vérifiez qu'il trouve les deux pods (champ endpoint en utilisant la commande kubectl describe)
- Vérifiez que l'application est bien disponible en ouvrant le port 30008 de votre nœud
- Créez un repertoire tp-3 dans Kubernetes-training (après l'avoir récupéré sur votre github) et copiez vos manifests à l'intérieur
- Enfin, poussez ce dossier sur github afin de conserver tous vos fichiers

Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- Gestion du réseau
- [Gestion du stockage](#)
- Introduction à helm
- Mini-projet



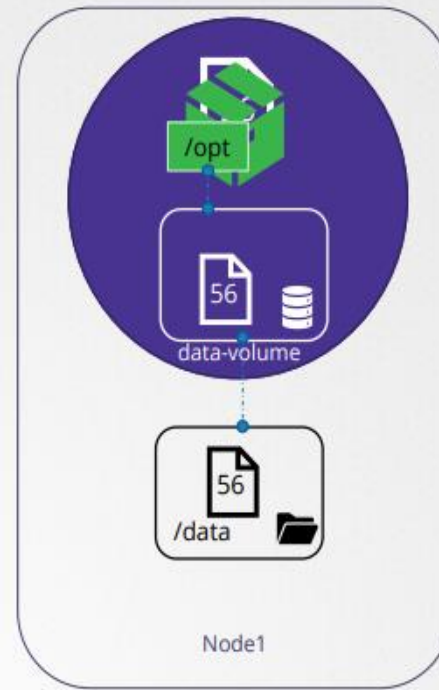


Gestion du stockage (1/5): Problématique

- Si le conteneur est supprimé
- Si on souhaite partager des données entre conteneur

Gestion du stockage (2/5): Volumes

```
apiVersion: v1
kind: Pod
metadata:
  name: random-number-generator
spec:
  containers:
  - image: alpine
    name: alpine
    command: ["/bin/sh", "-c"]
    args: ["shuf -i 0-100 -n 1 >> /opt/number.out;"]
    volumeMounts:
    - mountPath: /opt
      name: data-volume
  volumes:
  - name: data-volume
    hostPath:
      path: /data
      type: Directory
```



Gestion du stockage (3/5): Persistent Volume

pv-definition.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-vol1
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

▶ `kubectl create -f pv-definition.yaml`

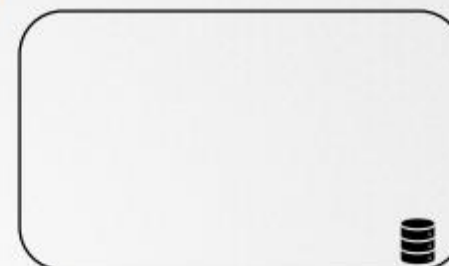
▶ `kubectl get persistentvolume`

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
pv-vol1	1Gi	RWO	Retain	Available				3m

ReadOnlyMany

ReadWriteOnce

ReadWriteMany



Persistent Volume (PV)

Gestion du stockage (4/5): Persistent Volume Claim

pvc-definition.yaml

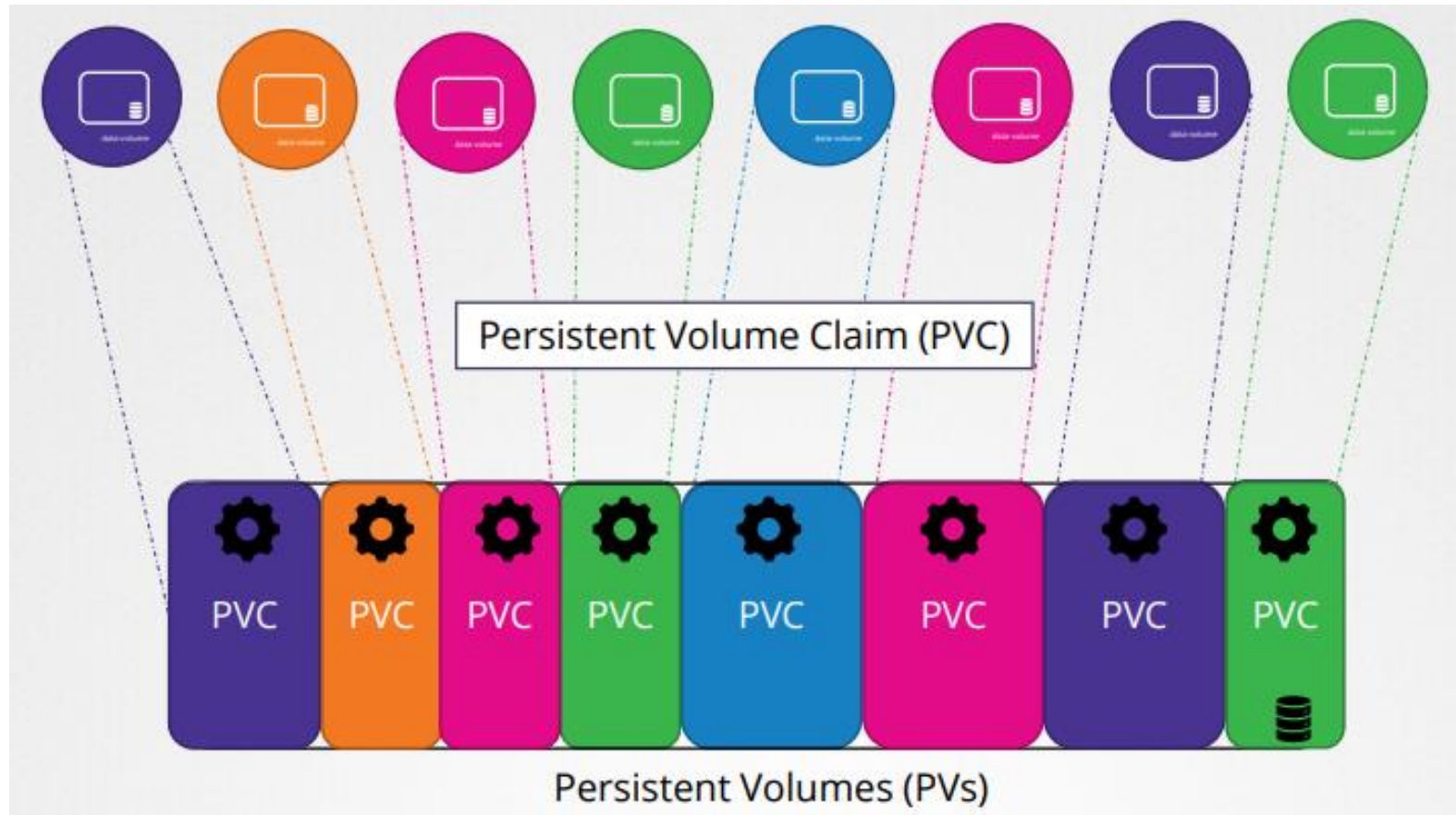
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
```

▶ `kubectl create -f pvc-definition.yaml`

pv-definition.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-vol1
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 1Gi
  awsElasticBlockStore:
    volumeID: <volume-id>
    fsType: ext4
```

Gestion du stockage (5/5): PV et PVC



TP-4: Stockage persistant

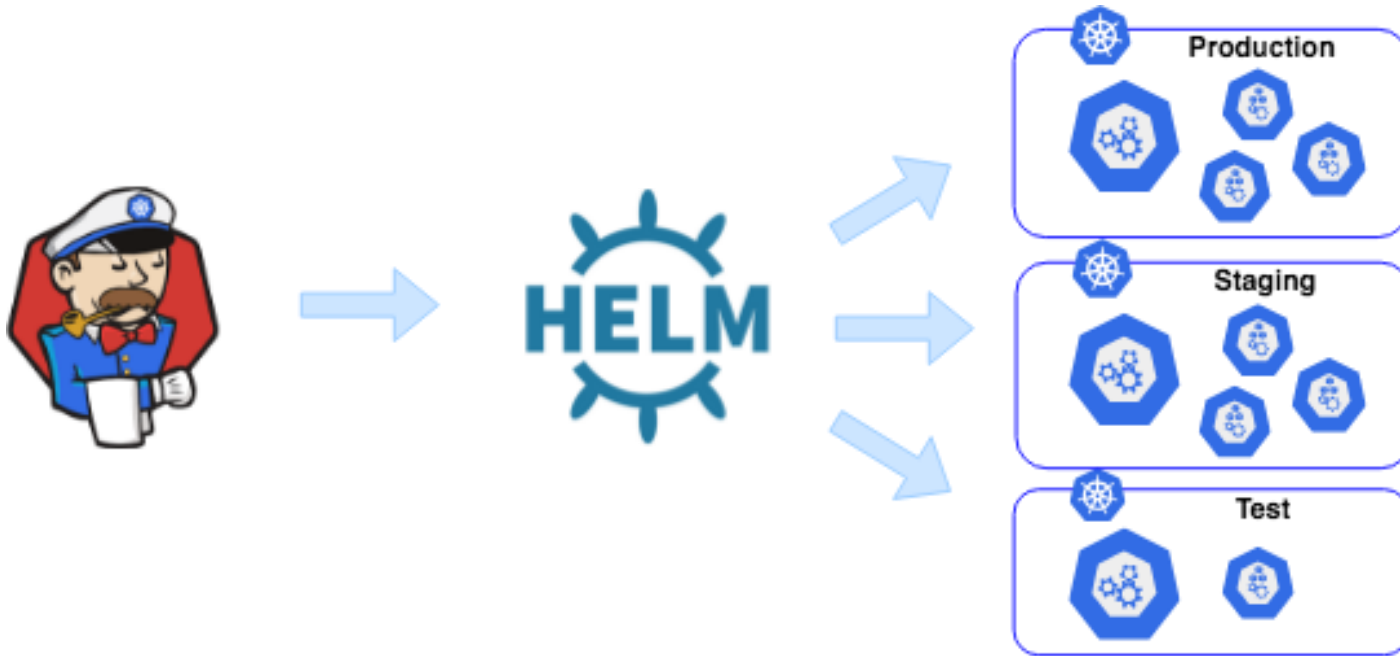
- Ecrivez un manifest `mysql-volume.yml` déployant un pod (nommé `mysql-volume`) `mysql`, avec les paramètre d'environnement suivants : nom bdd: `eazytraining`, login: `eazy`, mot de passe: `eazy`, mot de passe compte root: `password`
- Faites en sorte que le dossier contenant la base de données soit persistant en le montant sur votre nœud dans `/data-volume` en utilisant le principe de volumes
- Lancez la création de votre pod, vérifiez que votre pod a bien démarré et que ce dernier consomme effectivement le dossier local `/data-volume`
- Ecrivez `persistent-pv.yml` (volume persistant de taille 1 Go utilisant le dossier local `/data-pv` pour stocker les données) et `pvc.yml` (volume persistant claim de taille 100 Mo utilisant le PV créé précédemment pour stocker les données)
- Lancez la creation de vos pv et pvc et verifiez qu'ils sont bien créés et prêt à être consommés
- Ecrivez un manifest `mysql-pv.yml` déployant `mysql` (nommé `mysql-pv`) comme précédemment à la seule différence qu'il utilisera comme volume de stockage le PVC créé précédemment
- Vérifiez que votre pod consomme bien le stockage
- Créez un repertoire `tp-4` dans `Kubernetes-training` (après l'avoir récupéré sur votre github) et copiez vos manifests à l'interieur
- Enfin, poussez ce dossier sur github afin de conservez tous vos fichiers

Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- Gestion du réseau
- Gestion du stockage
- [Introduction à helm](#)
- Mini-projet



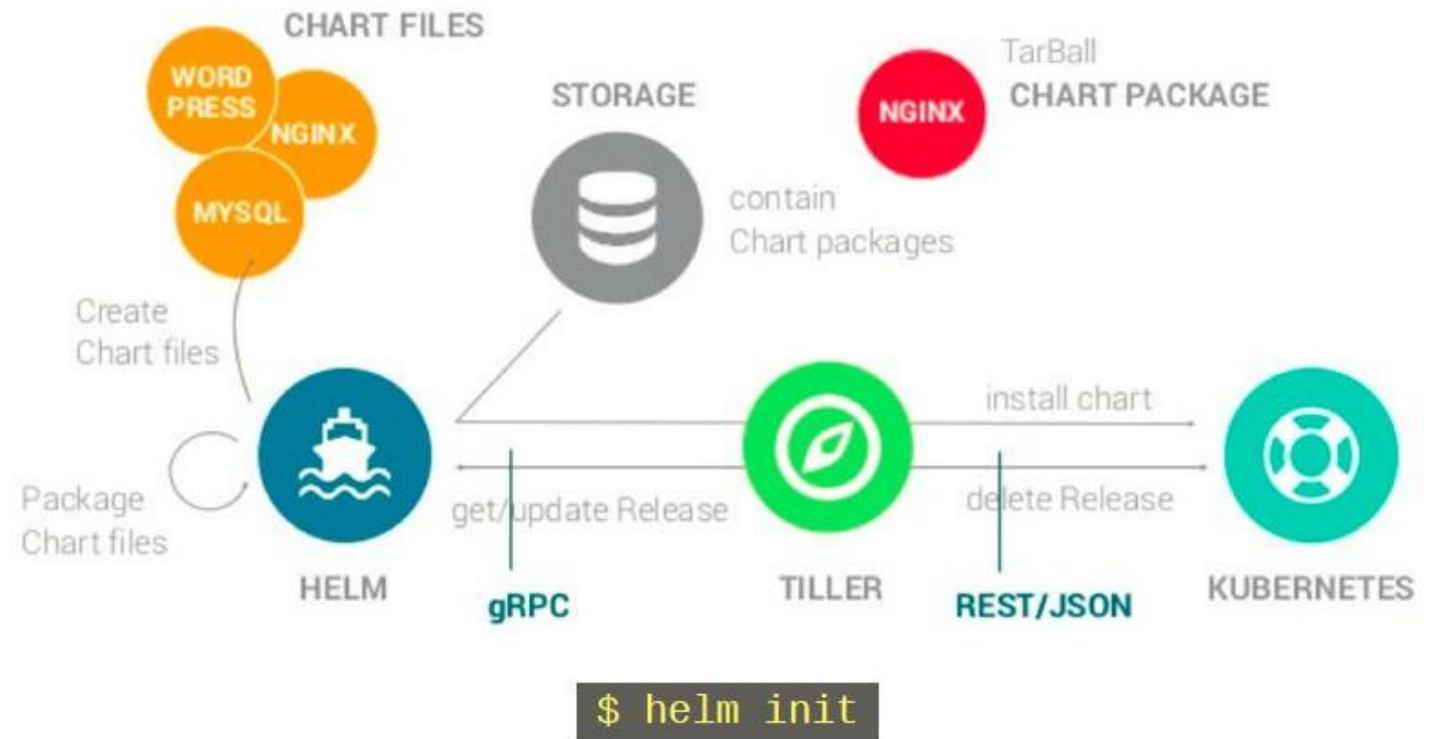
Introduction à helm (1/3): Objectifs



- Package du déploiement de votre application
- Réutilisation de manifest
- Versionning de toute l'application (service, volume, deployment ...)
- Comparable à des rôles ansible
- Bon début pour déployer facilement

Introduction à helm (2/3): Workflow

Helm Architecture



Introduction à helm (3/3): Chart

- Override template
- Golang template
- Versionning de vos releases (deploiement applicatif)
- Stable/incubator

my-chart /

```
|--- chart.yaml
|--- values.yaml
|--- requirements.yaml
|--- templates
|   |--- deployment.yaml
|   |--- service.yaml
|   |--- _helpers.tpl
```

```
~ helm install stable/redis --name my-redis --set cluster.slaveCount=2
```

TP-5: Déployez Wordpress

- Installez helm en utilisant la documentation suivante: <https://github.com/diranetafen/supinfo-kubernetes/tree/minikube/tp07#-install-helm->
- Utilisez le chart wordpress (<https://github.com/helm/charts/tree/master/stable/wordpress>) pour déployer cette application
- Nous vous conseillons d'utilisez le service de type nodeport en http, vous avez le choix du nodeport
- Nous vous conseillons aussi de surchargez les variables nécessaires en utilisant un fichier nommé values.yml où vous surchargerez les variables permettant de déployer l'application comme demandé
- Le mot de passe de l'utilisateur wordpress devra être « admin » et son mot de passe « password »
- Créez un repertoire tp-5 dans Kubernetes-training (après l'avoir récupéré sur votre github) et copiez vos manifests à l'interieur
- Enfin, poussez ce dossier sur github afin de conservez tous vos fichiers

Plan

- Présentation du formateur
- Introduction au DevOps et à l'orchestration
- Composants et Installation de Kubernetes
- Déployez vos premières applications
- Gestion du réseau
- Gestion du stockage
- Introduction à helm
- [Mini-projet](#)



Mini-projet: Déployez Wordpress à l'aide de manifests (et non par helm 😊)

- Déployez wordpress en suivant les étapes suivantes
 - Créez un deployment mysql avec un seul replicat
 - Créez un service de type clusterIP pour exposer vos pods mysql
 - Créez un deployment wordpress avec les bonnes variables d'environnement pour se connecter à la base de données mysql
 - Votre deployment devra stocker les données de wordpress sur un volme mounté dans le /data de votre nœud
 - Créez un service de type nodeport pour exposer le frontend wordpress
- Nous vous conseillons d'utiliser les manifests pour réaliser cet exercice
- Grâce à ce travail vous comprendrez mieux comment les fichiers contenu dans le chart wordpress
- A la fin de votre travail, poussez vos manifests sur github et envoyez nous le lien de votre repo à eazytrainingfr@gmail.com et nous vous dirons si votre solution respecte les bonnes pratiques et si votre solution bonne. Nous vous proposerons aussi notre solution/



Merci pour votre attention ! À
la prochaine sur EAZYTraining