INTRODUCTION AUX SYSTÈMES DISTRIBUÉS

INF36307 – SYSTÈMES DISTRIBUÉS

AGENDA – COURS 3 – KUBERNETES

1

Déploiement

Pods et services

2

Allocation de ressources

3

Mise à l'échelle

Adapter la taille de la flotte de services

4

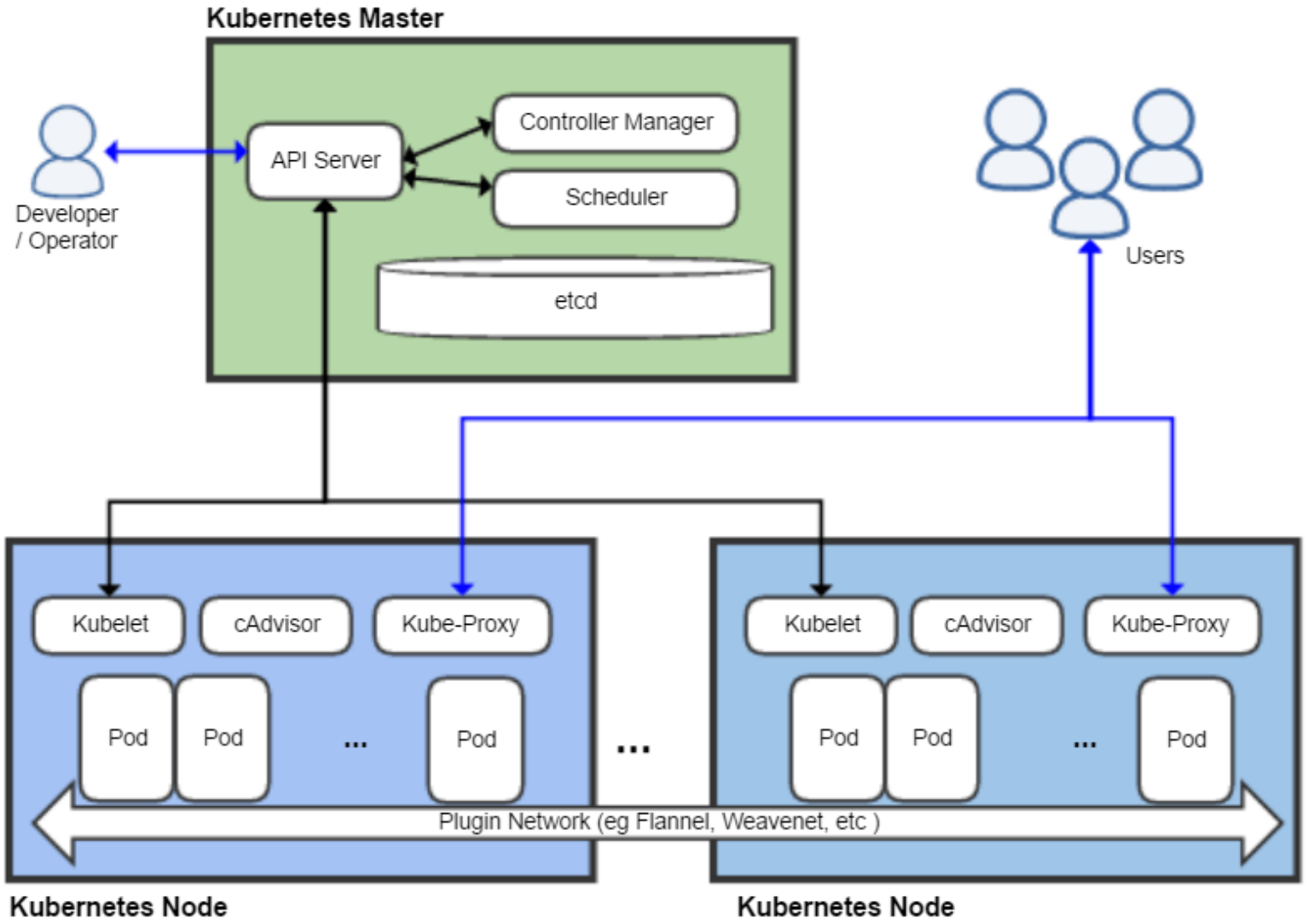
Regéneration

Pratique de l'auto guérison des services

KUBERNETES

- Une plateforme Open Source
- Facilite le déploiement de clusters de serveurs
- Les clusters hébergent des flottes de conteneurs
  - Deployés sur des serveurs
- Conçu par Google

kubernetes



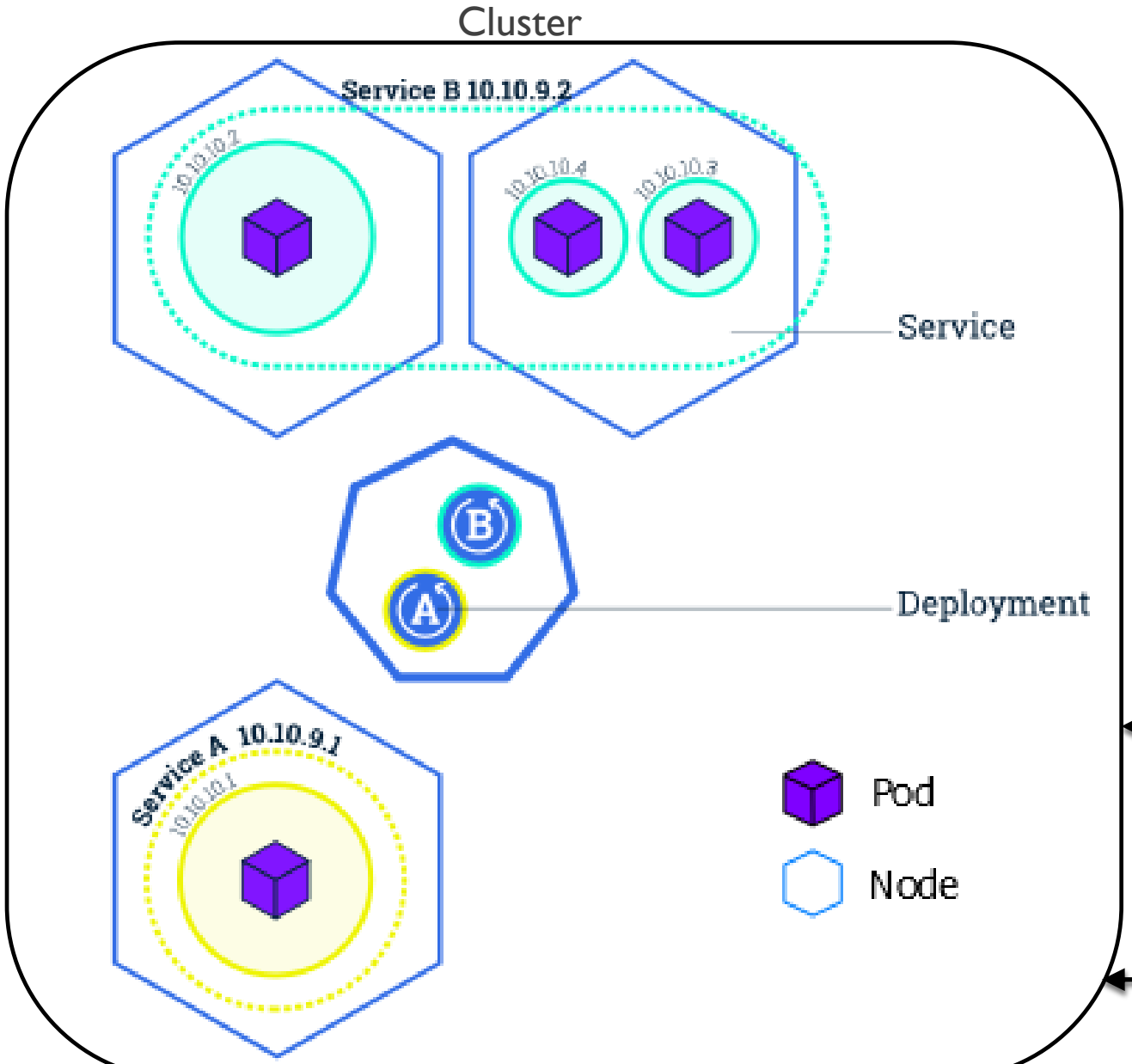
- **etcd**: stockage de configuration
- **Scheduler**: déploie un pods sur une **node** en fonction des ressources demandées et disponibles
- **Node** : une machine physique qui roule des conteneurs
- **Kubelet**: Gère le cycle de vie d'un pod
- **Cluster DNS** : Service discovery

KEBERNETES POD

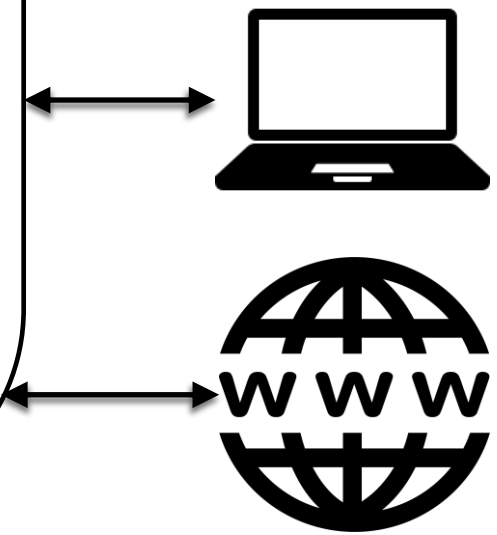
- Consiste en un ou plusieurs conteneurs co-localisées sur une machine
- Possède une adresse IP unique
- Peut définir des volumes

KUBERNETES SERVICE

- Un groupe de **PODS** travaillant ensemble
- Un **service** est requis pour exposer les **pods** au traffic extérieur
  - Ex: votre ordinaleur local ou l'internet public pour la production
- Inclus du Service discovery
- Inclus un Load balancer (round-robin) automatique pour distribuer le trafic sur les **pods**



- **Pod**: contient un ou plusieurs conteneurs
- **Node**: Serveur physique qui héberge des **Pods**
- **Cluster:** Une ou plusieurs **Nodes** orchestrées par un Kubernetes Master
- **Service** : Expose un ou plusieurs **Pods** hebergés sur une ou plusieurs **Nodes**, à l'extérieur d'un Cluster
- **Deployment**: Instance d'une configuration de un ou plusieurs **Pods** deployé dans un **Cluster**



4 TYPES D'EXPOSITION D'UN SERVICE

- **ClusterIP** : Mode par défault. IP interne au **Cluster**
  - Non accessible hors du cluster
- **NodePort**: Port disponible sur chaque **Nodes**
- **LoadBalancer:** IP externe associé à un load balancer
- **ExternalName:** Créer un nom de domaine qui permet l'accès au service

```
kubectl apply -f .
kubectl expose deployment nginx-deployment --type=NodePort
```

Hello world kubernetes

DEAMONSET

- Permet de bypasser le scheduler et de deployer un **pod** sur toutes les **nodes** physiques d'un cluster
  - Utilisé pour l'aggrégation de logs
  - Parfois utilisé pour les déploiement de base de donnée qui ont besoin d'un stockage persistent
    - Utilise la notion de label et d'affinité pour y arriver

Capture de packets distribuée

PROGRESSION

- **Deploiements**
- **Mise à l'échelle**
- **Allocation de ressources**
- **Auto-guérison**

MISE À L'ÉCHELLE HORIZONTALE

- Une fois un **deployment** lancé, il est possible de changer sa taille

```
kubectl scale --replicas=3 deployment/$NAME
```

Confirmer que le load balancing fonctione via les logs nginx.

PROGRESSION

- **Deploiements**
- **Mise à l'échelle**
- **Allocation de ressources**
- **Auto-guérison**

MISE À L'ÉCHELLE VERTICALE

- Kubernetes utilise les **limits** pour restreindre les ressource d'un **conteneur**
  - Mémoire
  - CPU
- **Requests** servent aussi au **scheduler** pour choisir la **node** qui possède assez de ressources pour héberger les **conteneurs** d'un **pod**
- **Requests** détermine les ressources alouées à un **conteneurs**, mais les **conteneurs** peuvent prendre plus tant qu'ils ne dépassent pas les **limits**

```
requests:
    memory: "64Mi"
    cpu: "250m"
```

```
limits:
    memory: "128Mi"
    cpu: "500m"
```

```
requests:
    memory: "64Mi"
    cpu: "250m"
spec:
  containers:
  - name: app
    image: images.example/app:uqar
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m“
      limits:
        memory: "128Mi"
        cpu: "500m"
```

MISE À L'ÉCHELLE VERTICALE

- Les unités pour le CPU sont en fraction de CPU
  - 2 = 2 CPU
  - 0.5 = un demi CPU
  - 500m = un demi CPU
- Les unités pour la mémoire sont en bytes
  - Ex: 128Mi

MISE À L'ÉCHELLE VERTICALE

- Les **limits** sur le CPU et la Mémoire sont les plus fréquemment utilisées
- Mais il en existe d'autres
  - `ephemeral-storage: "2Gi"`

Doubler la mémoire associée à un conteneur.

OOM kill : Lancer MySQL avec 128M de ram

PROGRESSION

- **Deploiements**
- **Mise à l'échelle**
- **Allocation de ressources**
- **Auto-guérison**

KUBERNETES SERVICE ET LABEL

- Permet de ne pas se soucier de la santé d'un **pod** ou de la **node** qui l'héberge
  - Si une **node** physique meurt, tous les **pods** sur cette **node** meurt aussi
  - Mais le **service** va recéduler automatiquement les **pods** configurés pour un **service** sur d'autres **nodes** disponibles
- Le Service Discovery dans un service kubernetes utilise les **labels**
  - Rends possible la découverte des **pods** même lorsqu'ils changent d'adresses IP

```
labels:
    app: nginx
```

```
selector:
    matchLabels:
      app: nginx
```

- Ajouter un **label** a un **pod**

```
kubectl label pod $POD_NAME university=uqar
```

```
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: mysql
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
```

VOLUME KUBENETES

- Un peu comme docker, kubernetes assigne un disque éphèmere au pods
- Mais il est possible de demander un disque persistant

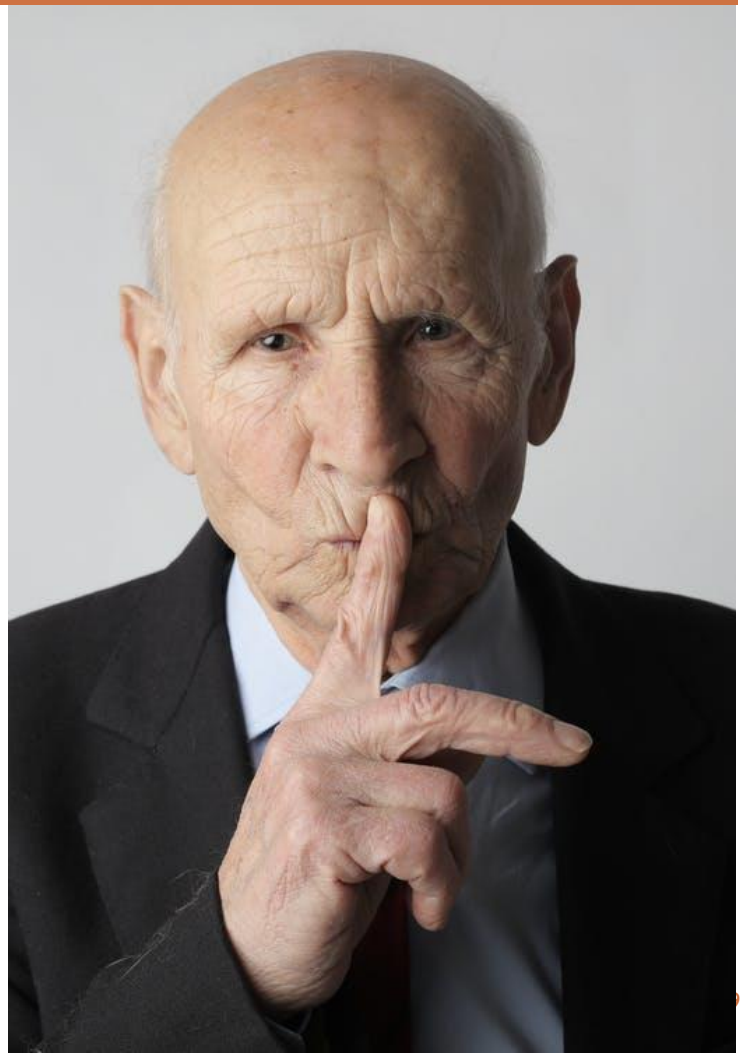
Convertir le service rest en kubernetes

LES SECRETS KUBERNETES

- Permet de rendre disponible au conteneur des informations sensibles (ex: password) sans les rendre visibles dans les fichiers de configuration
- `kubectl create secret generic mysql-db-secret --from-literal=password='test123'`

```
env:
- name: DB_PASSWORD
  valueFrom:
    secretKeyRef:
      name: mysql-pass
      key: password
```

Retirer le mot de pass dans le service rest en kubernetes



AUTO GUÉRISON

- **Kubelet** est en charge des vérifier l'état de santé des **pods** sur une **node**
- Kubelet va régénérer un pod perdu pour conserver l'état désiré dans la configuration

Tuer une des replicas de votre pods et observer la guérison

DEVOIR

LIRE SUR LES SYSTEMS DE GESTION DE QUEUE

SE FAMILIARISER AVEC **KAFKA**