PEO ATOS **Parcours : Devops infra**

Formation :

Kubernetes

Formateur

Brahim Hamdi
brahim.hamdi.consult@gmail.com

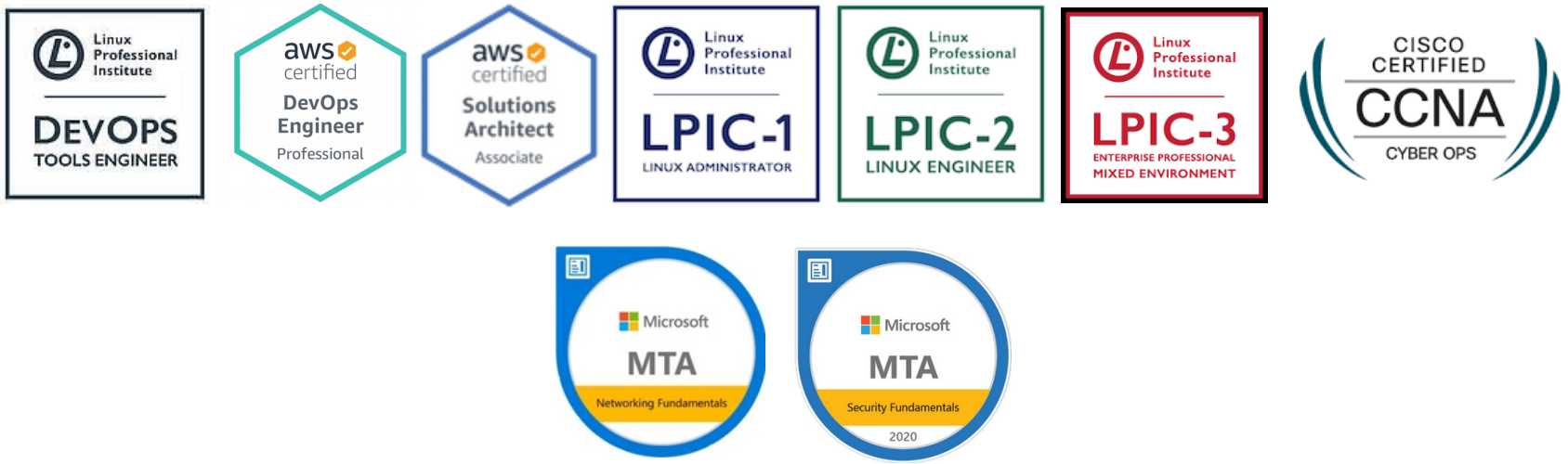
Plan

- Présentation de l’équipe
- Le plan de formation
- Objectifs de la formation
- Public concerné
- Connaissances requises
- Références bibliographiques

Présentation du formateur

Brahim HAMDI

- Consultant/Formateur DevOps & Cloud
- Expert DevOps, Cloud, CyberOps et Linux



Plan de la formation

- Notions Basiques
- Installation kubernetes
- Architecture de kubernetes
- Objets et contrôleurs
- Kubernetes concepts

Objectifs pédagogiques

- Comprendre les conteneurs et l’orchestration
- Comprendre les composants de kubernetes
- Réussir l’installation d’un cluster kubernetes
- Création des ressources dans le cluster
- Déployer une application avec kubernetes

Public concerné

- Développeurs
- DevOps
- Architectes

Connaissance requise

- Connaissances des commandes linux
- Connaissances des conteneurs
- Connaissances générale des principes du réseau et les systèmes d'opérations.

Références

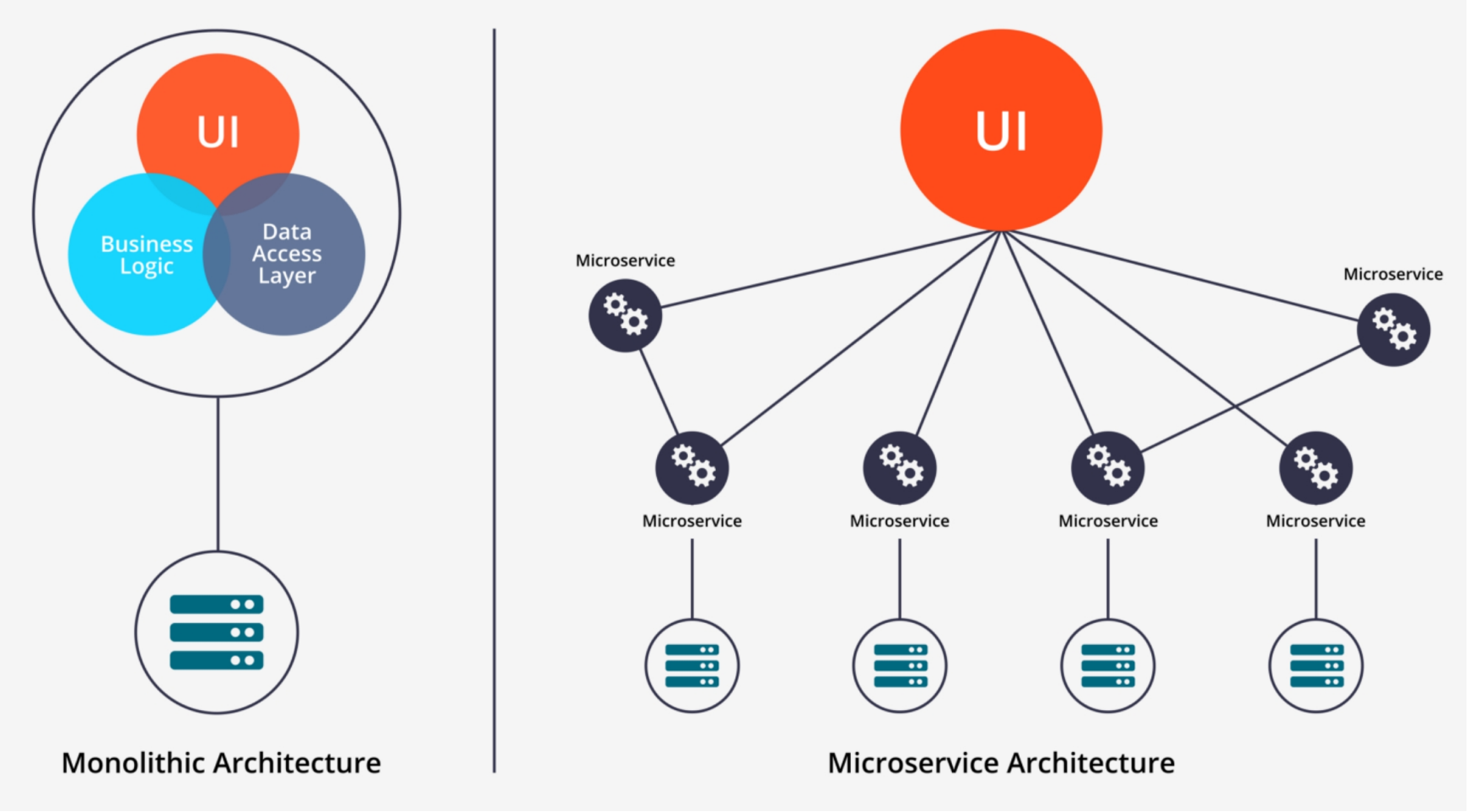
- *https://kubernetes.io*
- *https://www.docker.com*

Notions de base

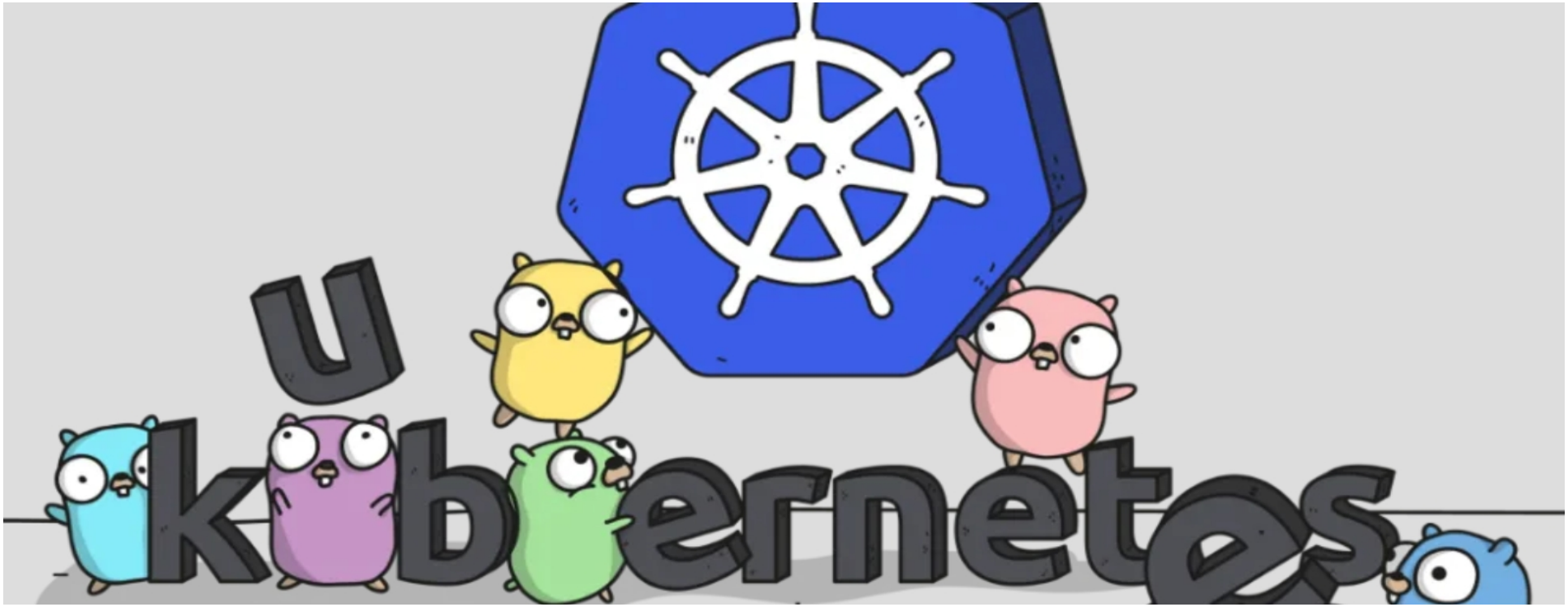
- Les microservices
- Kubernetes
- K8s vs docker swarm

Les microservices

Un modèle de conception de logiciel pour découpler les applications monolithiques en composants réutilisables.

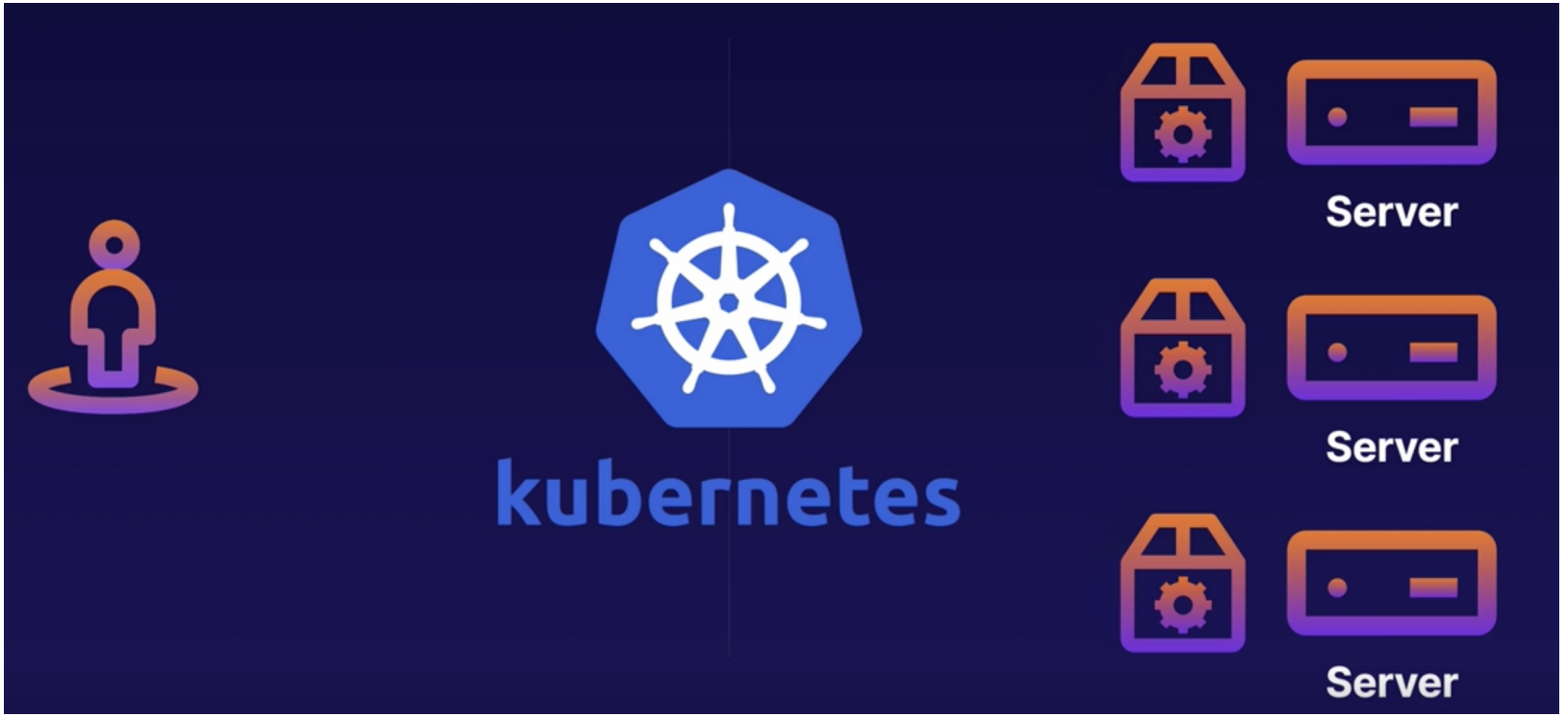


- Kubernetes, K8s, est un système open-source pour
  - Automatiser le déploiement des applications conteneurisée
  - Scaling et la gestion des applications conteneurisées.



- Déploiement « facile » des conteneurs automatisation

- Déploiement « facile » des multiple réplications dans multiple serveurs



Kubernetes vs Docker Swarm

	Kubernetes	Docker Swarm
Installation	Installation facile et flexible	Intégration dans l'environnement Docker déjà réalisé
Initialisation cluster	Compliquée ; mais une fois initialisé, le cluster est très fort	Très simple, mais cluster n'est pas très fort
Mise à l'échelle	Rapide dans la mise à l'échelle	Plus rapide que kubernetes
Mise à l'échelle automatique (autoscaling)	Oui	Non
Monitoring	Fait partie intégrante de l'application	Uniquement avec un logiciel supplémentaire
Répartition de charge	Possible avec une étape supplémentaire	Fait partie du concept

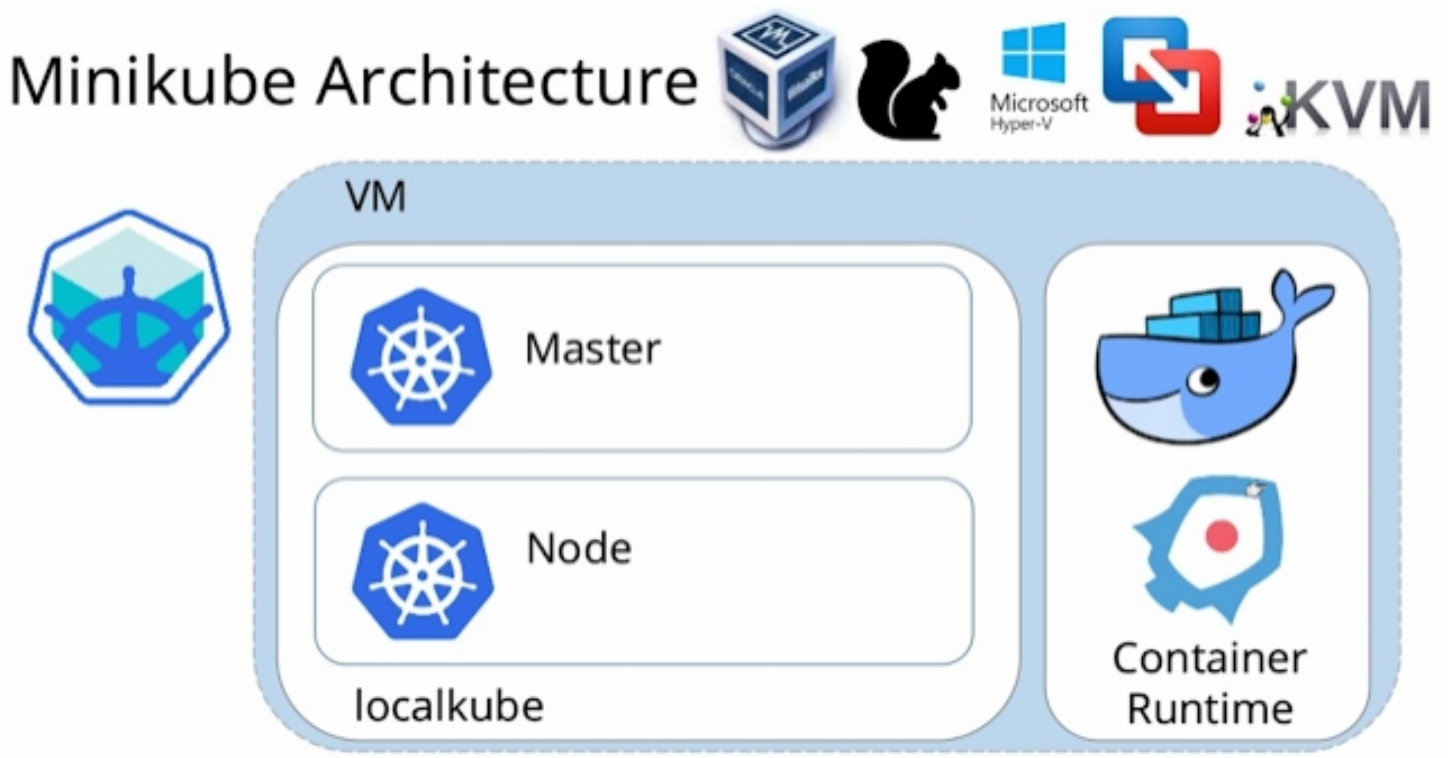
Kubernetes cluster

- Un cluster kubernetes
- Architecture & composants
- Installation

Cluster kubernetes

- Une collection de machines pour exécuter les conteneurs

- Minikube:


Minikube Architecture
Microsoft Hyper-V
KVM
VM
Master
Node
localkube
Container Runtime

Cluster kubernetes: Architecture & composants

- Multi nodes

Master
Control Plane

* Une collection des services qui contrôlent le cluster.
* Permet l'interaction des utilisateurs avec le cluster
* Monitoring de l'état du cluster

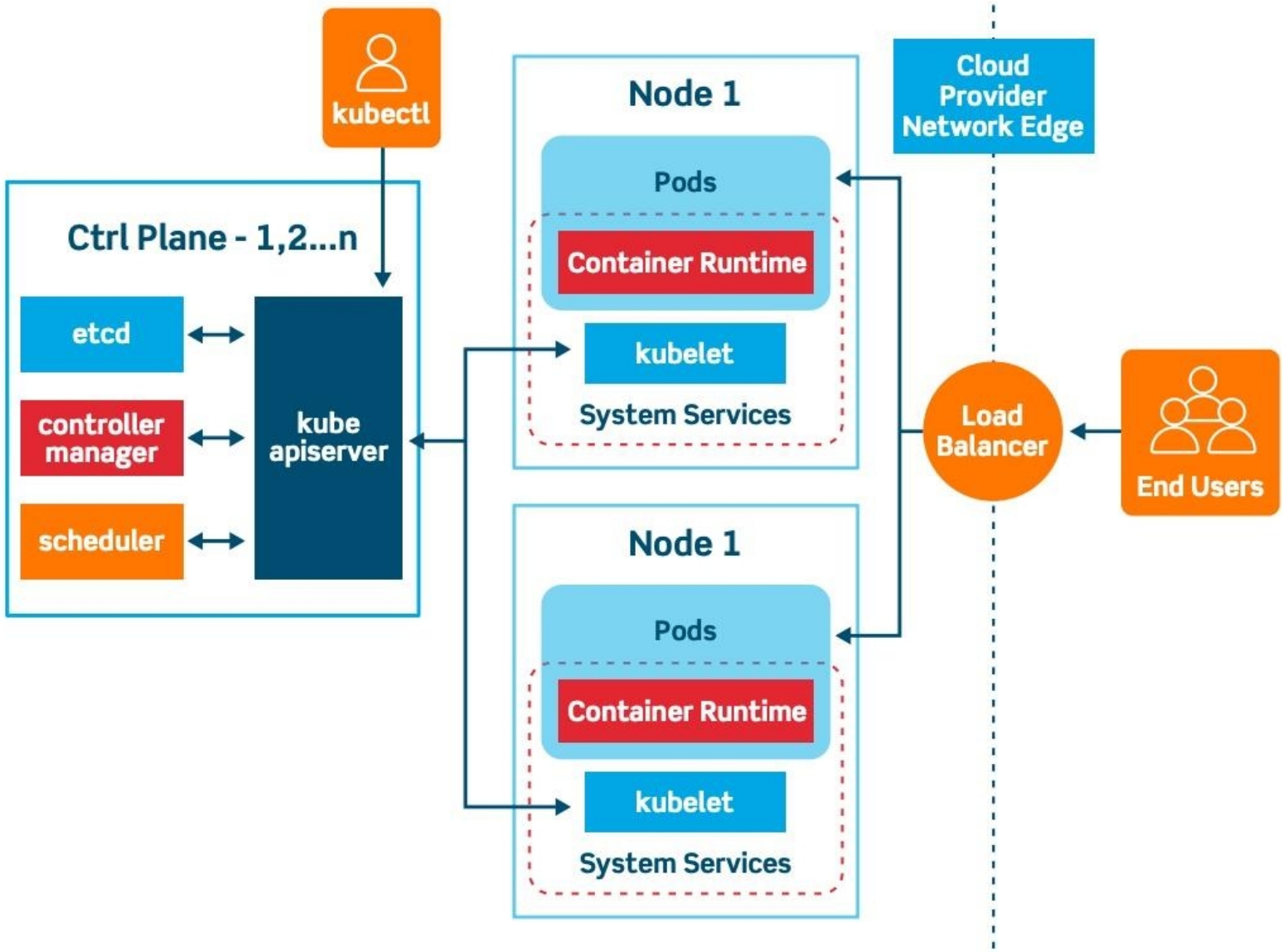
Node = worker

* La machine ou les conteneurs s'éxécutent
* Monitoring de l'état des conteneur et renvoyer le rapport des status au control plane

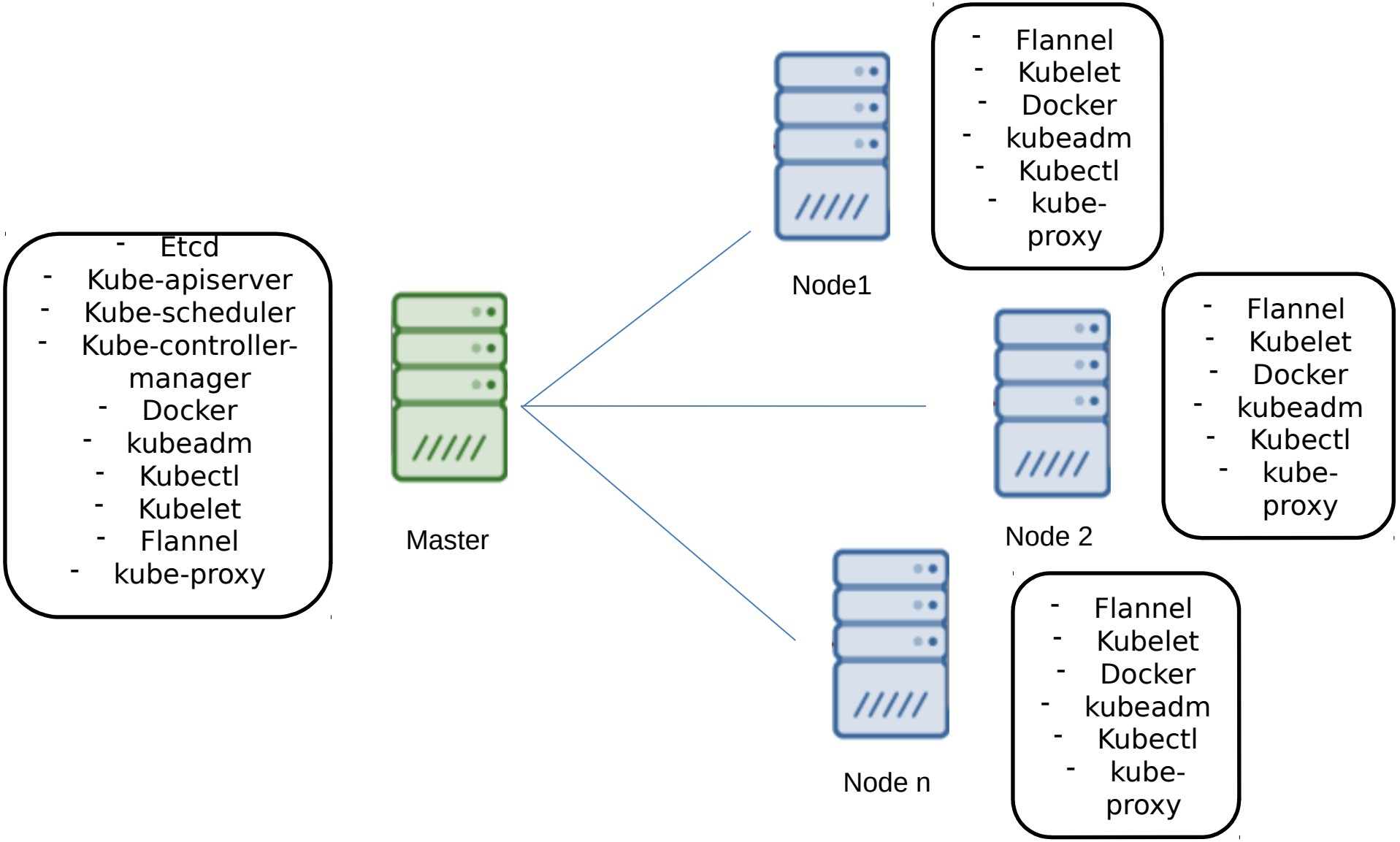
Node

Node

Cluster kubernetes: Architecture & composants



Cluster kubernetes: Architecture & composants



Cluster kubernetes: Architecture & composants

Composants:

- **Docker**: container runtime
- **Kubeadm:** un outil pour la gestion du cluster
- **Kubelet** : agent pour la gestion de processus d'éxecuter des conteneurs dans chaque node.
- **kubectl** : CLI outil pour l'interaction de l'utilisateur avec le cluster
- **Flannel**.: CNI : container network interface
- **Control plane** : juste dans le master : ensemble de services qui forment le master du kubernetes pour assurer la communication et l'intégration avec les autres nodes.
- **Kube-proxy** : maintient les règles réseau les nœuds, permettant une communication réseau vers les Pods depuis des sessions réseau à l'intérieur ou à l'extérieur du cluster.

Steps:

- Installation Docker
- Installation **kubeadm**, **kubelet** et **kubectl**
- Création du cluster
  - Initialisation sur Master
  - Joindre les nodes workers
- Configuration du réseau avec **Flannel**.

- **Kube master , initialisation du cluster**

sudo kubeadm init --apiserver-advertise-address=192.168.205.10 --pod-network-cidr=10.244.0.0/16

- **Kube node , joindre le cluster**

sudo kubeadm join 192.168.205.10:6443 --token 4vzlft….

- **Flannel : master !!**

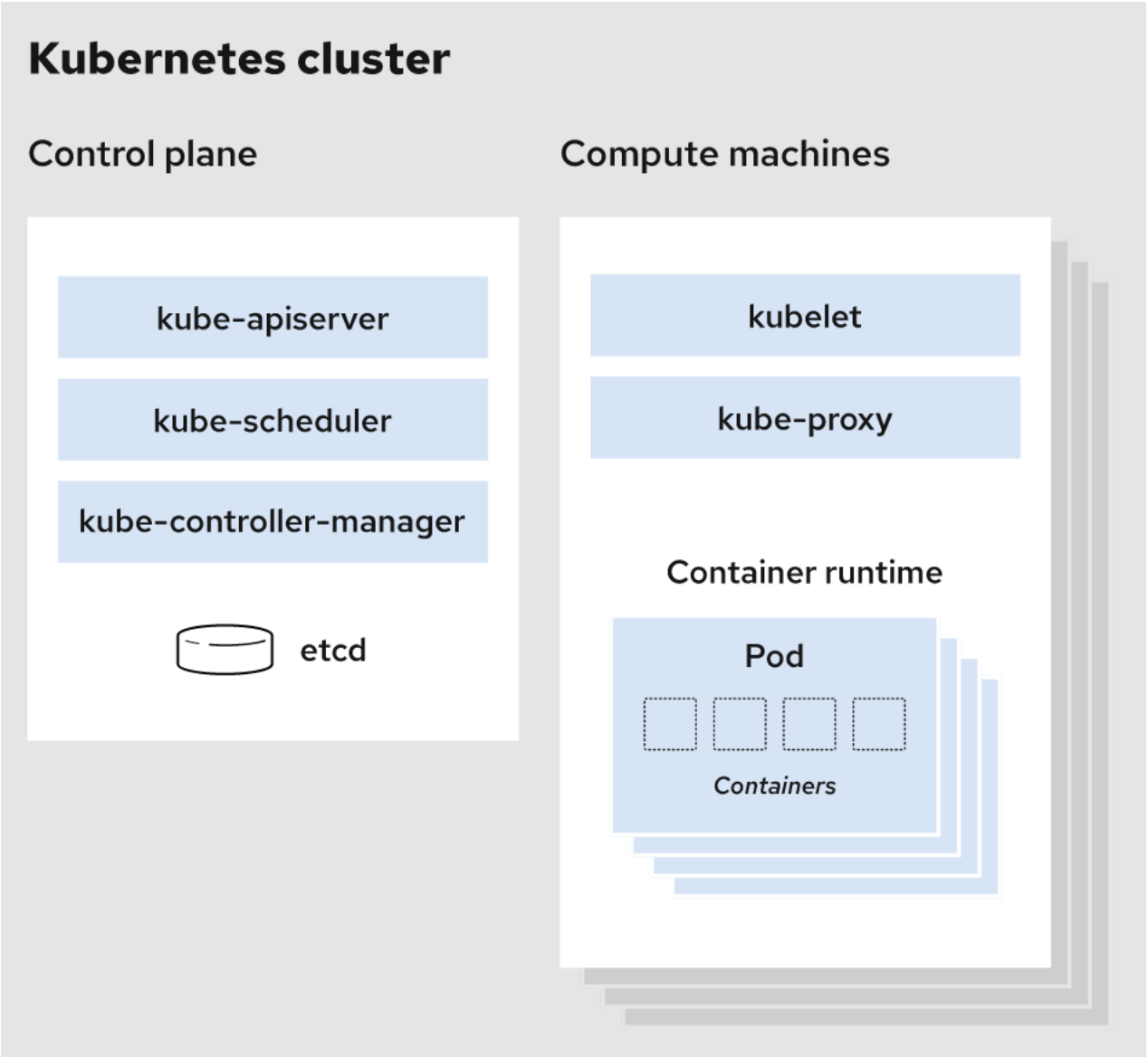
kubectl apply -f https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml

Kubernetes Nodes

- Master
- Nodes/workers

```
kubectl get nodes
```

```
kubectl describe node $node_name
```



Objets & contrôleurs K8S

- Objets kubernetes
- Contrôleurs kubernetes
- Pod
- Service
- ReplicaSet
- Deployement

Les objets kubernetes

- Les quatres principaux objets utilisés par kubernetes :
  - Les Pods
  - Les Services
  - Les volumes
  - Les namespaces

Contrôleurs kubernetes

- Un contrôleur est une abstraction de plus haut niveau qu'un objet simple et est basé sur ces derniers :
- Il va venir piloter des objets de plus bas niveau
- Les trois principaux contrôleurs sont :
  - Le ReplicatSet
  - Le Deployment
  - Le StatefulSet

Kubernetes concepts

- Le Pod
- Le Service
- Le ReplicatSet
- Le Deployment

Le Pod

- Le Pod représente la plus petite unité déployable et exécutable au sein d'un cluster.
- C'est la brique de base qui encapsule un ou plusieurs conteneurs
- Il a son propre réseau et volume interne

- Si on veut scaler horizontalement une application, il suffit d'avoir plusieurs instances, chacune sur un pod différent
- Dans la réalité, on ne travaille que rarement directement sur les pods, mais on utilise des contrôleurs pour gérer les pods.

Création d'un pod

```
cat << EOF | kubectl create -f -
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
EOF
```

```
kubectl get pods
```

```
kubectl describe pod nginx
```

```
kubectl delete pod nginx
```

Le Service

- Un service est un ensemble logique de pods ouverts au reste du cluster kubernetes
- En regroupant logiquement des pods ensemble et en leur fournissant une interface vers le reste du cluster, au moyen d’une IP, d’un port et d’un protocole (http par exemple).
- Le service se charge de router les requêtes vers les pods qu’il gère.
- Contrairement aux pods, un service est persistant et représente le service exposé au reste du cluster.
- Il permet d’être découvert via le mécanisme de « service dicovery ».
- Il peut aller plus loin en fournissant du load balancing par exemple.

Le ReplicaSet

- Le ReplicaSet pilote des pods, et s’assure qu’un certain nombre de pods sont lancés.
- Si un pod vient de mourir, le ReplicaSet va lancer un nouveau pod.
- Dans la réalité, on ne se sert que rarement directement des replicaSet, au profil d’un autre contrôleur, qui vient piloter un ReplicaSet : le Deloyment.

Le Deployment

- Un Deployment contrôle et pilote des RecplicaSets et des Pods.
- Il ajoute des fonctionnalités aux ReplicaSet :

  - Déployer des ReplicaSet : les ReplicaSets vont créer des pods en tâche de fond et le Deloyement va s'assurer que le déploiement se passe bien.
  - Déclarer un nouvel état des pods : le Deployment va piloter la mise à jour des pods, des anciens ReplicaSet vers les nouveaux (rolling update).
  - Revenir à une ancienne version de Deployment (rollback).
  - Mettre à l'échelle le Deployment pour gérer les montées en charge.
  - Mettre en pause un Deployment pour fixer des éventuelles erreurs.
  - Nettoyer les anciens ReplicaSets qui ne servent plus.
  - Et d'autres fonctionnalités pour des cas d'utilisation plus avancés.

Création de 2 ngnix pods :

```
cat << EOF | kubectl create -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.15.4
        ports:
        - containerPort: 80
EOF
```

Pod, Service, Deployment