# Kubernetes

Nicolas Lamirault

# About Me

 Senior Software Engineer

 nicolas.lamirault.xyz

 Open source fan

 @nlamirault

 github.com/nlamirault

 gitlab.com/nicolas-lamirault

# Les conteneurs

# Les soucis

? comment gérer le cycle de vie (stop, start, error)

? gestion des pannes, la maintenance, des noeuds qui hébergent les conteneurs

? la gestion des mise à jours, redéploiement, scalabilité

# Solutions

- Kubernetes (CNCF)
- Nomad (Hashcorp)
- Mesos
- Rancher

# Kubernetes

# Historique

- En provenance de Google, basé sur Borg
- Open Source en 2014
- Premier projet de la CNCF

# Capacités

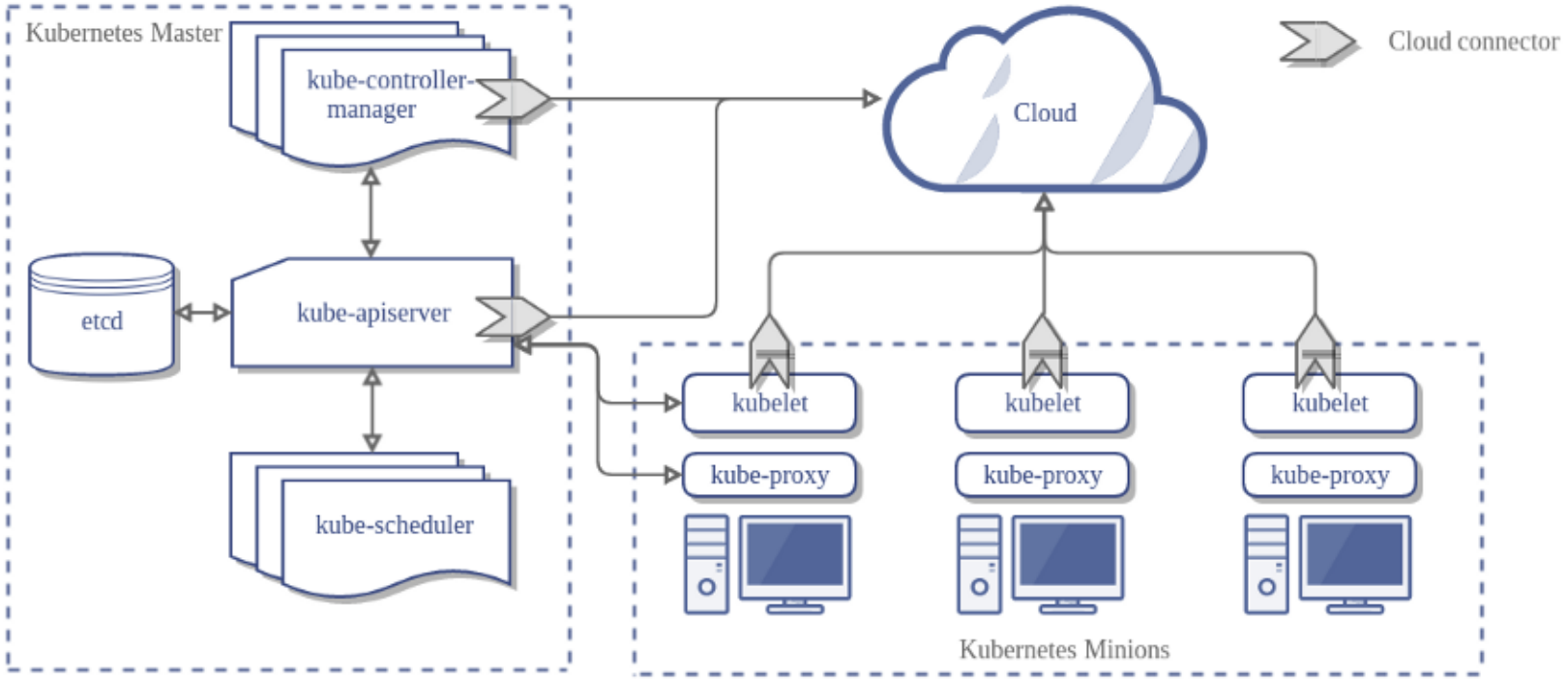
- Execute des composants en fonction de contraintes techniques: réseau, stockage, …
- Gestion des ressources (cpu, mémoire, stockage, …)
- Gestion du fail-over (des noeuds, composants, …)
- RBAC: Gestion des droits d’accès aux ressources

# Points importants

- Abstraction aux moteurs de conteneurs par la CRI (Docker, Rkt, CRI-O, Kata, ContainerD, ...)
- Interface pour les plugins réseaux : CNI (Weave, Flannel, ...)
- Kubernetes Distributions (https://www.cncf.io/certification/software-conformance/)
- Environnement de développement officiel: minikube

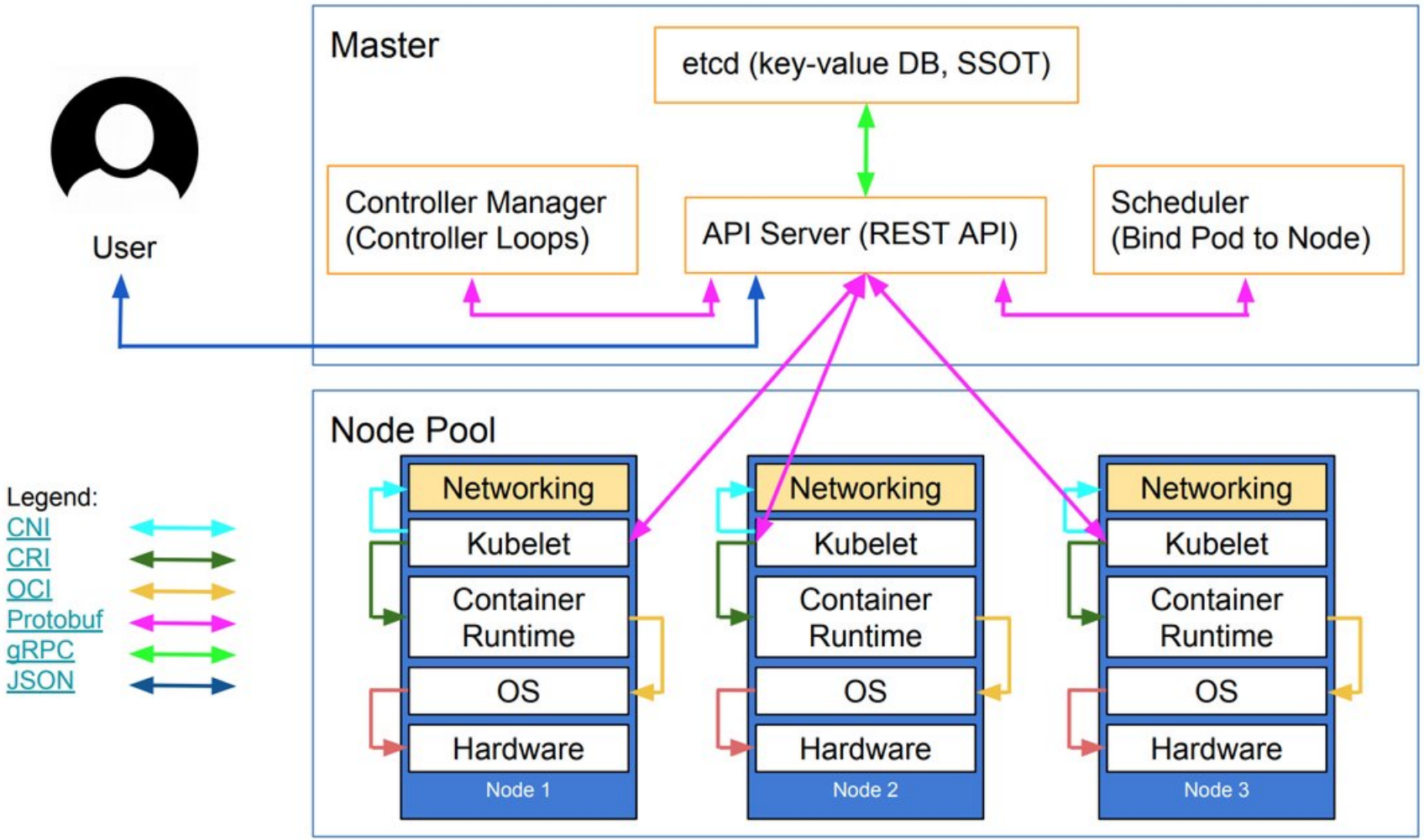
# Architecture

# Vue générale



# Composants

## Kubernetes' high-level component architecture



# Etcd

- Base de données clé/valeur distribuée (protocole Raft)
- Persistence des données du cluster
- ⚠ Faire des backups !

# Control Plane Components aka Master

- kube-apiserver
- kube-controller-manager
- kube-scheduler

# API Server

- Server qui permet la configuration d'objet Kubernetes (Pods, Service, Replication Controller, ...)
- Tous les communications du cluster passent par l'API Server
- Gère l'authentification, l'autorisation, la validation de la demande, le contrôle d'admission, ...

# Controller Manager

## Gère les différents controlleurs du cluster

- Node Controller : cycle de vie des noeuds.
- Replication Controller : supervise les pods, replicasets, ...
- Endpoints Controller : suit les pods, services (endpoints)
- ServiceAccount / TokenController : gère les ServiceAccount et Token

# Scheduler

- **S’occupe de trouver un noeud pour déployer un POD**

- on peut implémenter son propre Scheduler

# Nodes Components

- kubelet
- kube-proxy
- Container Runtime Engine

# Kubelet

- Crée et gère les conteneurs
- Il est contacté par l'API Server

# Kube Proxy

- Gère les règles réseaux sur chaque noeud
- Forwarding TCP/UDP et Load balancing entre les services et les backend

# Container Runtime Engine

## Exécute les conteneurs (compatible CRI)

- Containerd (Docker)
- Cri-o (Poussée par Redhat sur Openshift/OKD)
- Kata
- gVisor

# Container Network Interface

- **IP Address Management: allocation des adresses IP des conteneurs.**
- **Interface : gestion de l'interface réseau qui va porter le conteneur**

- Flannel
- Calico
- Weave

# Container Storage Interface

**› création, redimensionnement, attachement, montage [...] des volumes**

Les drivers: https://kubernetes-csi.github.io/docs/drivers.html

# Cloud-Controller-manager

- Fonctions relatives aux interactions avec les cloud providers.
- Extraction du code en provenance du controller-manager
- Pod qui sera démarré en suivant du bootstrap du cluster si besoin

# Clients

- kubectl : CLI qui permet de piloter un cluster Kubernetes
- dashboard: Web UI officielle

# Installation

# Kubeadm

- Officiel
- Pas de multi-master
- Ne gère pas CRI (Choix de la runtime)
- Ne gère pas CNI (Choix du plugin réseau)

# Cloud Public

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

# Autres

- **Local: Minikube**
- **Ansible: Kubespray**
- **Terraform : Typhoon**
- **Distributions : Openshift/OKD(Redhat), Rancher, …**
- **Kubernetes The Hard Way, par Kelsey Hightower**

Choix : https://kubernetes.io/docs/setup/pick-right-solution/

# Kubernetes / Ressources

# Resources / Core

- Namespaces
- Labels
- PODs
- Services

# Namespace

- **Nommer et cloisonner les différents environnements, projets, ...**
- **Quotas de resources sur des namespaces**

- **kube-system**: namespace utilisé par le système interne à Kubernetes
- **kube-public**: namespace spécial, utilisé pour initialisation et configuration
- **default**: namespace utilisé pour tout objet sans namespace spécifié

# Labels

## Clé/Valeur pour identifier les objets, les décrire

- Utiliser pour lier les objets entre eux
- Permet de filtrer les recherches (avec le CLI)
- Il faut en abuser :)

# PODs

- **Plus petit élément, composé de un ou plusieurs conteneurs**
- **Les conteneurs d'un pod fonctionnent ensemble et sur un même hôte**
- **Doivent être considérés comme jetable**

- Probes: livenessProbe, readinessProbe
- Lifecycle Hook: preStop, postStart)

# Service

- Matérialisation du service rendu
- Accéssibilité: ClusterIP, NodePort, LoadBalancer, External et Headless
- Les labels font le lien entre les pods et les services
- Accéssibilité: <service name>.<namespace>.svc.cluster.local

# Resources / Usage

- ReplicaSet
- Deployments
- Daemonset
- StatefulSet
- Job / Cronjob

# ReplicaSet

- S’assure que le bon nombre de pod est exécuté et fonctionne

# Deployment

- Description des applications mises en oeuvre sur le cluster
- Fonctionnalité de rollback et de mise à jour
- Gère les ReplicaSets

# Daemonset

**S’assure qu’une copie d’un POD est déployé sur chaque noeud du cluster**

idéal pour les services de type Log Forwarding, Health Monitoring, …

# StatefulSet

- TODO

# Job / CronJob

- **Un ou plusieurs pod s'execute et se termine correctement**
- **Possibilité d'executer les jobs de la même manière que Cron Unix**

Les PODs correspondants aux jobs ne sont pas nettoyés

# Stockage

- Volumes
- PersistentVolumes
- PersistentVolumeClaims
- StorageClass

# Volume

- Stockage qui suit le cycle de vie du POD
- Un POD peut avoir plusieurs types de volumes qui lui attaché
- Les conteneurs d’un même POD ont accés à ce(s) volume(s)

# PersistentVolume

- Représente un type de stockage
- C’est une ressource globale au cluster

# PersistentVolumeClaim

- **Une demande de stockage**
- **Est lié à un PersistentVolume**

Phases: Available, Bound, Released, Failed

# StorageClass

- Représente un type de stockage
- Utilisé pour fournir un stockage externe

# Configuration

- ConfigMap
- Secrets

# ConfigMap

- Gérer la configuration des applicatifs
- Accéssible: variables d’environnement, fichier de configuration, …

# Secret

- Identique que ConfigMap (stockage en base64) pour stocker des données sensibles
- Types: docker-registry (credentials), tls (certificat), Opaque

# Network

- Ingress

# Ingress Resource

- **Routeurs qui permettent d'exposer les services sur des IP publiques externes au cluster.**
- **Définition de règles de routage applicatives (HTTP/HTTPS)**
- **Un Ingress Controller doit être mis en place (NGinx, Traefik, Contour, …)**

# Limit / Quota

- LimitRange
- ResourceQuota

# LimitRange

- **Requêtes minimale, maximale, par défaut pour les pods dans un namespace**

# ResourceQuota

- **Quantité de ressources disponibles pour les pods dans un namespace**

# Security

- NetworkPolicy: restrictions réseau à appliquer
- PodSecurityPolicy ensemble de conditions que les pods doivent remplir pour être acceptés par le cluster

# Mais aussi

- Scaling: HorizontalPodAutoscaler, PodDisruptionBudget
- Cluster State: Node, Cluster, ComponentStatus, Event, …

# Authentification - Autorisation - Droits

# Authentification

- Basic Auth
- Token
- Certificats client

# Service Account

- **Identité sous laquelle sont exécutés les PODs**

- toujours un SA par défaut par namespace (default)

# Role, ClusterRole

- Actions que l’on peut effectuer sur les ressources
- Idem mais pour des ressources de type cluster

# RoleBinding, ClusterRoleBinding

- Qui peut effectuer les actions définies dans un Role ou ClusterRole

CLI et dashboard

# Utilisation

# Kubectl

- CLI qui permet d’interagir avec le cluster
- Multi cluster, multi contextes, multi utilisateur
- Choisir son contexte et interagir avec Kubernetes

# La configuration

```
$ kubectl config get-contexts
CURRENT   NAME                          CLUSTER     AUTHINFO            NAMESPACE
          atmos-stg                     stg         admin-stg
          minikube                      minikube    minikube
*         openstack-atmos-itg           openstack   openstack-fbpl6415  atmos-itg
          openstack-atmos-prod          openstack   openstack-fbpl6415  atmos-prod
          openstack-atmos-prp           openstack   openstack-fbpl6415  atmos-prp
          openstack-dt-tools-monitoring openstack   openstack-fbpl6415  dt-tools-monitoring
          vmware-atmos-dev              sf-prp      sf-prp-fbpl6415     atmos
          vmware-atmos-itg              sf-prod     sf-prod-fbpl6415    atmos-itg
          vmware-atmos-prod             sf-prod     sf-prod-fbpl6415    atmos
          vmware-atmos-prp              sf-prod     sf-prod-fbpl6415    atmos-prp
          vmware-nimbus-itg             sf-prod     sf-prod-fbpl6415    nimbus-itg
          vmware-nimbus-prod            sf-prod     sf-prod-fbpl6415    nimbus
          vmware-nimbus-prp             sf-prod     sf-prod-fbpl6415    nimbus-prp

$ kubectl config use-context openstack-atmos-prod
Switched to context "openstack-atmos-prod".
```

# Les ressources

```
$ kubectl get
You must specify the type of resource to get. Valid resource types include:

  * all
  * certificatesigningrequests (aka 'csr')
  * clusterrolebindings
  * clusterroles
  * componentstatuses (aka 'cs')
  * configmaps (aka 'cm')
  * controllerrevisions
  * cronjobs
  * customresourcedefinition (aka 'crd')
  * daemonsets (aka 'ds')
  * deployments (aka 'deploy')
  * endpoints (aka 'ep')
  * events (aka 'ev')
  * horizontalpodautoscalers (aka 'hpa')
  * ingresses (aka 'ing')
  * jobs
  * limitranges (aka 'limits')
  * namespaces (aka 'ns')
  * networkpolicies (aka 'netpol')
  * nodes (aka 'no')
  * persistentvolumeclaims (aka 'pvc')
  * persistentvolumes (aka 'pv')
  * poddisruptionbudgets (aka 'pdb')
  * podpreset
  * pods (aka 'po')
  * podsecuritypolicies (aka 'psp')
  * podtemplates
  * replicasets (aka 'rs')
  * replicationcontrollers (aka 'rc')
  * resourcequotas (aka 'quota')
  * rolebindings
  * roles
  * secrets
  * serviceaccounts (aka 'sa')
  * services (aka 'svc')
```

# Les services

```
$ kubectl get svc
NAME               TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)             AGE
atmosdb            ClusterIP   10.233.38.105   <none>        80/TCP              39d
doc                ClusterIP   10.233.30.140   <none>        80/TCP,9180/TCP     19h
keystone           ClusterIP   10.233.3.73     <none>        80/TCP              39d
mariadb            ClusterIP   10.233.47.5     <none>        3306/TCP            40d
mariadb-exporter   ClusterIP   10.233.52.59    <none>        9104/TCP            1h
portal             ClusterIP   10.233.18.201   <none>        80/TCP              34d
pushgateway        ClusterIP   10.233.34.165   <none>        80/TCP              40d
scorecards         ClusterIP   10.233.22.47    <none>        80/TCP              34d
```

# Les déploiements

```
$ kubectl get deployment
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
atmosdb            1         1         1            1           39d
doc                1         1         1            1           19h
keystone           1         1         1            1           39d
mariadb            1         1         1            1           40d
mariadb-exporter   1         1         1            1           1h
portal             1         1         1            1           34d
pushgateway        1         1         1            1           40d
scorecards         1         1         1            1           34d
```

# Kubernetes / Dev

# Bonnes pratiques Docker

- **Minimiser** les images Docker (Alpine, Bitnami Minideb, …)
- Utiliser un fichier .dockerignore
- Utiliser le multi-stage
- Minimiser le nombre de **layers**.
- Pas de **root** dans le conteneur
- …

# Bonnes pratiques Kubernetes

- Pas d'image latest
- Utiliser de nombreux labels de description
- Utiliser init-containers
- Utiliser le DNS interne : service.namespace ou service
- Exposer des métriques (Prometheus)
- Utiliser Readiness et liveness
- Mettre en place des limites et des quotas

# References

- CNCF: Cloud Native Computing Foundation
- CRI: Container Runtime Interface
- Docker best practices par Docker Inc
- Kubernetes best practices: Sandeep Dinesh
- Lucas Käldström (schéma d'Architecture)
- Original slide template - Dan Allen & Sarah White

# Nicolas Lamirault

 @nlamirault