istic Informatique Électronique

Kubernetes

understanding Kubernetes

Brice Ekane - (brice.ekane@univ-rennes.fr)

ISTIC Rennes - France

2024-2025

git clone https://github.com/bekane/tlc.git

Cours repris de Pr. Olivier Barais

Kubernetes: What's it do?

Agenda

- What is Kubernetes
- Kubernetes concepts
- Architecture
- Conclusion

Kubernetes VS Docker compose

Docker

- Docker is the container technology that allows you to containerize your applications.
- Docker is the core of using other technologies.

Kubernetes VS Docker compose

Docker Compose

- Docker Compose allows configuring and starting multiple Docker containers.
- Docker Compose is mostly used as a helper when you want to start multiple Docker containers and don't want to start each one separately using docker run ....
- Docker Compose is used for starting containers on the same host.
- Docker Compose is used instead of all optional parameters when building and running a single docker container.
- Docker Swarm

Kubernetes VS Docker compose

Kubernetes

- Kubernetes is a container orchestration tool developed by Google
- Kubernetes' goal is very similar to that for Docker Swarm

Kubernetes VS Docker compose

Kubernetes

- Kubernetes is a container orchestration tool developed by Google
- Kubernetes' goal is very similar to that for Docker Swarm

What is Kubernetes?

- Kubernetes is a container orchestrator like Docker Swarm, Mesos Marathon, Amazon ECS, Hashicorp Nomad. Container orchestrators are the tools which group hosts together to form a cluster, and help us make sure applications:
  - are fault-tolerant,
  - can scale, and do this on-demand
  - use resources optimally
  - can discover other applications automatically, and communicate with each other
  - are accessible from the external world
  - can update/rollback without any downtime.

What is Kubernetes?

- A highly collaborative open source project originally conceived by Google
- Google has 10+ years experience w/ containerized apps
- Sometimes called:
  - kube
  - k8s (that's 'k' + 8 letters + 's')
- Start, stop, update, and manage a cluster of machines running containers in a consistent and maintainable way.

What is Kubernetes?

- Particularly suited for horizontally scalable, stateless, or 'microservices' application architectures.
  - Does not mean others will not work or are ignored
- Additional functionality to make containers easier to use in a cluster (reachability and discovery).
- Kubernetes does NOT and will not expose all of the 'features' of the docker command line.

What is Kubernetes?

- Many people argue that Kubernetes is hard to learn.
- It's because it solves a series of problems and people try to understand without knowing all the prerequisites.
- This makes it complicated.
- Start putting the pieces of the puzzle together by reading about concepts/terms like the following.
- This process will help you understand the kind of problems Kubernetes tries to solve:

What is Kubernetes?

- This process will help you understand the kind of problems Kubernetes tries to solve:
  - 12-factor apps,
  - Automatic binpacking,
  - Self-healing mechanisms,
  - Horizontal scaling,
  - Service discovery and Load balancing,
  - Automated rollouts and rollbacks,
  - Blue-Green deployments / Canary deployments
  - Secrets and configuration management,
  - Storage orchestration
  - And because there are a lot of different things around containers and their management, keep an eye on the Cloud Native landscape: https://landscape.cncf.io/

KUBERNETES: CONCEPTS ET OBJETS

_

- Master
- Minion/Node
- Pod
- Replication Controller
- Deployments
- Ingress et Ingress
- controller
- NetworkPolicy
- Services
- Label
- Namespace

KUBERNETES : Les concepts

Master

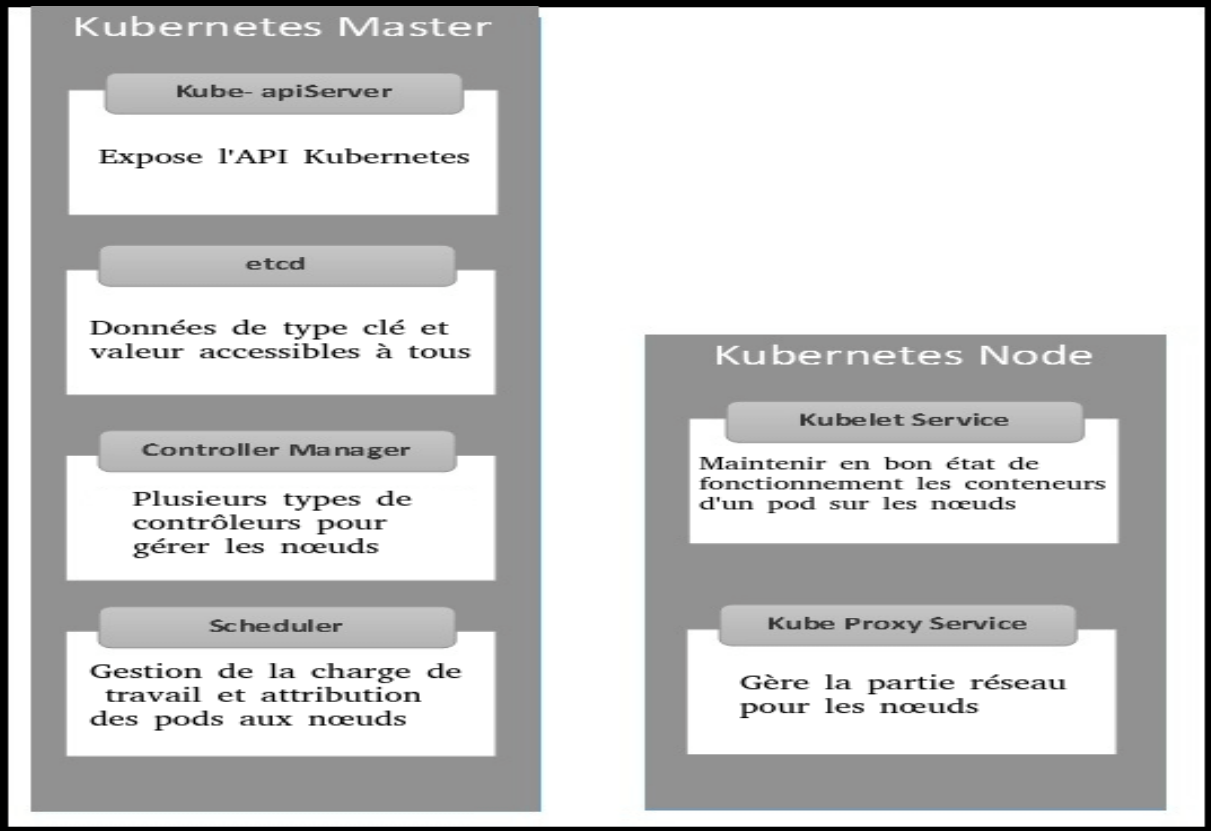
- Typically consists of:
  - kube-apiserver
  - kube-scheduler
  - kube-controller-manager
  - etcd
- Might contain:
  - kube-proxy
  - a network management utility

KUBERNETES : Minion - Node

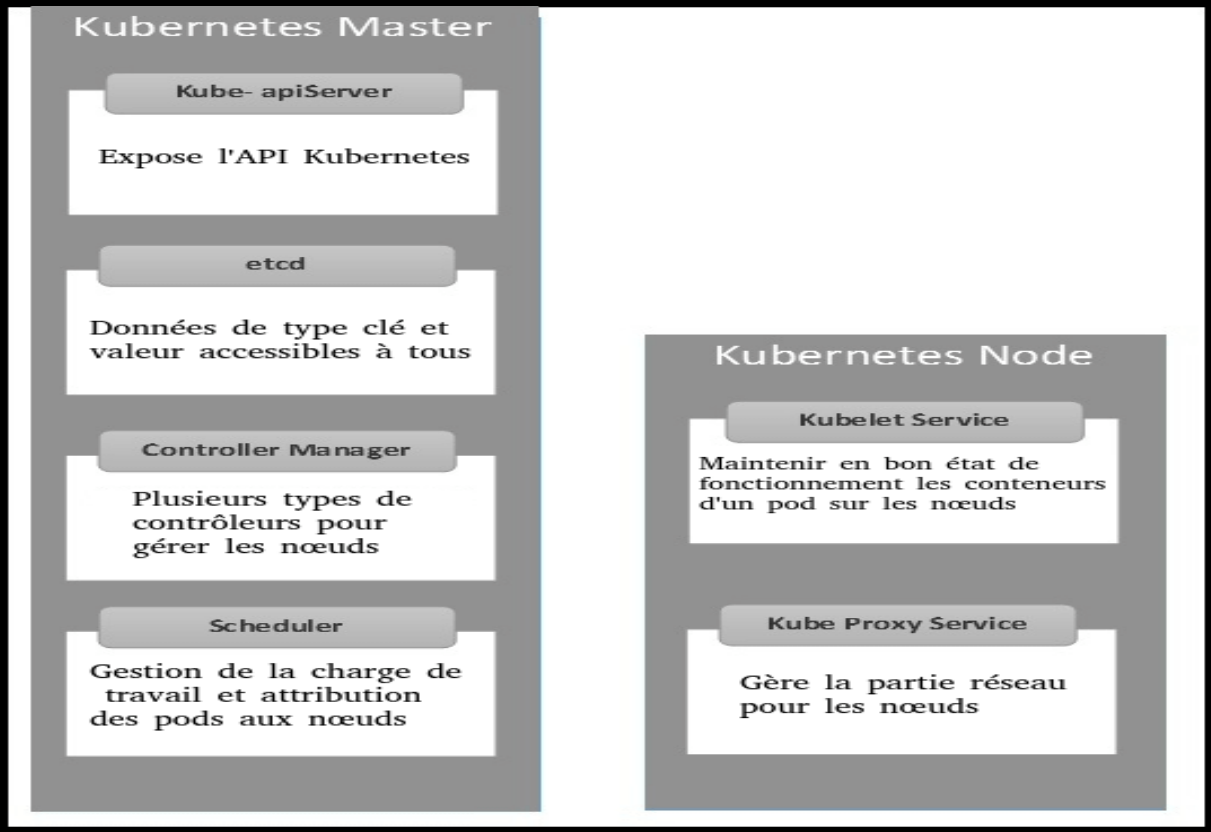
_

- Typically consists of:
  - kubelet
  - kube-proxy
  - cAdvisor
- Might contain:
  - a network management utility
- May be referred to by either name.

KUBERNETES : Systems and Binaries



KUBERNETES : Systems and Binaries



KUBERNETES : POD

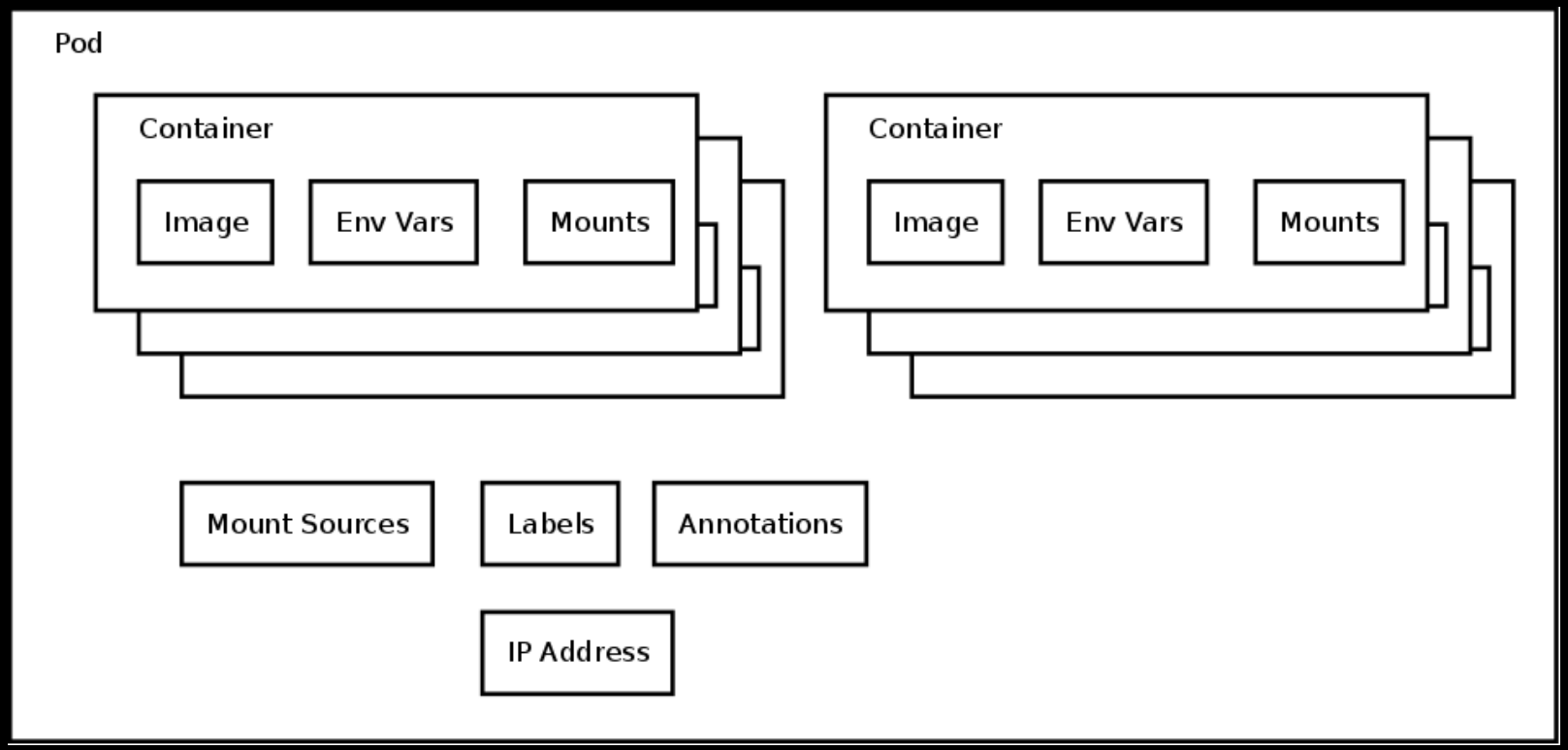
- Ensemble logique composé de un ou plusieurs conteneurs
- Les conteneurs d'un pod fonctionnent ensemble
- (instanciation et destruction) et sont orchestrés sur un même hôte
- Les conteneurs partagent certaines spécifications du POD :
  - La stack IP (network namespace)
  - Inter-process communication (PID namespace)
  - Volumes
- C'est la plus petite unité orchestrable dans Kubernetes

KUBERNETES : POD

_

- Single schedulable unit of work
- Can not move between machines
- Can not span machines
- One or more containers
- Shared network namespace
- Metadata about the container(s)
- Env vars – configuration for the container
- Every pod gets an unique IP
- Assigned by the container engine, not kube!

Pod


Pod
Container
Image
Env Vars
Mounts
Container
Image
Env Vars
Mounts
Mount Sources
Labels
Annotations
IP Address

KUBERNETES : POD

- Les PODs sont définis en YAML comme les fichiers
- docker-compose :

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

KUBERNETES: Replication Controller

_

- Consists of
  - Pod template
  - Count
  - Label Selector
- Kube will try to keep $count copies of pods matching the label selector running
- If too few copies are running the replication controller will start a new pod somewhere in the cluster

KUBERNETES: Replication Controller

KUBERNETES : SERVICES

_

- Abstraction des PODs et RCs, sous forme d'une VIP de service
- Rendre un ensemble de PODs accessibles depuis l'extérieur
- Load Balancing entre les PODs d'un même service

KUBERNETES : SERVICES

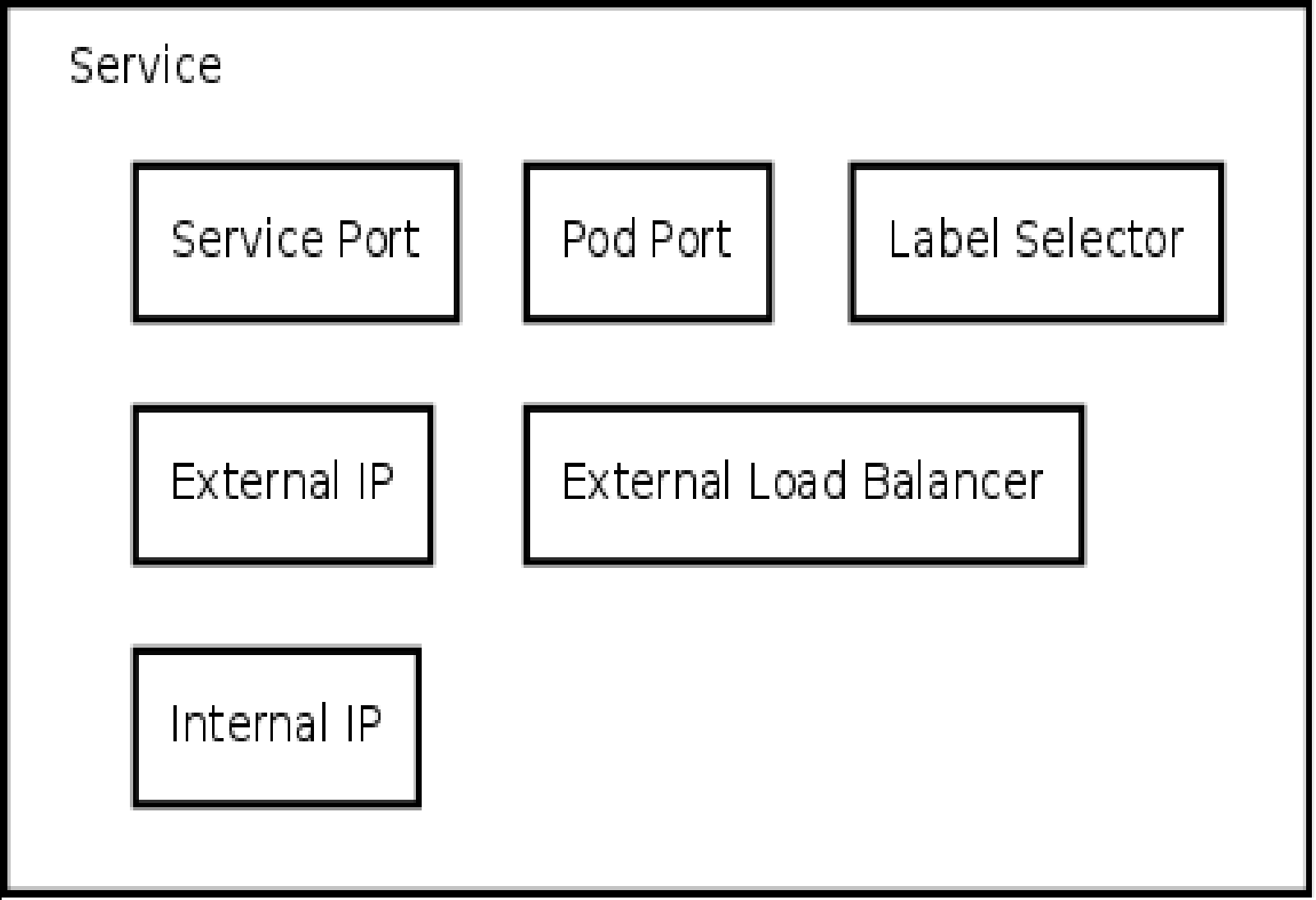
_

- Load Balancing : intégration avec des cloud provider :
  - AWS ELB
  - GCE
- Node Port forwarding : limitations
- ClusterIP : IP dans le réseau privé Kubernetes (VIP)
- IP Externes : le routage de l'IP publique vers le cluster est manuel

KUBERNETES : SERVICES

- How 'stuff' finds pods which could be anywhere
- Define:
  - What port in the container
  - Labels on pods which should respond to this type of request
- Can define:
  - What the 'internal' IP should be
  - What the 'external' IP should be
  - What port the service should listen on

KUBERNETES : SERVICES



KUBERNETES : SERVICES

- Exemple de service (on remarque la sélection sur le label):

```
{
        "kind": "Service",
        "apiVersion": "v1",
        "metadata": {
        "name": "example-service"
        },
        "spec": {
        "ports": [{
        "port": 8765,
        "targetPort": 9376
        }],
        "selector": {
        "app": "example"
        },
        "type": "LoadBalancer"
        }
```

KUBERNETES : LABELS

_

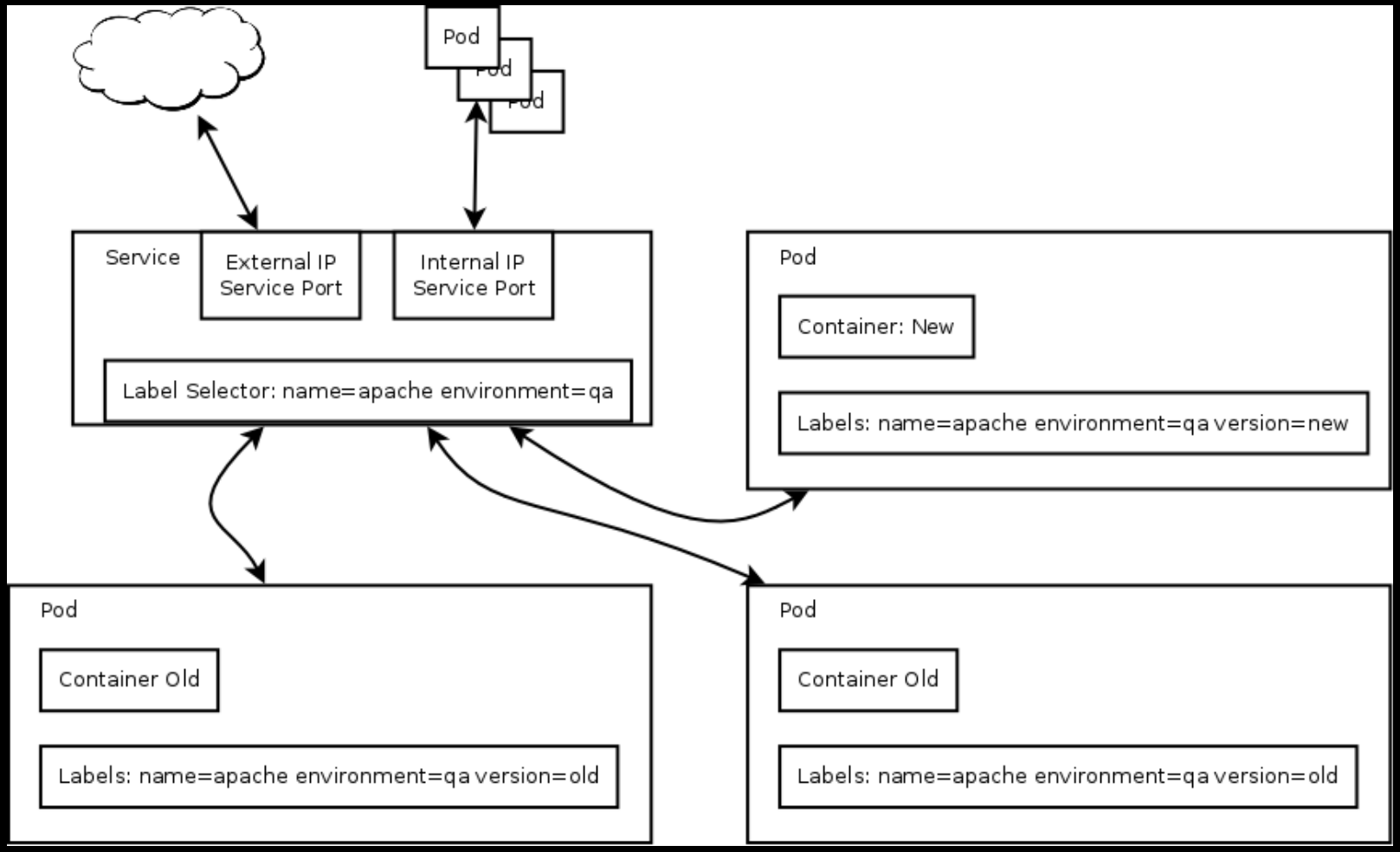
- Système de clé/valeur
- Organiser les différents objets de Kubernetes (PODs, RC, Services, etc.) d'une manière cohérente qui reflète la structure de l'application
- Corréler des éléments de Kubernetes : par exemple un service vers des PODs

KUBERNETES : LABELS

_

- List of key=value pairs
- Attached to all objects
- Currently used in 2 main places:
  - Matching pods to replication controllers
  - Matching pods to services
- Objects can be queried from the API server by label

Services and Labels



KUBERNETES : LABELS

Exemple de label

```
{
    "apiVersion": "v1",
    "kind": "Pod",
    "metadata": {
        "name": "nginx",
        "labels": {
            "app": "nginx"
        }
    },
    "spec": {
        "containers": [{
            "name": "nginx",
            "image": "nginx",
            "ports": [{
                "containerPort": 80
            }]
        }]
    }
}
```

KUBERNETES : NAMESPACES

- Fournissent une séparation logique des ressources par exemple :
  - Par utilisateurs
  - Par projet / applications
  - Autres...
- Les objets existent uniquement au sein d'un namespace donné
  - Évitent la collision de nom d'objets

KUBERNETES : NAMESPACES

- Attached to every object
- Pods in ns1 will not get service variable from ns2
- Users with permission to CRUD objects in ns1 may not have permissions to CRUS object in ns2
- The network is not segregated!

KUBERNETES : VOLUMES

_

- Fournir du stockage persistent aux PODs
- Fonctionnent de la même façon que les volumes Docker pour
  - les volumes hôte :
  - EmptyDir = volumes docker
  - HostPath = volumes hôte
- Support de multiples backend de stockage :
  - GCE : PD
  - AWS : EBS
  - GlusterFS / NFS
  - Ceph
  - iSCSI

KUBERNETES : VOLUMES

- On déclare d'abord le volume et on l'affecte à un service :

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: redis
spec:
  containers:
  - name: redis
    image: redis
    volumeMounts:
    - name: redis-persistent-storage
      mountPath: /data/redis
  volumes:
  - name: redis-persistent-storage
    emptyDir: {}
```

KUBERNETES : NETWORKING

- Les conteneurs peuvent communiquer sans NAT
  - Un nœud peut accéder aux conteneurs des autres nœuds sans NAT
- Nécessite une solution tierce :
  - Canal : Flannel + Calico
  - Weaves
  - OpenShift
  - OpenContrail
  - Romana
- Ces solutions implémentent CNI (Container Network Interface) Spécifications sur la configuration d'interface réseaux des conteneurs

Networking Setup

- Flannel
  - Super super easy configuration
  - Can create a vxlan overlay network
  - Can configure docker to launch pods in this overlay
  - Pods just work!
- There are many other solutions.
  - This one is easy.

KUBERNETES : NETWORKPOLICY

- Contrôle la communication entre les PODs au sein d'un namespace
- Pare-feu entre les éléments composant une application :

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
 name: test-network-policy
 namespace: default
spec:
 podSelector:
  matchLabels:
    role: db
 ingress:
  - from:
     - namespaceSelector:
        matchLabels:
         project: myproject
     - podSelector:
        matchLabels:
         role: frontend
   ports:
```

KUBERNETES : DEPLOYMENTS

- Permet d'assurer le fonctionnement d'un ensemble de PODs
  - Version, Update et Rollback
- Souvent combiné avec un objet de type service

KUBERNETES : INGRESS RESOURCE

- Définition de règles de routage applicatives (HTTP/HTTPS)
  - Traffic load balancing, SSL termination, name based virtual hosting
- Définies dans l'API et ensuite implémentées par un Ingress Controller

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  namespace: default
  name: traefik
  annotations:
    kubernetes.io/ingress.class: "traefik"
spec:
  rules:
    - host: traefik.archifleks.net
      http:
        paths:
          - backend:
              serviceName: traefik-console
              servicePort: 8080
```

KUBERNETES : INGRESS RESOURCE

Exemple

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  namespace: default
  name: traefik
  annotations:
    kubernetes.io/ingress.class: "traefik"
spec:
  rules:
    - host: traefik.archifleks.net
      http:
        paths:
          - backend:
              serviceName: traefik-console
              servicePort: 8080
```

KUBERNETES : INGRESS CONTROLLER

_

- Routeur applicatif de bordure (L7 Load Balancer)
- Implémente les Ingress Resources
- Plusieurs implémentations :
  - Traefik (`https://traefik.io/`)
  - nginx

KUBERNETES : RUN ET ADMINISTRATION

_

- WebUI (Kubernetes Dashboard)
- Kubectl (Outil CLI)
- Objets: Secret et ConfigMap : paramétrages, plus sécurisés que les variables d'environnements
  - Vault et consul

KUBERNETES : AUJOURD'HUI

- Version 1.19 : stable en production
- Solution complète et une des plus utilisées
- Éprouvée par Google

KUBERNETES : ARCHITECTURE

KUBERNETES : COMPOSANTS

_

- Kubernetes est écrit en Go, compilé statiquement.
- Un ensemble de binaires sans dépendances
- Faciles à conteneuriser et à packager
- Peut se déployer uniquement avec des conteneurs sans dépendances d'OS

KUBERNETES : COMPOSANTS

_

- **kube-apiserver** : API server qui permet la configuration d'objet Kubernetes (Pods, Service, Replication Controller, etc.)
- **kube-proxy** : Permet le forwarding TCP/UDP et le load balancing entre les services et les backend (Pods)
- **kube-scheduler** : Implémente les fonctionnalités de scheduling
- **kube-controller-manager** : Responsable de l'état du cluster, boucle infinie qui régule l'état du cluster afin d'atteindre un état désiré

KUBERNETES : COMPOSANTS

_

- **kubelet** : Service "agent" fonctionnant sur tous les nœuds et assure le fonctionnement des autres services
- **kubectl** : Client qui permet de piloter un cluster Kubernetes

KUBERNETES : KUBELET

_

- Service principal de Kubernetes
- Permet à Kubernetes de s'auto configurer :
  - Surveille un dossier contenant les manifests (fichiers YAML des différents composant de K8s).
  - Applique les modifications si besoin (upgrade, rollback).
  - Surveille l'état des services du cluster via l'API server (kube-apiserver).
- Dossier de manifest sur un noeud master :
  - /etc/kubernetes/manifests/

KUBERNETES : KUBELET

- Exemple du manifest kube-proxy :

```
apiVersion: v1
kind: Pod
metadata:
  name: kube-proxy
  namespace: kube-system
  annotations:
    rkt.alpha.kubernetes.io/stage1-name-override: coreos.com/rkt/stage1-fly
spec:
  hostNetwork: true
  containers:
  - name: kube-proxy
    image: quay.io/coreos/hyperkube:v1.3.6_coreos.0
    command:
    - /hyperkube
    - proxy
    - --master=http://127.0.0.1:8080
    - --proxy-mode=iptables
    securityContext:
```

KUBERNETES : KUBE-APISERVER

_

- Les configurations d'objets (Pods, Service, RC, etc.) se font via l'API server
- Un point d'accès à l'état du cluster aux autres composants via une API REST
- Tous les composants sont reliés à l'API server

KUBERNETES : KUBE-SCHEDULER

_

- Planifie les ressources sur le cluster
- En fonction de règles implicites (CPU, RAM, stockage disponible, etc.)
- En fonction de règles explicites (règles d'affinité et anti- affinité, labels, etc.)

KUBERNETES : KUBE-PROXY

_

- Responsable de la publication de services
- Utilise iptables
- Route les paquets à destination des PODs et réalise le load balancing TCP/UDP

KUBERNETES : KUBE-CONTROLLER-MANAGER

_

- Boucle infinie qui contrôle l'état d'un cluster
- Effectue des opérations pour atteindre un état donné
- De base dans Kubernetes : replication controller, endpoints controller, namespace controller et serviceaccounts controller

KUBERNETES : NETWORK-POLICY-CONTROLLER

_

- Implémente l'objet NetworkPolicy
- Contrôle la communication entre les PODs
- Externe à Kubernetes et implémenté par la solution de Networking choisie :
  - OpenShift
  - OpenContrail
  - Romana
  - Calico

KUBERNETES : CONCLUSION

KUBERNETES : CONCLUSION

- **Node** : Un node ("noeud" en fr) est une machine de travail du cluster Kubernetes. Ce sont des unités de travail qui peuvent être physiques, virtuelles mais aussi des instances cloud.
- **Pod** : Il s'agit de l'unité la plus petite de K8s, un pod encapsule le ou les conteneur(s) formant votre application conteneurisée partageant ainsi la même stack réseau (chaque pod se voit attribuer une adresse IP unique) et le même stockage, plus précisément un volume partagé (tous les conteneurs du pod peuvent accéder aux volumes partagés, ce qui permet à ces conteneurs de partager plus facilement des données).

KUBERNETES : CONCLUSION

_

- **Replicas** : c'est le nombre d'instances d'un Pod ("réplique" en fr)
- **ReplicaSet** : s'assure que les réplicas spécifiés sont actifs
- **Deployment** : défini l'état désiré et fournit des mises à jour déclaratives de vos Pods et ReplicaSets.
- **Service** : Un service peut être défini comme un ensemble logique de pods exposés en tant que service réseau. C'est un niveau d'abstraction au-dessus du pod, qui fournit une adresse IP et un nom DNS unique pour un ensemble de pods. Avec les Services, il est très facile de gérer la configuration de Load Balancing

Communication avec le cluster

_

- kubeclt (ligne de commande)
- Dashboard (interface web)
- API (Rest)