CentraleSupélec

TP KUBERNETES

kubernetes

Initiation aux conteneurs et à leur déploiement

WRAP UP DOCKER

Historique, vous-souvenez vous de l'année de sortie de Docker ? Et de son fondateur ?

A quoi sert Docker ?

Est-ce que vous vous souvenez des 2 grandes fonctions du kernel Linux permettant d'implémenter les containers ?

Immutabilité des containers et persistance : qu'avez vous retenu dans le TP ?

Remarques sur la construction / build des images ?

- ADD et UNZIP https://docs.docker.com/engine/ref

Historique

Développé à l'origine par Google (évolution de BORG)

Nette accélération du projet en 2014 au passage en Open Source

Stagnation de l'offre fermée Swarm

Appropriation de Kubernetes par Google, Azure et AWS dans leurs offres cloud

Intégration de Kubernetes dans les offres PaaS (OpenShift, VMware, Rancher, Cloud Foundry, Scaleway, OVH, Digital Ocean…)

Kubernetes first commit : https://github.com/kubernetes/kubernetes/commit/2c4b3a562ce34cddc3f8218a2c4d11c7310e6d56
Timeline de l'histoire de Kubernetes : https://blog.risingstack.com/the-history-of-kubernetes/

Historique

kubernetes
Terme de recherche

apache mesos
Terme de recherche

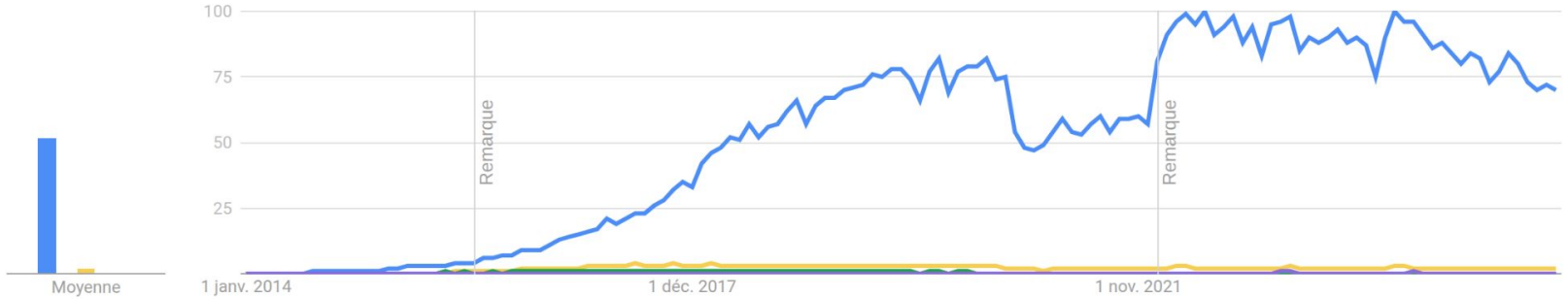
docker swarm
Terme de recherche

pivotal cloud foun...
Terme de recherche

hashicorp nomad
Terme de recherche

Dans tous les pays | 01/01/2014 – 17/07/2025 | Informatique et électronique | Recherche sur le Web

Évolution de l'intérêt pour cette recherche



Pourquoi un orchestrateur ?

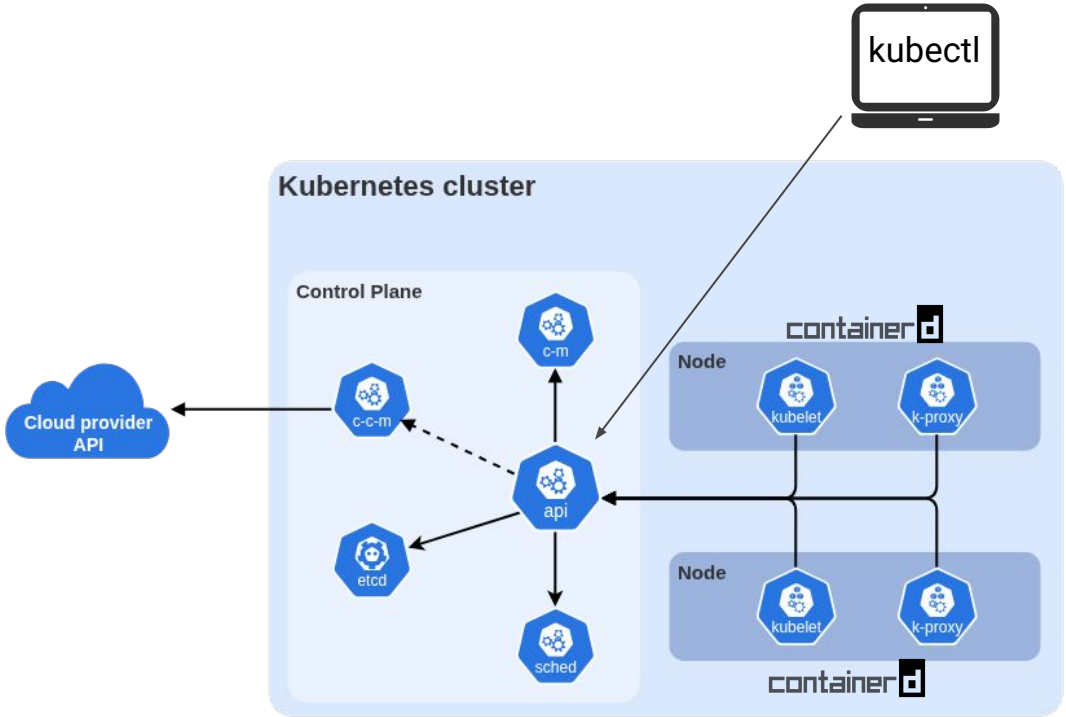
- Gestion complète du cycle de vie des conteneurs par des API
  - Création
  - Relance en cas d'erreur
  - Scalabilité et haute disponibilité (déploiement sur plusieurs nodes)
  - Mise à jour des applications et stratégie de déploiement (rollingOut strategy)
  - https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
- Gestion de la réconciliation entre l'état désiré et l'état réel/courant
  - Utilisation de configurations déclaratives
  - https://kubernetes.io/docs/concepts/overview/working-with-objects/object-management/

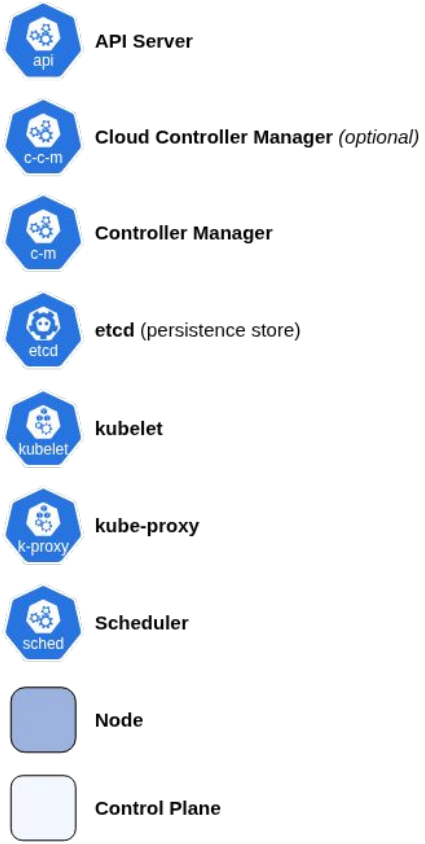
CentraleSupélec

KUBERNETES

Principes de fonctionnement

Architecture générale





https://kubernetes.io/docs/concepts/overview/components/

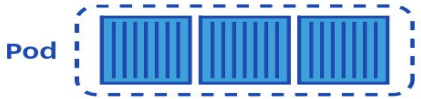
Les objets / concepts Kubernetes

- Pod
- Service
- Deployment / replica set / stateful set / jobs
- ConfigMaps / Secrets / volumes claim
- Label
- *(Et tous les autres : quotas, rbac, netpol, endpoints, crd, psp, ingress, …)*

Pod

La plus petite unité de déploiement Kubernetes

- Posé sur un Node
- Contient 1 ou plusieurs containers
- Mappé avec des ressources du cluster
  - Adresse IP
  - Volumes (secret, configMap, PVC)



Ne dispose que d'une seule adresse IP pour tous les containers

Communication entre container via localhost sur les ports

- Attention au conflit de port !

Le Pod est décrit dans un PodTemplate

Multi-containers : SideCar container (proxy, log forwarder, …) ; initContainer
Anti-pattern : Webserver & Database (cycles de vie différent)

Service

Un Service est une abstraction réseau permettant d'accéder à un ou plusieurs pods (exemple ferme de serveurs web) qui sont exposés par ce service.

Le nombre de Pods peut varier mais le Service reste unique et ne change pas (l'implémentation est en général réalisée par iptables)

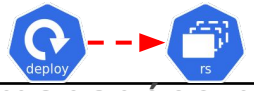
- Le service est mis à jour dynamiquement par Kubernetes pour qu'il répartisse les requêtes entrantes vers les différents pods qu'il référence. Dès qu'un pod disparaît, le service ne lui envoie plus de requêtes (idem s'il n'est pas ready) - **/!\ dépendance au control plane**

3 types de services :

- NodePort
- ClusterIP
- LoadBalancer (uniquement chez les cloud providers)

Le service fonctionne avec des sélecteurs (selector) et des labels (cf. label) pour identifier les pods vers lesquels il doit transmettre les requêtes.

Deployment / replica set



Prend en charge le déploiement de ressources qui sont managées par le control plane

Le *Deployment* va gérer les replicaSets et permet les opérations :

- Création de ReplicaSet suivant les RollingOut strategy
- Mises à jour et rollback (mode canary, blue/green, legacy)

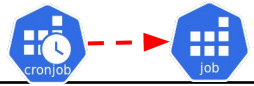
Le deployment peut être “scalé“ automatiquement par un HPA (Horizontal Pod Autoscaler) qui prend en compte la charge CPU/RAM.

Le replica set est un ensemble de Pods de nature identique qui s'exécutent en même temps (par exemple une ferme de serveurs web)

L’état désiré du ReplicaSet (nombre de pods) est maintenu par Kubernetes

- Surveille l’état réel
- Si différence entre le réel et le souhaité, alors ajustement automatique

stateful set / jobs



Un stateful set permet de créer un ensemble de pod ayant chacun :

- Un nom fixe
- Un volume qui sera toujours attaché à ce pod

Ceci permet de réaliser, par exemple, des clusters de base de données ayant chacun leur volume. La synchronisation des données se faisant au niveau applicatif.

Un deployment ne permet que de partager un même volume entre tous les pods du replicaSet.

Un job est une instanciation d'un pod qui est arrêté une fois son exécution terminée. (Le lancement de jobs peut être schédulé via les KubeCronJobs)

Dans un deployement (replica|stateful set) un pod n'a jamais de fin, s'il se termine, l'orchestrateur le relance.

NameSpaces

Un namespace permet d'héberger plusieurs projets au sein d'un même cluster avec une isolation totale ou partielle

La plupart des ressources Kubernetes sont déployées dans un namespace (Namespace Scoped ressources) et sont soumises aux règles définies par l'administrateur (Quotas, limites)

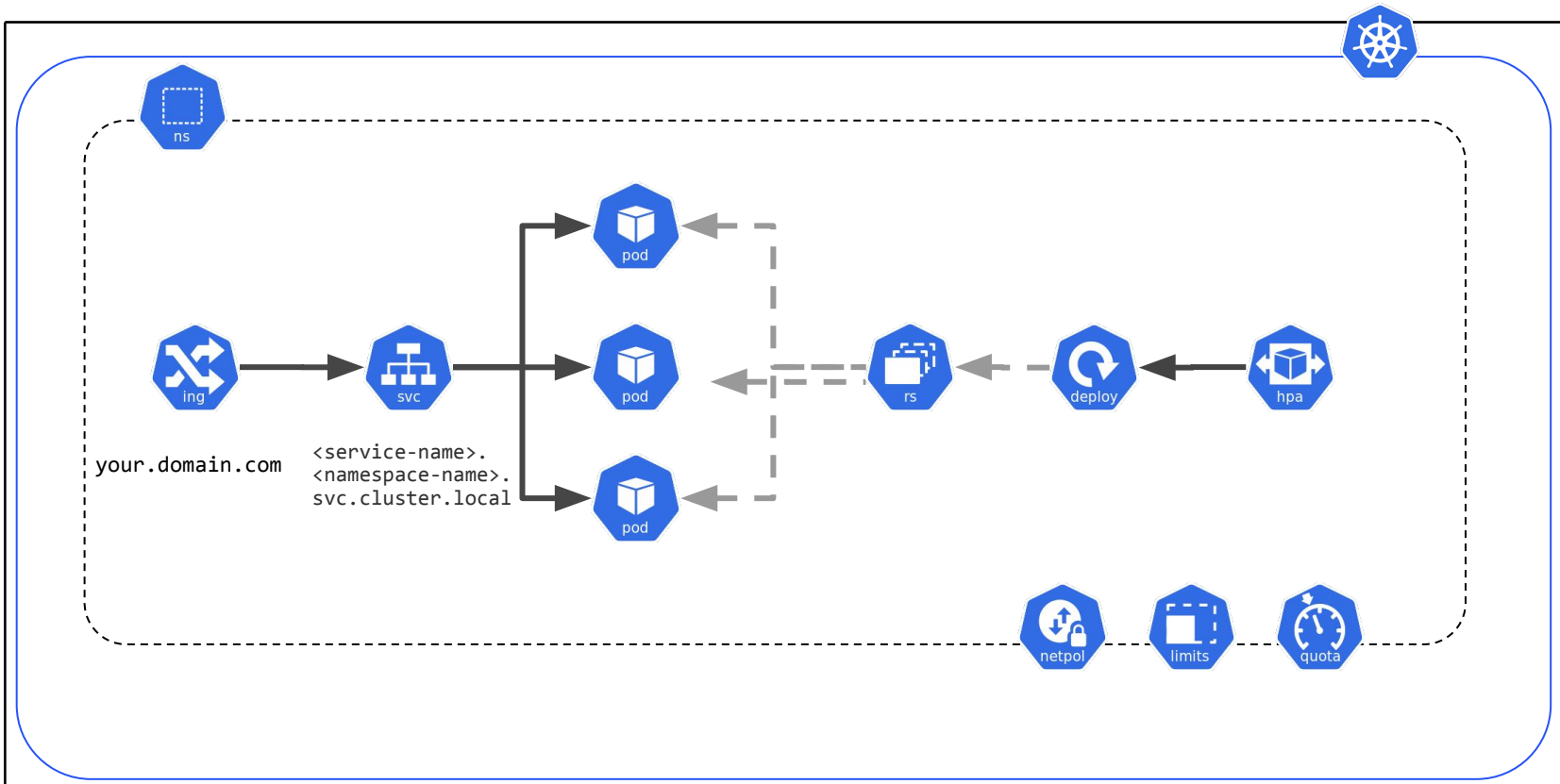
Il est possible de gérer les flux réseaux entre les namespaces via des networkPolicies (qui correspondent à un firewall), les fonctionnalités disponibles dépendent du plugin réseau que vous utilisez (CNI)

à noter : les services sont accessibles via des record DNS internes liés au namespaces :

- `<service-name>.<namespace-name>.svc.cluster.local`
- if a container just uses `<service-name>`, it will resolve to the service which is local to a namespace.

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

Exposed Application

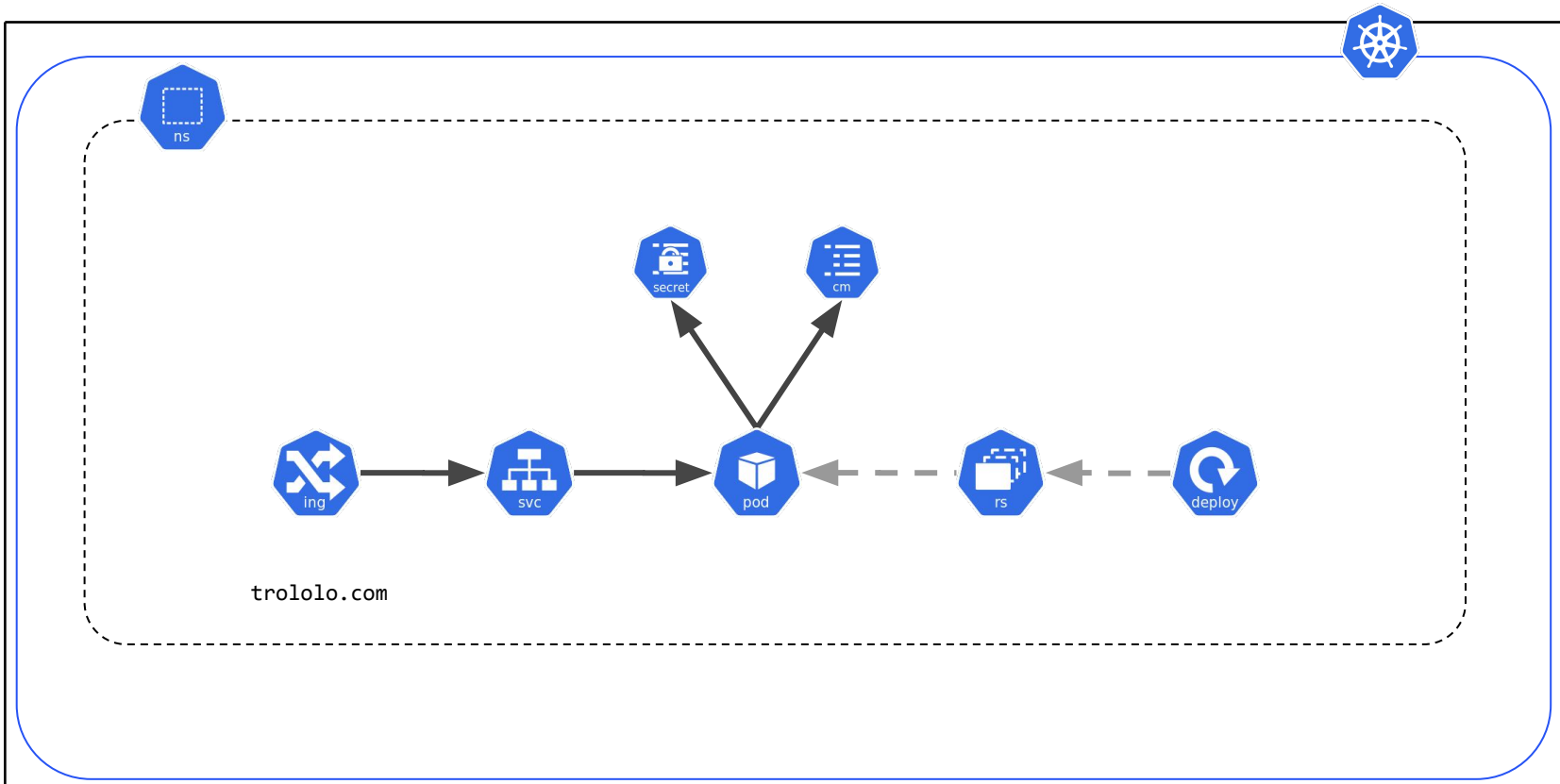


Volumes

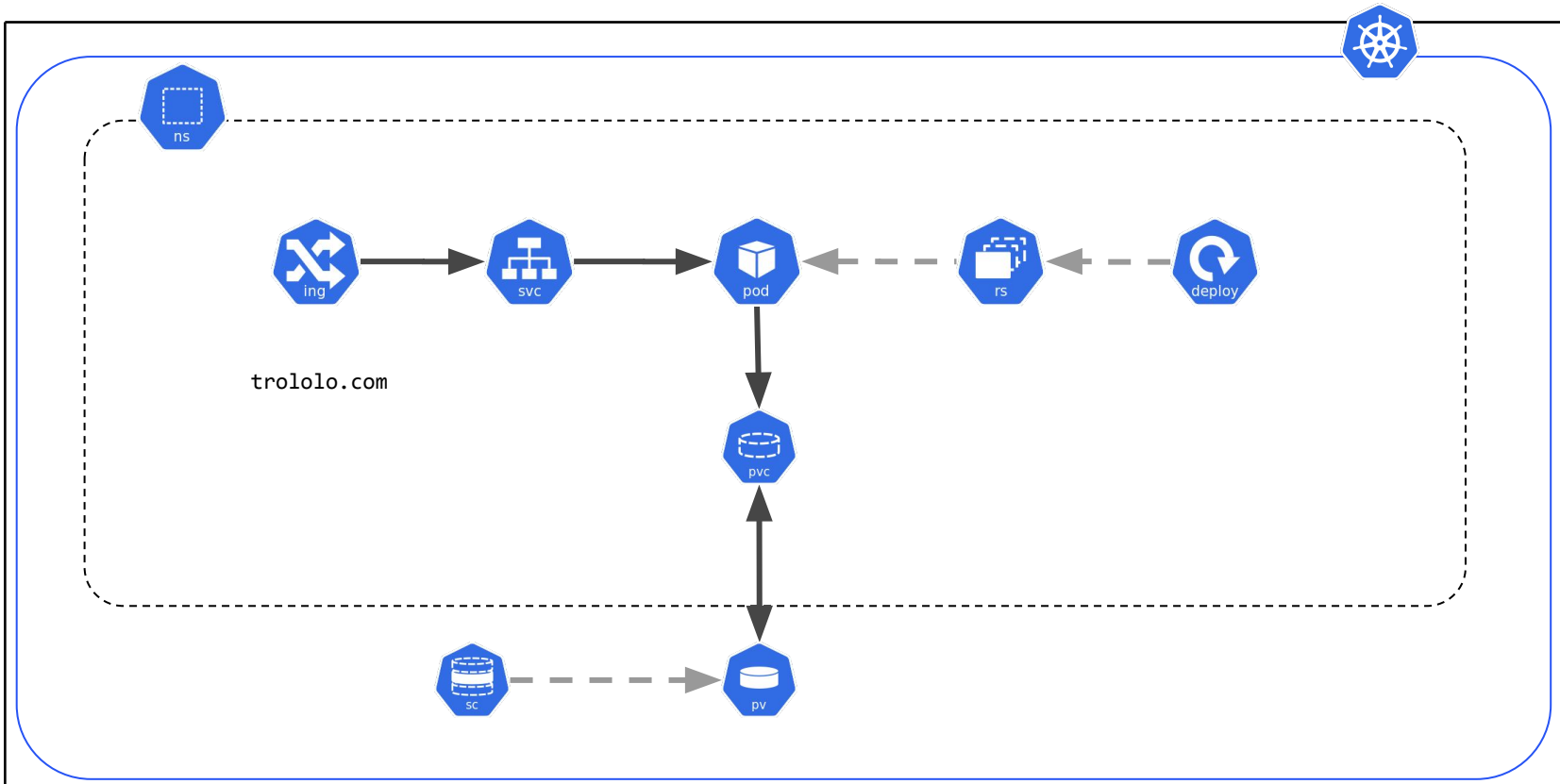
Afin de stocker des données de configuration, données sensibles, données applicatives, Kubernetes proposent plusieurs concepts :

- ConfigMap pour stocker les données de configuration
  - Ensemble de clef-valeur (une valeur peut représenter un fichier complet)
- Secrets pour stocker en base64 des données sensibles
  - sécurité limitée à l'intérieur du cluster car les admins peuvent voir les valeurs
  - Les secrets peuvent être associés aux serviceAccounts (et automatiquement montés dans des pods)
- PersistentVolumeClaim pour disposer d'un volume de stockage sur un backend (NFS, GlusterFS, CEPH, …) hébergeant les persistentVolumes
  - Le PVC référence une storageClass (qui pilote le backend de storage compatible)
  - Les utilisateurs manipulent et référencent uniquement les PVC pour demander un volume et l'attacher à un pod.
  - Le claim référence une storageClass et crée lui même les volumes lors du scale
- EmptyDir pour stocker des données volatiles
  - Il est également possible d'écrire dans un répertoire du conteneur, les données seront perdues à chaque redémarrage du pod.

Application with configuration



Application with persistent storage



Label

Un qualificateur textuel (clef/valeur) permettant de repérer un/des objets Kubernetes

La sélection d'objets se fait par des équations logiques sur les Labels. En effet, les ressources sont éphémères et ont souvent des noms avec des UID (exemple pod)

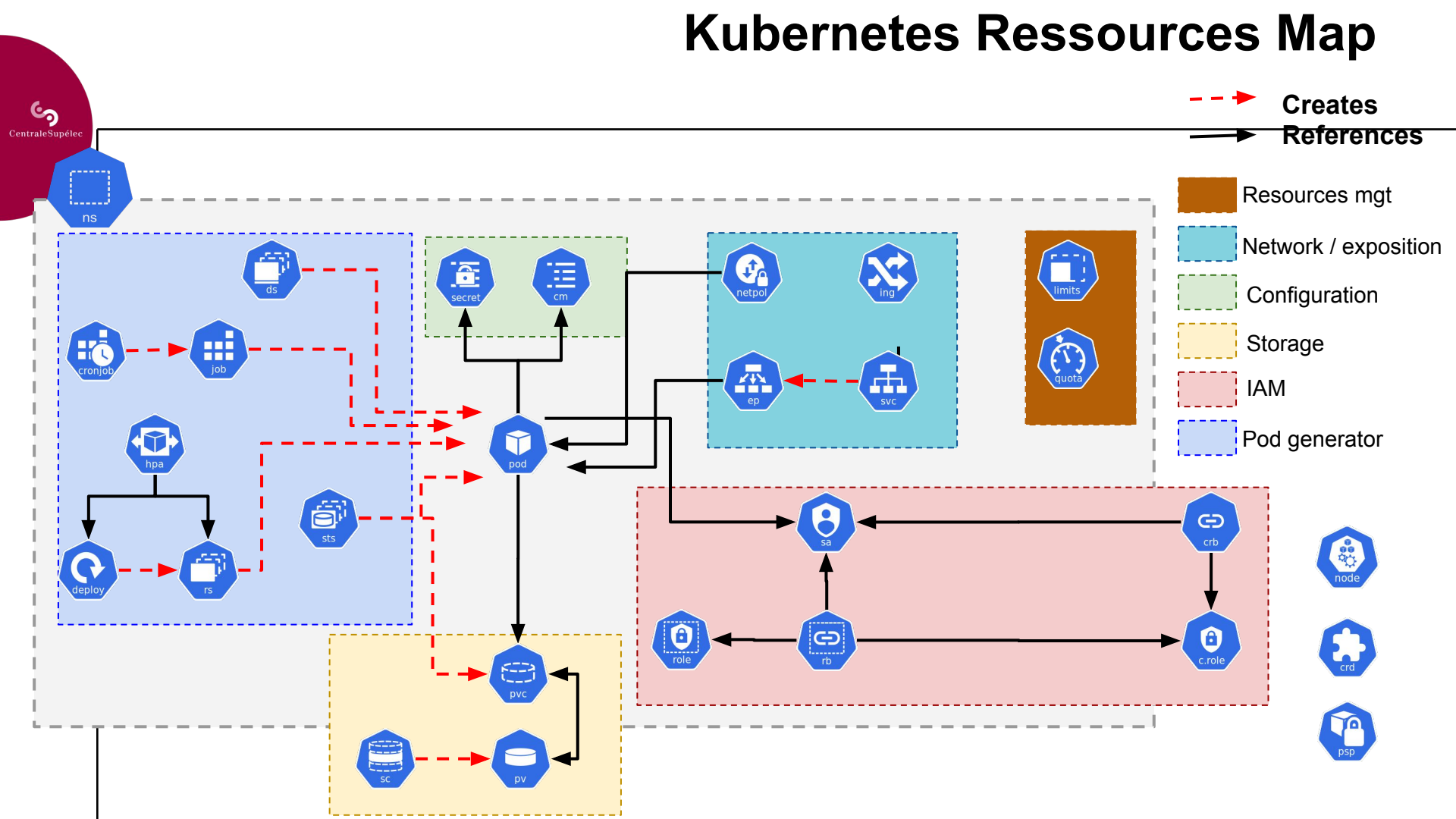
```
template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
```

La Sélection se fait par les opérateurs logiques

- par égalité : = != et par combinaison logique : && ,
- par appartenance à un ensemble : in, notin, exists

```
environment = production / environment in (production, qa) /
tier != frontend  / tier notin (frontend, backend) / partition /
!partition
kubectl get pods -l app=bgd
```

Kubernetes Ressources Map



Composants additionnels

N’oubliez pas : par défaut, le code de kubernetes de fournit pas certains composants indispensables ou optionnels, exemples :

Indispensables :

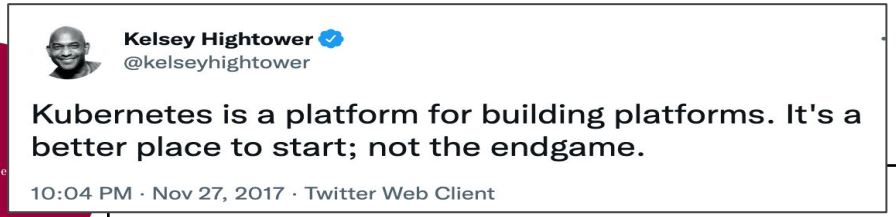
- Network (CNI) : Cilium, Calico
- Container registry : Harbor
- DNS : CoreDNS

Optionnels :

- Ingress Controller : Contour, Nginx, HAproxy
- Service Mesh : LinkerD, Istio
- Logs : Fluentd
- Metrics : Prometheus

L’ensemble du paysage autour de Kubernetes (cloud native landscape), attention on peut vite s’y perdre : https://landscape.cncf.io/

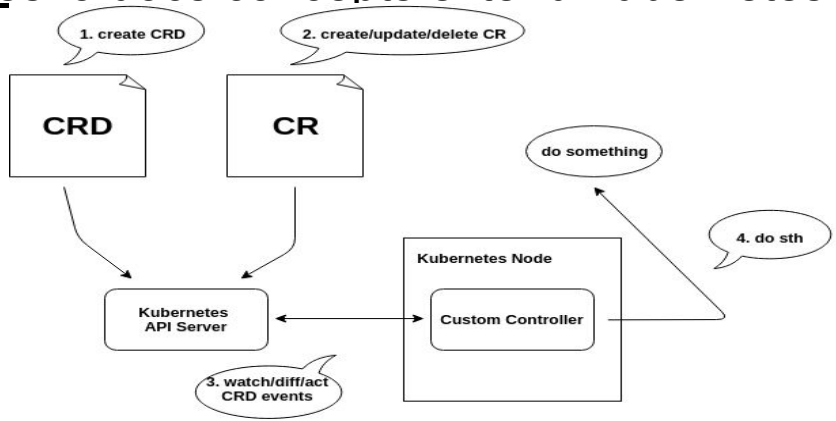
Extension de Kubernetes



Une des grandes forces de Kubernetes est son extensibilité

La manière la plus puissante d'étendre Kubernetes : CustomResourcesDefinition + controller (aussi appelés operator)

https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/

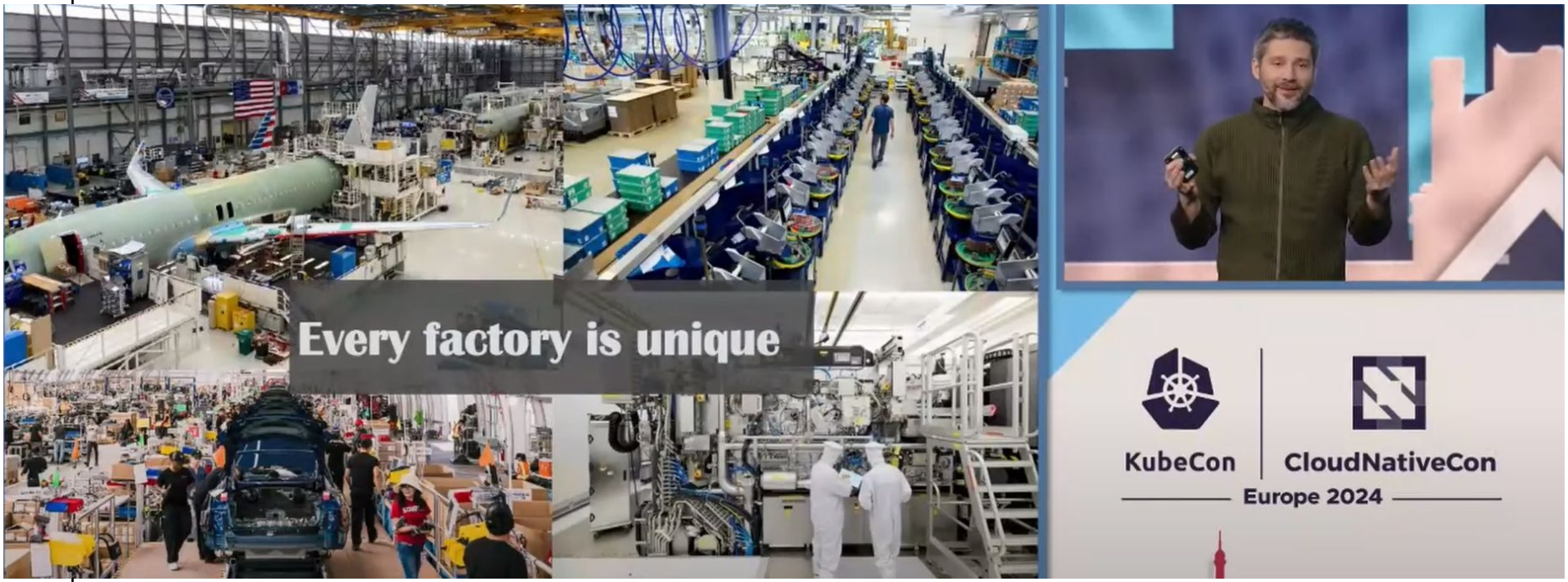


https://michalswi.medium.com/introduction-to-kubernetes-api-the-way-to-understand-the-concept-of-kubernetes-operators-ed667385caf4

Building platform to build products

https://kccnceu2024.sched.com/event/1YhKQ

Building platform to build products



Questions

Annexes - Dimensionnement / Déploiement

Types de déploiements :

- Cloud service Provider
- OnPremises
  - VMs (KVM, VMware ESXi, NUTANIX AHV, …)
  - BareMetal
- L'installer peut provisionner l'infra (calls API) ou alors on utilise des outils comme Terraform
- HA nécessite 3 noeuds etcd (idéalement répartis dans 3 zones/DC)

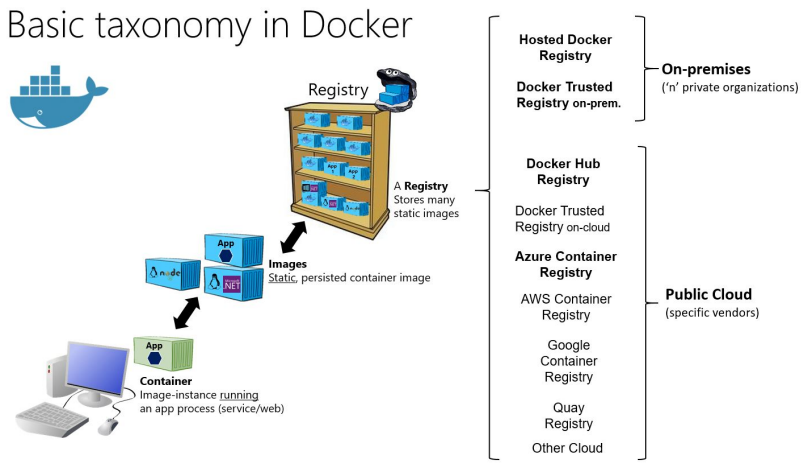
Architecture : possibles critères de segmentation des clusters (non exhaustif) :

- 1 cluster pour un projet / application
  1 cluster par environnement/stage (DEV,PROD)
- 1 cluster par zone/niveau de sécurité
- 1 cluster par département métier
- 1 cluster par type de workload (stateful, legacy, monster pods, …)

Sizing et types de workers :

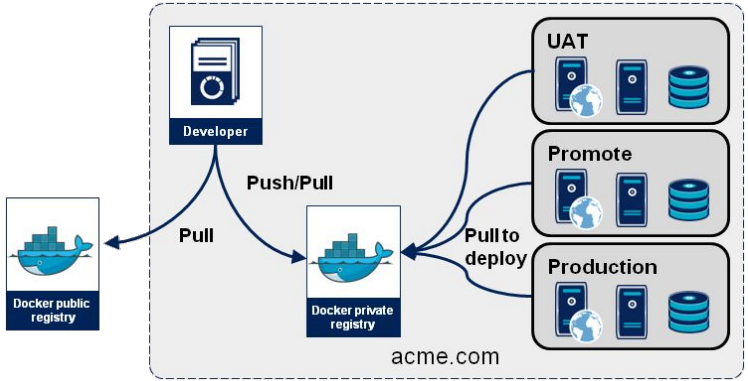
- clusters avec un seul type de workers ou mixte
- Workers spécifiques (GPU, …) – utilisation des nodes selectors (https://kubernetes.io/docs/concepts/scheduling-eviction/assign-pod-node/)
- Pas plus de 110 pods par worker (https://kubernetes.io/docs/setup/best-practices/cluster-large/)

Annexes - container registry



https://docs.microsoft.com/fr-fr/dotnet/architecture/microservices/container-docker-introduction/docker-containers-images-registries

https://blog.octo.com/en/docker-registry-first-steps/

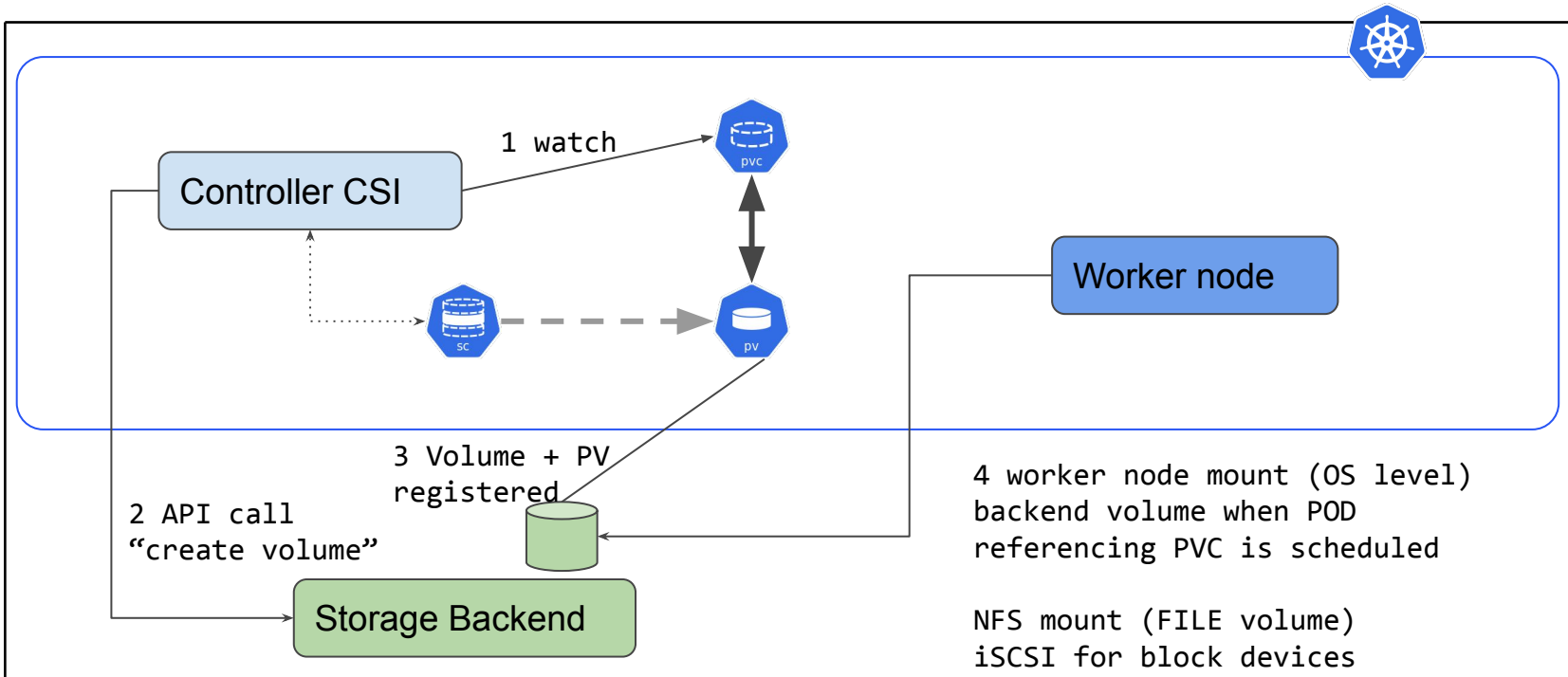


Annexes - Kubernetes en entreprise

Les entreprises grands comptes vont rechercher certaines propriétés dans une solution Kubernetes (qu'on ne trouve pas simplement dans les distributions "Vanilla")

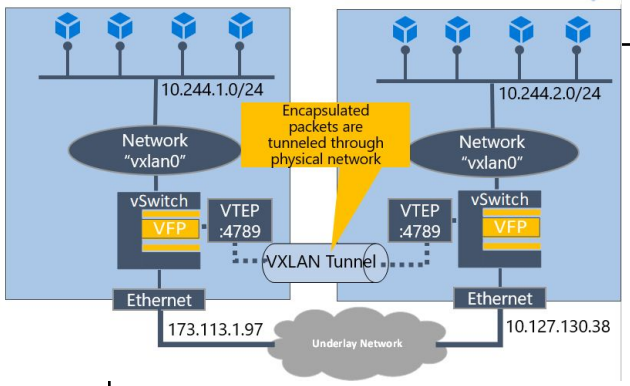
- Features Sécurité
  - Scans d'images docker au niveau du registry
  - Possibilité d'installation "offline" (sans connexion INTERNET) avec registry interne et système de mirroring
  - Hardening des OS et des règles Kubernetes (RBAC, NetworkPolicies, SCC/Gatekeeper, …)
- Intégration et maintenance des composants plates-forme
  - Plugin réseau CNI
  - Ingress Controller
  - Logging / Monitoring
  - CI/CD pour build des images + registry
  - … (cf. cloud native landscape)
- Hybrid Cloud
  - Installation OnPremises et sur cloud Provider
  - Console et outillage de management multi-clusters sur différentes infrastructures
- Exemples de solutions :
  - RedHat Openshift
  - VMware Tanzu
  - GKS OnPrem (Anthos)

Annexes - CSI storage



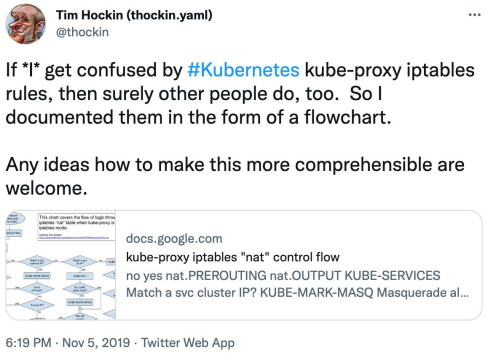
Annexes - CNI and networking

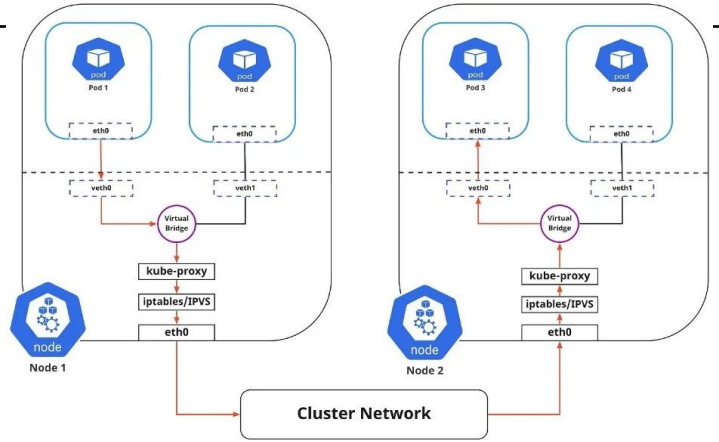
https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/



Overlay Network
(cluster/service/pods network)

https://argonsys.com/microsoft-cloud/library/introducing-kubernetes-overlay-networking-for-windows/





Pod to service communication (kube-proxy)

https://opensource.com/article/22/6/kubernetes-networking-fundamentals

https://itnext.io/kubernetes-networking-behind-the-scenes-39a1ab1792bb

Play with iptables to see what kube-proxy does :

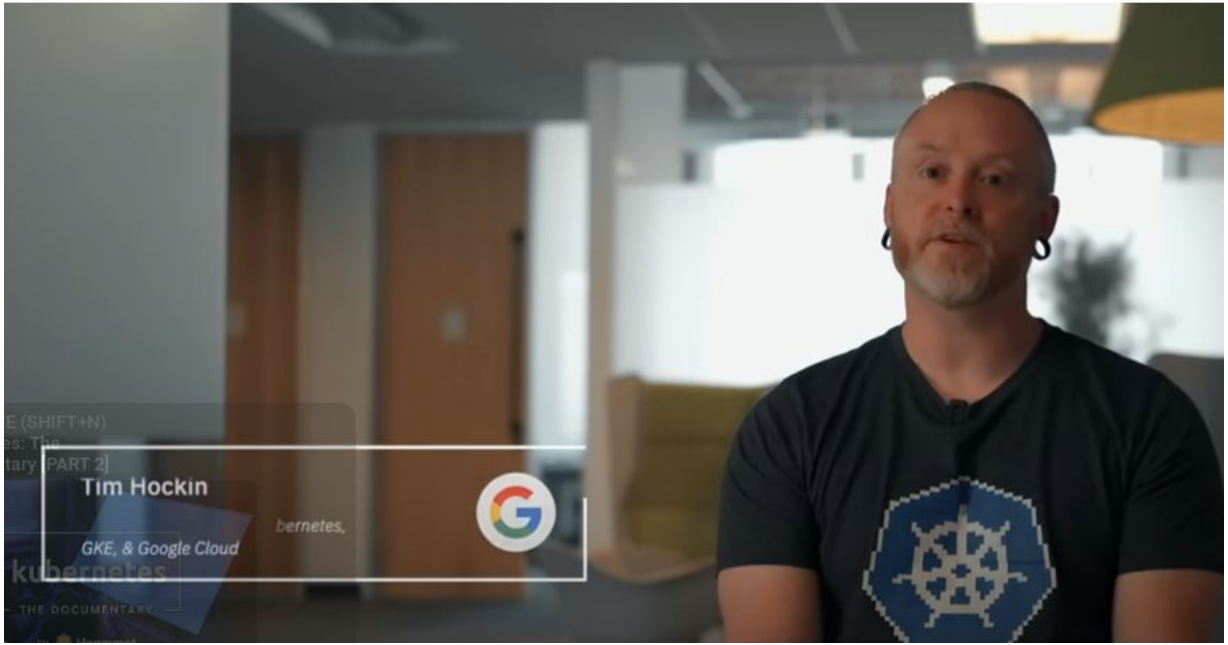
```
sudo iptables -t nat -L KUBE-SERVICES -n  | column -t
```

https://ronaknathani.com/blog/2020/07/kubernetes-nodeport-and-iptables-rules/

https://docs.google.com/drawings/d/1MtWL8qRTs6PlnJrW4dh

Annexes - Kube documentary

https://www.youtube.com/watch?v=BE77h7dmoQU PART 1
https://www.youtube.com/watch?v=318eIlq37PE PART 2



CentraleSupélec