KUBERNETES

IDEO-LAB & Lockheed LLC

NOVEMBER 2024

Author : Guillaume Oneill

AGENDA GENERAL

- ECOSYSTEME
- PRESENTATION
- OPENSHIFT

IDEO·LAB

C’est Quoi ?

🎤 Pitch oral (explication à des débutants)

> “Kubernetes, souvent abrégé *K8s*, c’est un peu le chef d’orchestre du monde des conteneurs.
>
> Imagine qu’on ait des dizaines ou des centaines d’applications tournant dans des conteneurs — par exemple des images Docker. Sans organisation, ce serait vite le chaos : il faudrait lancer chaque conteneur à la main, surveiller les pannes, gérer les mises à jour, la charge, le réseau, la sécurité, etc.
>
> Kubernetes automatise tout cela.
> C’est une **plateforme d’orchestration** qui s’assure que les applications sont toujours disponibles, correctement réparties sur les serveurs, et qu’elles peuvent monter ou descendre en charge automatiquement.
>
> Techniquement, il répartit les conteneurs sur des “nœuds” (les serveurs du cluster), les surveille, les relance s’ils tombent, et expose tes applications vers l’extérieur via des services ou des points d’entrée (Ingress).
>
> On parle d’un système **déclaratif** : tu décris dans un fichier YAML l’état que tu veux (“je veux 3 instances de mon API Django”) et Kubernetes se charge de le maintenir.
>
> C’est aujourd’hui le standard mondial pour déployer des applications modernes, scalables et résilientes, qu’elles tournent sur le cloud (AWS, GCP, Azure) ou sur des serveurs privés.
>
> En résumé, Kubernetes c’est la **tour de contrôle** des applications conteneurisées : il orchestre, surveille, ajuste et automatise — pour que tout tourne sans interruption.”

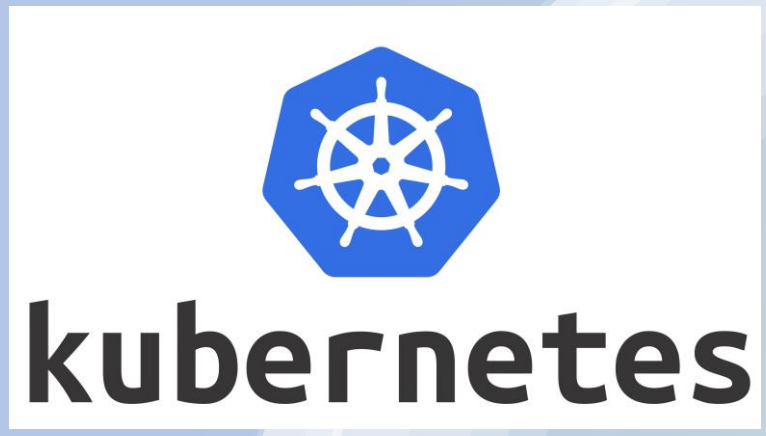
IDEO·LAB

Pour les Pro

💼 Pitch professionnel (1 min)

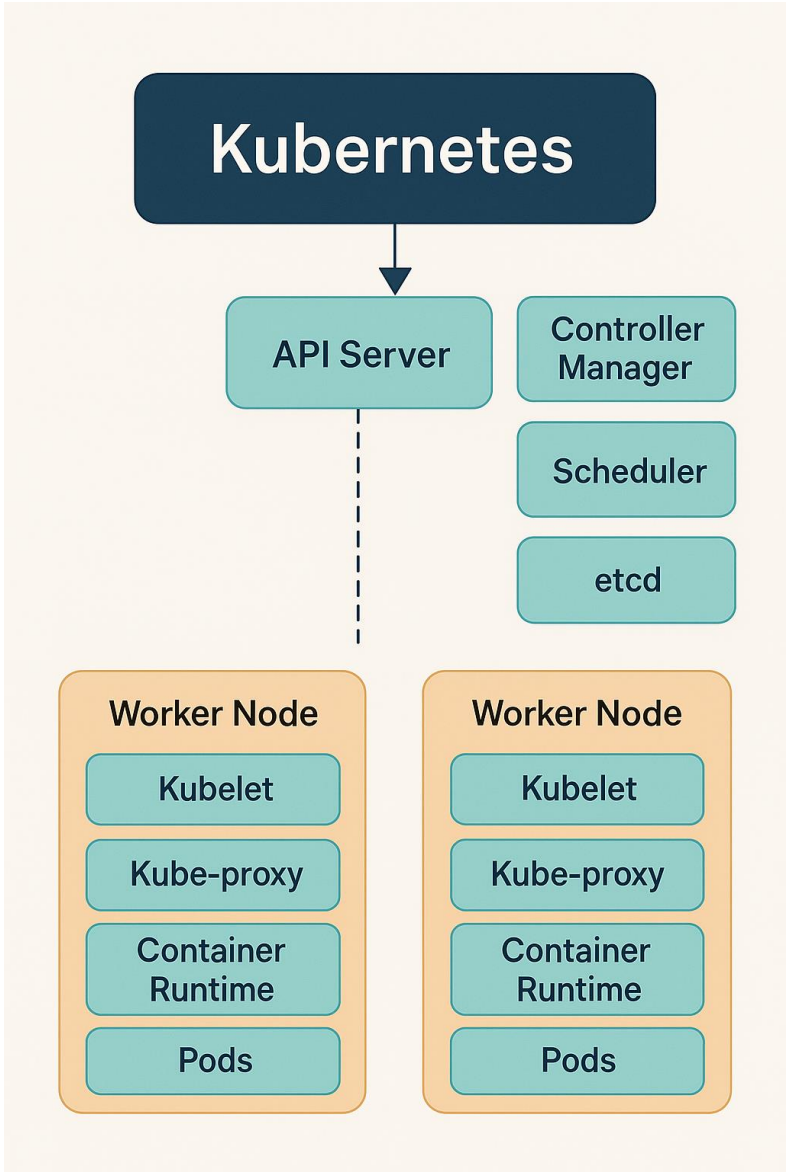
> "Kubernetes, c'est une plateforme d'orchestration de conteneurs open source, conçue par Google.
> Elle permet d'automatiser le déploiement, la mise à l'échelle et la gestion d'applications conteneurisées.
>
> Plutôt que de gérer manuellement des conteneurs Docker, Kubernetes les répartit automatiquement sur les serveurs disponibles, s'assure qu'ils tournent toujours, redémarre ceux qui tombent, et permet de monter ou descendre en charge selon la demande.
>
> En pratique, on définit l'état désiré dans des fichiers YAML — et Kubernetes maintient cet état de manière continue.
>
> C'est aujourd'hui l'un des piliers de l'écosystème DevOps moderne, utilisé dans tous les grands environnements cloud pour garantir la **scalabilité, la résilience et l'automatisation** des déploiements applicatifs."

IDEO·LAB

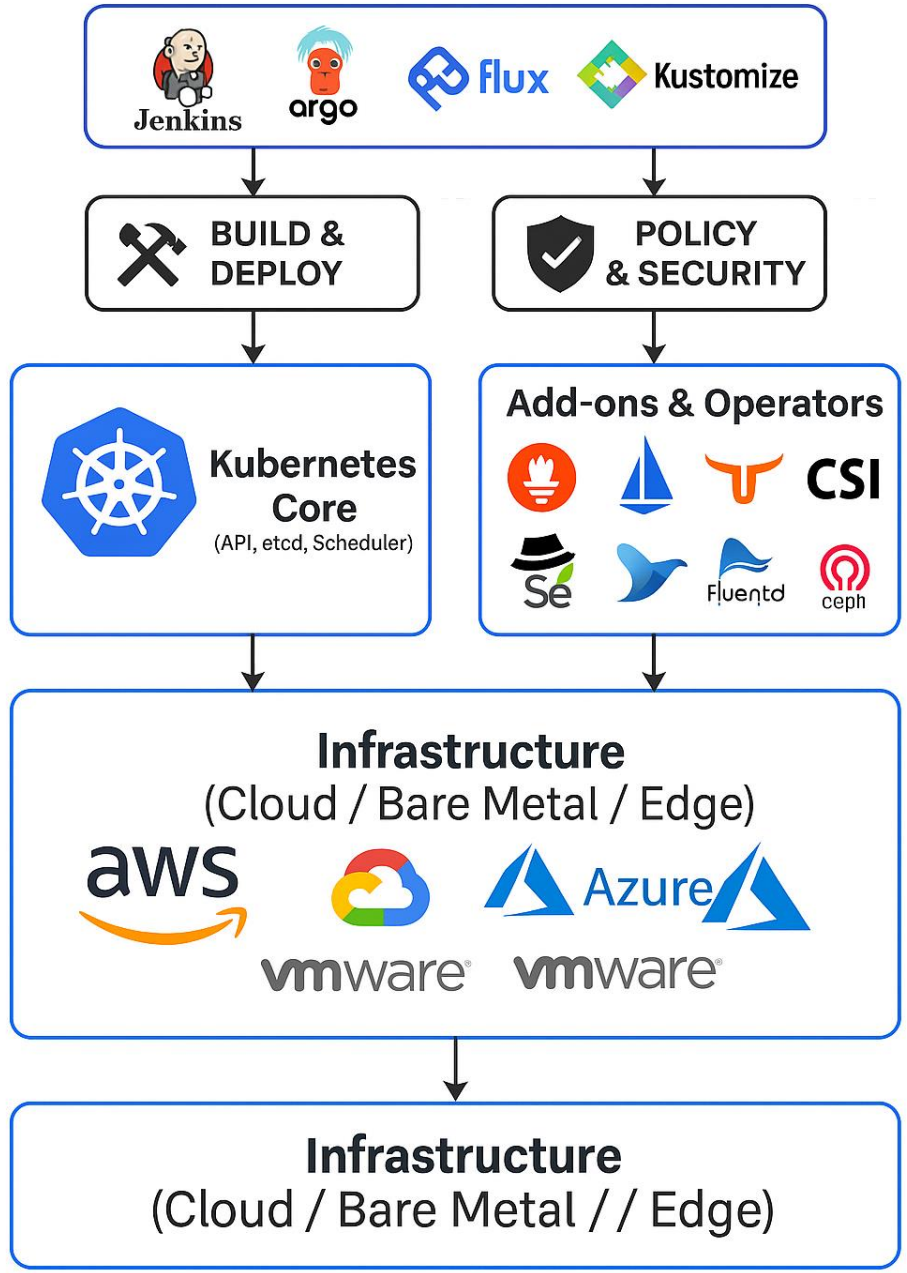


ECOSYSTEME

IDEO·LAB


Kubernetes
API Server
Controller Manager
Scheduler
etcd
Worker Node
Kubelet
Kube-proxy
Container Runtime
Pods
Worker Node
Kubelet
Kube-proxy
Container Runtime
Pods

IDEO·LAB


Jenkins
argo
Flux
Kustomize
BUILD & DEPLOY
POLICY & SECURITY
Kubernetes Core
(API, etcd, Scheduler)
Add-ons & Operators
CSI
Fluentd
ceph
Infrastructure
(Cloud / Bare Metal / Edge)
aws
Azure
vmware
vmware
Infrastructure
(Cloud / Bare Metal / / Edge)

AGENDA ECOSYSTEME KUBERNETES

- 1. Les couches fondamentales de l’écosystème Kubernetes
- 2. Les composants d’écosystème les plus utilisés
  - 2.1 — Infrastructure & déploiement
  - 2.2 — Packaging et gestion des applis
  - 2.3 — Réseau & communication
  - 2.4 — Stockage et persistance
  - 2.5 — Monitoring, logs & alerting
  - 2.6 — Sécurité et gouvernance
  - 2.7 — CI/CD et automation
  - 2.8 — Observabilité et gestion

IDEO·LAB

AGENDA ECOSYSTEME KUBERNETES - 2

- 3. Typologies d’utilisation de Kubernetes
- 4. Schéma mental global de l’écosystème
- 5. En résumé

IDEO·LAB

🧩 1. Les couches fondamentales de l'écosystème Kubernetes

Kubernetes s'intègre dans une **stack complète** en 6 niveaux :

Niveau	Élément	Exemples / Outils associés	Rôle
1. Infrastructure	Cloud ou serveur	AWS, GCP, Azure, OVH, VMware, Bare Metal	Fournit les VM ou nœuds où tourne K8s
2. Container runtime	Docker, containerd, CRI-O	Exécute les conteneurs dans les pods	
3. Kubernetes Core	kube-api-server, scheduler, controller, etcd	Orchestration du cluster	
4. Networking (CNI)	Calico, Flannel, Cilium, Weave	Réseau entre pods, services, sécurité réseau	
5. Storage (CSI)	Ceph, Longhorn, NFS, AWS EBS, GCP PD	Volumes persistants pour données	
6. Add-ons & services	Ingress, monitoring, logging, CI/CD,	Les outils qui rendent Kubernetes	

⚙️ 2. Les composants d'écosystème les plus utilisés

🧭 2.1 — Infrastructure & déploiement

Outil	Description
kubeadm	Outil natif pour initialiser un cluster K8s
kops	Pour créer des clusters sur AWS
kind / minikube / microk8s	Kubernetes local pour développement
Terraform / Pulumi	Infrastructure as Code (création de clusters, réseaux, load balancers)
Cluster API (CAPI)	Standard Kubernetes pour gérer les clusters K8s eux-mêmes

IDEO·LAB

🚀 2.2 — Packaging et gestion des applis

Outil	Description
Helm 🪶	Gestionnaire de packages K8s (templates YAML, versioning, upgrades)
Kustomize	Surcharge de manifestes YAML sans Helm
Argo CD / FluxCD	Déploiement continu GitOps (synchronise le code Git avec le cluster)
Skaffold	Automatisation du build et du déploiement (pour devs)

🌐 2.3 — Réseau & communication

Outil / composant	Description
CNI (Container Network Interface)	Plugin réseau : Calico, Flannel, Cilium
CoreDNS	DNS interne pour le cluster
Ingress Controller	Expose HTTP/HTTPS (Nginx, Traefik, HAProxy, Istio Gateway)
Service Mesh	Gestion fine du trafic, sécurité, observabilité (Istio, Linkerd, Consul)

💾 2.4 — Stockage et persistance

Type	Outil / plugin	Description
CSI (Container Storage Interface)	Longhorn, Ceph-CSI, Rook	Stockage dynamique pour Pods
External volumes	AWS EBS, GCP PD, Azure Disk	Stockage géré par le cloud
Object storage	MinIO, S3, GCS	Pour fichiers et backups
Database Operators	Crunchy Postgres Operator, MySQL Operator	Gestion d'instances DB dans K8s

---

📊 2.5 — Monitoring, logs & alerting

Domaine	Outils	Description
Monitoring	Prometheus, Grafana, Metrics Server	Collecte métriques & dashboards
Logs	Fluentd, Loki, Elasticsearch, Kibana	Agrégation et visualisation des logs
Tracing	Jaeger, OpenTelemetry	Suivi des requêtes distribuées
Alerting	Alertmanager	Alertes selon les métriques Prometheus

IDEO·LAB

🔐 2.6 — Sécurité et gouvernance

Domaine	Outils / concepts	Description
Authentification	RBAC, OIDC, Service Accounts	Contrôle des accès
Admission Control	Kyverno, OPA Gatekeeper	Politique et conformité
Secrets & clés	Sealed Secrets, Vault, External Secrets	Sécurisation des secrets
Network Policies	Calico, Cilium	Firewall interne entre Pods
Image scanning	Trivy, Aqua Security	Détection vulnérabilités images Docker

---

🧪 2.7 — CI/CD et automation

Étape	Outils	Description
Build	Jenkins, GitHub Actions, GitLab CI	Construction images & tests
Deploy	Argo CD, FluxCD	Déploiement GitOps
Test	Sonobuoy, LitmusChaos	Tests de conformité & résilience
Autoscaling	HPA (Horizontal Pod Autoscaler), KEDA	Ajustement automatique des pods

IDEO·LAB

🧩 1. Les couches fondamentales de l'écosystème Kubernetes

Kubernetes s'intègre dans une **stack complète** en 6 niveaux :

Niveau	Élément	Exemples / Outils associés	Rôle
1. Infrastructure	Cloud ou serveur	AWS, GCP, Azure, OVH, VMware, Bare Metal	Fournit les VM ou nœuds où tourne K8s
2. Container runtime	Docker, containerd, CRI-O	Exécute les conteneurs dans les pods	
3. Kubernetes Core	kube-api-server, scheduler, controller, etcd	Orchestration du cluster	
4. Networking (CNI)	Calico, Flannel, Cilium, Weave	Réseau entre pods, services, sécurité réseau	
5. Storage (CSI)	Ceph, Longhorn, NFS, AWS EBS, GCP PD	Volumes persistants pour données	
6. Add-ons & services	Ingress, monitoring, logging, CI/CD, Helm, etc.	Les outils qui rendent Kubernetes "productif"	

IDEO·LAB

🧠 2.8 — Observabilité et gestion

Outil	Description
Lens	Interface graphique de gestion de cluster
k9s	Interface CLI interactive
Octant	Dashboard pour visualiser ressources
Rancher	Plateforme multi-cluster
OpenShift (RedHat)	Distribution Kubernetes complète (avec CI/CD, sécurité, GUI)

IDEO·LAB

🏗️ 3. Typologies d'utilisation de Kubernetes

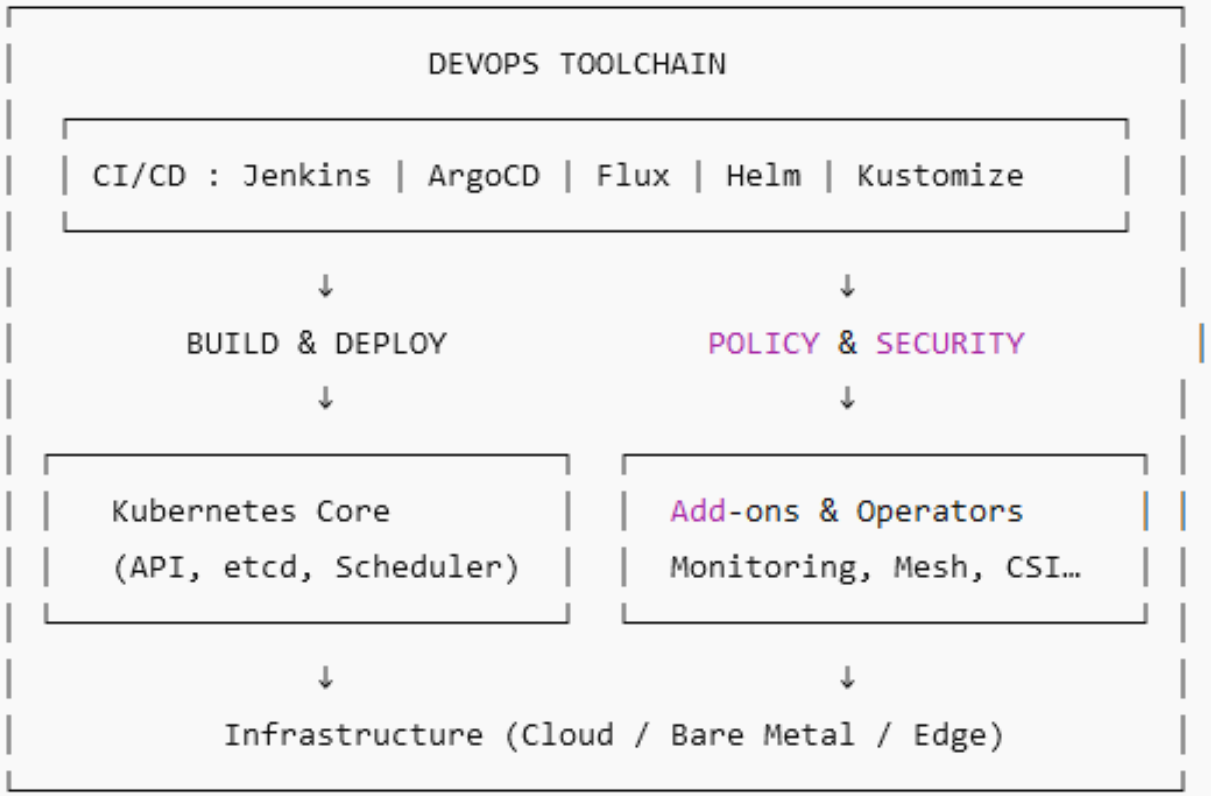
Usage	Description	Exemple
Cloud-Native Apps	Déploiement d'apps microservices	API Django + React + PostgreSQL
Big Data / ML Ops	Jobs Spark, TensorFlow, Kubeflow	Formation modèles IA
Edge Computing	Mini-clusters K8s pour IoT	K3s, MicroK8s
CI/CD Pipelines	Agents build/test containerisés	Jenkins on K8s
Multi-tenant Platforms	Plusieurs équipes isolées par namespaces	SaaS multi-client

IDEO·LAB

🧬 4. Schéma mental global de l'écosystème

pgsql

Copier le code

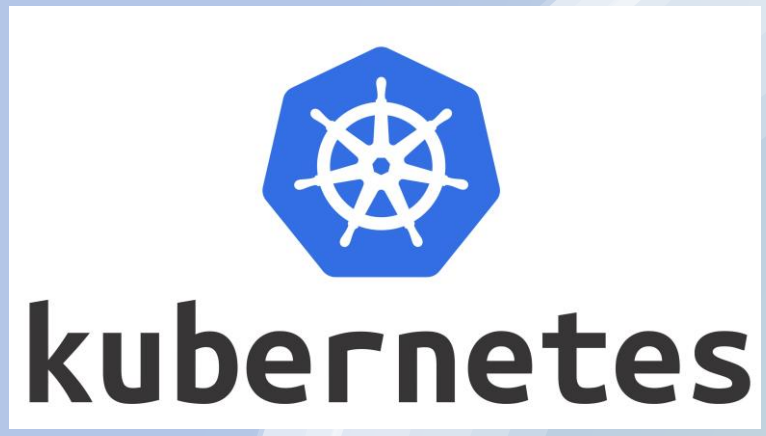


IDEO·LAB

🧭 5. En résumé

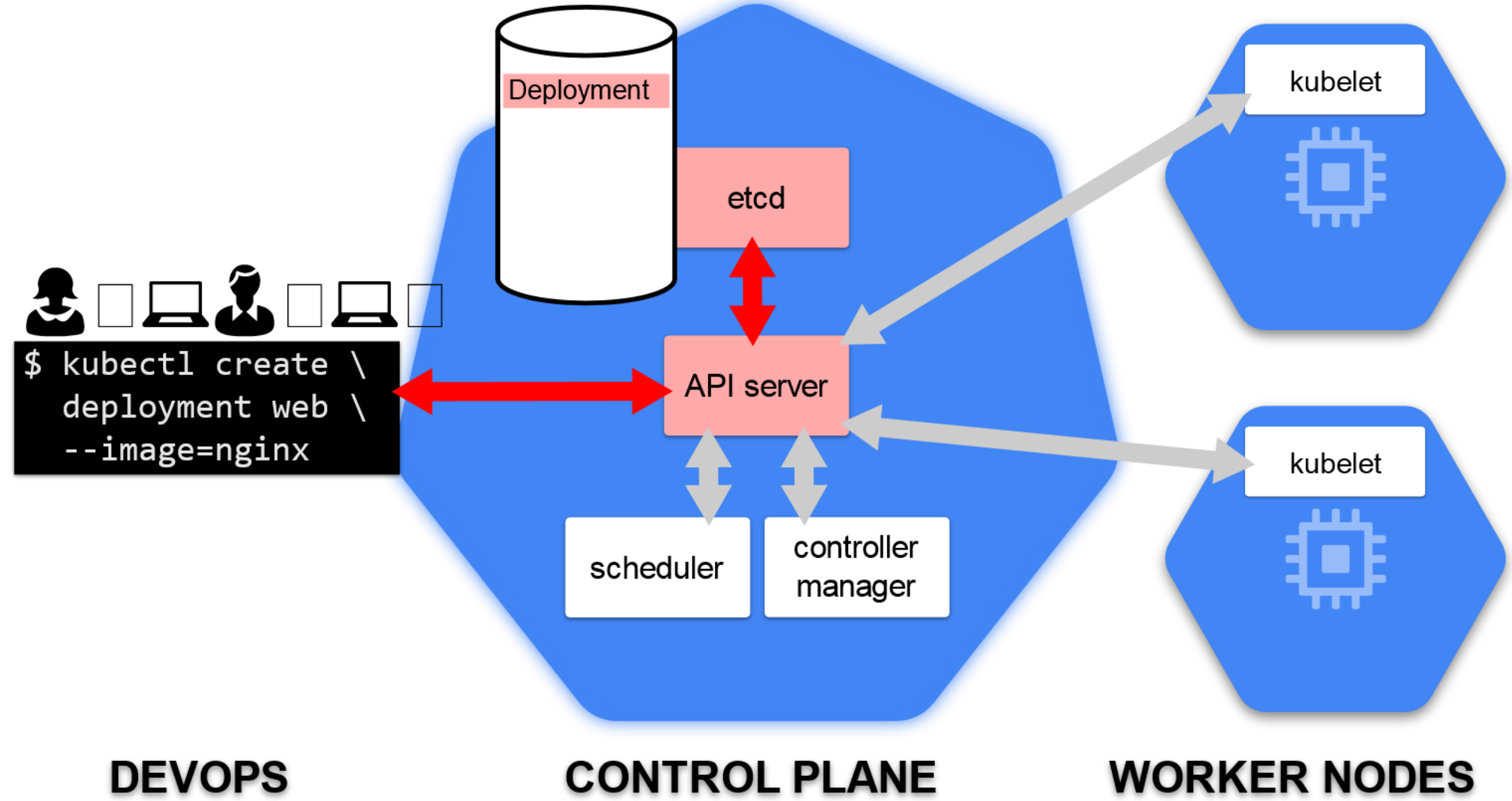
Catégorie	Outils clés	Rôle
Orchestration	Kubernetes	Gère les conteneurs
Packaging	Helm / Kustomize	Déploiement des applis
Réseau	Calico / Cilium / Ingress	Communication interne/externe
Stockage	Longhorn / Ceph / EBS	Persistance des données
Monitoring	Prometheus / Grafana	Observabilité
CI/CD	Jenkins / ArgoCD / Flux	Automatisation des déploiements
Sécurité	Vault / Kyverno / RBAC	Protection et conformité
Interface	Lens / Rancher	Administration visuelle

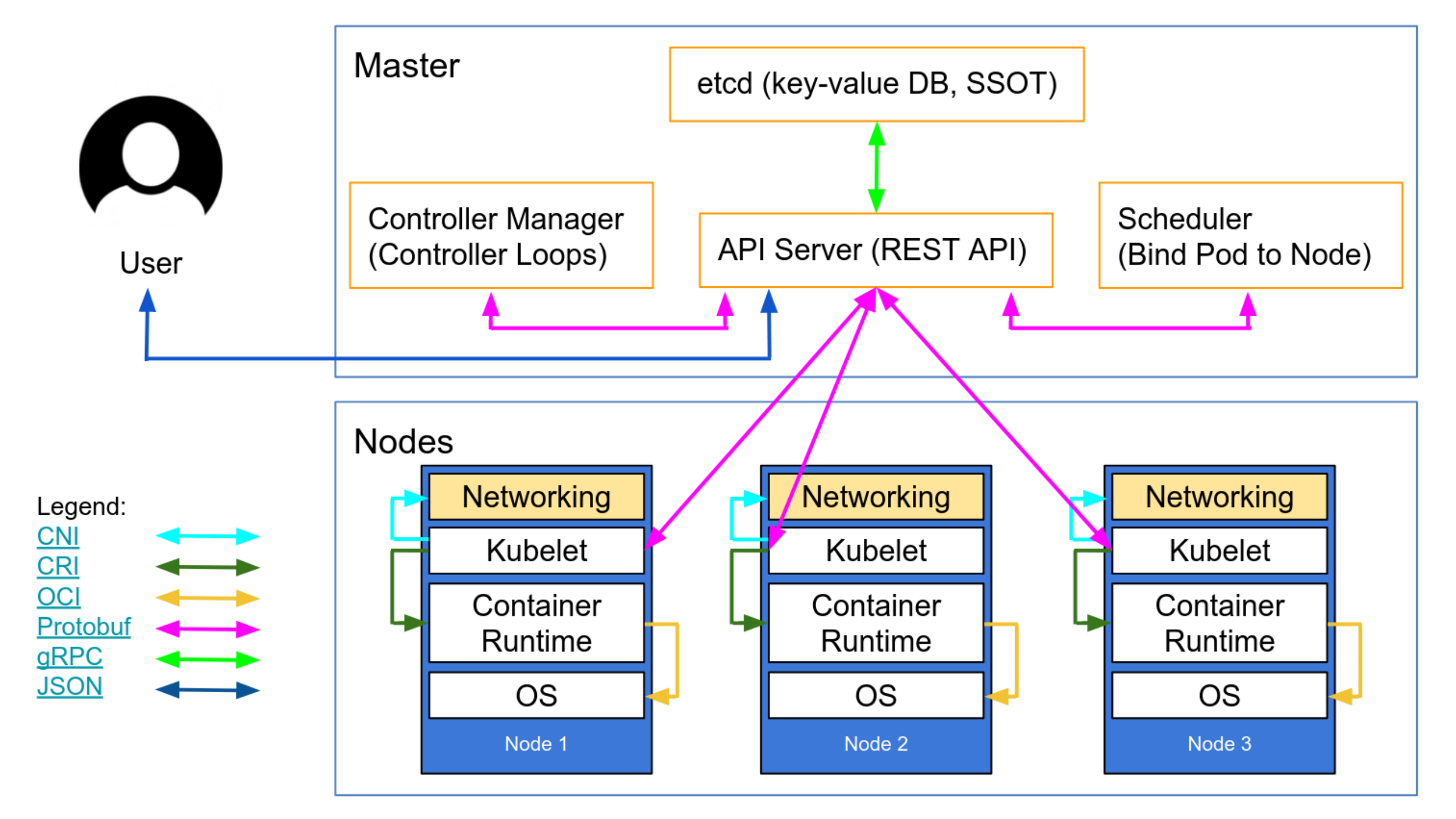
IDEO·LAB



PRESENTATION DE KUBERNETES

IDEO·LAB


Deployment
etcd
API server
scheduler
controller manager
kubelet
kubelet
$ kubectl create \
deployment web \
--image=nginx
DEVOPS
CONTROL PLANE
WORKER NODES


Master
etcd (key-value DB, SSOT)
Controller Manager
(Controller Loops)
API Server (REST API)
Scheduler
(Bind Pod to Node)
User
Nodes
Networking
Kubelet
Container
Runtime
OS
Node 1
Networking
Kubelet
Container
Runtime
OS
Node 2
Networking
Kubelet
Container
Runtime
OS
Node 3
Legend:
CNI
CRI
OCI
Protobuf
gRPC
JSON

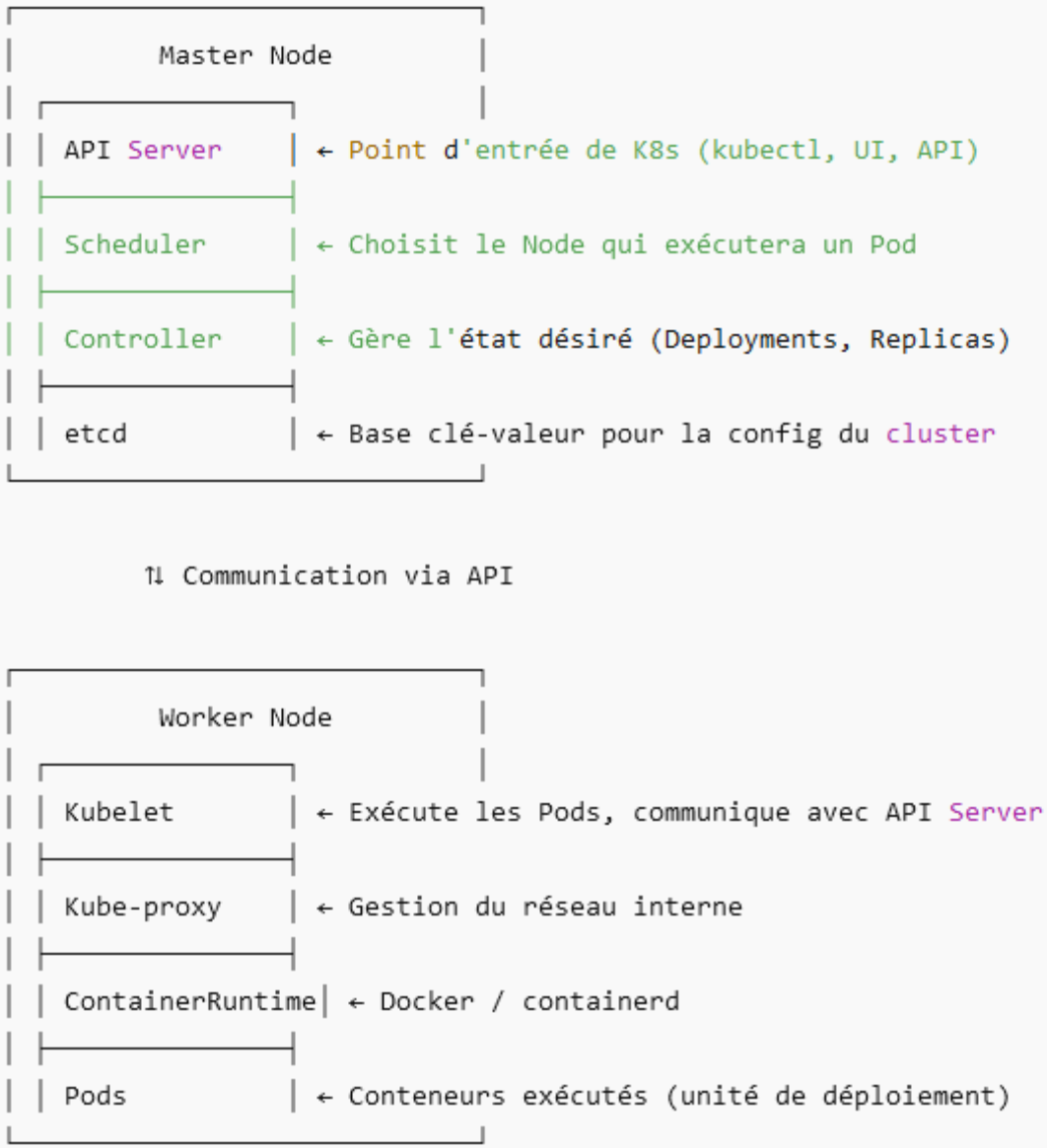
🧱 Étape 1 — Comprendre et Expliquer l'Architecture Kubernetes

Kubernetes (a.k.a **K8s**) est une **plateforme d'orchestration de conteneurs**, conçue pour automatiser le déploiement, la mise à l'échelle et la gestion d'applications conteneurisées.

IDEO·LAB

pgsql

Copier le code



🔹 Composants principaux

Type	Composant	Rôle
Control Plane	API Server	Gère la communication interne & externe
	etcd	Stocke l'état du cluster
	Controller Manager	Garantit l'état désiré des ressources
	Scheduler	Affecte les pods aux nœuds disponibles
Worker Nodes	Kubelet	Exécute les conteneurs
	Kube-proxy	Gère la mise en réseau entre Pods
	Container runtime	Docker, containerd, CRI-O
Réseau	CNI	Calico, Flannel, Weave, etc.
Ingress	Nginx Ingress Controller	Gère les entrées HTTP/HTTPS vers les services

⚙️ Étape 2 — Questions Techniques Populaires + Réponses Synthétiques

🔹 Niveau 1 — Fondamentaux

1. **Qu'est-ce qu'un Pod ?**

   → L'unité de base dans Kubernetes. Contient un ou plusieurs conteneurs qui partagent le même espace réseau et stockage.

2. **Quelle différence entre Pod, Deployment et Service ?**
   - Pod = unité de base (exécution)
   - Deployment = gère la réplication et le déploiement des Pods
   - Service = expose les Pods vers l'extérieur ou en interne (ClusterIP, NodePort, LoadBalancer)

3. **Qu'est-ce qu'un Namespace ?**

   → Une séparation logique pour isoler les ressources (multi-environnements, équipes, etc.).

4. **Que fait le Scheduler ?**

   → Il choisit sur quel Node exécuter un Pod selon les ressources disponibles et les règles d'affinité.

5. **Comment stocker les données persistantes ?**

   → Via des **PersistentVolume (PV)** et **PersistentVolumeClaim (PVC)**, abstractions du stockage.

IDEO·LAB

🔹 Niveau 2 — Intermédiaire

6. **Qu'est-ce qu'un ReplicaSet ?**

   → Garantit un nombre constant d'instances (Pods).

   Le Deployment utilise un ReplicaSet sous le capot.

7. **Différence entre ConfigMap et Secret ?**

   → Les deux stockent des configurations, mais les **Secrets** sont encodés en Base64 (plus sûrs).

8. **C'est quoi un DaemonSet ?**

   → Déploie un Pod sur **chaque Node** du cluster (ex: agent de monitoring, Fluentd, etc.)

9. **C'est quoi un StatefulSet ?**

   → Pour les applications **stateful** (base de données, Kafka, etc.) avec identités persistantes.

10. **Comment exposer une application vers l'extérieur ?**

- **Service NodePort / LoadBalancer**
- **Ingress Controller** (pour HTTP/HTTPS, nom de domaine, SSL)

IDEO·LAB

🔹 Niveau 3 — Avancé

11. Comment faire un rolling update ?

→ `kubectl rollout restart deployment/myapp`

ou via le manifeste `spec.strategy.type: RollingUpdate`

12. Qu'est-ce que Helm ?

→ Un **gestionnaire de paquets pour Kubernetes** (templates YAML versionnés, variables, releases).

13. Comment monitorer un cluster ?

→ Stack classique : **Prometheus + Grafana + Alertmanager**

14. Comment sécuriser Kubernetes ?

- RBAC (Role-Based Access Control)
- Network Policies
- Admission Controllers
- Secrets chiffrés (KMS)

15. Que se passe-t-il quand un Pod échoue ?

→ Le **Controller Manager** le recrée automatiquement pour maintenir l'état désiré.

IDEO·LAB

🚀 Étape 3 — Exemple de Projet (fictif mais crédible)

🔷 Projet : "Deployer une API Django scalable sur Kubernetes"

Contexte :

- Application Django containerisée avec Docker
- Base PostgreSQL externe (RDS)
- Trafic variable → besoin d'auto-scaling

Architecture Kubernetes :

- Namespace : `prod-django`
- Deployment : `django-api` (3 replicas, autoscaling HPA)
- Service : `django-service` (ClusterIP)
- Ingress : `nginx-ingress` pour exposer l'API HTTPS
- ConfigMap : variables d'environnement (DEBUG, ALLOWED_HOSTS)
- Secret : credentials DB + clé API
- HPA (Horizontal Pod Autoscaler) : scale de 3 à 10 Pods selon CPU > 70%
- Monitoring : Prometheus + Grafana

Avantage :

- Scalabilité automatique
- Déploiement CI/CD via GitHub Actions + Helm
- Zéro downtime lors des updates

IDEO·LAB

📌 Étape 4 — Pitch (présentation 1 min)

> "Je travaille depuis plusieurs années dans des environnements conteneurisés, et Kubernetes s'est imposé naturellement comme la solution d'orchestration la plus robuste.
> Je maîtrise l'architecture du Control Plane, la gestion des Pods, Services, Ingress, et j'ai déjà déployé des applications complexes — comme une API Django et un frontend React — sur des clusters K8s avec auto-scaling et monitoring complet.
> Ce que j'aime dans Kubernetes, c'est sa logique déclarative et son intégration naturelle dans les pipelines CI/CD. Aujourd'hui, je cherche à rejoindre une équipe où je pourrais continuer à industrialiser les déploiements, automatiser et améliorer la résilience des plateformes cloud."

IDEO·LAB

LES LOGICIELS ASSOCIES

IDEO·LAB

🧱 Plan général du Top Kubernetes Ecosystem

1️⃣ Orchestration

- **Kubernetes** : le cœur de l'écosystème → architecture, fonctionnement, avantages.

2️⃣ Packaging & Déploiement

- **Helm** 🪶 : gestionnaire de packages (charts, templates, versions)
- **Kustomize** : gestion de configurations YAML sans dépendance externe

---

3️⃣ Réseau

- **Calico** 🐆 : sécurité et politiques réseau (Network Policies)
- **Cilium** 🧬 : observabilité et sécurité eBPF
- **Ingress NGINX / Traefik** 🌐 : exposition HTTP/HTTPS des services

---

4️⃣ Stockage & Persistance

- **Longhorn** 🐮 : stockage distribué pour Kubernetes
- **Ceph / Rook** 🧱 : stockage objet, bloc et fichier
- **EBS / CSI Plugins** : stockage dynamique cloud (AWS, Azure, GCP)

5️⃣ Monitoring & Observabilité

- **Prometheus** 📈 : collecte métriques et alertes
- **Grafana** 📊 : visualisation et dashboards
- **Loki / ELK Stack** : centralisation des logs
- **Jaeger / OpenTelemetry** 🔍 : traçabilité distribuée

6️⃣ CI/CD

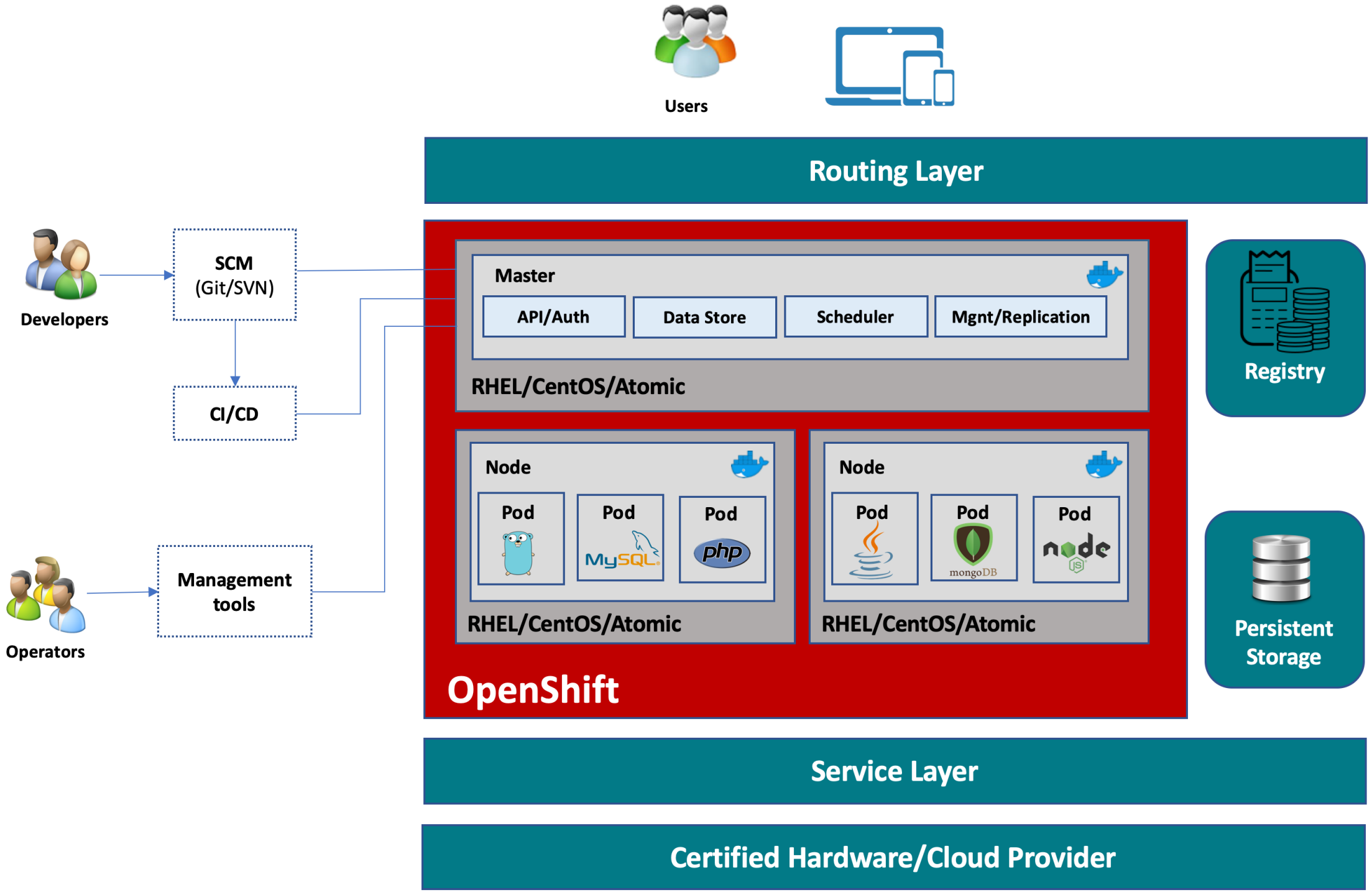
- **Jenkins** ⚙️ : moteur CI historique (intégration via agents K8s)
- **Argo CD** 🚀 : GitOps natif pour Kubernetes
- **FluxCD** 🔁 : alternative légère à ArgoCD (GitOps)
- **Tekton / Jenkins X** 🧩 : pipelines Kubernetes natifs

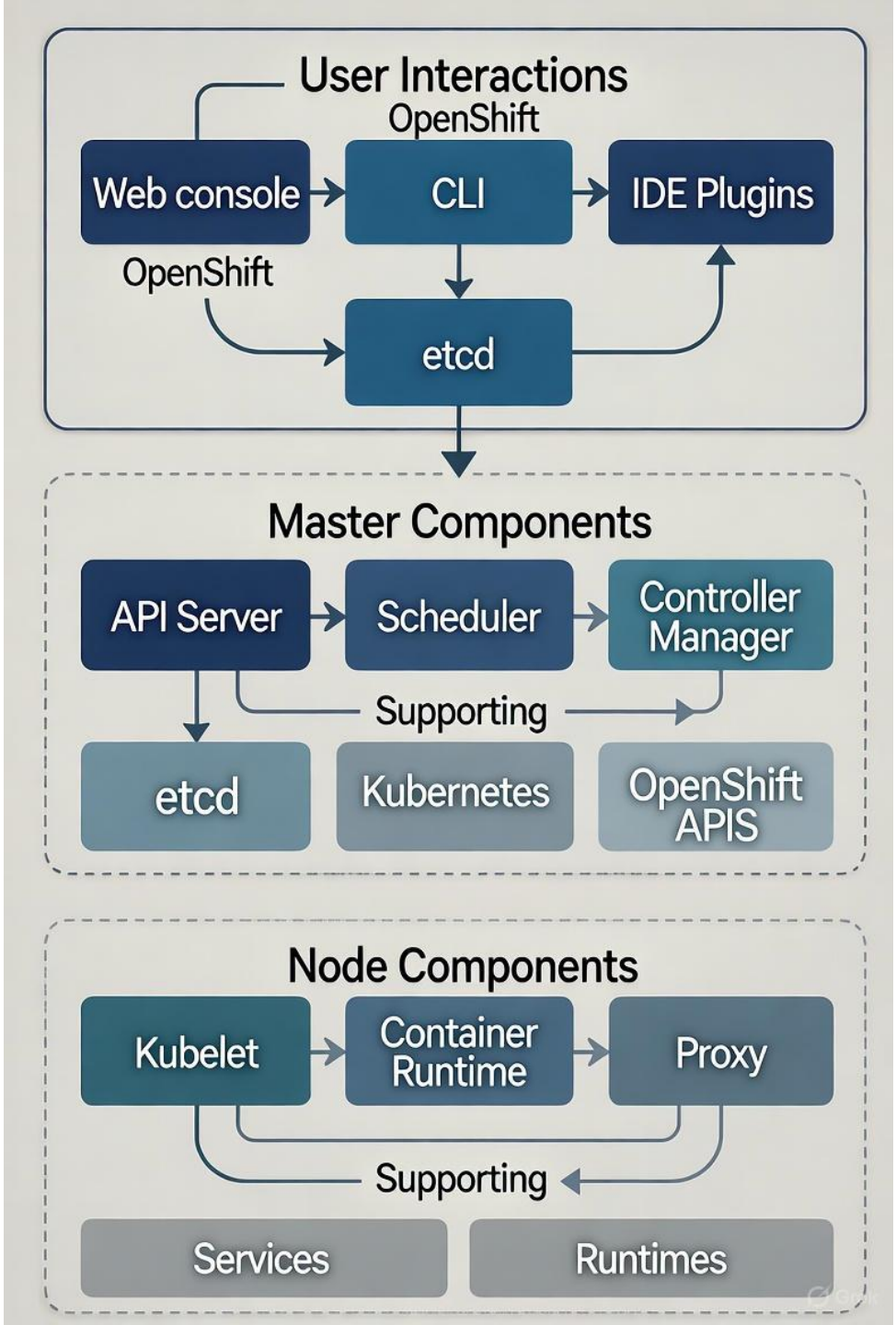
7️⃣ Sécurité & Conformité

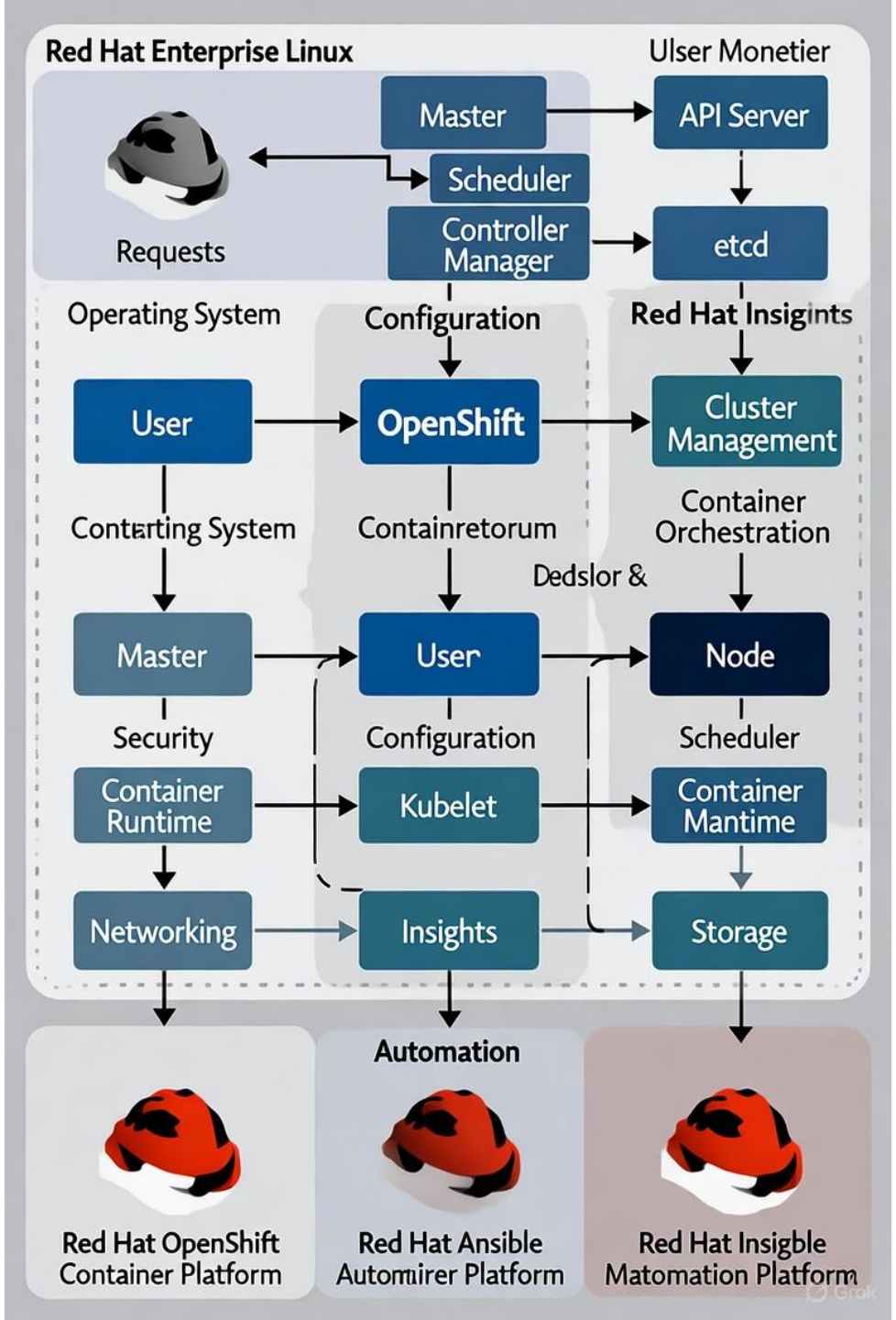
- **Vault (HashiCorp)** 🔐 : gestion des secrets
- **Kyverno** 👹 : politiques de conformité déclaratives (YAML)
- **RBAC / NetworkPolicies** 🔒 : contrôle d'accès et isolement
- **Trivy / Falco / OPA Gatekeeper** 🧠 : scan de vulnérabilités & politiques dynamiques

OPENSHIFT

IDEO·LAB


Users
Routing Layer
Developers
SCM
(Git/SVN)
CI/CD
Operators
Management
tools
Master
API/Auth
Data Store
Scheduler
Mgnt/Replication
RHEL/CentOS/Atomic
Node
Pod
Pod
Pod
RHEL/CentOS/Atomic
Node
Pod
Pod
Pod
RHEL/CentOS/Atomic
OpenShift
Registry
Persistent
Storage
Service Layer
Certified Hardware/Cloud Provider


User Interactions
OpenShift
Web console
CLI
IDE Plugins
OpenShift
etcd
Master Components
API Server
Scheduler
Controller Manager
Supporting
etcd
Kubernetes
OpenShift APIS
Node Components
Kubelet
Container Runtime
Proxy
Supporting
Services
Runtimes


Red Hat Enterprise Linux
Ulser Monetier
Master
API Server
Scheduler
Requests
Controller Manager
etcd
Operating System
Configuration
Red Hat Insigints
User
OpenShift
Cluster Management
Contarting System
Containretorum
Container Orchestration
Dedslor &
Master
User
Node
Security
Configuration
Scheduler
Container Runtime
Kubelet
Container Mantime
Networking
Insights
Storage
Automation
Red Hat OpenShift Container Platform
Red Hat Ansible Automirer Platform
Red Hat Insigble Matomation Platform

AGENDA OPENSHIFT

- 1. Introduction générale
- 2. Architecture et composants principaux
- 3. Versions et déclinaisons
- 4. Infrastructure et système d’exploitation
- 5. Installation (setup complet)
- 6. Sécurité intégrée (point fort OpenShift)
- 7. CI/CD & DevOps intégré
- 8. Observabilité et supervision
- 9. Licences et tarification
- 10. Commandes CLI essentielles (oc)
- 11. Intégration multi-cloud et hybridation
- 12. Cas d’usage typiques
- 13. Documentation & ressources
- 14. Comparatif rapide OpenShift vs Kubernetes pur
- 15. Avantages et limites
- 16. Exemple d’architecture OpenShift sur Red Hat
- 17. Conclusion

IDEO·LAB

🧩 1. Introduction générale

OpenShift est une **plateforme PaaS (Platform-as-a-Service)** basée sur **Kubernetes** et développée par **Red Hat.**

Elle permet de **déployer, gérer et automatiser** des applications conteneurisées (Docker/Podman) dans des environnements hybrides (on-premise + cloud).

Elle s'appuie sur :

- **Kubernetes** pour l'orchestration des conteneurs
- **CRI-O** (remplaçant de Docker) pour le runtime
- **OpenShift API & CLI (oc)** pour la gestion avancée
- **Source-to-Image (S2I)** pour construire automatiquement les images à partir du code source
- **Operators** pour la gestion automatisée du cycle de vie des services

> 👉 En résumé : OpenShift = Kubernetes + Sécurité + Intégration continue + Gouvernance + Support Red Hat

🏗️ 2. Architecture et composants principaux

OpenShift repose sur une **architecture modulaire**, très proche de Kubernetes, mais enrichie par Red Hat :

🔹 Nœuds principaux

Rôle	Description
Master Nodes (Control Plane)	Hébergent `etcd`, le scheduler, les API, et les contrôleurs Kubernetes.
Worker Nodes	Hébergent les pods applicatifs et les services utilisateurs.
Infrastructure Nodes (optionnel)	Hébergent les composants internes : registre d'images, métriques, logging, etc.

🔹 Composants OpenShift spécifiques

- **OpenShift API Server** : surcouche au Kube-API Server avec RBAC avancé et politiques de sécurité renforcées.
- **OpenShift Web Console** : interface graphique d'administration et de déploiement.
- **Image Registry intégré (Quay/Registry)** : dépôt d'images Docker/OCI sécurisé et interne.
- **Router (HAProxy)** : gère le routage HTTP/HTTPS, le load balancing et les ingress.
- **OAuth Server** : gère les authentifications (LDAP, GitHub, SSO, Kerberos, etc.).
- **Operators Framework** : automatisation de l'installation et des mises à jour des services (DB, monitoring, etc.).
- **Source-to-Image (S2I)** : build automatisé d'applications depuis un repo Git.

3. Versions et déclinaisons

Version	Description
OpenShift Container Platform (OCP)	Version commerciale, supportée, installée sur site (on-premise).
OKD (Origin Kubernetes Distribution)	Version communautaire open source d'OpenShift.
OpenShift Online	Version hébergée par Red Hat (PaaS public).
OpenShift Dedicated	Version managée par Red Hat sur AWS ou Google Cloud.

🧱 4. Infrastructure et système d'exploitation

🔸 OS natif : Red Hat Enterprise Linux CoreOS (RHCOS)

- Dérivé de Fedora CoreOS, immutable et optimisé pour Kubernetes.
- Comprend CRI-O, kubelet, et les agents OpenShift.
- Mis à jour via **rpm-ostree** pour garantir des mises à jour atomiques et rollback faciles.

🔸 Compatibilité :

- Fonctionne sur : **RHEL**, **RHCOS**, **VMware**, **AWS**, **Azure**, **GCP**, **Bare Metal**.
- Intègre **SELinux**, **firewalld**, **systemd**, et **Podman** nativement.

INSTALLATION

1. **Préparation du cluster**
   - Configurer DNS, DHCP, NTP et accès Internet.
   - Déployer les certificats SSL et les clés SSH.
   - Configurer les nœuds RHCOS via ignition files.
2. **Installer via l'outil** `openshift-install`

```bash
./openshift-install create cluster --dir=cluster-config --log-level=debug
```

3. **Configurer le CLI**

```bash
export KUBECONFIG=~/cluster-config/auth/kubeconfig
oc login -u kubeadmin -p <mot_de_passe> https://api.cluster.example.com:6443
```

4. **Vérification du cluster**

```bash
oc get nodes
oc get pods -A
```

5. **Déploiement d'une app exemple**

```bash
oc new-project demo
oc new-app --name=hello https://github.com/sclorg/nodejs-ex.git
oc expose svc/hello
```

IDEO·LAB

🔐 6. Sécurité intégrée (point fort OpenShift)

Fonction	Description
Security Context Constraints (SCC)	Contrôle des permissions des pods (root, volumes, capabilities).
SELinux	Isolement renforcé des conteneurs.
OAuth2 / LDAP / SSO	Authentification unifiée et intégrée.
Role-Based Access Control (RBAC)	Permissions granulaire sur ressources Kubernetes.
NetworkPolicies	Contrôle du trafic inter-pods.
Compliance Operator	Analyse CIS / NIST / PCI DSS / GDPR.

> 🔐 OpenShift est **certifié ISO 27001, SOC 2, FedRAMP, HIPAA**, etc.

IDEO·LAB

🚀 7. CI/CD & DevOps intégré

🔹 Pipelines

- Basé sur **Tekton Pipelines** → 100 % Kubernetes-native.
- YAML déclaratif, intégré à la console web.
- Intégration GitHub, GitLab, Jenkins, Bitbucket.

🔹 GitOps

- Basé sur **Argo CD** → synchronisation automatique Git ↔ cluster.

🔹 Build automatisé

- Source-to-Image (S2I) :
  - Détection automatique du langage (Python, Node.js, Go, Java, PHP...)
  - Build → Push → Déploiement automatique.

    Exemple :

```bash
oc new-app python:3.9~https://github.com/user/repo.git
```

IDEO·LAB

📊 8. Observabilité et supervision

Module	Description
Prometheus	Collecte des métriques du cluster.
Grafana	Tableaux de bord personnalisables.
Alertmanager	Gestion des alertes (email, Slack, PagerDuty).
ElasticSearch + Fluentd + Kibana (EFK)	Logging centralisé.
Cluster Monitoring Operator	Automatisation de la supervision et du reporting.

💰 9. Licences et tarification

Version	Coût	Support
OKD	Gratuit	Communautaire
OCP Standard	~$5 000 / an / cluster	Support 8x5 Red Hat
OCP Premium	~$10 000 / an / cluster	Support 24x7
OpenShift Dedicated (AWS/GCP)	À partir de $0.076 / core-hour	Managé par Red Hat

🧱 10. Commandes CLI essentielles (`oc`)

Commande	Description
`oc login`	Se connecter à un cluster
`oc new-project <nom>`	Créer un projet
`oc new-app`	Créer une application
`oc get pods`	Lister les pods
`oc logs <pod>`	Voir les logs
`oc describe pod <nom>`	Détails d'un pod
`oc delete all --all`	Supprimer tout dans le projet
`oc adm top nodes`	Voir les ressources consommées
`oc whoami`	Vérifier l'utilisateur connecté

IDEO·LAB

☁️ 11. Intégration multi-cloud et hybridation

- Compatible **AWS**, **Azure**, **GCP**, **IBM Cloud**, **VMware**, **Bare Metal**, **OpenStack**.
- Fonctionne avec **Red Hat Advanced Cluster Management (RHACM)** pour gérer plusieurs clusters.
- Support du **Edge Computing** et de **MicroShift** (version légère pour IoT / Edge).

🧠 12. Cas d'usage typiques

Secteur	Utilisation
Banques / FinTech	Déploiement sécurisé d'applications microservices (PCI DSS).
Industrie	Automatisation des pipelines CI/CD.
Télécom	Déploiement d'infrastructures 5G avec Edge nodes.
Santé	Hébergement d'applications HIPAA compliant.
Cloud hybride	Interconnexion multi-sites via RHACM.

IDEO·LAB

📚 13. Documentation & ressources

- Documentation officielle Red Hat ↗
- OKD Community ↗
- OpenShift GitHub ↗
- Interactive Learning Portal ↗
- Red Hat Marketplace ↗

🧩 14. Comparatif rapide OpenShift vs Kubernetes pur

Critère	OpenShift	Kubernetes
Installation	Simplifiée (installeur intégré)	Manuelle (kubeadm, kops, etc.)
Sécurité	SELinux, SCC, RBAC strict	RBAC basique
Web Console	Intégrée	À installer (Lens, Dashboard)
Registre d'images	Intégré	Externe (DockerHub, Harbor)
Build S2I	Oui	Non
Support	Red Hat	Communautaire
CI/CD intégré	Tekton / Argo CD	Non natif

IDEO·LAB

🧭 15. Avantages et limites

✅ Points forts :

- Sécurité **entreprise-grade** (SELinux, SCC)
- Interface web complète
- CI/CD et GitOps intégrés
- Support professionnel Red Hat
- Très adapté aux environnements hybrides

⚠️ Limites :

- Coût élevé pour petites structures
- Installation plus lourde que k3s ou EKS
- Architecture parfois complexe (Operators multiples)

IDEO·LAB

16. Exemple d’architecture OpenShift sur Red Hat

mermaid

Copier le code

```
graph TD
A[Utilisateur DevOps] -->|CLI/Web| B[API Server]
B --> C[Controller Manager]
B --> D[Scheduler]
B --> E[etcd (base cluster)]
B --> F[Worker Nodes]
F --> G[Pods App 1, App 2, ...]
F --> H[Registry interne]
F --> I[Router HAProxy]
J[Red Hat Enterprise Linux CoreOS] --> F
K[Monitoring (Prometheus + Grafana)] --> B
```

IDEO·LAB

🧩 17. Conclusion

OpenShift est **bien plus qu'un Kubernetes packagé** :

c'est une **plateforme de conteneurisation complète**, sécurisée, automatisée, et supportée par Red Hat, pensée pour les entreprises exigeant fiabilité, conformité et évolutivité.

> 🔧 En résumé :
> **OpenShift = Kubernetes + Sécurité + CI/CD + Support Red Hat + Cloud hybride**

IDEO·LAB