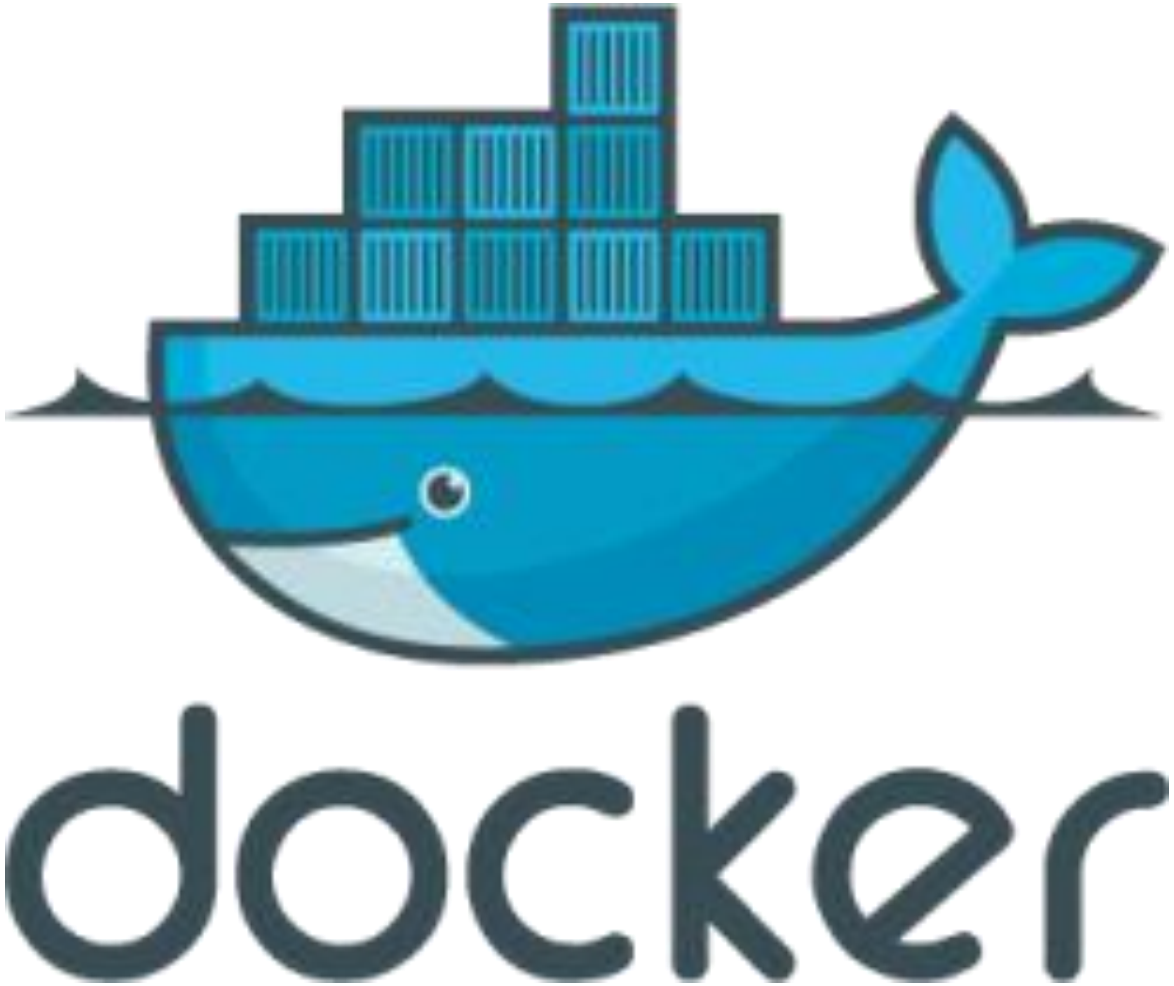
Par Achille MONGA

Docker Avancé



Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

Sécurisation du daemon docker

Ingress - Expose application

Observabilité

Orchestration

Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

Sécurisation du daemon docker

Ingress - Expose application

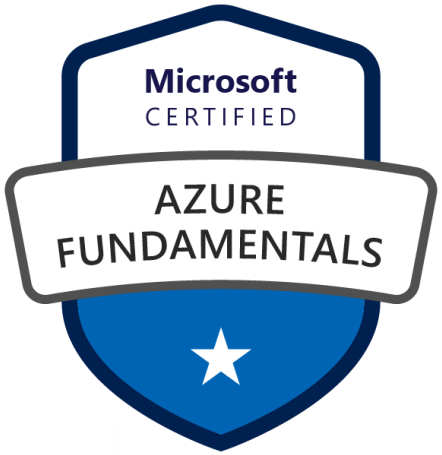
Observabilité

Orchestration

Présentation du formateur

- Achille MONGA (Cloud Architect, DevOps, Ingénieur Logiciels Embarqués)
- LDLC
- RENAULT
- SNCF
- ENEDIS
- ADEUNIS

@achillemonga



Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

Sécurisation du daemon docker

Ingress - Expose application

Observabilité

Orchestration

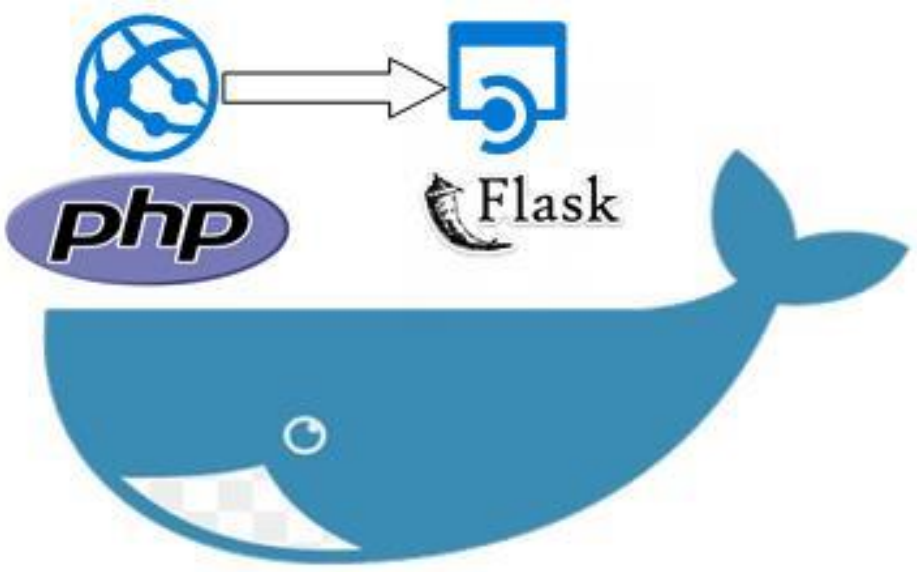
TP-1: déploiement Wordpress

- Démarrer un conteneur wordpress avec la commande run
  - Créer un réseau dans lequel le conteneur sera déployé
  - Démarrer un conteneur mysql qui sera utilisé pour stocker les config wordpress
    - Créer un volume où seront persister les données de la BD situé dans /var/lib/mysql
    - Utiliser les variables : db_name=wordpress_sparks, user=sparks, password=sparks
  - Démarrer le conteneur wordpress de sorte qu'il utilise la le conteneur de BD
    - Créer un volume où seront persister les données de wordpress situé dans /var/www/html

- Exécuter Wordpress via docker compose
  - Utiliser composerize pour écrire votre fichier iac

TP-2: student list

- Suivre les instructions pour conteneuriser puis exécuter l'application https://github.com/MTA1711/student-list.git



Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

Sécurisation du daemon docker

Ingress - Expose application

Observabilité

Orchestration

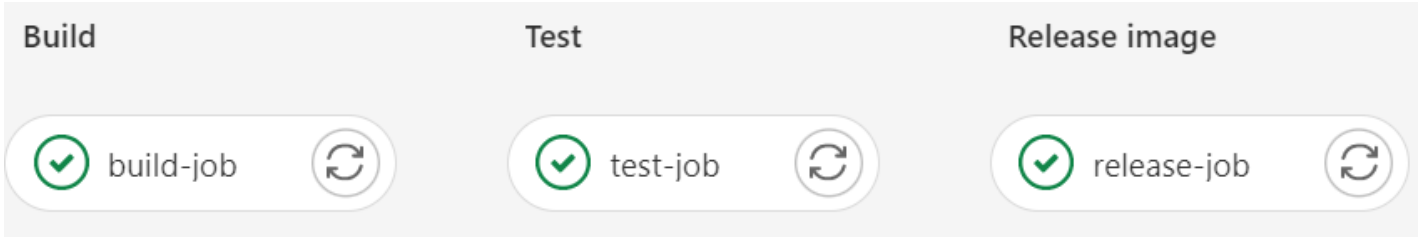
Optimisation du Dockerfile

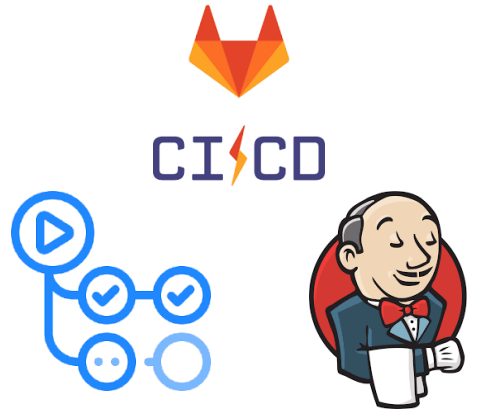
- Utilisation du cache lors de la construction de l'image
  - Les Layers qui changent régulièrement doivent être les dernières de l'image
- Avoir un utilisateur dédié pour l'execution. Eviter le root user
- Toujours utiliser une version spécifique pour l'image de base
- Un seul process par container
- Avoir une image la plus petite possible
  - Accélère le déploiement
  - Utiliser le build multi-stage
  - Priviligier les versions lite pour les images de base
  - Utiliser les images officielles

```
FROM node:14 AS builder
RUN apt-get update
WORKDIR /usr/app

COPY package*.json ./
RUN npm install

COPY . ./
RUN npm run build

FROM nginx:stable-alpine
COPY --from=builder /usr/app/dist /usr/share/nginx/html
```

Pipeline CI/CD

- Automatisation des releases des applications
  - Construire, tester, release des images via des pipelines CI/CD

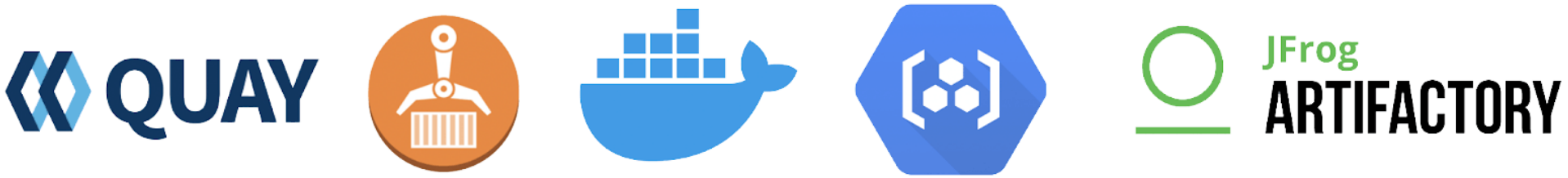




- Utiliser un artifact repository pour stocker vos images

Private registry

- Sécurité
- Déploiement rapide
- Possibilité de mettre en place du SSO/RAC



Scan des images

- Sécurité, limite la surface d'attaque
- Détection de vulnérabilités
- Mettre en œuvre l'analyse au sein des pipelines et/ou des registres

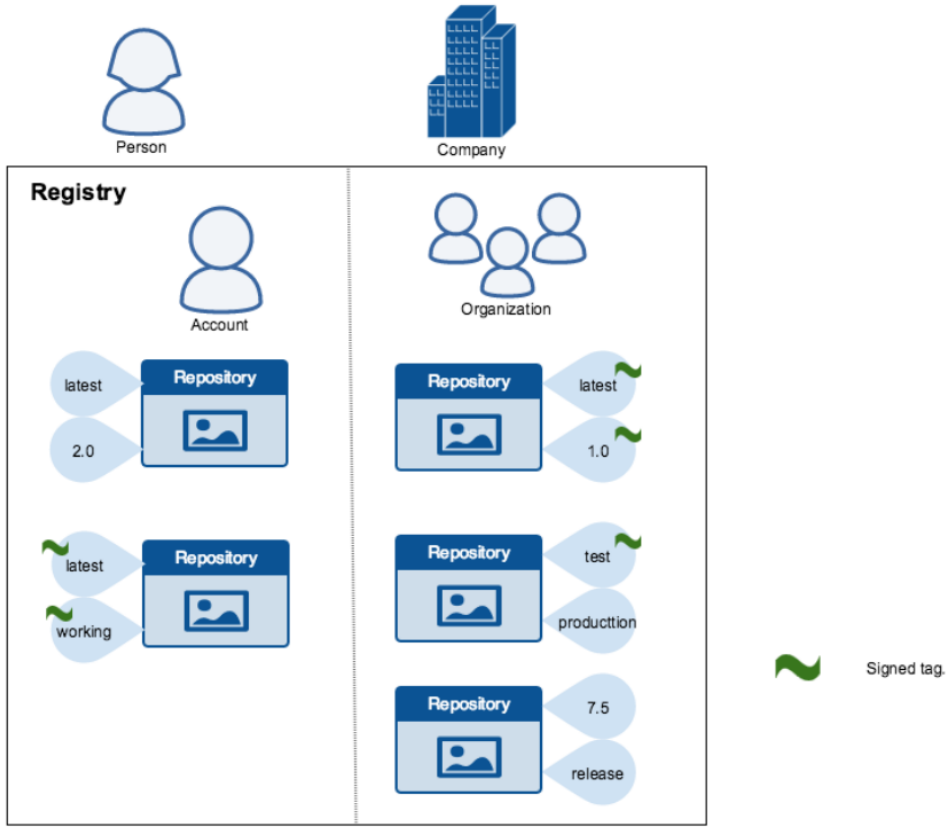
```
[vagrant@dmaster ~]$ docker scan --version
Version:    v0.17.0
Git commit: 061fe0a
Provider:   Snyk (1.827.0 (standalone))
```

```
[vagrant@dmaster ~]$ docker scan nginx:alpine

Testing nginx:alpine...

✗ Low severity vulnerability found in openssl/libcrypto1.1
  Description: CVE-2022-2097
  Info: https://snyk.io/vuln/SNYK-ALPINE316-OPENSSL-2941806
```

clair

Docker content trust: DCT

- Signature l'image
- Permettre la vérification de la provenance des images
- Améliorer la sécurité
- Empêcher l'exécution d'image non signées



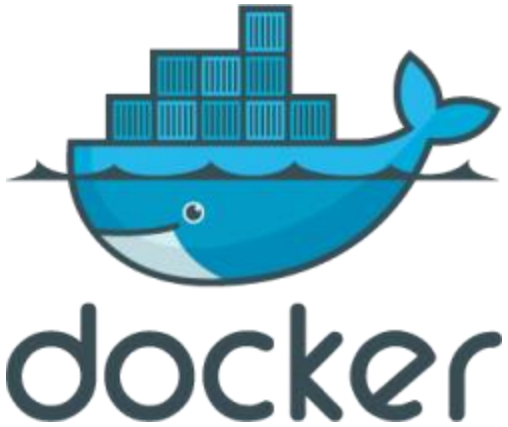

```
[vagrant@dmaster ~]$ docker trust key generate master_key
Generating key for master_key...
Enter passphrase for new master_key key with ID 12211e9:
Repeat passphrase for new master_key key with ID 12211e9:
Successfully generated and loaded private key. Corresponding public key available: /home/vagrant/master_key.pub
```

TP-3: Optimisation Image

- Appliquer les bonnes pratiques pour avoir une image production ready de l'application https://github.com/MTA1711/contact-manager-angular-material.git

TP-4: Scan Image avec docker scan

- Scanner l’image officielle de nginx pour avoir la liste des vulnérabilités
- Scanner l’image finale de l’application de gestion des contacts



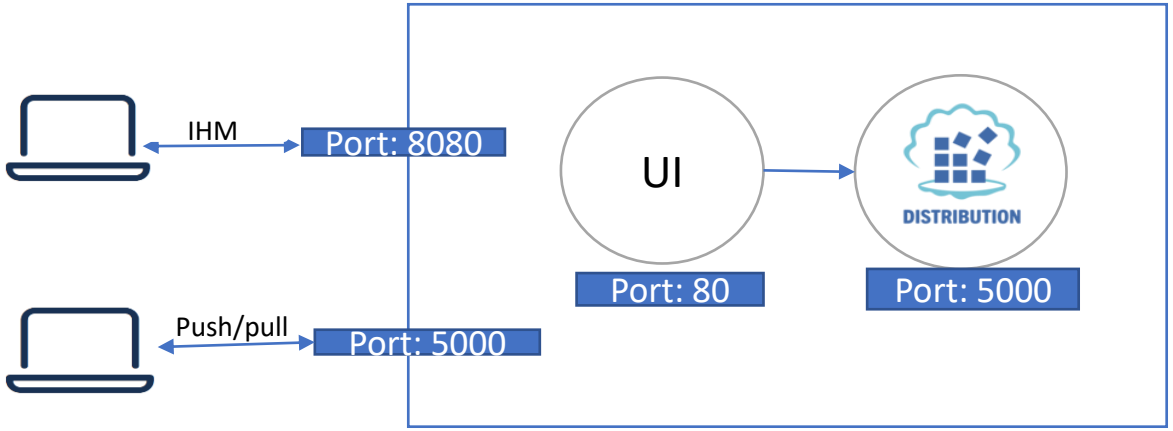
TP-5: Mise en place d’une CI-CD pour la release de l’image

- Créer une chaine CI/CD pour construire et release notre application de gestion de contact
  - Utiliser les github actions
  - Décrire les Jobs pour build, tester puis release de l’application sur le docker hub
  - Ajouter une action de scan avec Trivy si possible

GitHub Actions

TP-6: Gestion avancée des images: setup d'un registre privé

- Transformer la VM Dregistry en un private repository docker
  - Créer le réseau où seront déployés les conteneurs
  - Déployer l'image https://hub.docker.com/_/registry pour transformer la VM en repository
    - Créer un volume où seront stocké les images et monter le dans /var/lib/registry du conteneur
    - Utiliser la variable d'env REGISTRY_STORAGE_DELETE_ENABLE et la mettre à true et utiliser les variables pour activer le CORS
  - Déployer l'image https://hub.docker.com/r/joxit/docker-registry-ui pour avoir une UI qui va nous permettre de gérer notre repo
    - Utiliser les variables d'env REGISTRY_TITLE, REGISTRY_URL, DELETE_IMAGE, SINGLE_REGISTRY
  - Déployer l'image de l'api de student-list dans notre registry
  - Visualiser l'état du registry
  - Créer un fichier docker compose pour l'automatisation



Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

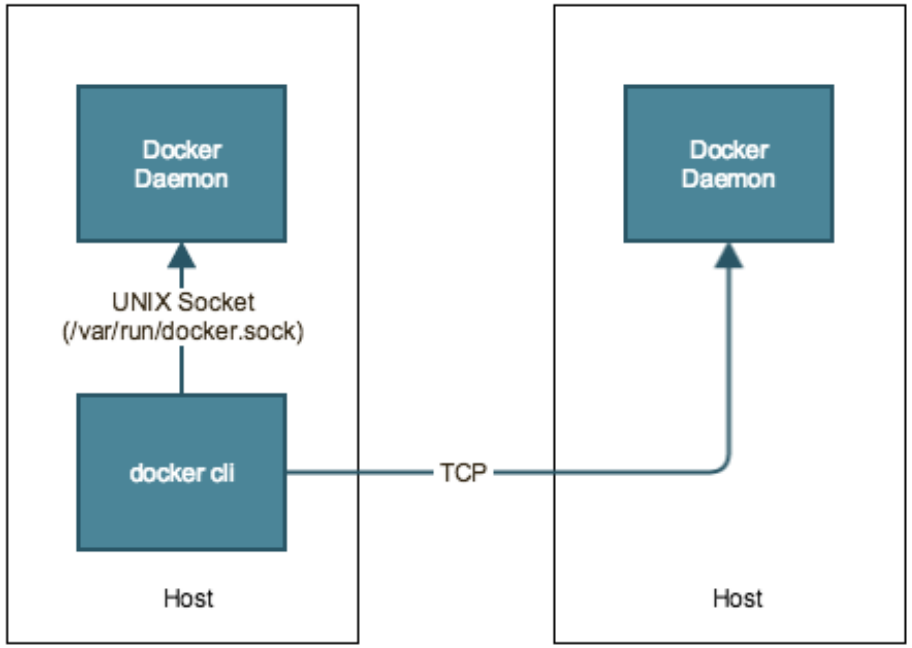
Sécurisation du daemon docker

Ingress - Expose application

Observabilité

Orchestration

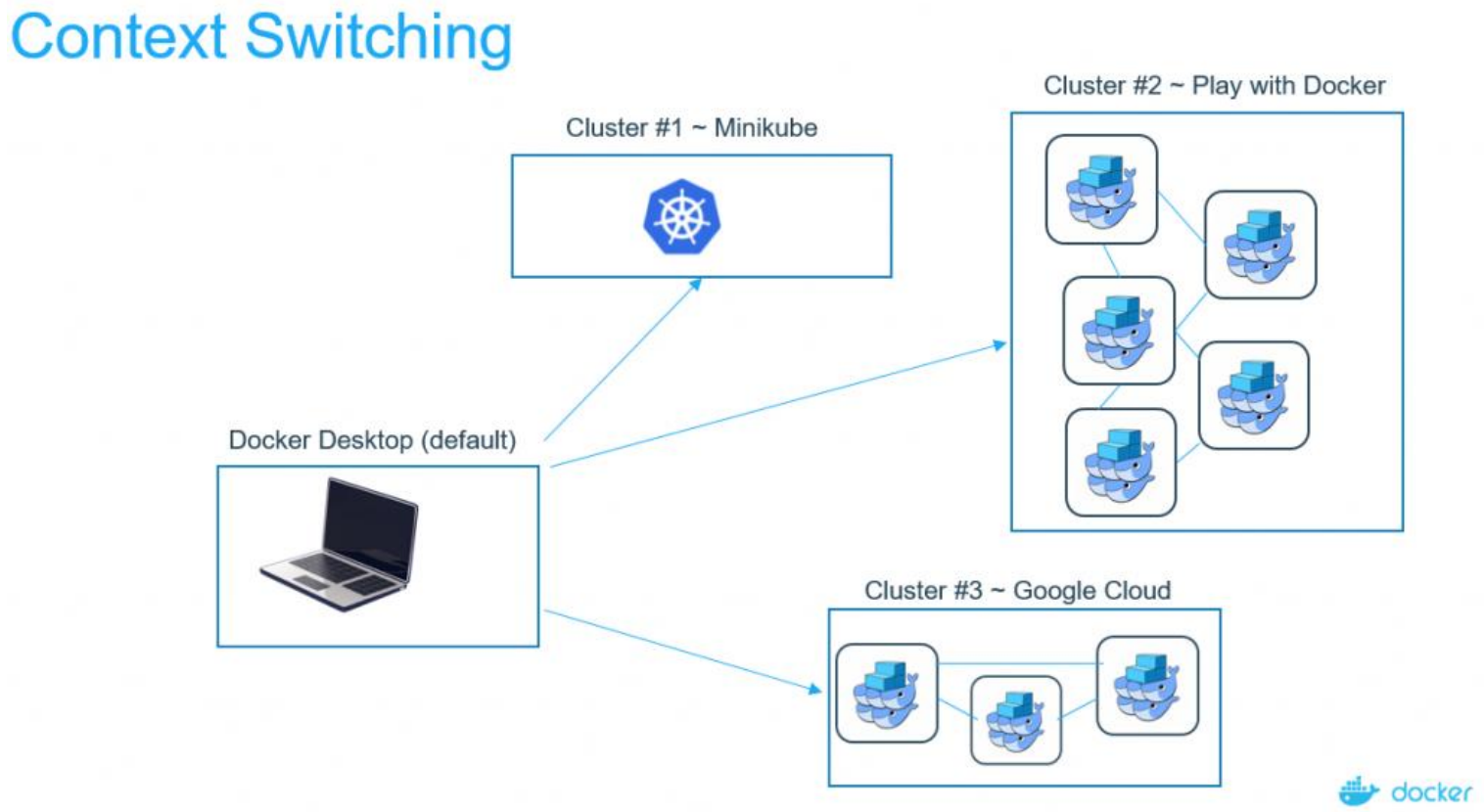
Sécurisation docker: Socket docker



- Port **2375** pour les communications non sécurisées
- Port **2376** pour les communications sécurisées

```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375 --containerd=/run/containerd/containerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always
```

Sécurisation docker: les contextes



Sécurisation docker: TLS Auth

```
$ dockerd \
    --tlsverify \
    --tlscacert=ca.pem \
    --tlscert=server-cert.pem \
    --tlskey=server-key.pem \
    -H=0.0.0.0:2376
```

```
$ docker --tlsverify \
    --tlscacert=ca.pem \
    --tlscert=cert.pem \
    --tlskey=key.pem \
    -H=$HOST:2376 version
```

- Besoin d'une autorité de certification
- export DOCKER_HOST=tcp://$HOST:2376 DOCKER_TLS_VERIFY=1

Sécurisation docker: Resource constraints

```
[vagrant@dmaster ~]$ docker stats --no-stream
CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT   MEM %     NET I/O   BLOCK I/O   PIDS
```

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
6dabef87438b	hops_app_1	0.01%	125.8MiB / 31.3GiB	0.39%	1.1GB / 151MB	30.2MB / 0B	13
e10951cfef27	sec0419b_mysql_1	0.11%	234.1MiB / 31.3GiB	0.73%	7.42MB / 7.23MB	52MB / 39.5MB	36
5df760550484	sec0419b_app_1	0.01%	55.72MiB / 31.3GiB	0.17%	8.14MB / 6.47MB	50.8MB / 2.11MB	7
4c18d0250bce	hops_mysql_1	0.32%	912.6MiB / 31.3GiB	2.85%	292MB / 2.21GB	86.1MB / 3.42GB	38
5cac82e13be5	portainer_portainer_1	0.03%	26.99MiB / 31.3GiB	0.08%	1.56MB / 6.07MB	36MB / 23.5MB	16
e201222e0859	brt1119b_mysql_1	0.40%	446.2MiB / 31.3GiB	1.39%	5.12MB / 8.33MB	119MB / 72.4MB	45
f3fcb9e08e13	brt1119b_app_1	0.01%	96.42MiB / 31.3GiB	0.30%	9.75MB / 9.11MB	70.5MB / 1.93MB	12
b8bb07d57869	heron-web_app_1	0.01%	8.527MiB / 31.3GiB	0.03%	961kB / 0B	4.7MB / 0B	82
24208d8c94ec	atp0820_app_1	0.01%	24.04MiB / 31.3GiB	0.07%	961kB / 0B	34.1MB / 0B	6

Sécurisation docker: Capabilities

- Principe du moindre privilège
- Linux capabilities

CAP_CHOWN: Avoir la capacité de changer le propriétaire d'un fichier;
CAP_DAC_OVERRIDE: Passer outre le contrôle d'accès (Posix ACL);
CAP_FSETID: Avoir la capacité d'utiliser chmod sans limitation;
CAP_FOWNER: Outrepasser le besoin d'être propriétaire du fichier;
CAP_MKNOD: Avoir la capacité d'utiliser des fichiers spéciaux;
CAP*NET*RAW: Avoir la capacité d'utiliser les sockets raw et packet (snifiing, binding);
CAP_SETGID: Avoir la capacité de changer le GID;
CAP_SETUID: Avoir la capacité de changer l'UID;
CAP_SETFCAP: Avoir la capacité de modifier les capacités d'un fichier;
CAP_SETPCAP: Avoir la capacité de modifier les capacités d'un autre processus;
CAP*NET*BIND_SERVICE:, Avoir la capacité d'écouter sur un port inférieur à 1024;
CAP*SYS*CHROOT: Avoir la capacité de faire un change root;
CAP_KILL: Avoir la capacité de killer un processus;
CAP*AUDIT*WRITE: Avoir la capacité d'écrire des logs Kernels (par exemple pour changer un password);

```
docker run -d \
 --cap-drop=chown \
 --cap-drop=dac_override \
 --cap-drop=fowner \
 --cap-drop=fsetid \
 --cap-drop=kill \
 --cap-drop=setpcap \
 --cap-drop=mknod \
 --cap-drop=setfcap \
 --publish=2222:22 \
 --name serveurssh \
 --hostname serveurssh \
 jmp/sshd
```

TP-7: Sécurisation de la socket docker pour les remote user

- Exposer la socket docker de la machine dworker
- Sur la machine dmaster créer un contexte pour la machine dworker
- Déployer une image de serveur web sur la machine dworker

Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

Sécurisation du daemon docker

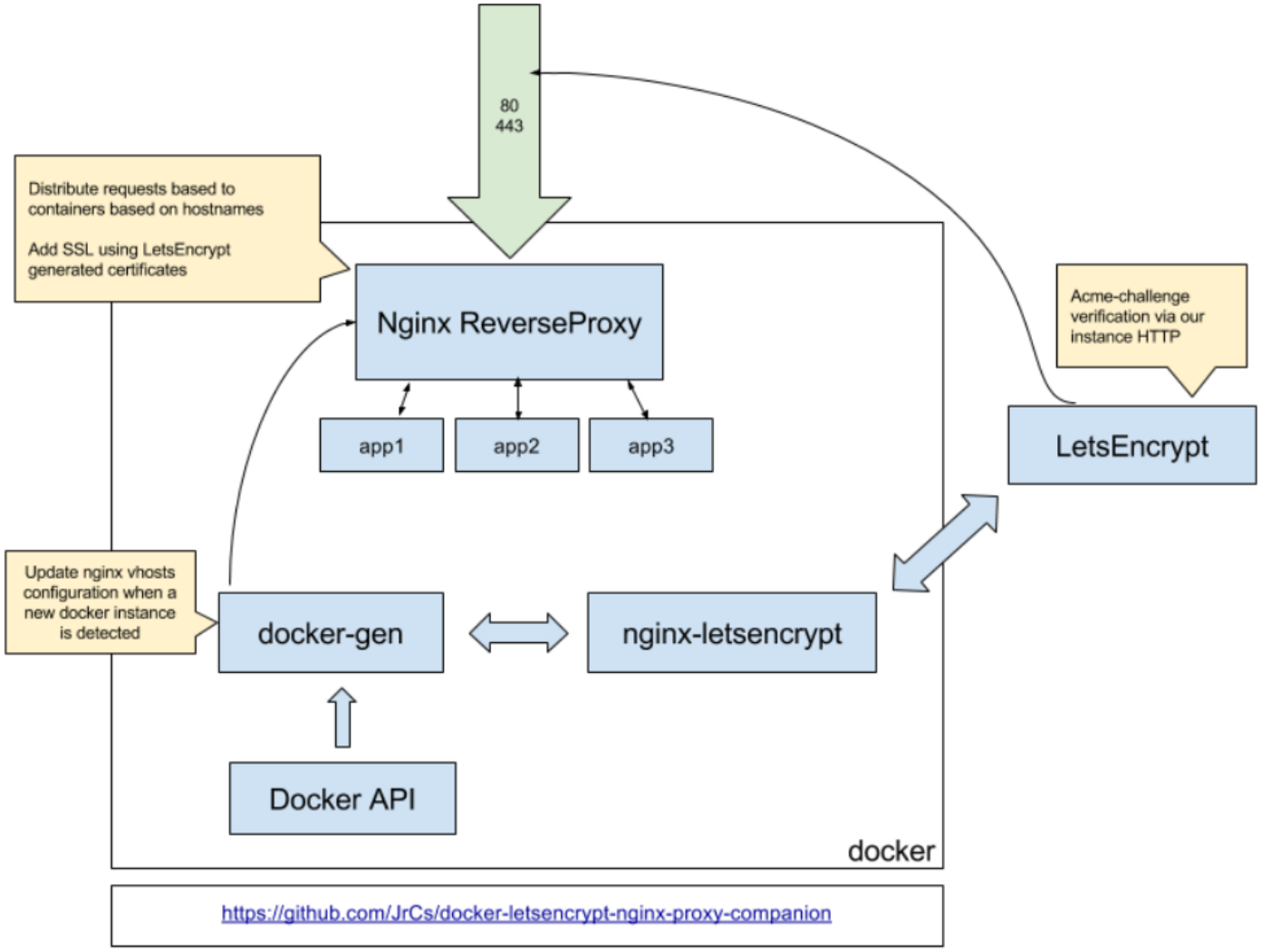
Ingress - Expose application

Observabilité

Orchestration

Ingress

- Plusieurs applications sur un même hôte.
- Plusieurs apps écoutant le même port.
- Partage des ressources matérielles.
- Plusieurs DNS pointant la même IP.



TP-8: Exposer plusieurs app sur le serveur

- Démarrer **nginx-proxy** avec la bonne configuration https://hub.docker.com/r/nginxproxy/nginx-proxy
- Déployer les applications wordpress et student-list en déclarant l'env var **VIRTUAL_HOST**
- Mettre à jour le fichier hosts de votre PC (il faut être admin) en créant des entrées DNS pour les apps
  - wordpress.eazytraining.io
  - student-list.eazytraining.io
- Tester via le navigateur
- Automatiser le déploiement de notre serveur via docker-compose

Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

Sécurisation du daemon docker

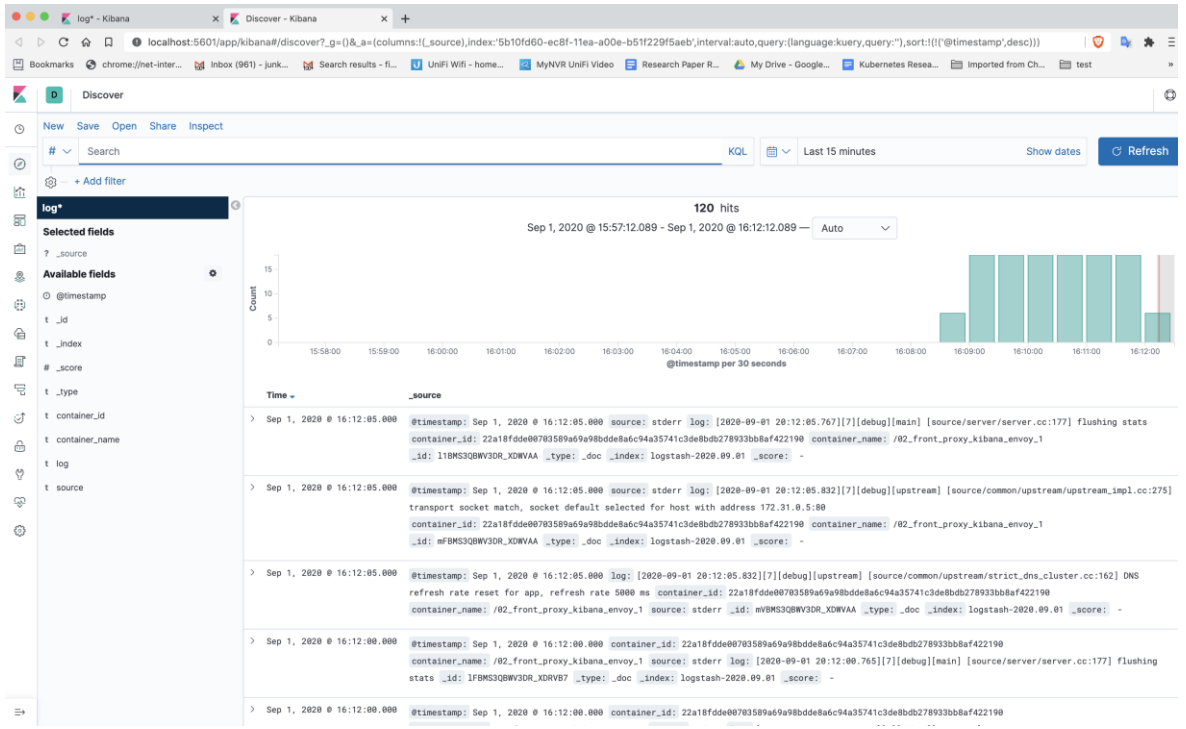
Ingress - Expose application

Observabilité

Orchestration

Observabilité

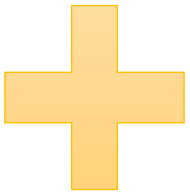
- Vérifier la stabilité du système
- Surveiller les métriques
- Debug en cas de fail



Observabilité: Monitoring

- Collecter les métriques des applications et du serveur
  - Cpu, disk, memory, IO, customs metrics
- Visualisation utilisation des ressources
- Debug

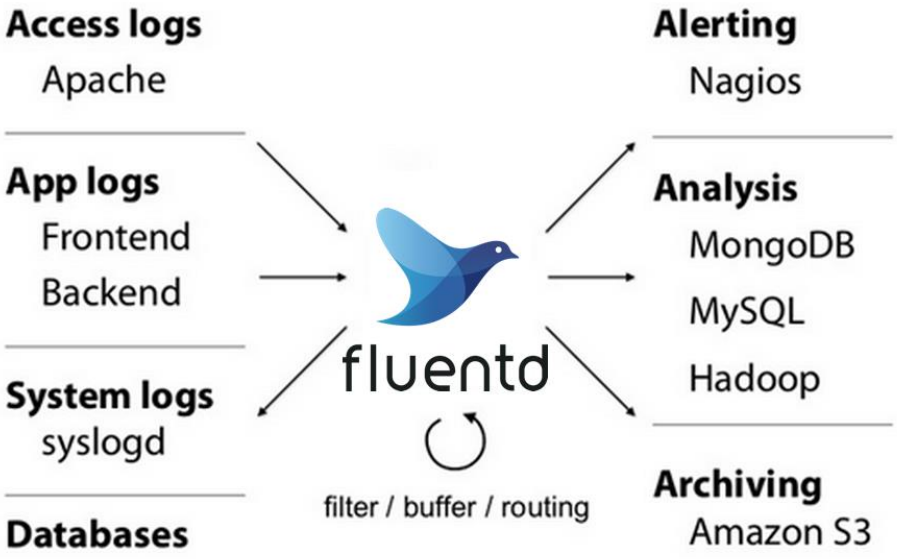
```
[vagrant@master ~]$ docker stats --no-stream
CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT     MEM %     NET I/O
0cb2ce1d7546   task-client-angular   0.30%     364.1MiB / 1.795GiB   19.81%    656B / 0B
ec5d5d097568   apache              0.01%     11.26MiB / 1.795GiB   0.61%     656B / 0B
e559f528df7b   nginx               0.00%     2.086MiB / 1.795GiB   0.11%     2.81kB / 1.7kB
```



Observabilité: Logging

- Collecter les logs des différents conteneurs
- Logs centralisés
- Faciliter les opérations de recherche sur les logs
- Déployer la stack EFK

```
[vagrant@master ~]$ docker logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/07/09 11:27:16 [notice] 1#1: using the "epoll" event method
2022/07/09 11:27:16 [notice] 1#1: nginx/1.23.0
```



TP-9: Monitoring et gestion des logs

- Monitoring
  - Déployer Graphana et Prometheus https://github.com/stefanprodan/dockprom
  - Visualiser les métriques du serveur

- Logging
  - Déployer la stack EFK https://docs.fluentd.org/container-deployment/docker-compose pour monitorer votre serveur
  - Générer des logs de connexion sur le serveur web apache créé.
  - Accéder à l'interface de Kibina pour voir les logs

Plan

Présentation du formateur

Rappel par la pratique

Gestion avancée des images

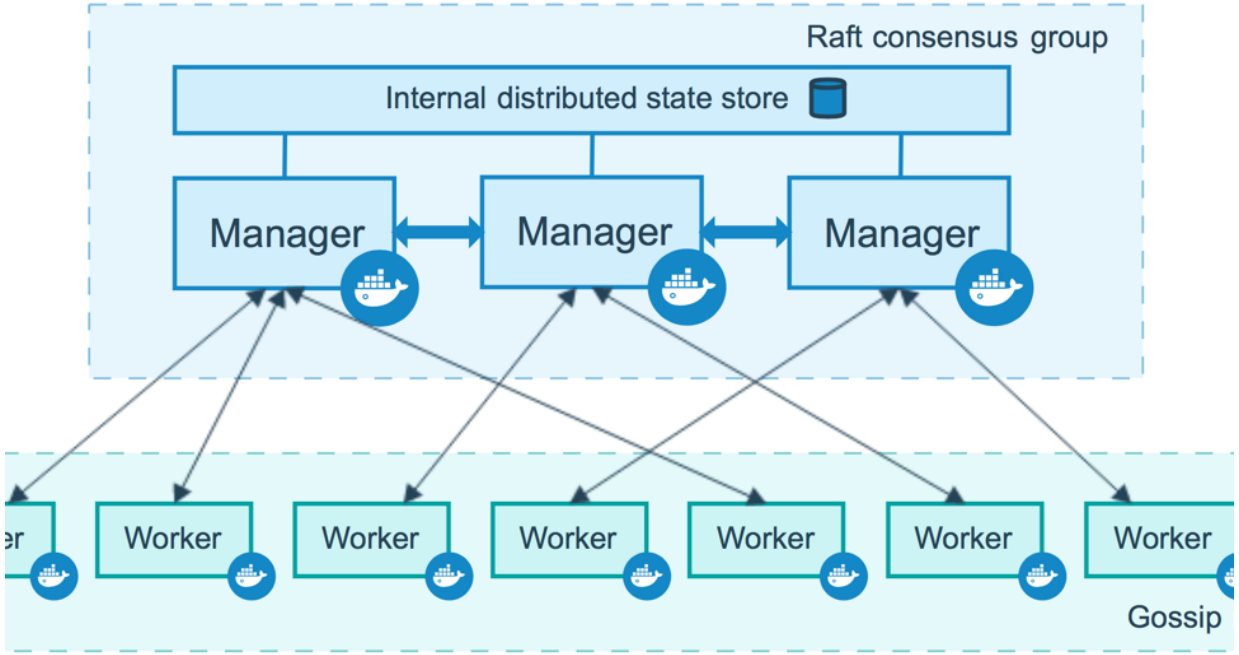
Sécurisation du daemon docker

Ingress - Expose application

Observabilité

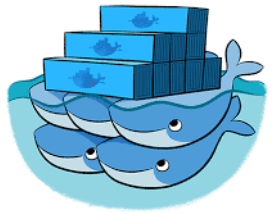
Orchestration

Orchestration: pourquoi ?



- Etat des applications /conteneurs
- Auto-discovery
- Rolling Update / Blue-Green deployment / rollbacks
- Scalabilité des services
- Load balancing
- Secret Management
- Fail-over app

Solutions d’orchestration



Apache MESOS™

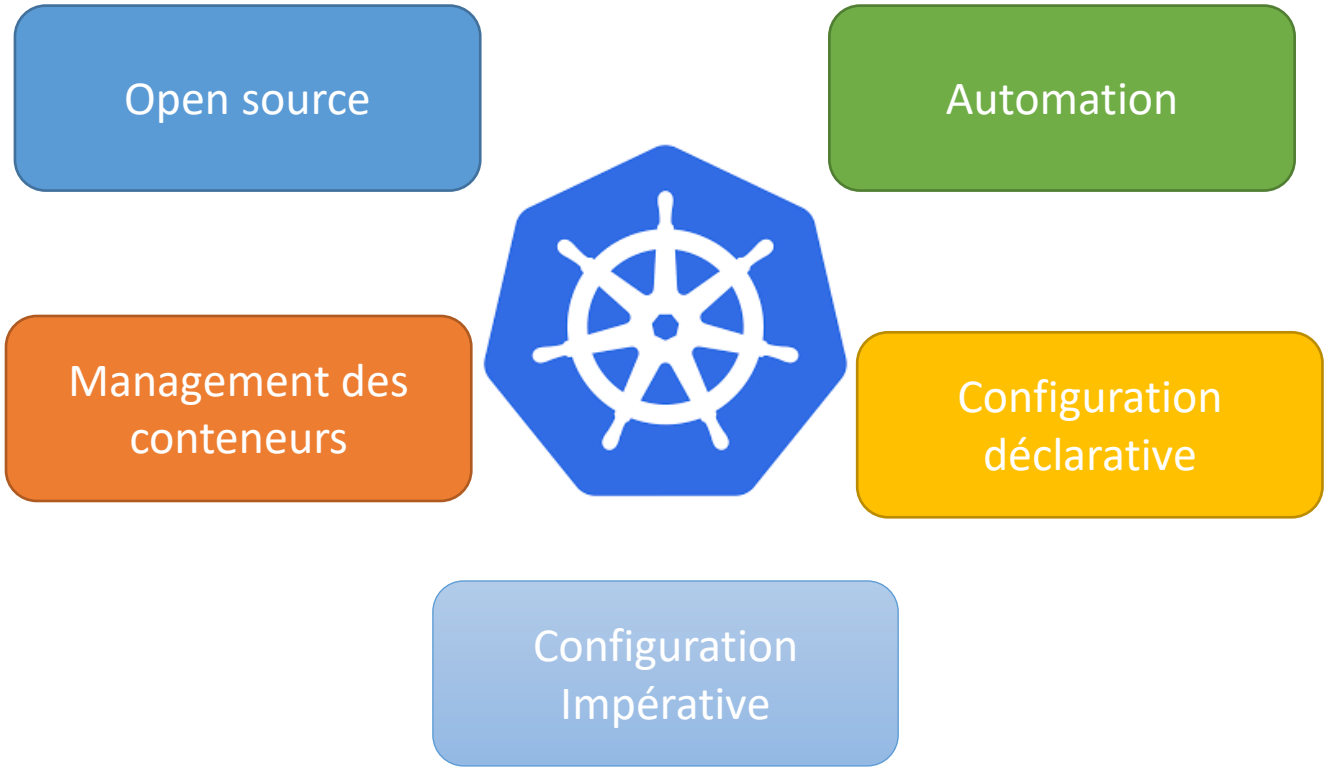
RANCHER

Google Kubernetes Engine

Amazon EKS

AZURE KUBERNETES SERVICE

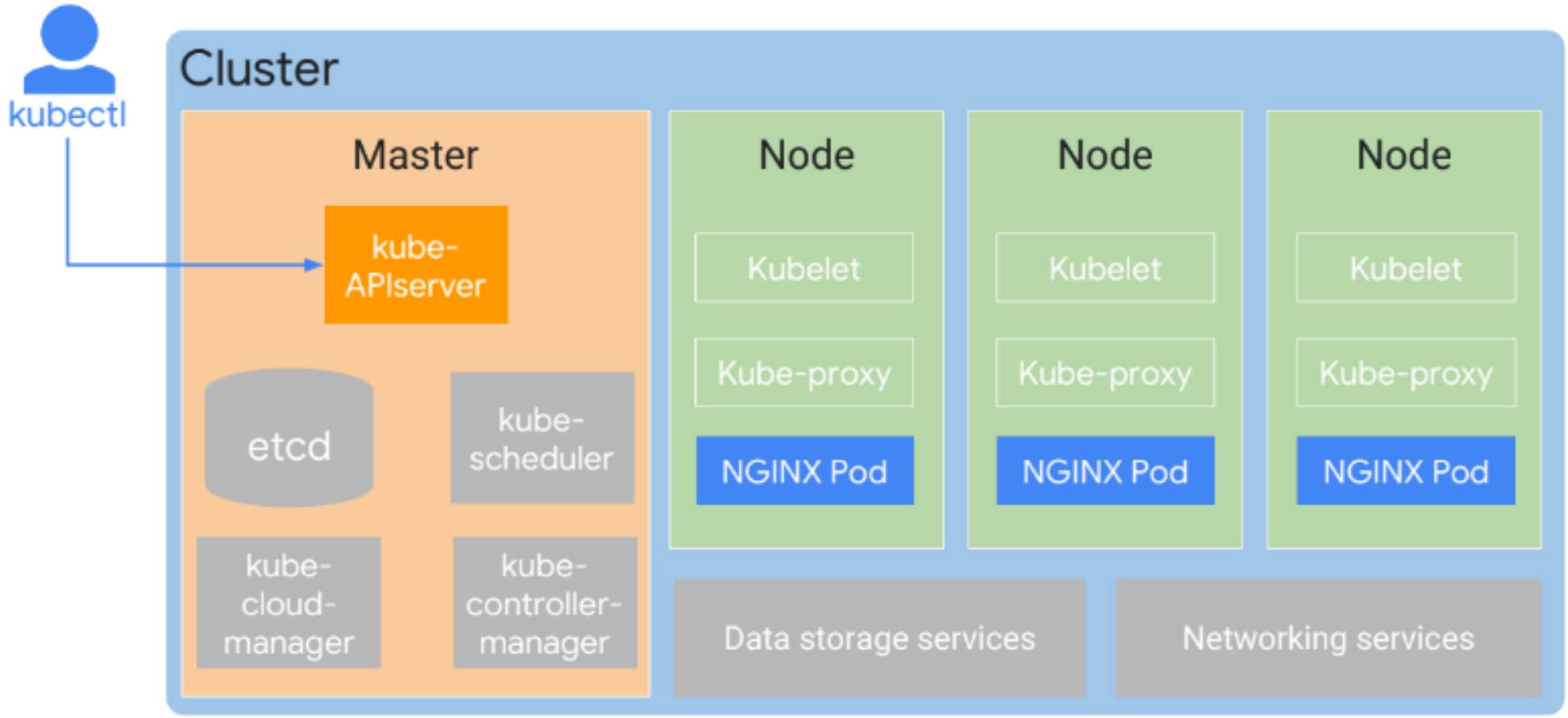
Kubernetes


Open source
Automation
Management des conteneurs
Configuration déclarative
Configuration Impérative

Kubernetes features

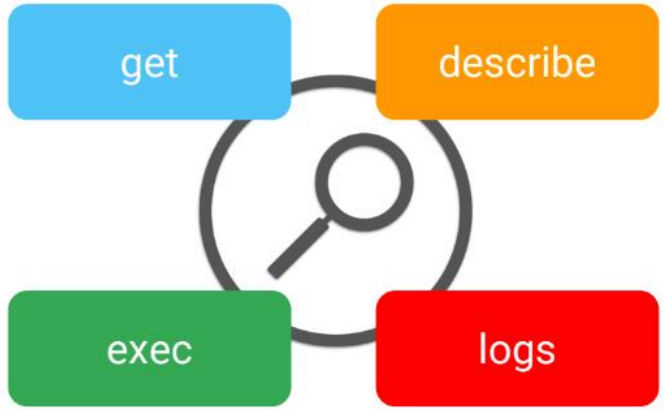
- Supporte différents types de workload
  - Stateful, stateless, batch jobs, daemon
- Gère l’autoscaling
- Gestion des ressources
- Extensible
- Portable

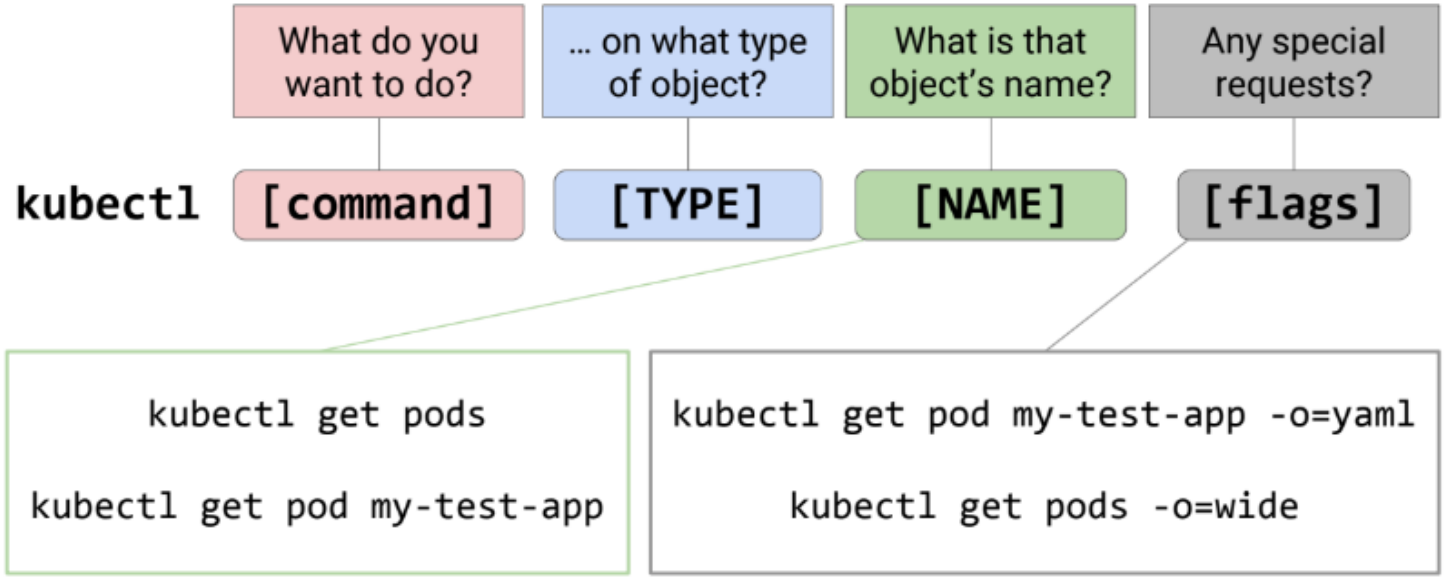
Kubernetes composants



Kubernetes composants: kubectl

- Utilitaire utilisé pour contrôler le cluster
- Permet de manager des objets K8s
- Permet l'inspection et l'export des configs
- Introspection d'application

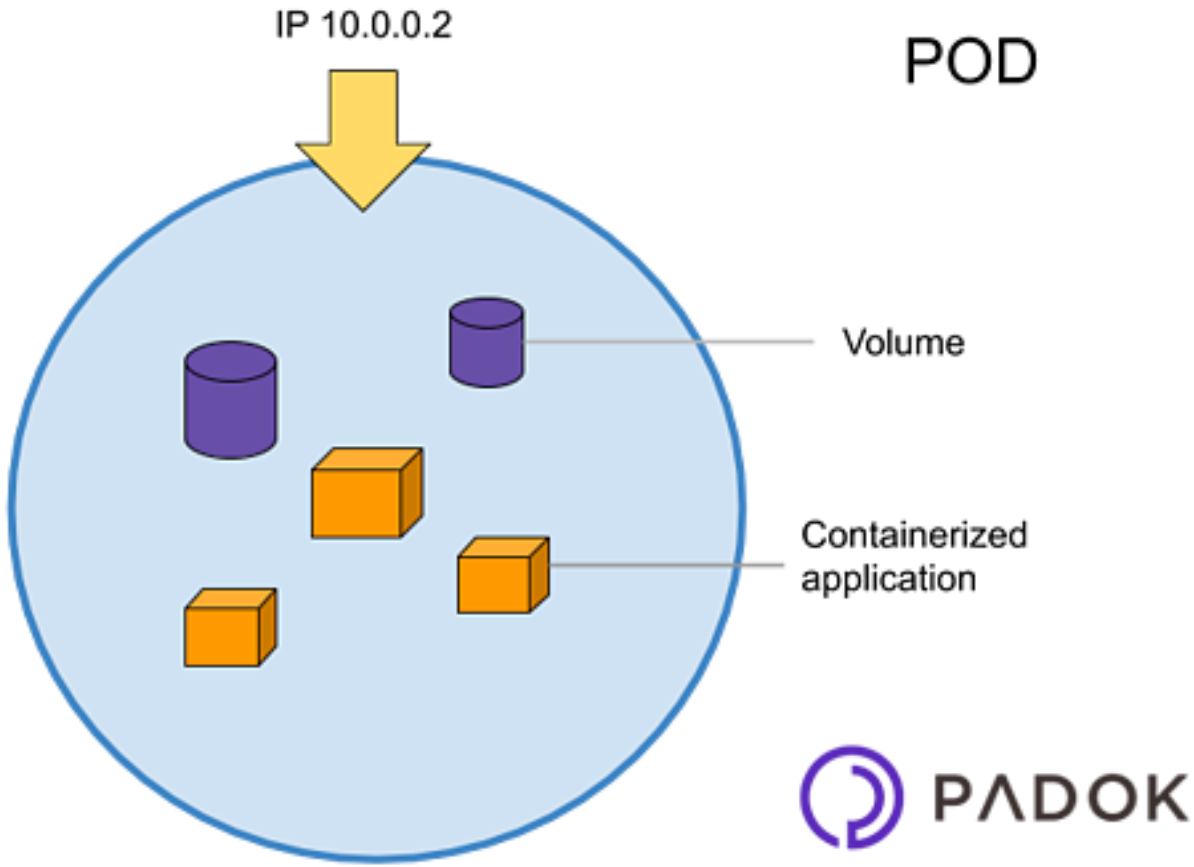




Les objets de Kubernetes: pod

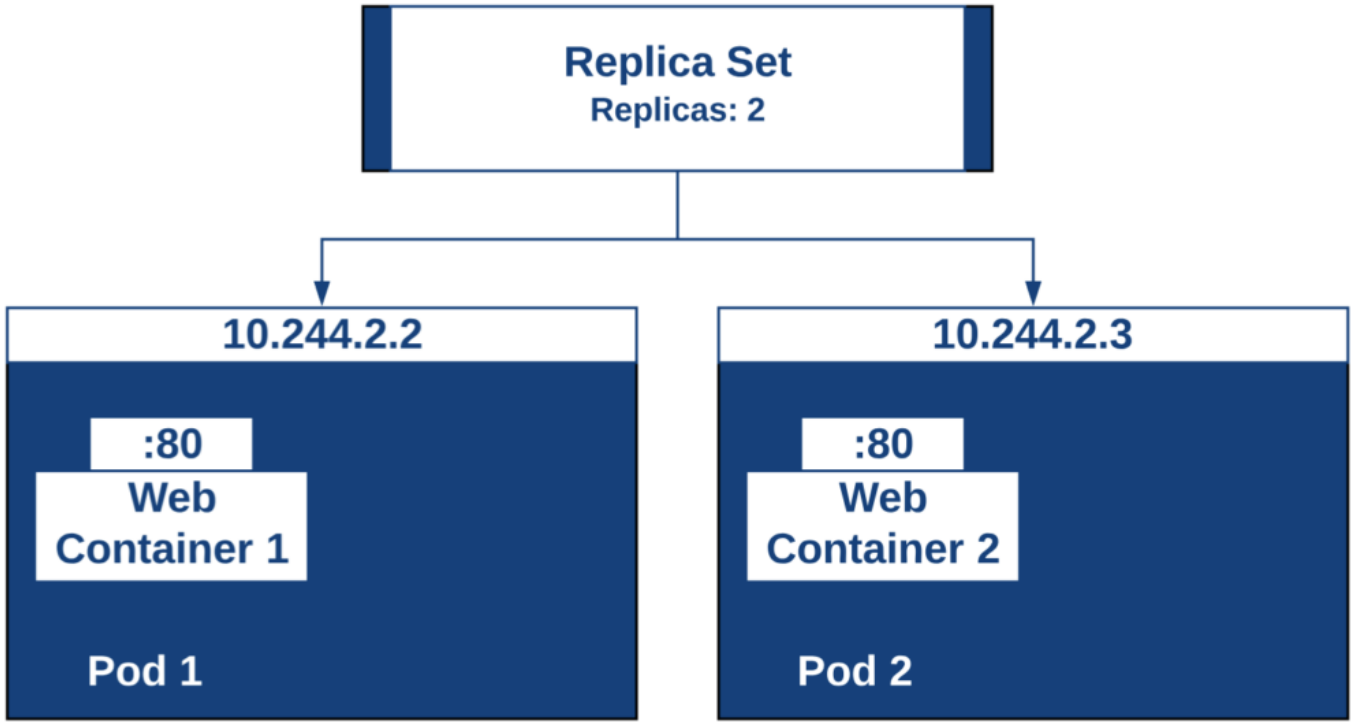
- Plus petite unité d'exécution
- Espace réseau
- Espace de volume
- Données Ephémères

```yaml
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name:  nginx
    image: nginx:latest
```



Les objets de Kubernetes: Replica set

- Scalabilité
- Résilience



Kubernetes: Bonnes pratiques

- Utiliser les namespaces
- Readiness et liveness probes
- Mettre en place les limites de resources pour les différents objets
- Déployer les pods comme deployment, replicaset ou statefuset
- Utiliser des charts helm pour vos déploiements
- Faire du Gitops