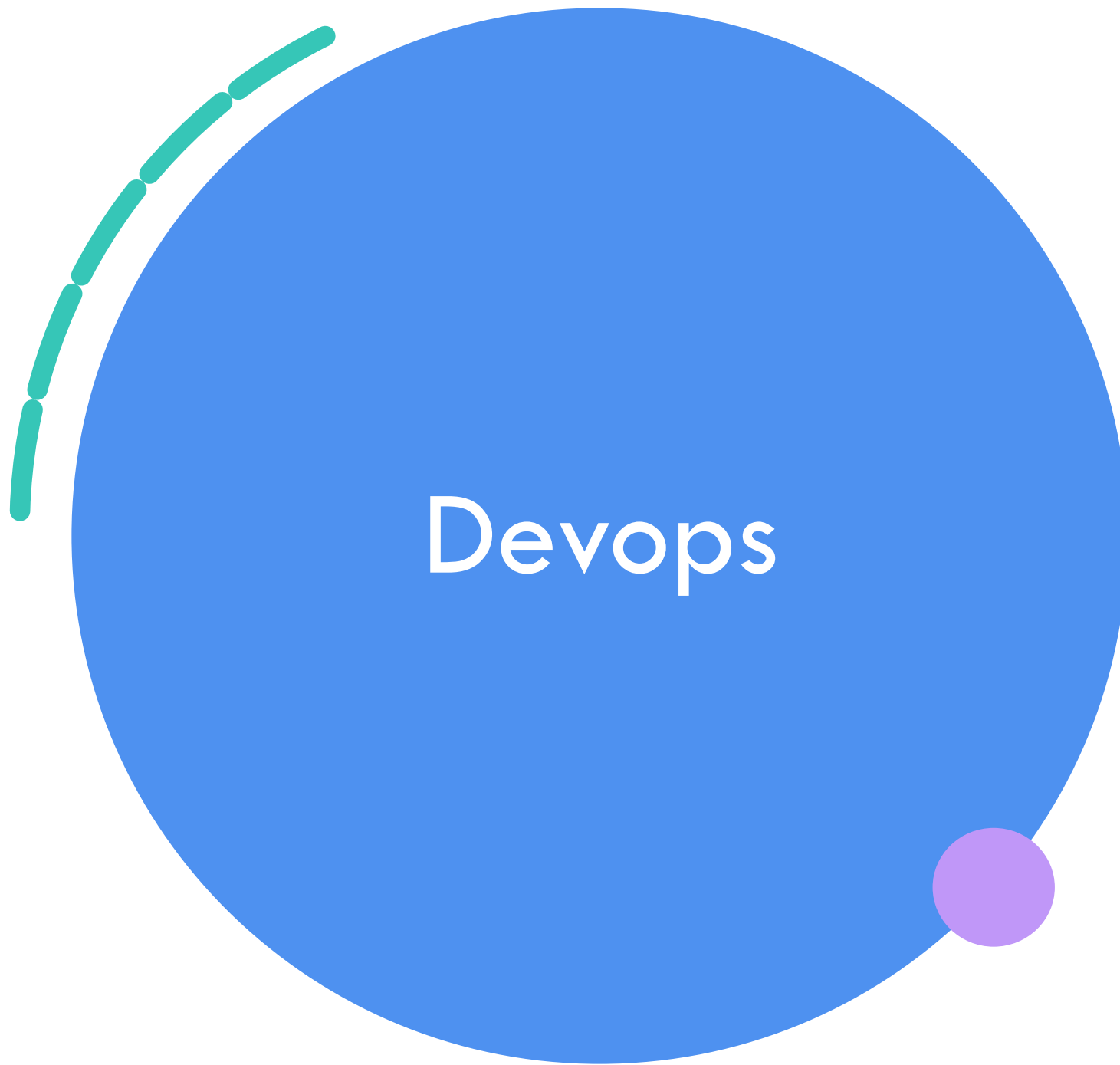
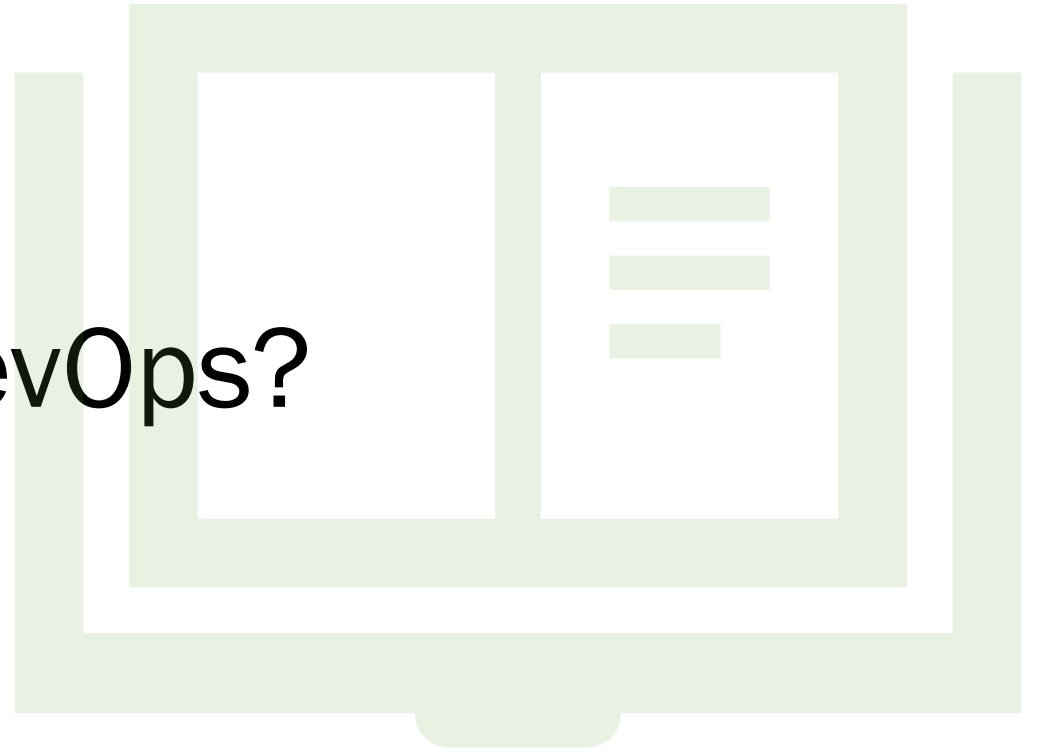




Devops



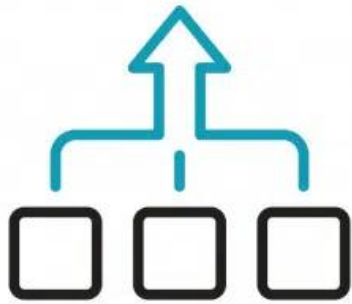
Qu'est ce que le DevOps?





En 2008, certains membres du mouvement agile tel que Patrick Debois et Andrew Shafer ont cherché à appliquer les préceptes de l'agilité a la branche infrastructure des systèmes d'information

Concepts fondamentaux



Un objectif commun :

Il convient de définir un objectif commun pour l'ensemble des équipes de façon simple et claire il doit être communiqué de façon régulière à l'ensemble des parties prenantes.



Responsabilisation :

Chaque participant doit porter la responsabilité de ses actions et de leurs conséquences positives ou négatives, et communiquer sur l'avancement de ses tâches en toute transparence



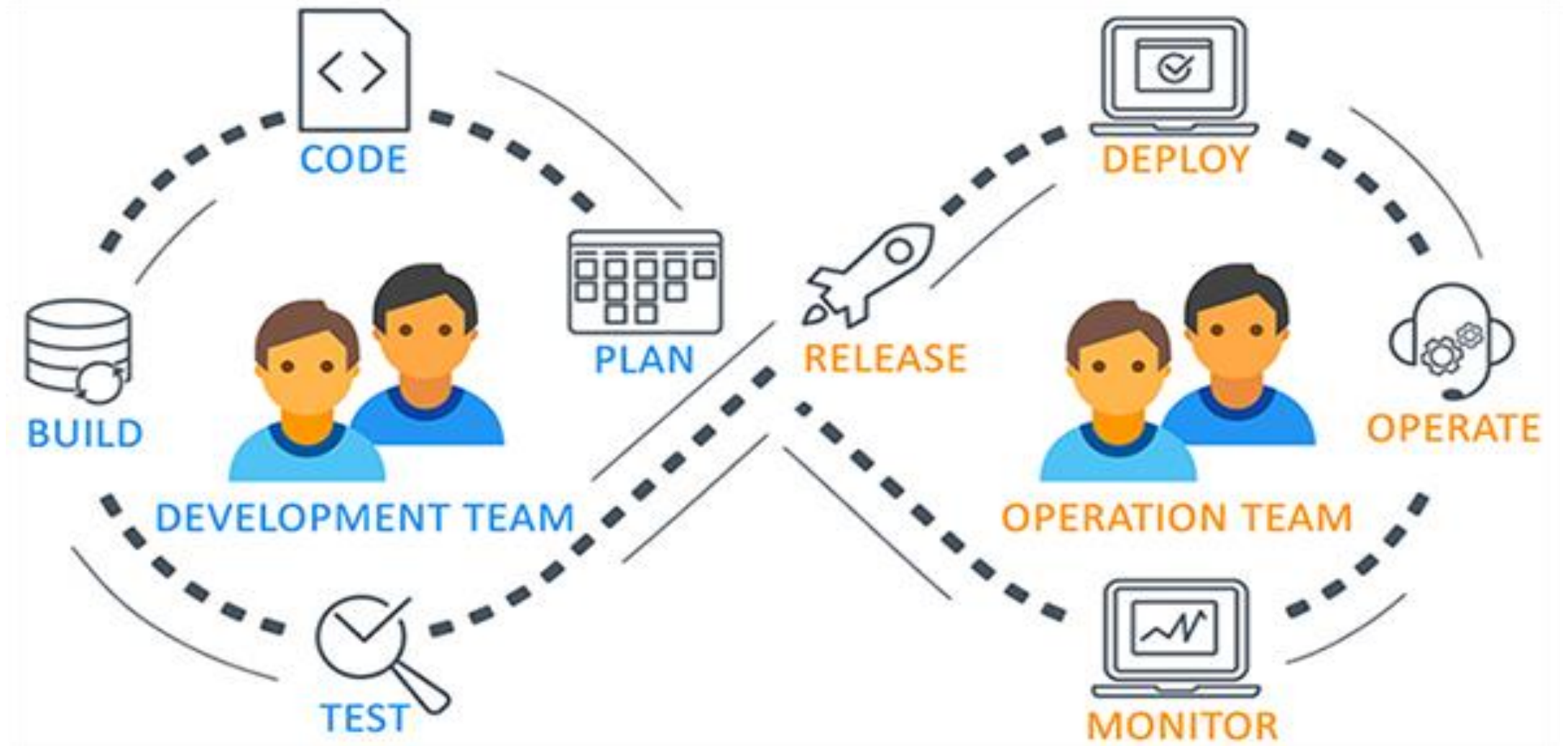
Amélioration continue:

Il est recommandé d'intervenir sur le système par petites itérations et de capitaliser sur les réussites et les échecs de chaque étape.



Source de vérité unique :

Il est préconisé de stocker l'ensemble des informations relatives à une application au plus proche de l'application elle-même.



Une méthode de travail où les équipes de développement, d'exploitation informatique et de sécurité s'associent pour créer, tester une application et fournir un retour d'information régulier tout au long du cycle de vie du développement logiciel.

Qu'est ce que l'approche CI/CD en DevOps?



Dans le cadre du développement et de l'exploitation d'applications , le CI/CD rationalise le codage, les tests et le déploiement des applications en offrant un référentiel unique pour stocker le travail et des outils d'automatisation pour combiner et tester le code de manière cohérente afin de garantir son bon fonctionnement

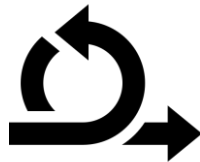
Avantages du DevOps?



Accroît la coopération et la compréhension entre les départements d'une organisation, entre les départements des opérations et du développement



Augmente la cadence de production et de déploiement des développement logiciels par le biais de système de production



Offre une grande agilité, ce qui se traduit par une amélioration de la qualité grâce a une vérification quasi-continue des produit dès les premiers stades du développement et à la réduction des erreurs potentielles.



Une plus grande de sécurité des applications développées, grâce notamment à l'automatisation des procédures de contrôle des menaces et des vulnérabilités.

Différents outils DevOps

Gestion du code source	Git, Subversion, Github, Gitlab ainsi que Bitbucket
D'intégrations continue et de déploiement continu	Jenkins, GitlabCI, Bamboo, TeamCity, Concourse, CircleCI ou Travis CI
Pour l'automatisation	Terraform, Ansible, Puppet et Salt
Monitoring et d'alerte	Prometheus et Grafana
Gestion de projet	Jira et Trello
Conteneurs	Docker, RKT
Cloud providers	Google Cloud platform, AWS et Azure



Linux

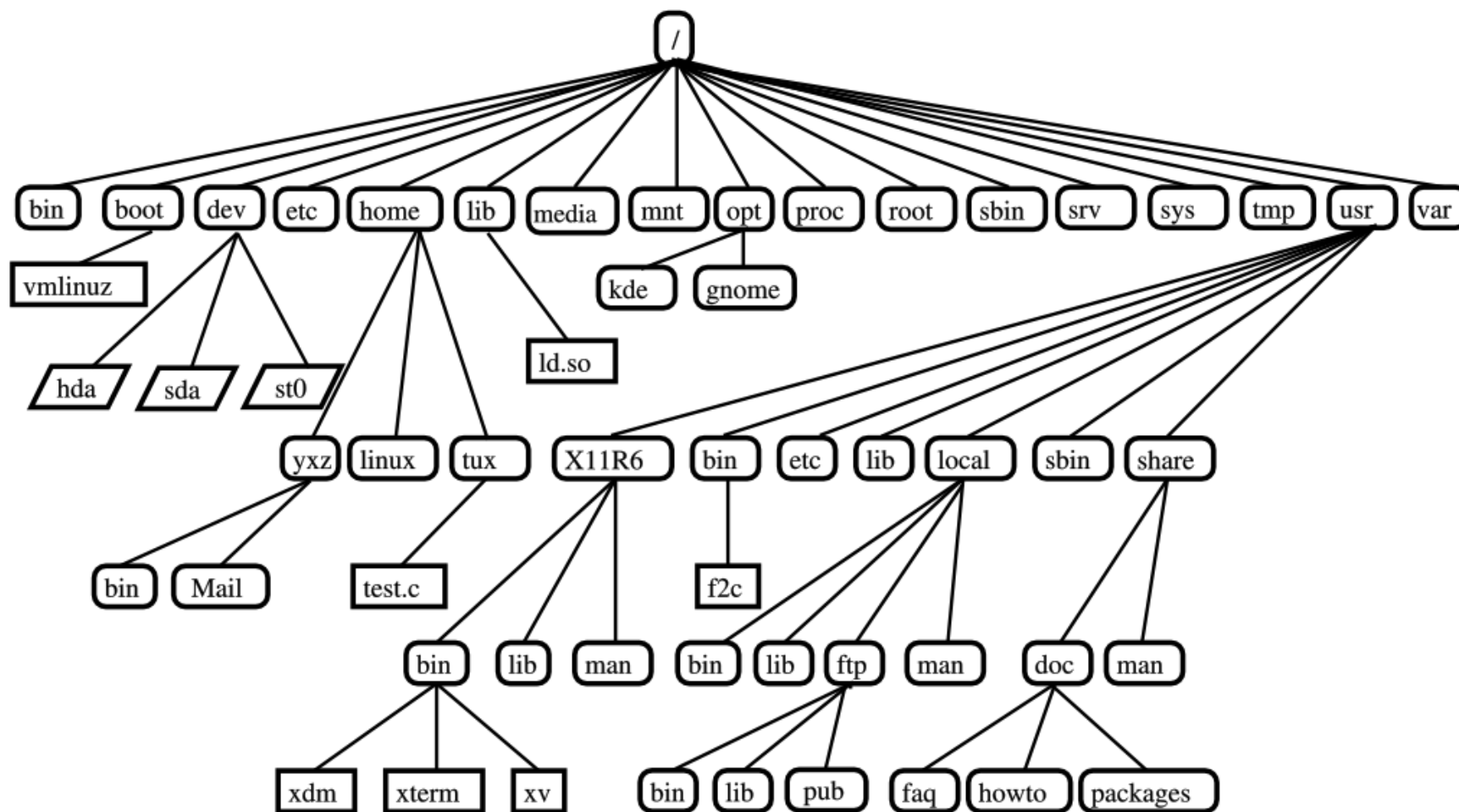


Qu'est-ce que Linux ?



Linux a été crée en 1991 par un développeur finlandais, Linus Torvalds. Le but de ce dernier était de produire un noyau de système d'exploitation libre pour les ordinateur personnels. Il a réussi, et ce noyau n'a cessé de croitre au fil des ans.

Linux utilise le standard FHS pour définir son arborescence. Ce standard propose une structure de répertoires dont chacun possède un rôle spécifique défini dans FHS





Les commandes linux

La commandes **pwd** :

permet d'afficher l'emplacement ou on se situe actuellement dans la hiérarchie

```
[root@centos ~]# pwd  
/root
```

La commandes **cd** :

permet de changer de répertoire courant et de se situer sur un autre

```
[root@centos ~]# cd /home/  
[root@centos home]# pwd  
/home
```

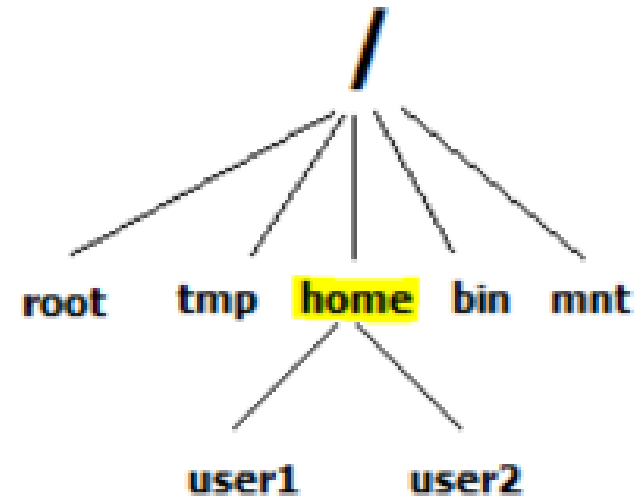


Les commandes linux

La commandes **ls** :

permet de lister les fichier disponible dans un répertoire, si appelé sans arguments, ls liste mes fichier du répertoire courant

```
[root@centos home]# ls  
user1 user2
```





Les commandes linux

La commandes **mkdir** :
permet de créer un répertoire

```
[root@centos home]# cd /tmp
[root@centos tmp]# pwd
/tmp
[root@centos tmp]# mkdir repertoiretest
[root@centos tmp]# ls
repertoiretest
fichiertemp
```

La commandes **rmdir** :
permet de supprimer un répertoire

```
[root@centos tmp]# rmdir repertoiretest
[root@centos tmp]# ls
fichiertemp
```



Les commandes linux

La commandes **touch** :

Permet de changer la date du dernier accès ou modification d'un fichier, mais permet également de créer un fichier vide.

```
[root@centos home]# cd /tmp
[root@centos tmp]# pwd
/tmp
[root@centos tmp]# mkdir repertoiretest
[root@centos tmp]# ls
repertoiretest
fichiertemp
```



Les commandes linux

La commandes **cp** :

Permet de copier un fichier ou un répertoire

```
[root@centos tmp]# cp fichier fichier2  
[root@centos tmp]# ls  
fichier  
fichier2  
fichiertemp
```



Les commandes linux

La commandes **rm** :

Permet de supprimer un fichier ou un répertoire.

```
[root@centos tmp]# rm -f fichier fichier2  
[root@centos tmp]# ls  
fichiertemp
```

À utiliser avec précaution car avec l'option **-f** ou **rf**, cette commande peut endommager voir supprimer tout votre système de fichier de manière irréversible



Les commandes linux

La commandes **cat :**
Permet d'afficher le contenu d'un fichier

```
[root@ip8 tmp]# cat /home/fichier4  
[root@ip8 tmp]#
```

```
[root@centos tmp]# cat /etc/resolv.conf  
# Generated by NetworkManager  
nameserver 8.8.8.8
```



Les commandes linux

La commandes **echo** :

Permet d'afficher une ligne

```
[root@centos tmp]# echo "tutoriel Linux de sitedetout"  
tutoriel Linux de sitedetout
```

Permet aussi d'écire du contenu dans un fichier moyennant le signe « > » pour écraser le contenu du fichier ou « >> » pour suffixer le contenu du fichier

```
[root@centos tmp]# cat /home/fichier4  
[root@centos tmp]# echo contenu > /home/fichier4  
[root@centos tmp]# cat /home/fichier4  
contenu  
[root@centos tmp]# echo contenu2 > /home/fichier4  
[root@centos tmp]# cat /home/fichier4  
contenu2  
[root@centos tmp]# echo contenu3 >> /home/fichier4  
[root@centos tmp]# cat /home/fichier4  
contenu2  
contenu3
```



Les commandes linux

La commandes **man** :

Permet d'afficher le manuel d'aide de n'importe quelle autre commande disponible :

```
LS(1)                                Manuel de l'utilisateur Linux                                LS(1)

NOM
ls, dir, vdir - Afficher le contenu d'un répertoire

SYNOPSIS
ls [options] [fichier...]
dir [fichier...]
vdir [fichier...]

Options POSIX : [-CFRadilqrutl] [--]

Options GNU (forme courte) : [-labcdfgiklmnopqrstuvwXABCFGHILNQRSUX] [-w cols] [-T cols] [-I motif] [--full-time]
[--show-control-chars] [--block-size=taille] [--format={long,verbose,commas,across,vertical,single-column}]
[--sort={none,time,size,extension}] [--time={atime,access,use,ctime,status}] [--color={none,auto,always}] [--help]
[--version] [--]

DESCRIPTION
La commande ls affiche tout d'abord l'ensemble de ses arguments fichiers autres que des répertoires. Puis ls affiche
l'ensemble des fichiers contenus dans chaque répertoire indiqué. Si aucun argument autre qu'une option n'est fourni,
l'argument « . » (répertoire en cours) est pris par défaut. Avec l'option -d, les répertoires fournis en argument ne
sont pas considérés comme des répertoires (on affiche leurs noms et pas leurs contenus). Un fichier n'est affiché que
si son nom ne commence pas par un point, ou si l'option -a est fournie.

Chacune des listes de fichiers (fichiers autres que des répertoires, et contenu de chaque répertoire) est triée
séparément en fonction de la séquence d'ordre de la localisation en cours. Lorsque l'option -l est fournie, chaque
liste est précédée d'une ligne indiquant la taille totale des fichiers de la liste, exprimée en blocs de 512 ou
1024 octets.

Le résultat est envoyé sur la sortie standard, un élément par ligne, sauf si un affichage multi-colonnes est demandé
avec l'option -C. Toutefois, lorsque la sortie standard est un terminal, l'affichage peut se faire sur une ou plusieurs
colonnes suivant l'implémentation. Les options -l et -C forcent un affichage sur une seule colonne ou plusieurs,
respectivement.

OPTIONS POSIX
-C Présenter les fichiers en colonnes, triés verticalement.

:|
```



Les commandes linux

La commandes **ln** :

Cette commande est utilisé pour créer un lien matériel ou symbolique (raccourci) vers un fichier ou un répertoire

```
[root@centos tmp]# touch fichiersource
[root@centos tmp]# echo "contenu" > fichiersource
[root@centos tmp]# cat fichiersource
contenu
[root@centos tmp]# ln -s fichiersource fichiercible
[root@centos tmp]# cat fichiercible
contenu
[root@centos tmp]# echo "contenu2" >> fichiersource
[root@centos tmp]# cat fi
fichiercible fichiersource firefox_webadm/
[root@centos tmp]# cat fichiersource
contenu
contenu2
[root@centos tmp]# cat fichiercible
contenu
contenu2
```



Les commandes linux

La commandes **sudo** :

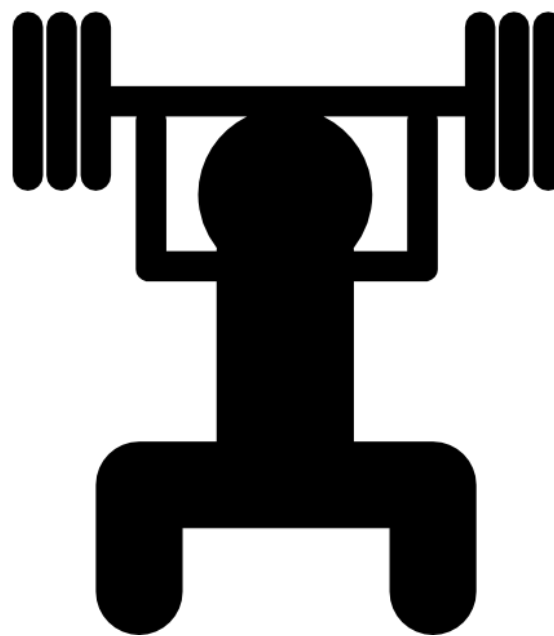
Permet à certains utilisateur la possibilité de lancer une commande en tant qu'administrateur

```
adrien@linux: ~ $ ls /root
ls: impossible d ouvrir le répertoire /root: Permission non accordée
adrien@linux: ~ $ sudo ls /root
[sudo] password for adrien:
script-de-root.sh
```

La commandes **apt-get** :

Permet d'effectuer l'installation et la désinstallation de paquet en provenance d'un dépôt APT.

```
sudo apt-get install <packet> <packet 2> etc ...
```



Exercise



Les utilisateur et droit linux

Un système d'exploitation GNU/Linux a trois types d'utilisateurs et trois type de droits distincts :

- Le propriétaire du fichier (user)
 - Le groupe du propriétaire du fichier (group)
 - Les autres utilisateurs, ou encore le reste du monde (others)
- r : droit de lecture (read)
 - w : droit d'écriture (write)
 - X : droit d'exécution (execute)



Les utilisateur et droit linux

La commandes **ls -l** :

Permet d'afficher les droits des fichiers et des répertoires

```
[don@localhost ~]$ ls -l
total 32
drwxr-xr-x. 2 don don 4096 Aug 15 17:20 Desktop
drwxr-xr-x. 2 don don 4096 Aug 15 17:20 Documents
drwxr-xr-x. 2 don don 4096 Sep 18 12:14 Downloads
drwxr-xr-x. 2 don don 4096 Aug 15 17:20 Music
-rw-rw-r--. 1 don don    0 Sep 27 10:41 one.txt
drwxr-xr-x. 2 don don 4096 Sep 18 12:06 Pictures
drwxr-xr-x. 2 don don 4096 Aug 15 17:20 Public
drwxr-xr-x. 2 don don 4096 Aug 15 17:20 Templates
-rw-rw-r--. 1 don don    0 Sep 27 10:41 three.txt
-rw-rw-r--. 1 don don    0 Sep 27 10:41 two.txt
drwxr-xr-x. 2 don don 4096 Aug 15 17:20 Videos
```



Les utilisateur et droit linux

Correspondances des droits en binaire/octale et leurs significations

Position Binaire	Valeur octale	Droits	Signification
000	0	- - -	Aucun droit
001	1	- - x	Exécutable
010	2	- w -	Ecriture
011	3	- w x	Ecrire et exécuter
100	4	r - -	Lire
101	5	r - x	Lire et exécuter
110	6	r w -	Lire et écrire
111	7	r w x	Lire écrire et exécuter



Les utilisateur et droit linux

Récapitulatif des différentes valeurs possibles fréquemment utilisées

644 - Lecture, écriture pour le propriétaire / Lecture pour les autres			
Valeur par défaut d'un fichier sous GNU/Linux			
Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w -	r - -	r - -
Position Binaire	110	100	100
Valeur Octale	6	4	4



Les utilisateur et droit linux

666 - Lecture, écriture pour tout le monde

Déconseillé

Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w -	r w -	r w -
Position Binaire	110	110	110
Valeur Octale	6	6	6

700 - Lecture, écriture, execution juste pour le propriétaire

Valeur par défaut d'un dossier sous GNU/Linux

Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w x	- - -	- - -
Position Binaire	111	000	000
Valeur Octale	7	0	0



Les utilisateur et droit linux

705 - Le propriétaire à tous les droits / Le groupe aucun / Les autres lire et executer

Préconisé par certains providers pour le répertoire du site, si celui-ci est inaccessible (message : "Forbidden...")

Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w x	- - -	r - x
Position Binaire	111	000	101
Valeur Octale	7	0	5

La ligne de code à entrer en FTP (sur Filezilla) en cas de souci avec certains providers est : **site chmod 705 /**



Les utilisateur et droit linux

755 - Le propriétaire à tous les droits / Les autres lire et executer

Utile pour des scripts par exemple et certains fichiers d'un site web

Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w x	r - x	r - x
Position Binaire	111	101	101
Valeur Octale	7	5	5

764 - Tous droits pour le propriétaire / Lecture, écriture pour le groupe / Lecture seule pour les autres

Parfois utile pour des fichiers d'un site web appartenant au groupe [www-data](#)

Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w x	r w -	r - -
Position Binaire	111	110	100
Valeur Octale	7	6	4



Les utilisateur et droit linux

774 - Tous les droits pour le propriétaire et le groupe / Lecture seule pour les autres

Utile pour certains fichiers d'un site sur un serveur local de développement

Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w x	r w x	r - -
Position Binaire	111	111	100
Valeur Octale	7	7	4

775 - Tous les droits pour le propriétaire et le groupe / Lecture et exécution pour les autres

Très pratique pour se simplifier la vie avec la gestion d'un site en développement sur un serveur local (dans le dossier media)

Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w x	r w x	r - x
Position Binaire	111	111	101
Valeur Octale	7	7	5



Les utilisateur et droit linux

777 - Tous les droits pour tous

Fortement déconseillé ! Mais peut-être nécessaire pour le cache de CMS en local (par exemple) sur un [serveur Lamp](#)

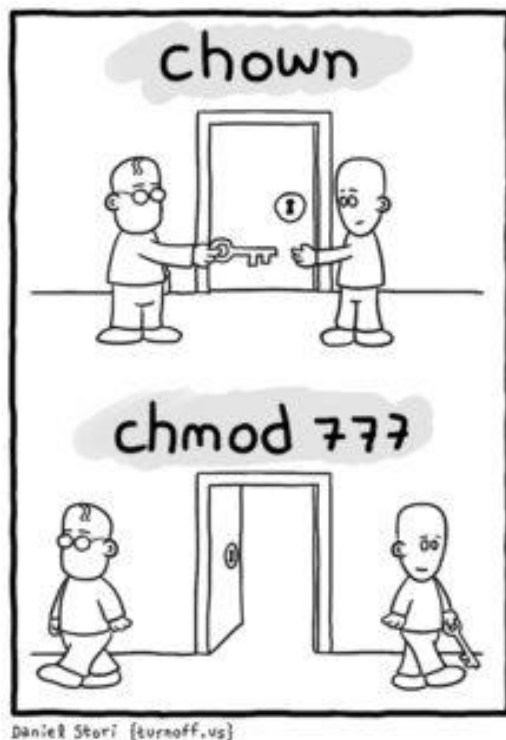
Type d'utilisateurs	Propriétaire	Groupe	Les autres
Droits	r w x	r w x	r w x
Position Binaire	111	111	111
Valeur Octale	7	7	7



Les utilisateur et droit linux

La commandes **chown** et **chmod** :

chown définit le propriétaire d'un fichier, **chmod** définit qui peut faire quoi

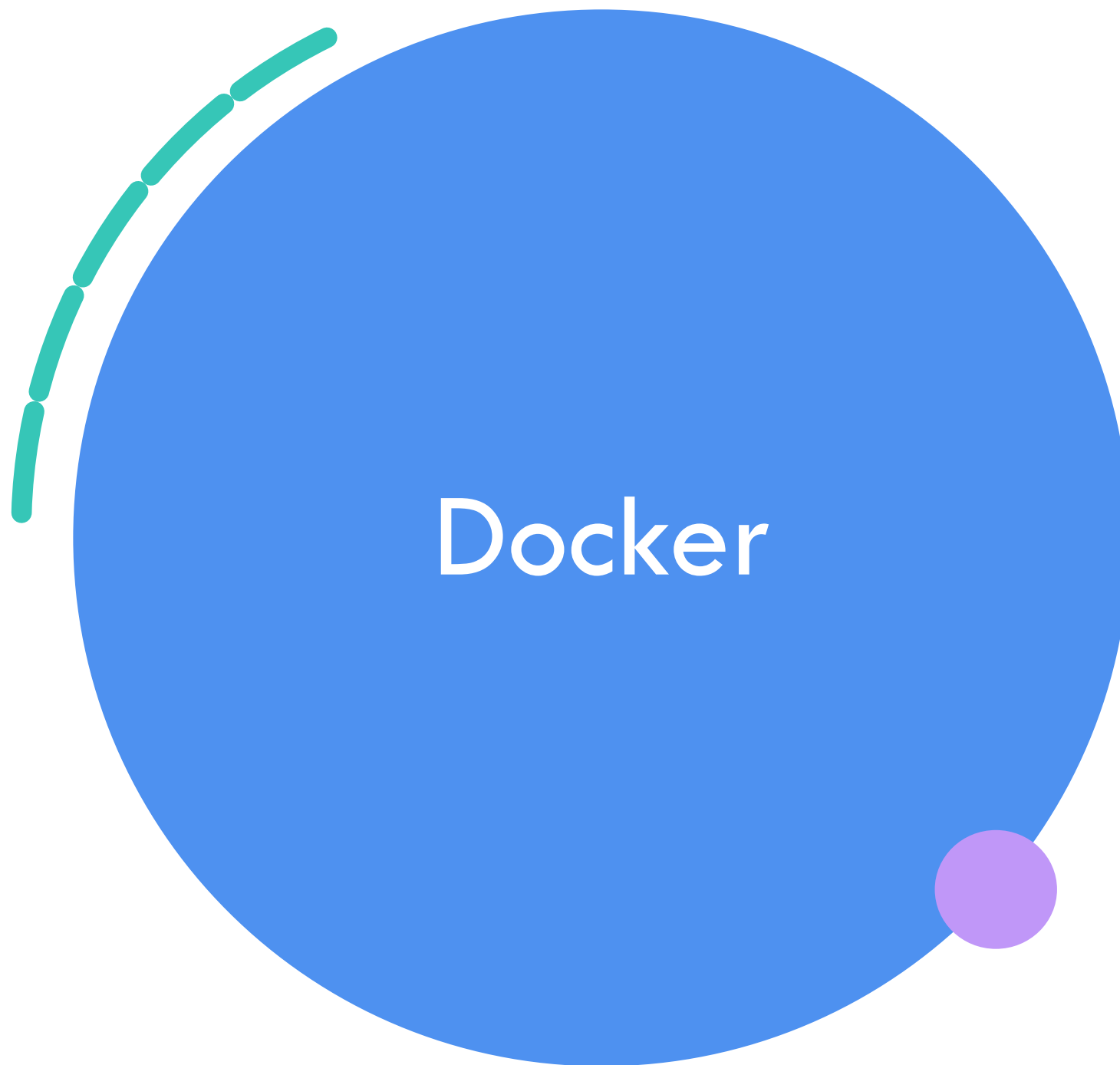


Exemple, si vous souhaitez vous réapproprier les droits sur l'ensemble d'un dossier et que vous êtes utilisateur « martin » :

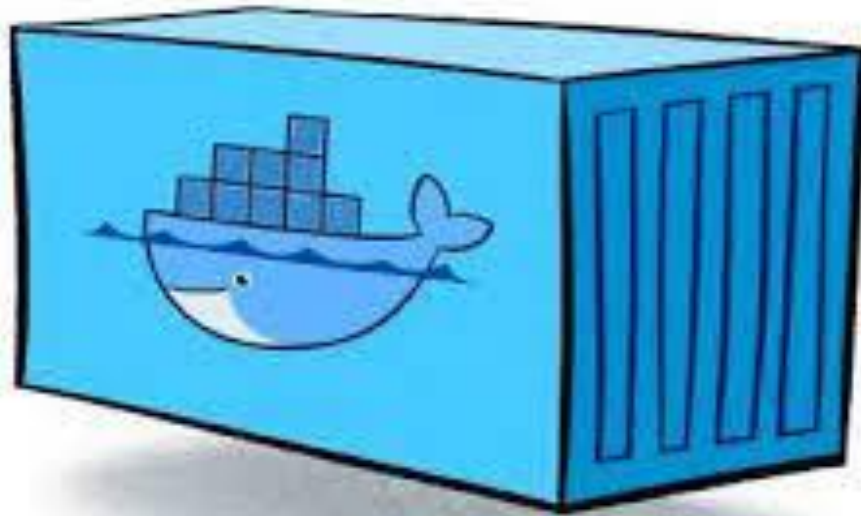
```
sudo chown -R martin /chemin/vers/dossier
```

Pour récupérer les droits sur l'ensemble du dossier on lance :

```
sudo chmod -R 755 /chemin/vers/dossier
```

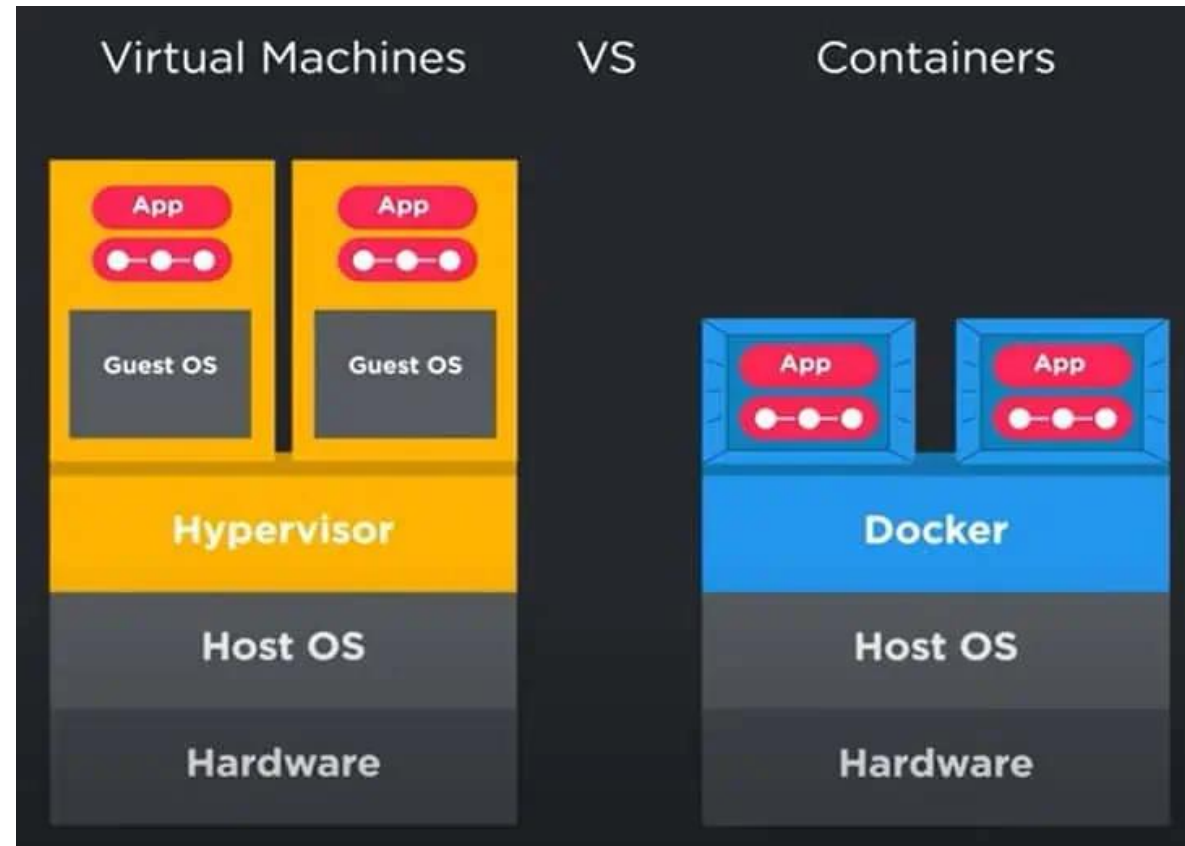


Qu'est ce qu'un conteneur?

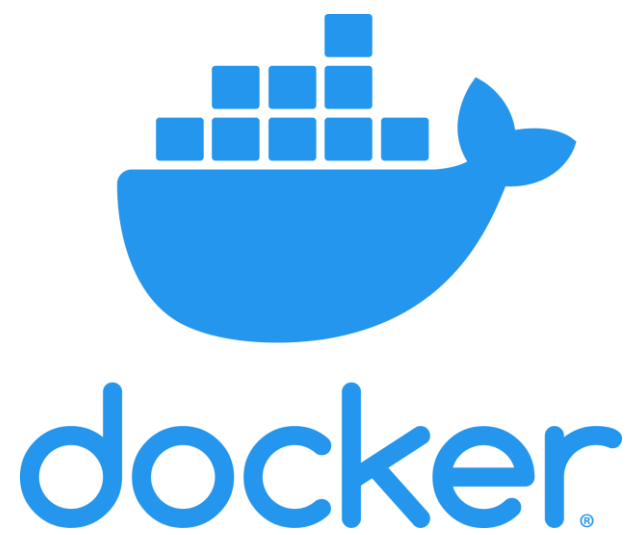


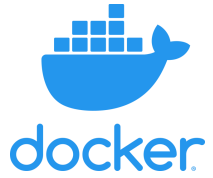
- Un environnement permettant d'isoler des applications et leurs dépendances respective.
- Il peut être relié à d'autres conteneurs.
- Vont seulement puiser les ressources nécessaires à l'exécution de l'application dans l'os principal la machine physique.
- Permet donc de simuler plusieurs système sur le même os.

Containers vs VM

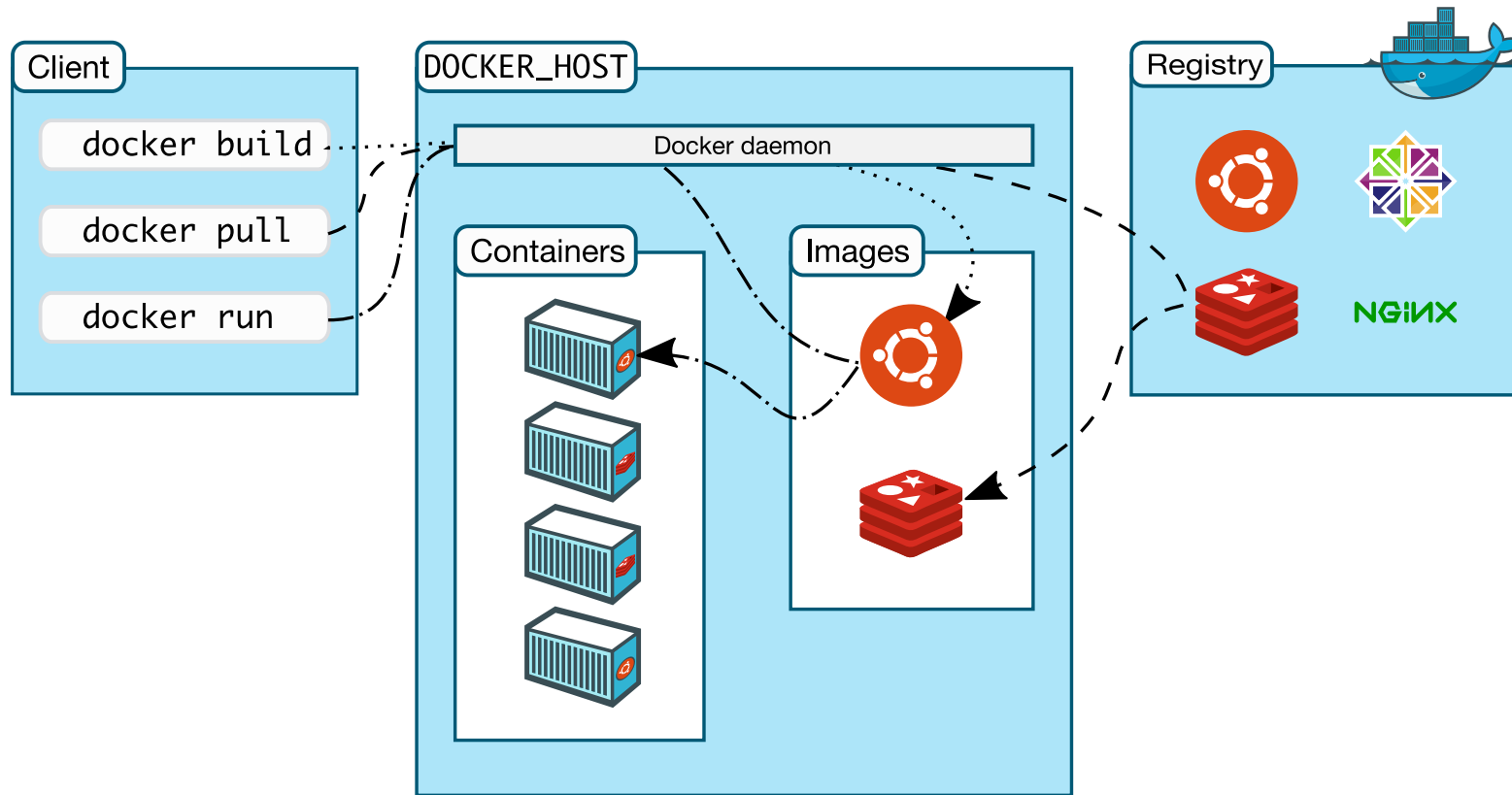


Une machine virtuelle va recréer un serveur complet pour chaque application avec son propre os. Docker va isoler l'application tout en utilisant l'os de son hôte

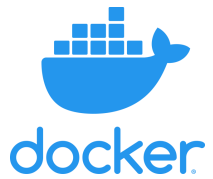




Qu'est ce qu'un docker?



Docker est une technologie de Virtualisation par conteneurs. Il permet d'automatiser le déploiement d'applications et de leurs dépendances dans différents conteneurs isolés.



Commande docker cli

Connaître la version de Docker :

```
docker -- version
```

Lancer un conteneur :

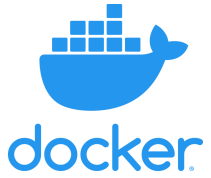
```
docker run -d -p 80:80 docker/getting-started
```

Lister tous les conteneurs :

```
docker ps -a
```

Lister tous les images présentes sur le système :

```
docker images
```



Commande docker cli

Construire une image à partir d'un Dockerfile :

```
docker build
```

Supprimer un conteneur :

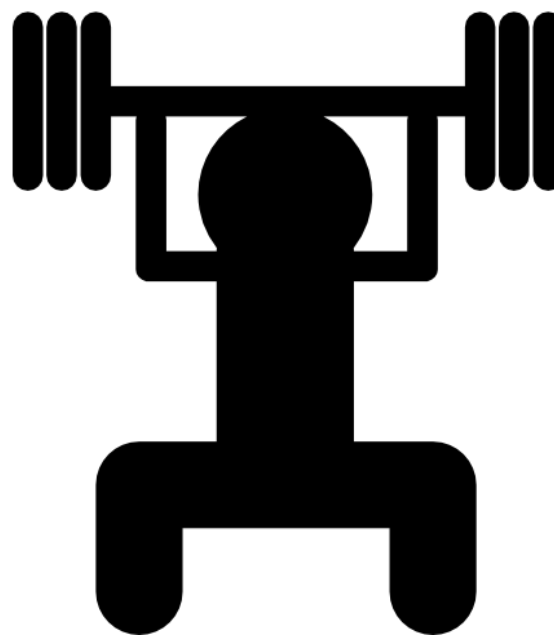
```
docker rm ID_du_conteneur
```

Supprimer une image :

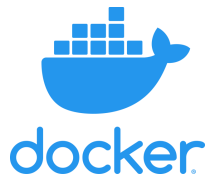
```
docker rmi ID_de_l_image
```

Afficher les logs d'un conteneur avec leurs détails :

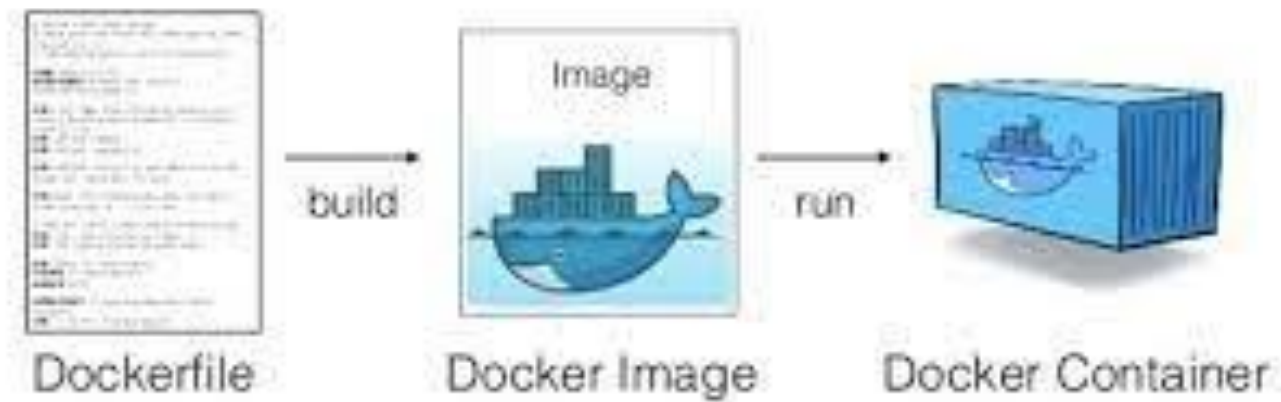
```
docker logs --details
```



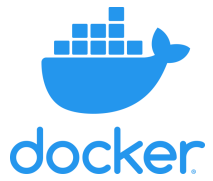
Exercise



Docker image



Une image est un modèle pour les conteneurs. On y retrouve tout le code source de l'application à conteneuriser, ses dépendances, les bibliothèques à utiliser ainsi que d'autres outils nécessaires à la création du conteneur.



Dockerfile

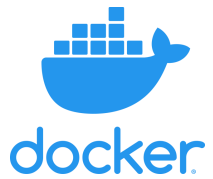
```
# Sample contents of Dockerfile
# Stage 1
FROM microsoft/aspnetcore-build:2.0 AS build-env
WORKDIR /source

# caches restore result by copying csproj file separately
COPY *.csproj .
RUN dotnet restore

# copies the rest of your code
COPY . .
RUN dotnet publish --output /app/ --configuration Release

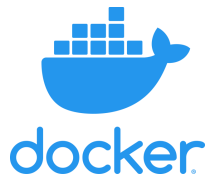
# Stage 2
FROM microsoft/aspnetcore
WORKDIR /app
COPY --from=builder /app .
ENTRYPOINT ["dotnet", "dockertest.dll"]
```

Un dockerfile est un fichier txt qui va décrire les étapes de la création d'une image Docker



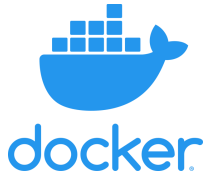
Les instructions Dockerfile

Instruction	Description	Commentaire
FROM	Définir l'image de base	Doit apparaître comme la première instruction ; une seule entrée par stade de build
ENV	Définit les variables d'environnement pour le build process et le container runtime	—
ARG	Déclare les paramètres de ligne de commande pour le build process	Peut apparaître avant l'instruction FROM
WORKDIR	Modifie le répertoire actuel	—
USER	Modifie le statut de membre de l'utilisateur et de groupe	—
COPY	Copie les fichiers et répertoires sur l'image	Crée une nouvelle couche



Les instructions Dockerfile

ADD	Copie les fichiers et répertoires sur l'image	Crée une nouvelle couche ; utilisation déconseillée
RUN	Exécute la commande dans l'image durant le build process	Crée une nouvelle couche
CMD	Définit les arguments par défaut pour le lancement du conteneur	Une seule entrée par stade de build
ENTRYPOINT	Définit la commande par défaut pour le lancement du conteneur	Une seule entrée par stade de build
EXPOSE	Définit les assignations de port pour le conteneur exécuté	Les ports doivent être exposés lors du lancement du conteneur
VOLUME	Inclut le répertoire dans l'image en tant que volume lors du lancement du conteneur dans le système hôte	—



L'instruction CMD et Entrypoint

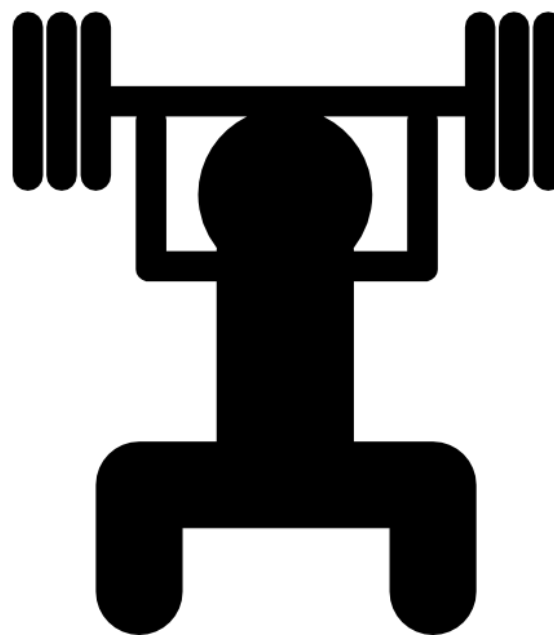
Les instructions CMD et ENTRYPOINT exécutent une commande lorsque le conteneur est lancé. Il existe une différence subtile entre les deux instructions.

```
ENTRYPOINT ["echo", "Hello"]  
CMD ["World"]
```

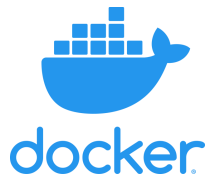
```
# Sortie "Hello World"  
docker run my_image  
# Sortie "Hello Moon"  
docker run my_image Moon
```

- ENTRYPOINT est utilisé pour créer un conteneur qui réalise toujours la même action lorsqu'il est lancé.
- CMD est utilisé pour créer un conteneur qui exécute une action définie au démarrage sans aucun paramètre supplémentaire.

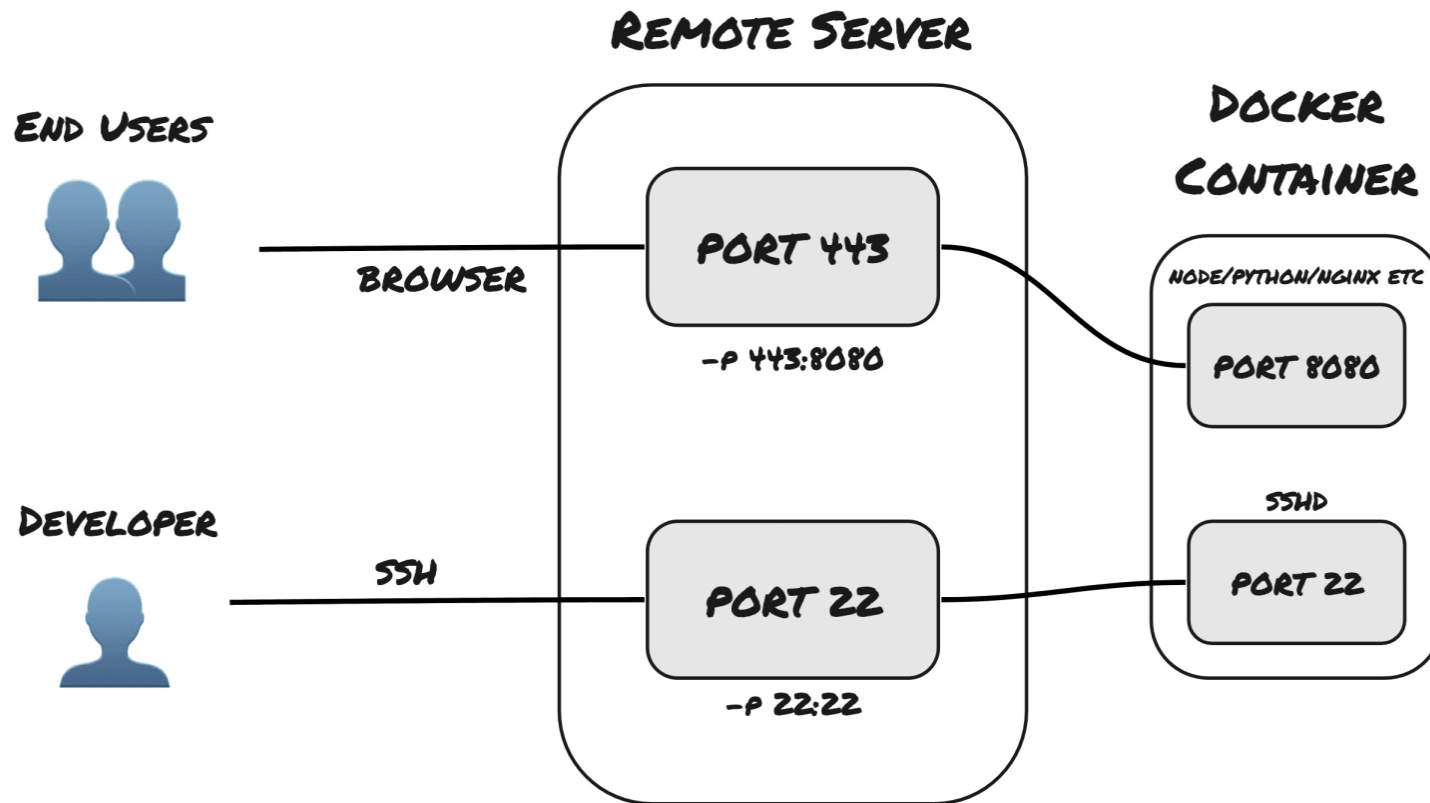
Lorsque vous combinez ces instructions, ENTRYPOINT définit l'action par défaut à réaliser lorsque le conteneur est lancé, tandis que CMD définit les paramètres dont il est facile de se passer pour l'action.



Exercise



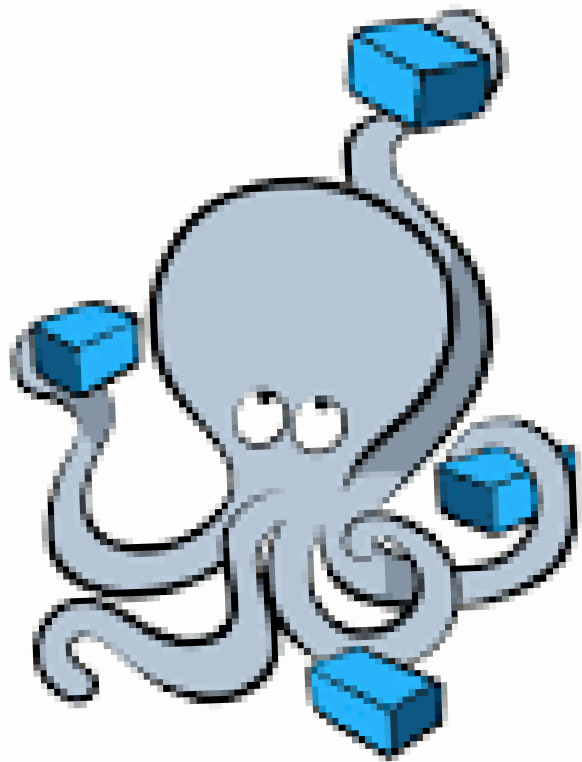
Docker port



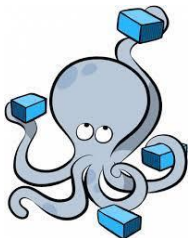
L'option `-p` va me permettre de spécifier le port de ma machine et lui dire vers quel port du conteneur je veux faire la liaison. De cette façon je vais pouvoir aller aux pages web via mon navigateur.

miro

```
docker run -d -p 80:80 docker/getting-started
```

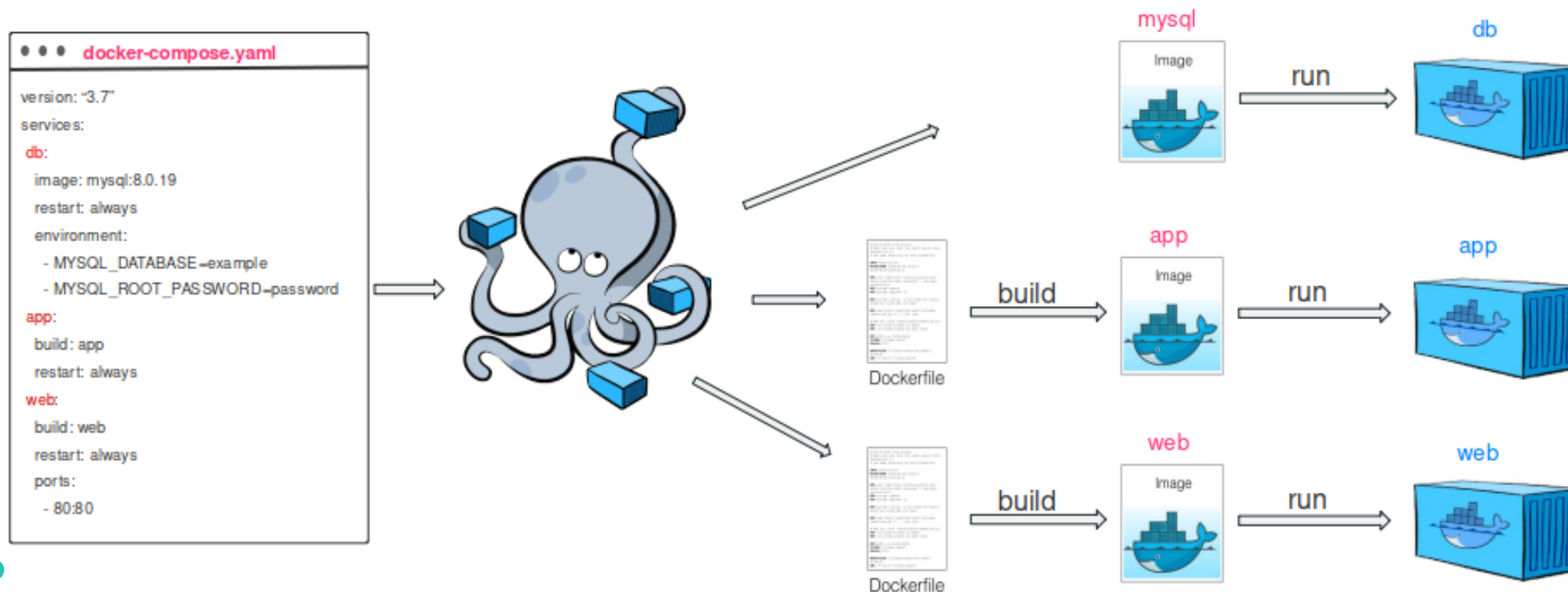


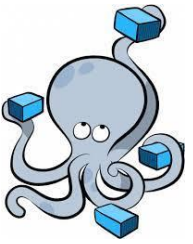
docker
Compose



Docker-compose

Docker-compose fonctionne à partir d'un fichier yaml, dans lequel on définit tous les services que l'on souhaite. Quand on lance la commande d'exécution, le Daemon Docker va lire le docker-compose.yml afin de monter chaque container avec les paramètres que l'on a choisis.





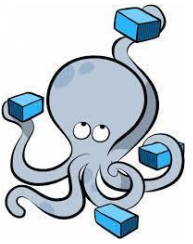
Docker-compose.yml

```
version: '3'
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress

volumes:
  db_data: {}
```

- **image** qui permet de spécifier l'image source pour le conteneur.
- **build** qui permet de spécifier le Dockerfile source pour créer l'image du conteneur.
- **volume** qui permet de spécifier les points de montage entre le système hôte et les conteneurs.
- **restart** qui permet de définir le comportement du conteneur en cas d'arrêt du processus.
- **environment** qui permet de définir les variables d'environnement.
- **depends_on** qui permet de dire que le conteneur dépend d'un autre conteneur.
- **ports** qui permet de définir les ports disponibles entre la machine hôte et le conteneur.



Docker-compose cli

Le cli de Docker Compose et celui de Docker sont très proche. Mais on va voir les principales commandes que vous pourriez avoir besoin.

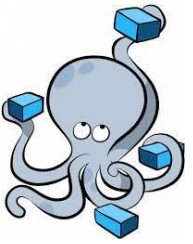
```
docker-compose -up
```

Cette commande va permettre de lancer la création de l'ensemble des conteneurs, comme la commande docker run.

vous pouvez rajouter l'argument `-d` pour faire tourner le conteneurs en tâche de fond

```
docker-compose ps
```

Cette commande permet de voir si l'ensemble des conteneurs sont dans un état fonctionnel et prêts à être utilisés.



Docker-compose cli

```
docker-compose logs
```

Cette commande retourne les logs de vos conteneurs.

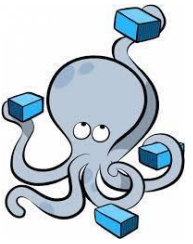
Rajouter les arguments `-f --tail 4` permet de voir l'ensemble des logs sur les différents conteneurs de façon continue tout en limitant l'affichage aux 4 premières lignes

```
docker-compose stop
```

Cette commande permet d'arrêter la stack Docker Composer mais elle ne va pas supprimer les différentes ressource créées par la stack.

```
docker-compose down
```

Cette commande va elle permettre de détruire l'ensemble des ressources créées.



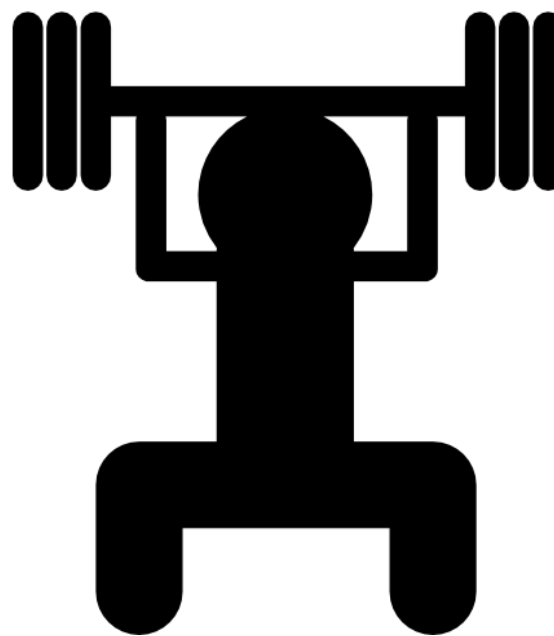
Docker-compose cli

```
docker-compose config
```

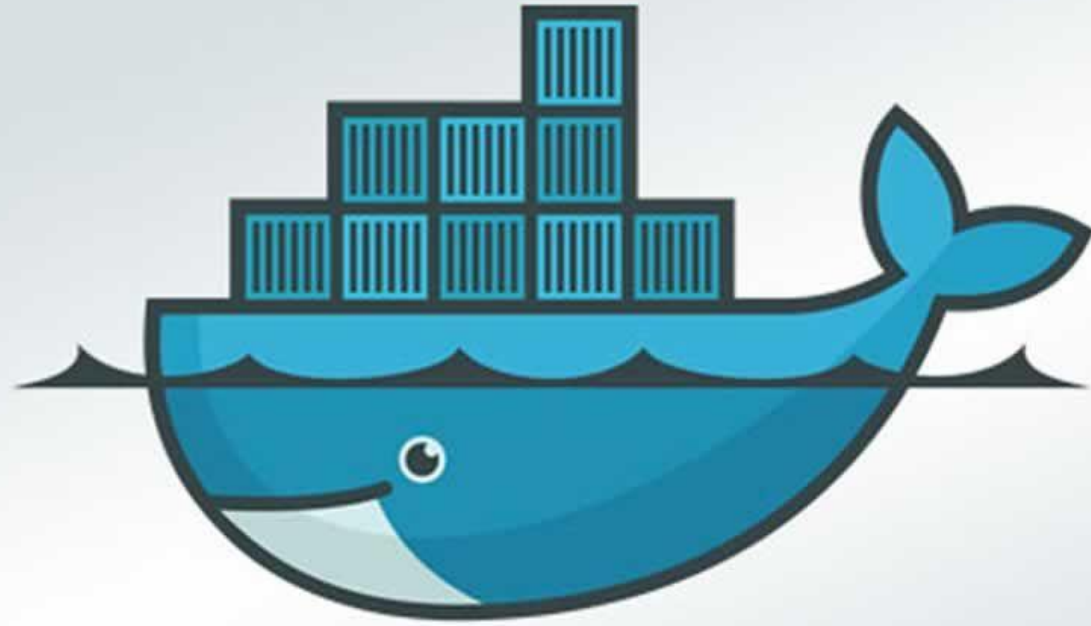
Cette commande va permettre de valider la syntaxe de votre fichier, et ainsi d'être certain de son bon fonctionnement.

Exemple :

```
→ docker-compose config  
ERROR: The Compose file './docker-compose.yml' is invalid because:  
Unsupported config option for services.db: 'images'
```



Exercise



docker HUB



Docker hub

Grace à docker hub on peut télécharger ou publier nos images de nos conteneur depuis un repository. Il y a deux type de repository privé et public :

- Les repository public sont accessibles à tous
- Les repository privé est accessible au propriétaire et ces associé

```
docker login
```

Va vous permettre de vous connecter au hub

```
docker tag Source_Image Target_Image
```

Va créer un tag avec l'image indiquer que vous pourrez ensuite push
exemple:

```
docker tag a5ed81cbcd28 lordmams/reptest:ubuntu
```



Docker hub

Une fois le tag créer vous pourrez le push sur le hub.

```
docker push NOM_REPO:TagName
```

```
docker push lordmams/repotest:ubuntu
```

TAG

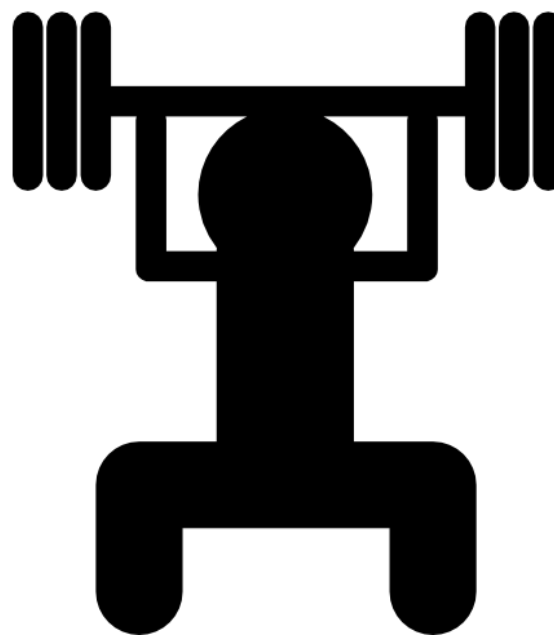
[ubuntu](#)

Last pushed 12 minutes ago by [lordmams](#)

```
docker pull lordmams/repotes...
```

DIGEST	OS/ARCH	LAST PULL	COMPRESSED SIZE ⓘ
f657815e324e	linux/amd64	---	80.83 MB

Et le récupérer avec un docker pull.



Exercise





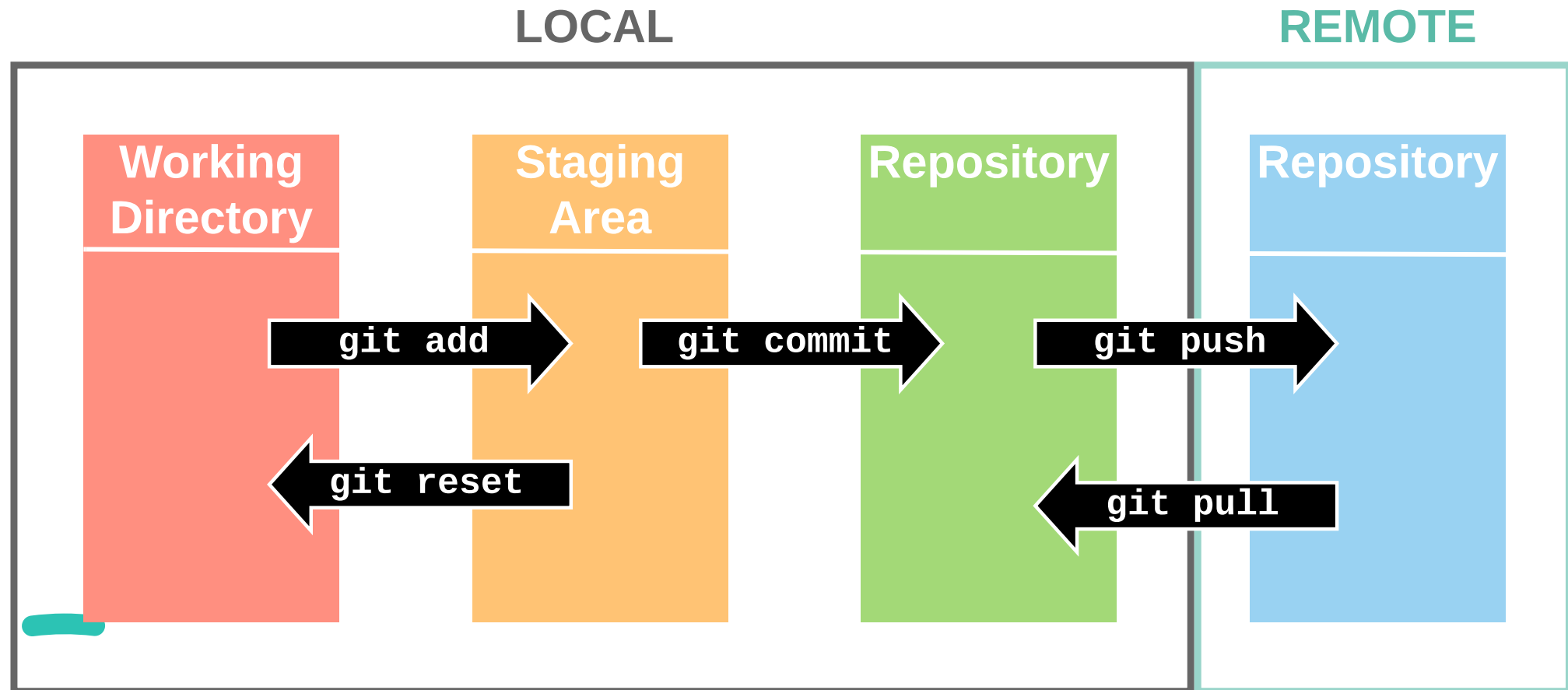
Qu'est-ce que GIT ?



C'est le système de contrôle de version le plus largement utilisé aujourd'hui . C'est un projet open source qui à été développé en 2005 par Linus Torvalds. Il part du principe de plutôt que de consacrer un seul emplacement pour l'historique complet des version du logiciel comme c'était souvent le cas dans les système de contrôle de version ayant fait leur temps comme CVS, dans Git, chaque copie de travail du code est également un dépôt qui contient l'historique complet de tous les changements.



Les commandes git de base





Les commandes git de base

La commandes **git init** :

Permet de créer un nouveau dépôt GIT

La commandes **git statut** :

Permet d'afficher la liste des fichier modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés.

La commandes **git checkout** :

Permet de soit créer des branches ou pour basculer entre elles.

```
command git checkout -b <nom-branche>
```

```
git checkout <nom-branche>
```



Les commandes git de base

La commandes **git init** :

Permet de créer un nouveau dépôt GIT

La commandes **git statut** :

Permet d'afficher la liste des fichier modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés.

La commandes **git checkout** :

Permet de soit créer des branches ou pour basculer entre elles.

```
command git checkout -b <nom-branche>
```

```
git checkout <nom-branche>
```



Les commandes git de base

La commandes `git diff` :
Permet de lister les conflits.

```
git diff --base <nom-fichier>
```

```
git diff <branche-source> <branche-cible>
```

```
git diff
```

La commandes `git stash` :
Permet d'enregistrer les changements qui ne doivent pas être commit immédiatement.
C'est un commit temporaire

```
git stash
```



Les commandes git de base

La commandes **git archive** :

Permet à un utilisateur de créer un fichier zip ou tar contenant les composant d'un arbre du dépôt.

```
git archive --format=tar master
```

La commandes **git prune** :

Permet de supprimés les fichiers qui n'ont pas de pointeurs entrants

La commandes **git rebase** :

Permet la ré application des commits sur une autre branche

```
git rebase master
```



Les commandes git de base

La commandes **git archive** :

Permet à un utilisateur de créer un fichier zip ou tar contenant les composant d'un arbre du dépôt.

```
git archive --format=tar master
```

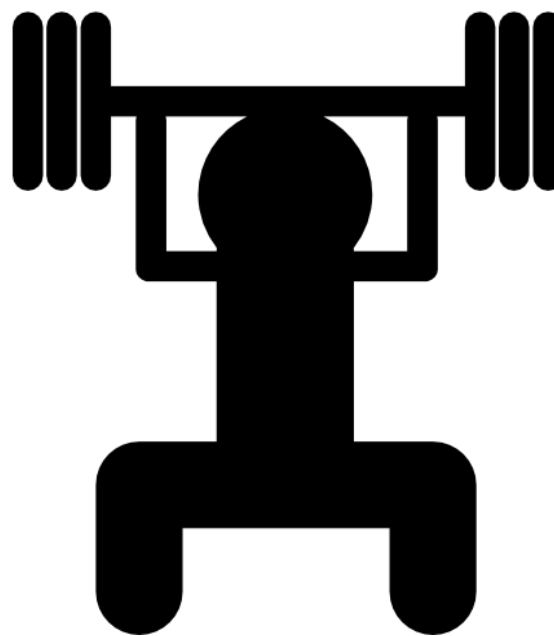
La commandes **git prune** :

Permet de supprimés les fichiers qui n'ont pas de pointeurs entrants

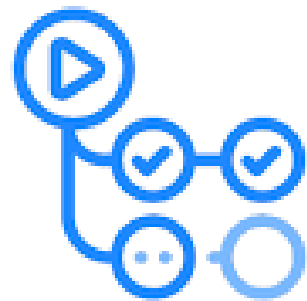
La commandes **git rebase** :

Permet la ré application des commit sur une autre branche

```
git rebase master
```



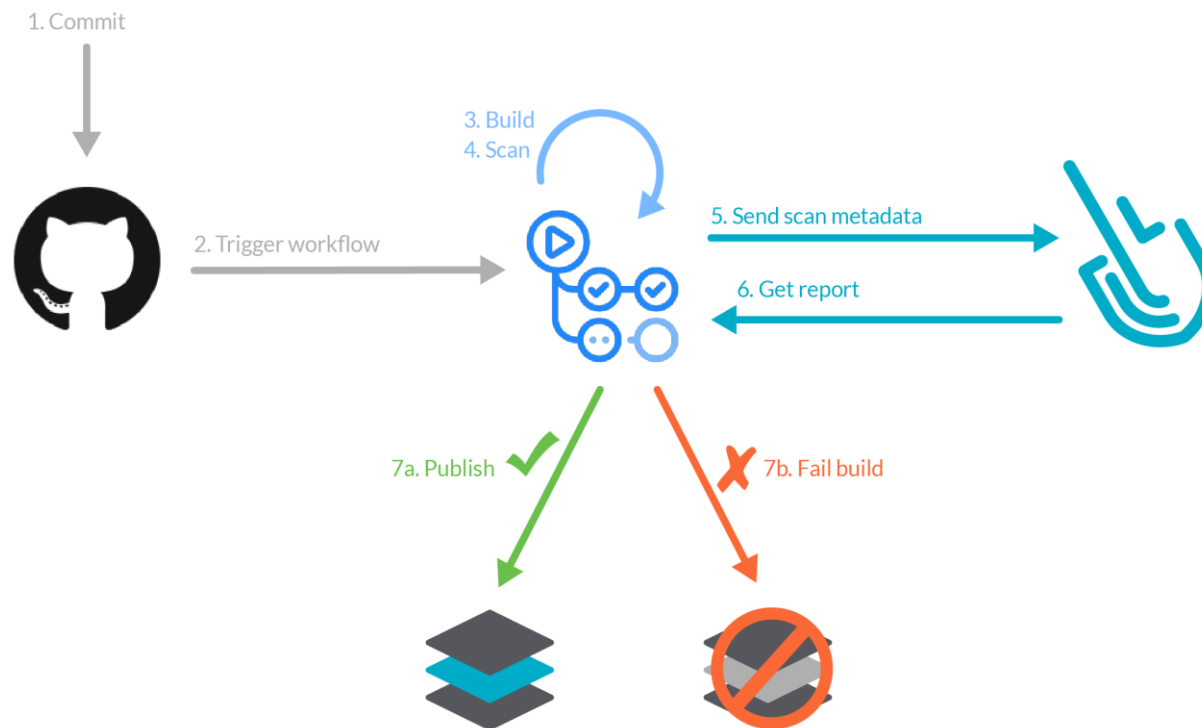
Exercise



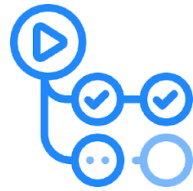
GitHub Actions



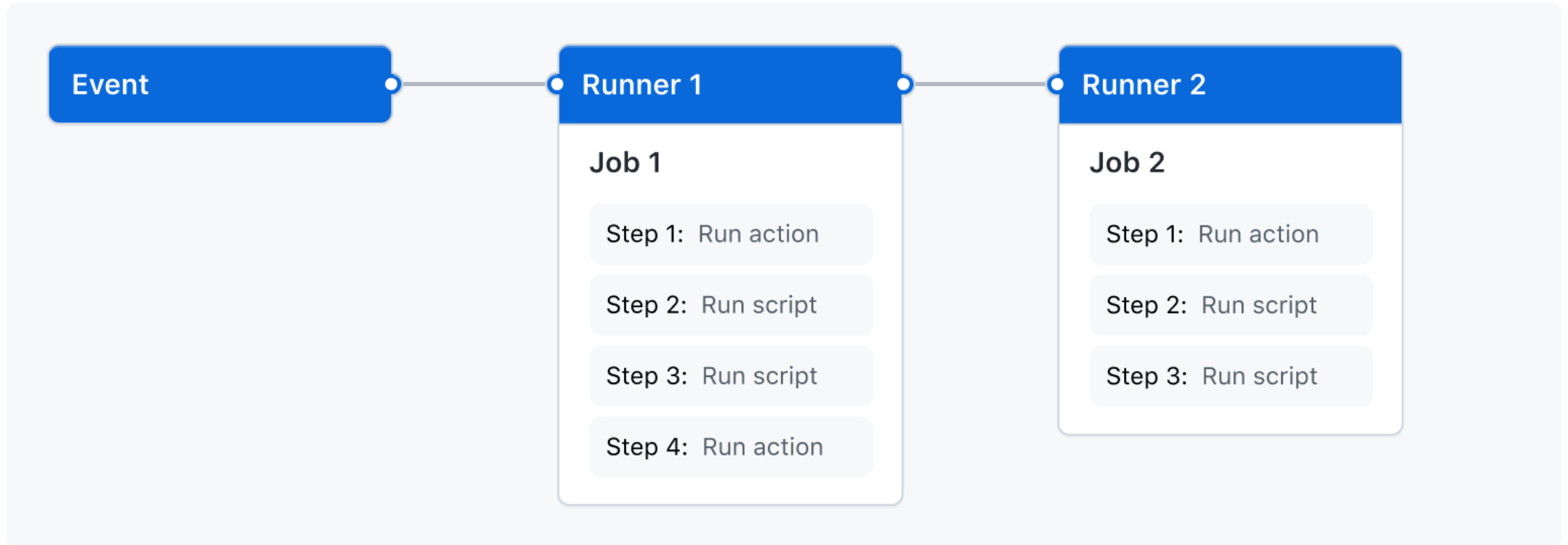
Github action



GitHub Actions est une palte-forme d'intégrations continues (CI/CD) qui permet d'automatisé notre pipeline de construction, de test et de déploiement. Il va permettre d'exécuter des flux de travail lorsque d'autres événements se produisent notre référentiel.



Github action



On peut configurer le workflow pour qu'il se lance lorsqu'un événement se déclenche dans notre repository comme les pull request. Le workflow contient un ou plusieurs jobs qui peuvent être lancés l'un à la suite de l'autre ou en parallèle. Chaque job va être dans son propre Runner ou à l'intérieur d'un conteneur et il pourra définir un ou plusieurs étapes pour soit lancer un script que l'on a défini ou déclencher une action