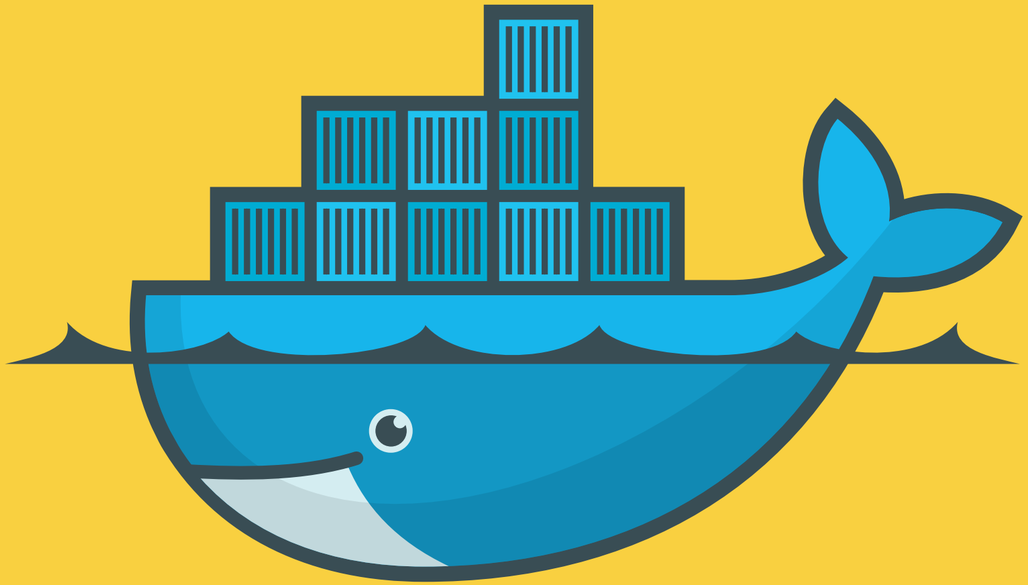
SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat



DOCKER

Plateforme de conteneurs

SOLSTICE
Coopérative d'entrepreneurs

Sébastien Grivolat

Cahier Docker

points à retenir

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

Bases de docker

Pourquoi docker :

- besoin d'uniformisation dans une équipe,
- besoin d'avoir un env stable entre dev/test/recette/prod,
- monter une stack technique complexe rapidement,
- ne pas dépendre des versions locales de certaines lib,
- scalabilité, isolation, "sécurité".

Différences avec une VM :

- plus léger,
- s'appuie sur l'OS hôte,
- mais dépend des binaires compilés pour l'archi hôte
  (instructions CPU natives : pas VM = pas d'émulation).

Limitations :

- peu adapté aux IHM,
- pas de multi OS sur un seul hôte,
- duplication de lib avec l'hôte.

install avec ubuntu server

- sudo apt-get install docker
- sudo usermod -aG docker ${USER}
- su - ${USER}
- groups
  - => il doit y avoir docker
- docker ps
  - => pas d'erreur

install avec un autre linux

Sur une autre distrib, si pas de dépot officiel ou pour mettre la version docker-ce :

- sudo apt install apt-transport-https ca-certificates curl software-properties-common
- curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
- echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
- sudo apt update
- apt-cache policy docker-ce
- sudo apt install docker-ce
- sudo systemctl status docker

sinon => RTFM

docker commandes courantes

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **run NOM_IMAGE**
  - télécharge si l'image n'est pas en local et démarre le conteneur : docker run hello-world
- **pull NOM_IMAGE**
  - télécharge l'image depuis le dépôt : docker pull ubuntu
- **ps**
  - statuts des conteneurs, par défaut les actifs : docker ps ou actifs et arrêtés : docker ps -a
- **stop NAME/ID**
  - arrêter un conteneur : docker stop gitea
  - faire Ctrl-C si celui-ci n'est pas en tache de fond
- **start NAME/ID**
  - démarrer un conteneur : docker start gitea
- **logs NAME/ID**
  - traces d'un conteneur : docker logs gitea
- **rm NAME/ID**
  - supprimer un conteneur : docker rm gitea
- **exec OPTIONS NAME/ID CMD**
  - pour un conteneur actif, lancer une commande dans ce conteneur : docker exec gitea sh

docker run options standards

- quand ':' est utilisé à **gauche** se sont les **valeurs de l'hôte** et à **droite** du **conteneur**.
- **-i ou --interactive**
  - pour avoir shell, lancer une commande ou pour des images spécifiques :
    docker run -i ubuntu sh
- **-v ou --volume**
  - mapper le système de fichier de l'hôte dans le conteneur, soit un répertoire (par défaut) soit un fichier spécifique : docker run -it -v /home/${USER}:/home/ubuntu ubuntu
- **--network**
  - spécifier le réseau virtuel pour le conteneur : docker run --network host ubuntu
  - l'objectif est de partager un réseau entre plusieurs conteneurs
- **-p ou --publish**
  - une image peut exposer par défaut les ports du conteneur, si ce n'est pas le cas ou que l'on souhaite un port spécifique : docker run -p 8022:22 ubuntu
  - -P ou --publish-all ouvre tous les ports du conteneur sur des ports de hôtes aléatoires
  - **Attention si il y a une VM** qui est l'hôte docker il faut aussi mapper le port exposé par docker en dehors de la VM pour y accéder de l'extérieur (port pour ssh, pour un navigateur web).

docker gestion des "objets"

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **docker image** : gérer les images docker, options :
  - **pull IMAGE** télécharger une image : docker image pull gitea/gitea
  - **rm ID/NAME** supprimer une image : docker image rm gitea
  - **prune** supprimer les images (par défaut non utilisées) : docker image prune
  - **ls** lister les images : docker image ls
- **docker network** : gérer les réseaux docker, options :
  - **ls** lister les réseaux : docker network ls
  - **rm ID/NAME** supprimer un réseau : docker network rm host
- **docker container** : gérer les conteneurs docker, options :
  - **ls** lister les conteneurs actifs : docker container ls
  - **ls -a** lister tous les conteneurs : docker container ls -a
  - **rm ID/NAME** supprimer un conteneur : docker container rm gitea
- **docker system** : gérer docker, options :
  - **df** voir l'occupation du disque dur : docker system df
  - **prune** supprimer les objets non utilisés : docker system prune
  - **prune -a** supprimer plus d'objets non utilisés : docker system prune -a
  - **df --help** avoir de l'aide sur df : docker system df --help

- **Tip** :
  - arrêter tous les conteneurs :
  - docker stop $(docker container ls -q)
  - forcer l'arrêt de tous les conteneurs :
  - docker kill $(docker container ls -q)

docker registry

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **Registry** :
  - c'est le github des images docker, évite l'étape de build de l'image depuis le Dockerfile
  - mettre à disposition de l'équipe ou de tous une image, ou travailler sur plusieurs environnements
  - versionner les images
- **Privée ou publique :**
  - privée en auto-herbergée
  - privée chez un fournisseur
  - publique sur le hub via la création d'un compte
- **Usage** :
  - **docker login URL_REGISTRY** : disposer de credentials docker login localhost:5000
  - **docker tag ID URL_REGISTRY/TAG** : formaliser avec l'équipe le nommage des tags
    docker tag 46331d942d63 localhost:5000/ma_version
  - **docker push URL_REGISTRY/TAG** : mettre à disposition une image
    docker push localhost:5000/ma_version
  - **docker pull URL_REGISTRY/TAG** : télécharger une image
  - docker pull localhost:5000/ma_version
  - **docker images** : vérifier que l'on a bien l'image pullée

Dockerfile

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **Dockerfile** : docker build -t test:v0.1 .
  - fichier de config, pour créer une image via une phase de 'build', '#' pour commenter
  - les mots clés peuvent se répéter plusieurs fois et s'appliquent potentiellement aux lignes suivantes
- **FROM :**
  - identifier l'image source : FROM alpine
- **CMD / ENTRYPOINT (avec/sans shell) :**
  - commande de lancement du conteneur
  - CMD echo "Bonjour !"
  - ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
- **WORKDIR :**
  - changer le répertoire courant
  - WORKDIR /app
- **COPY :**
  - copier des fichiers
  - COPY go.mod .
  - COPY ["go.mod", "go.sum", "./"]
- **préférer la syntaxe tableau quand elle est possible**
- **RUN :**
  - exécuter une commande
  - limiter le nombre de RUN, utiliser '&&' ou '|'
  - RUN go mod download
  - RUN curl http://sou.rce/file.tgz | tar -xjC /tmp/ file.tgz && make -C /tmp/app
- **EXPOSE :**
  - ouvre un port dans le conteneur
  - EXPOSE 8080
- **ENV :**
  - déclaration de variables
  - ENV HTTP_PORT=8081
- **ADD, VOLUME, USER, ...**
  - voir le manuel utilisateur

quelques bonnes pratiques

Dockerfile :

- avoir un fichier propre et optimisé
- limiter le nombre de layers et notamment les lignes 'RUN'
- nettoyer à chaque étape (fichiers générés, lib de compil, variables temporaires)
- utiliser des images optimisées et optimales pour le projet cible
- utiliser le 'multistages' lorsque c'est pertinent (comme : image de compil et image de run)
- séparer dans des conteneurs différents les applicatifs (par exemple pas de front et bdd dans le même conteneur)
- faire du 'multilayers' si le projet le nécessite
- vérifier la taille finale de l'image

quelques bonnes pratiques

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

Usage de docker :

- plusieurs commandes possibles pour une action, trouver votre fonctionnement
- utiliser des 'volumes' pour persister les données, si besoin les mapper sur le filesystem hôte
- préférer des chemins relatifs (plusieurs versions possibles et transfert facile de poste hôte)
- pour la prod ne pas utiliser d'image 'latest' mais spécifier un tag (pour la reproductibilité)
- nommer efficacement les conteneurs
- utiliser des variables lorsque c'est pertinent
- utiliser un '.env' externe pour paramétrer ce qui n'est pas "docker" (comme peupler la bdd avec des tests)
- nettoyer régulièrement ses anciens objets docker
- lors du changement de source pour une image similaire, nettoyer pour éviter les conflits
- régler l'horloge de ses conteneurs (NTP)
- faire des fichiers docker (Dockerfile, docker-compose.yml, scripts, ...) clairs, lisibles, optimisés et si nécessaire commentés (via relecture, veille, ...)
- tester avant publication/diffusion

quelques bonnes pratiques

Sécurité :

- choisir un user et son groupe (pas 'root' dans le conteneur)
- limiter les communications entre conteneurs (par exemple différents networks : bdd-api api-front)
- n'exposer que des ports et des volumes nécessaires
- Dockerfile : supprimer les infos sensibles dans le layer qui les crée
- vérifier la source de ses images (nombre de téléchargements, nombre de versions, correctifs, qualité du README, ...)
- utiliser 'docker secrets' pour stocker les données sensibles
- signer ses images de prod avec 'docker trust'
- être attentif à la sécurité, faire de la veille
- appliquer les bonnes pratiques classiques (mises à jour, protéger aussi l'OS hôte, avoir des moyens adaptés aux données, ...)

docker-compose

Surcouche à docker :

- multi-conteneurs
- centralisation des configurations
- environnement et paramètres dans un fichier de config
- commandes quasi identiques à celles "docker"

docker-compose.yml :

- syntaxe simplifiée… ou pas : yaml
- nom par défaut du fichier de config

install avec linux

- s'appuie sur une installation docker fonctionnelle
- sudo apt-get install docker-compose

docker-compose commandes courantes

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **up**
  - télécharge l'image en local et/ou démarre le conteneur
  - docker-compose up --remove-orphans
- **ps**
  - statuts des conteneurs correspondants au .yml courant uniquement
  - docker-compose ps
- **logs**
  - traces des conteneurs
  - docker-compose logs -f --tail 5
- **down**
  - arrêter les conteneurs et supprimer les objets
  - docker-compose down
- **exec**
  - lance un commande dans le conteneur actif
  - docker-compose exec NAME sh
- **restart**
  - redémarre un ou tous les conteneurs
- **build**
  - lance la compilation des conteneurs
  - docker-compose build
- **pull**
  - télécharge les images depuis le dépôt
  - docker-compose pull
- **stop**
  - arrêter les conteneurs
  - docker-compose stop
- **config**
  - tester la syntaxe du docker-compose.yml
  - docker-compose config
- **- options -**
  - globalement identiques à celles des commandes docker équivalentes
  - -a -d notamment, voir la doc

docker-compose.yml syntaxe

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- https://docs.docker.com/compose/compose-file/
- **services** :
  - puis le **nom** du conteneur comme clé, par exemple "**web**" :
  - **image**: XXX/YYY:version : l'image du conteneur
  - **environment**: les paramètres fonctionnels
    - - XXX=YYY
    - - ZZZ:0
  - **volumes**: les mapping des fichiers de l'hôte (fichier ou répertoire)
    - - path_hote:path_conteneur
    - - path_hote:...
  - **ports**: les mapping des ports de l'hôte (à gche) et du conteneur (à dte)
    - - p_hote:p_conteneur
    - - p_hote:...
  - **links**: dépendances/renommages d'autres conteneurs du le même yml
    - - name_ext:name_int
  - **build:** répertoire et options pour créer l'image (RTFM)
  - **command**: la commande de remplacement de celle par défaut dans l'image (RTFM)

```
version: "3"

networks:
  gitea:

services:
  web:
    image: gitea/gitea:1.17.1
    environment:
      - USER_UID=1000
    restart: always
    networks:
      - gitea
    volumes:
      - ./gitea:/data
    ports:
      - "3000:3000"
```

docker-compose.yml syntaxe

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- https://docs.docker.com/compose/compose-file/
- **version**:
  - numéro informatif de la version de la syntaxe utilisée, pas d'impact sur l'exécution
- **networks**:
  - RTMF https://docs.docker.com/compose/compose-file/#networks-top-level-element
- **volumes**:
  - RTMF https://docs.docker.com/compose/compose-file/#volumes-top-level-element
- **restart** (under **services**: & **container**:) defines the policy that the platform will apply on container termination.
  - **no**:
    - The default restart policy. Does not restart a container under any circumstances.
  - **always**:
    - The policy always restarts the container until its removal.
  - **on-failure**:
    - The policy restarts a container if the exit code indicates an error.
  - **unless-stopped**:
    - The policy restarts a container irrespective of the exit code but will stop restarting when the service is stopped or removed.

tips linux

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **Ctrl-R**
  - recherche de commande en arrière dans l'historique
- **history**
  - afficher toutes les commandes déjà exécutées
- **wget localhost:XXX / curl localhost:XXX**
  - pour vérifier si mon serveur web est actif, fonctionne avec une url
- **cat > MON_FICHIER.TXT**
  - écrire un fichier, fin avec Ctrl-D
- **nano MON_FICHIER.TXT ou vi MON_FICHIER.TXT**
  - éditer rapidement mon fichier sans éditeur type IDE disponible
- **ssh -p XX user@machine**
  - se connecter à distance ou à une VM depuis l'espace de travail habituel (yes pour accepter le certificat)
  - permet aussi de partager des fichiers depuis une machine distante (par exemple avec filezilla ou vs codium)
- **pwd**
  - connaître le répertoire courant
- **sudo netstat -tulpa**
  - voir les connexions réseau et les ports ouverts
- **lsof -p 42**
  - voir les fichiers ouverts par le processus ID 42
- **df :**
  - place disponible sur les partitions
  - df -h
- **du :**
  - usage disque pour une ressource
  - du -h
- **sort :**
  - trier l'entrée
  - du -h | sort -h
- **wc :**
  - compter des mots
  - ls | wc -l

tips linux

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

- **grep XXX ou grep -v ZZZ**
  - chercher XXX
  - chercher tout sauf ZZZ
  - sudo netstat -tulpa | grep http | grep -v firefox
- **docker run --help**
  - avoir l'aide en ligne sur une commande
- **sh ou bash** dans un conteneur (via exec) :
  - **ping** : vers les autres conteneurs, avec leur nom
  - **ip a** : adresse du conteneur
  - **ps aux** : lister les processus en cours dans le conteneur
  - **netstat -tulpa** : ports exposés par le conteneur
  - **/var/log ou ailleurs** : accéder à des logs internes non exposés (par volume)
  - **changements internes** de conf ou de contenu : attention aux volumes en read-only, les modifs sont perdues au redémarrage du conteneur, donc pour tester en live uniquement
- **sudo dmesg** :
  - affiche les traces de la machine linux
- **reboot**
  - redémarre la machine
  - sudo reboot
  - (dans le doute…)
- **shutdown**
  - arrête la machine
  - à éviter à distance sans accès physique
  - sudo shutdown -h now
- **rmdir FOLDER**
  - supprimer un répertoire vide
- **rm -rf FOLDER**
  - supprimer sans confirmation un répertoire et tout son contenu
  - ATTENTION IRRÉVERSIBLE RISQUE DE PERTE DE DONNÉES

Exemples & TP

Jour 1

- **tp #1 : démarrage**
  - install et commandes de base, hello-world, nginx (commandes déjà vues)

- **tp #2.1 nginx + volume**
  - docker run -d --name=nginx -p 80:80 -v $(pwd)/ngConfig:/config linuxserver/nginx
  - edition du fichier ngConfig/www/index.html repris en live par le conteneur

- **tp #2.2 nitter : afficher des tweets + network**
  - docker run -v $(pwd)/nitter-redis:/data -d --network host redis:6-alpine redis-server --save 60 1 --loglevel warning
  - créer le fichier nitter.conf dans le répertoire courant d'après le git (copie) https://github.com/zedeus/nitter/blob/master/nitter.example.conf
  - docker run -v $(pwd)/nitter.example.conf:/src/nitter.conf -d --network host zedeus/nitter:latest

- **tp #3 gitea : faire du git sur un serveur web + rendu du TP**
  - docker run -d -e MYSQL_ROOT_PASSWORD=secret -e MYSQL_DATABASE=gitea -p 3306:3306 mysql
  - docker run -d -p 3000:3000 gitea/gitea
  - mettre l'ip du conteneur de bdd dans la conf bdd de gitea (a priori 172.17.0.1)

SOLSTICE
Coopérative d'entrepreneurs
Sébastien Grivolat

Exemples & TP *voir les .yml

- **tp #1.1 : objets**
  - commandes : image, container, network et system (cf la page dédiée)
- **tp #1.2 docker-compose**
  - les commandes principales (cf la page dédiée)
- **tp #2 exemples avec compose**
  - gitea/gitea : https://docs.gitea.io/en-us/install-with-docker
  - nitter : https://github.com/zedeus/nitter
  - redis : https://hub.docker.com/_/redis
  - bookstack : https://docs.linuxserver.io/images/docker-bookstack
  - plik : https://github.com/root-gg/plik
  - wordpress : https://hub.docker.com/_/wordpress
  - mysql : https://hub.docker.com/_/mysql
  - phpmyadmin : https://hub.docker.com/_/phpmyadmin
  - mariadb : https://hub.docker.com/_/mariadb
  - postgres : https://hub.docker.com/_/postgres
  - pgadmin : https://hub.docker.com/r/dpage/pgadmin4
  - prestashop : https://hub.docker.com/r/prestashop/prestashop
  - heimdall : https://docs.linuxserver.io/images/docker-heimdall
  - firefox : https://docs.linuxserver.io/images/docker-firefox
  - vscode : https://github.com/INsReady/docker-vscode-php

Exemples & TP *voir les .yml

Jour 3

- **tp #1 : registry**
  - commandes : login, tag, push (cf la page dédiée)
  - registry : https://hub.docker.com/_/registry

- **tp #2 exemples avec compose**
  - yakforms : https://hub.docker.com/r/nouts/yakforms
  - tuleap : tuleap/tuleap-community-edition - Docker Image | Docker Hub
  - portainer : portainer/portainer-ce - Docker Image | Docker Hub
  - mattermost : https://hub.docker.com/r/mattermost/mattermost-team-edition
  - nextcloud : https://hub.docker.com/_/nextcloud
  - collabora : https://hub.docker.com/r/collabora/code
  - ldap : https://hub.docker.com/r/linuxserver/ldap-auth
  - ldap_gui : https://hub.docker.com/r/wheelybird/ldap-user-manager

Exemples & TP *voir les .yml

Jour 4

- **tp #1 : Dockerfile**
  - syntaxe (cf la page dédiée)
  - git clone https://github.com/olliefr/docker-gs-ping
  - docker build -t test-go .
- **tp #2 exemple avec votre code**
  - récupération de vos sources
  - copie dans l'image
  - ligne de commande de compilation/build/téléchargement des dépendances
  - ajout d'un volume pour les logs ou accéder aux sources
  - vérification du fonctionnement et de la possibilité de modifier le source
- t**p #3 build & repository**
  - pusher l'image que l'on vient de créer
  - la récupérer d'un autre poste
- **tp #4 docker-compose & build**
  - python : https://hub.docker.com/_/python
  - apache : https://hub.docker.com/_/httpd
  - tomcat : https://hub.docker.com/_/tomcat
  - jmeter : https://hub.docker.com/r/justb4/jmeter
  - apache spark : https://hub.docker.com/r/apache/spark
  - rudderlabs server : https://github.com/rudderlabs/rudder-server
  - mongodb : https://hub.docker.com/_/mongo

Merci