A la découverte des Docker Dev Environments

Date : 21 avril 2022
Speakers : Djordje Lukic (Docker) et Guillaume Lours (Docker)
Format : conférence (45mn)

2 speakers dont un à distance en webcam. Tous les 2 travaillent chez Docker depuis 3 ans.

Docker essaie d'améliorer la boucle de feedback interne de développement, notamment l'onboarding de nouveaux développeurs. Ils notent la difficulté à travailler sur plusieurs projets Git en même temps.

- Git madness
    - Le changement de branches pour les revues perturbe l'IDE (ex : chgt de dépendances Maven) : du coup on fait des review dans le navigateur
- IDE > code
    - Faire tourner le code à l'intérieur d'un conteneur permet d'y intégrer tout le contexte (ex : variables d'environnement). Docker Dev permet de prendre un instantané et de le partager avec les autres dévs
- Consitency
    - Démarrage rapide d'un environnement de développement afin de faciliter l'onboarding d'un nouvel arrivant.

Docker Dev Environment se rapproche de GitPod ou Codespaces. A la différence de ces dernières qui sont des solutions purement Cloud, Docker Dev Env s'installe en local et permet d'être utilisé sur son poste de dév.
Intérêts :

- Local First
- Agnostique de l'IDE et du serveur Git
- Extends beyond code
- Auto-magic
- Share with teams in one click
- Construit par dessus Docker Compose très apprécié des dévs

La démo commence avec Docker Desktop installé.
Le menu « *Dev Environments* » est en preview depuis plusieurs mois.
Guillaume commence par créer un sample. Il choisit le repo Git associé : dockersamples/single-dev-env de GitHub
Docker Desktop clone le repo Git dans le volume d'un container dédié, détecte le langage (Java, Go) pour déterminer l'image prédéfinie (En Java : JDK + Maven).
Ouverture du projet sample en GO dans VS Code.

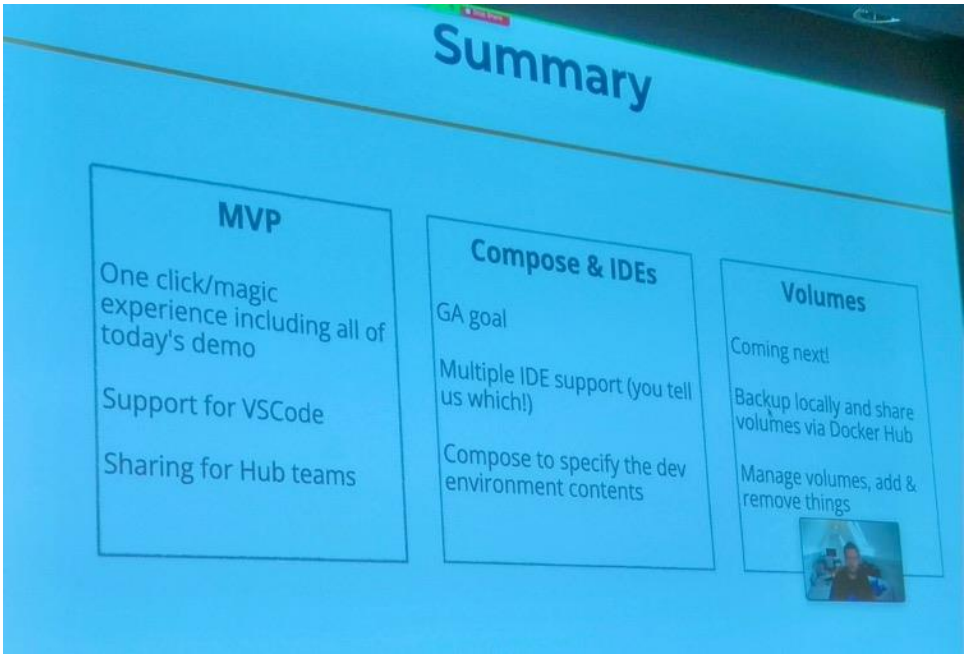
Guillaume crée une branche *devoxx*.
La config locale du poste de dév est montée dans le conteneur (exemple des git config)
Les dévs containers de Microsoft permettent de brancher l'IDE sur le serveur VSCode.

Guillaume partage son projet avec Djordje via Docker Hub. Le code source du volume est récupéré. Une nouvelle image est créée avec le code source, les dépendances, la config de l'IDE.
Dans Docker Desktop, on voit 2 volumes : serveur VS Code (non partageable car propriétaire) + volume du code source.

Djordje colle le nom de l'image et du tag Docker dans Docker Desktop.
Guillaume est sur Macbook M1 sur ARM64 et non Djordje. Docker Desktop utilise alors qemu-aarch64 pour émuler cette architecture (plus lent).

Djordje précise qu'il est possible de configurer l'image Docker (Dockerfile ou chemin image) à utiliser pour le dév dans le fichier .docker/config.json
Les **credentials Git** ne sont bien évidemment pas partagés.

Démonstration du second sample illustrant l'utilisation de Docker Compose : proxy nginx, backend et base de données.
Docker Desktop effectue un compose up pour construire toutes les images, lance tous les conteneurs. L'application web est accessible dans le navigateur.
Pour travailler sur le backend, le conteneur du backend est remplacé par un conteneur de développement. Docker Desktop peut rebrancher le conteneur de base par le conteneur de base dans Compose.
Le fichier compose est accessible dans .docke/docker-compose.yaml
C'est un fichier Compose des plus classiques.
Le Dockerfile est utilisé à la fois pour le dév et la prod .



Encore en preview, cette fonctionnalité n'est disponible qu'aux détenteurs de licences Docker Desktop. Cela va changer prochainement. (ndlr : j'y ai accès sans licence sur mon Macbook perso).
Se référer à la roadmap du repo Git afin de savoir quels sont les prochains IDE qui seront supportés (IntelliJ ?), pousser ses attentes et voter.

L’équipe Docker a de nombreuses idées :

- Une évolution de la spécs compose file pourrait permettre d’ajouter une section spéciale au dév
- Support de la génération vers Codespaces ou GitPod et leur import
- Partage de volume pour, par exemple, partager la base de données entre le dév et la CI

Anecdote : au cours de dév de Docker Dev Environment, les dévs Docker avaient créé beaucoup de volumes : du coup ils ont ajouté une UI pour la gestion des volumes.

Questions :

- Besoin remonté : pouvoir configurer l’environnement de dév en fonction du profil du développeur : dév backend / frontend, QA, Devops.
- Le support du multi-repo est posé. L’équipe n’a pas encore la bonne solution. Peut-être avoir un repo à part.
- Les privates registries Docker ne sont pas encore supportés mais sera adressé d’ici la GA
- Quel est le business model derrière cette fonctionnalité ? Compris dans l’usage de la licence Docker Desktop.
- Comment le développeur B modifie le code du dév A et souhaite le renvoyer au dév A ? Tout simplement en le repartageant
- Une issue IntelliJ empêchait de faire la même chose qu’avec VSCode

Teasing : une extension Docker Desktop est en cours de dév pour faire de la diff d’images. Sera annoncée autour de la Docker Con.