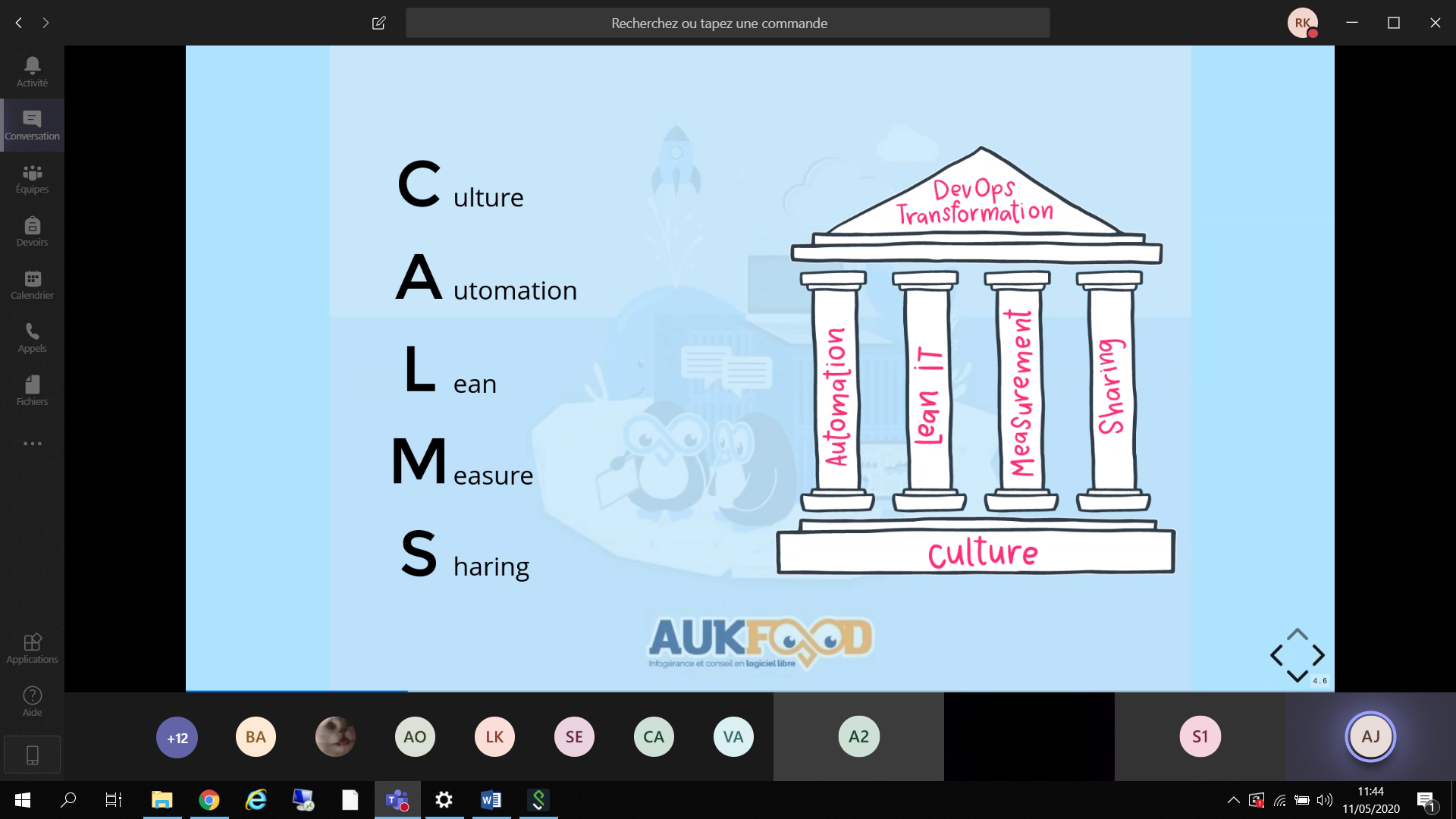
Le DevOps

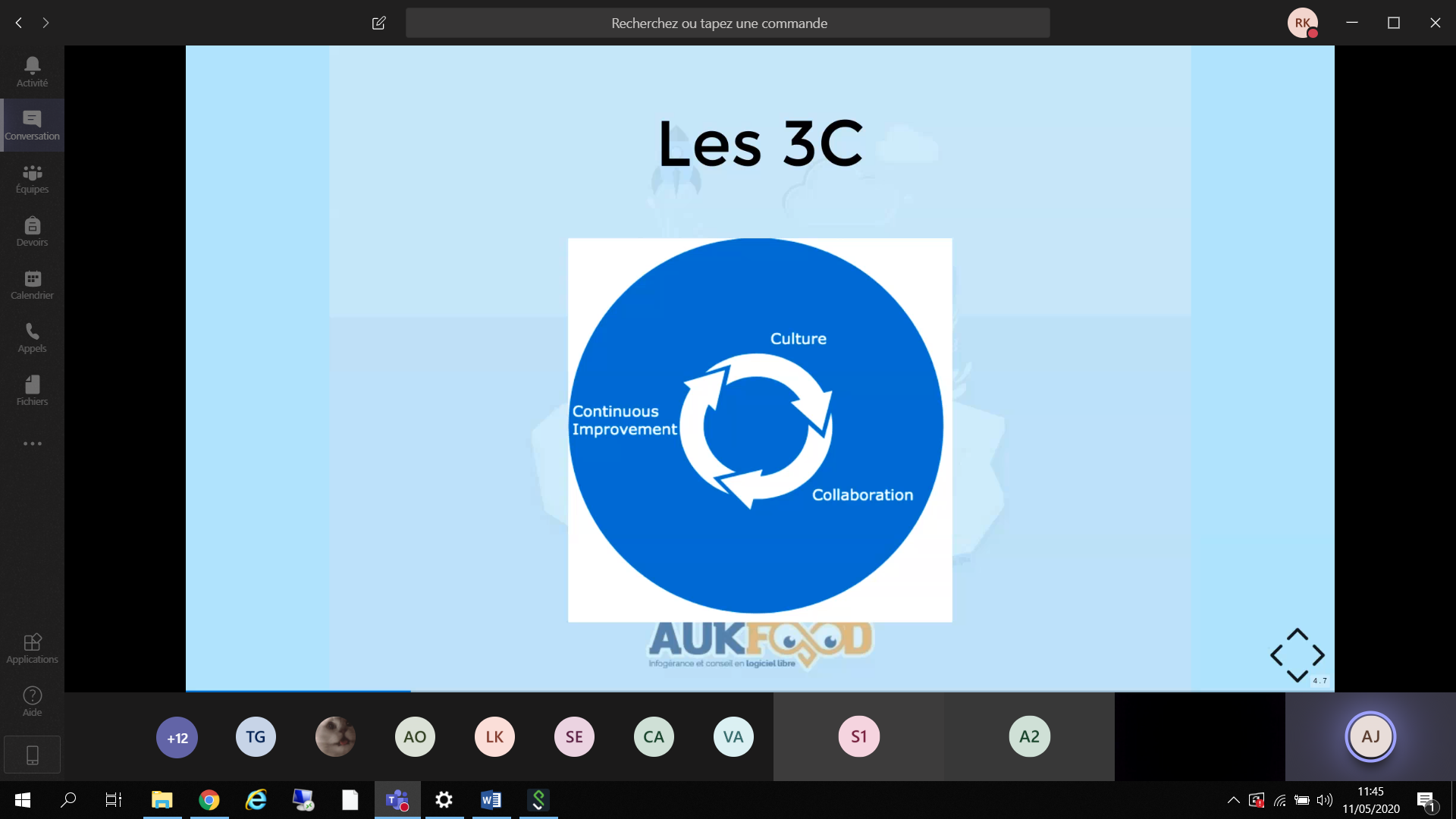
## Le DevOps c’est quoi ? C’est qui ? C’est qu’est ce ?

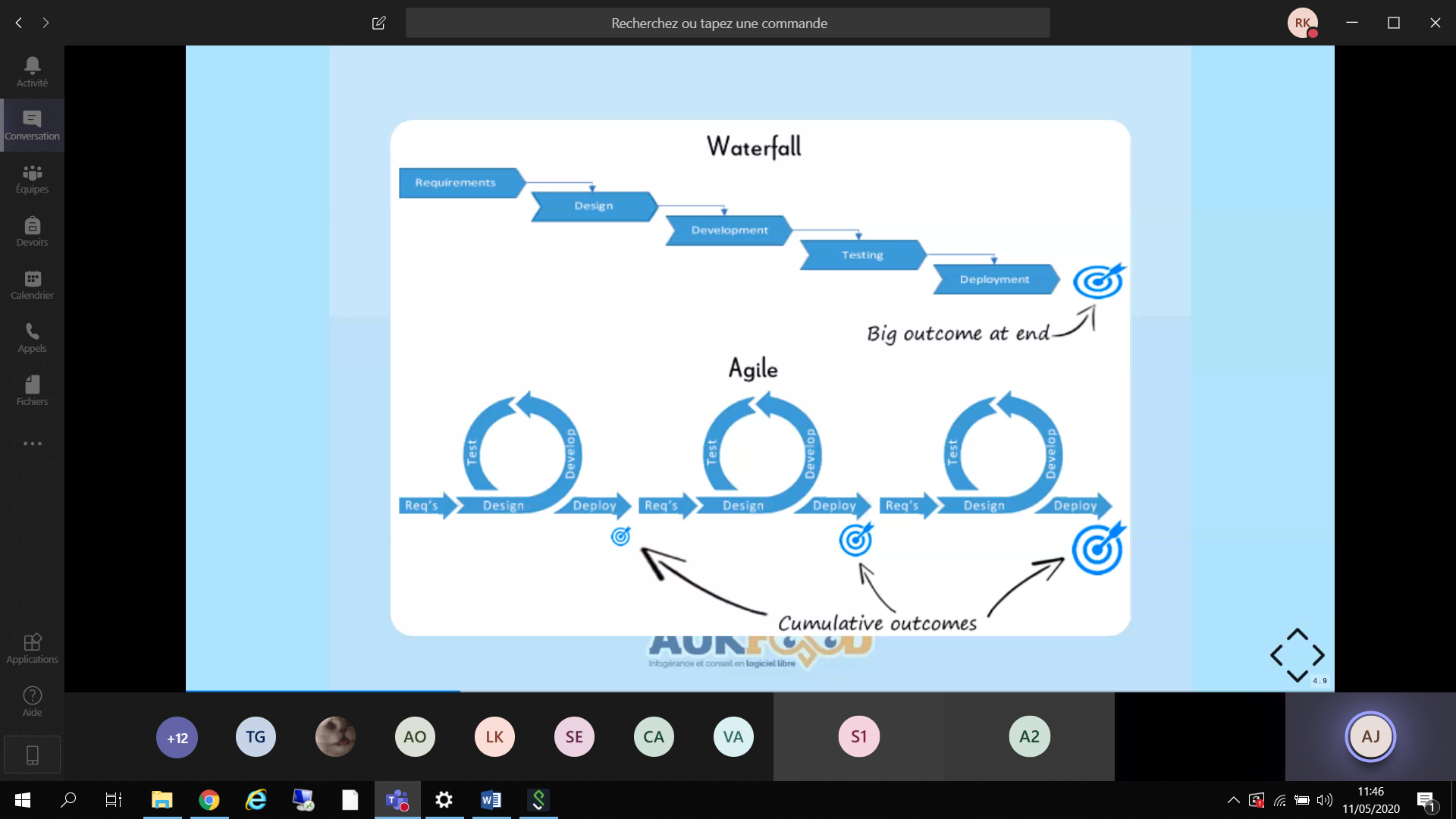
Au debut : une philosophie.

Plusieurs méthodes pour la mettre en pratique :

* The 3 ways : 
* Principe CALMS :
  + **Culture** ; Etre en communication et symbiose entre les équipes
  + **Automatisation** ; Tout automatiser
  + **Lean** ;Se focaliser sur la prise de valeur
  + **Mesure** ; a base de KPI
  + **Share** : Partager l’experience, le feedback



* Les 3C :
  + 



A chaque fin d’iteration le client est impliqué

## Les outils utiles :



Yarm

Runner = Service qui fais tourné les jobs qui passent par les Pipeline. Nécessaire pour faire tourner l’intégration continue.

<https://www.cvedetails.com/> : rechercher des failles dans les dépendances et package donner par par exemple sonarQube.

## 12 factors app : <https://www.youtube.com/watch?v=qlF378oDqW8> + voir les fichiers dans le dossier

1. CODEBASE

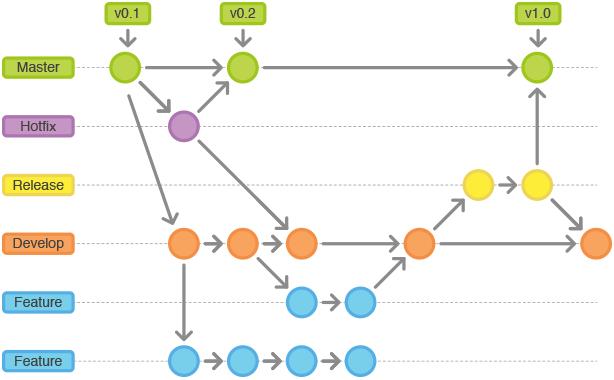
Avoir une seule version principale du code de l'application, stocké, géré et versionné sur un outil de versionning comme git.

L'important est de tout tracer à travers les différents déploiements, et que le codebase ne représente qu'une seule application.

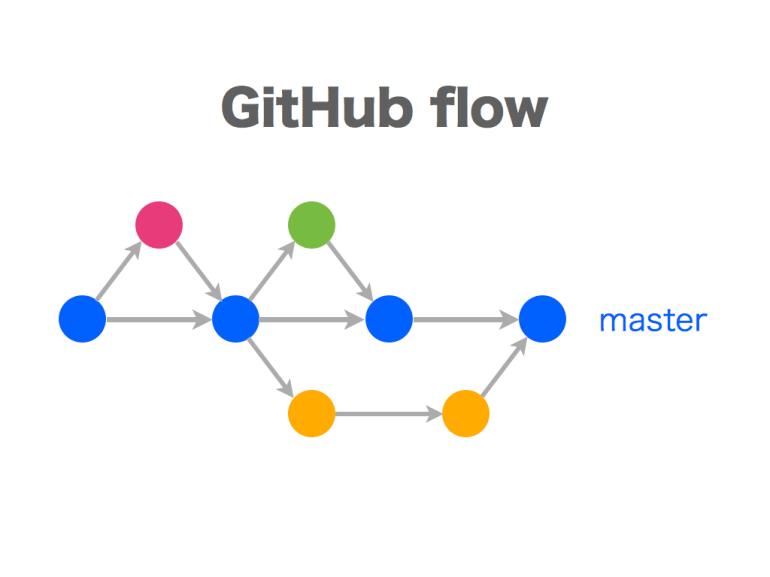
De sorte à ce que chaque déploiement soit de la même base pour tout le monde et puisse être

Tout est à récupérer, chaque commit et chaque fichier de configuration.

### Le gitFlow :



### Le gitHub flow :



1. DEPENDANCE

Garder la déclaration de toutes ses dépendance aussi regroupée et explicite que possible

Eventuellement récupérer directement les dépendances en package au sein de l'application pour éviter tout problème potentiel d'accès à ces dépendances

Ca simplifie grandement l'initialisation du projet pour les développeurs arrivant sur le projet en n'ayant besoin que du compilateur du langage et de l'outil de gestion des dépendances.

Utilise un manifeste de déclaration des dépendances, et un outil d'isolation des dépendances pour s'assurer qu'aucune dépendance implicite ne provienne du système.

En Ruby par exemple, on peut utiliser Gemfile pour le manifeste et bundle exec pour l'isolation. En Python, Pip est utilisé pour le manifeste et Virtualenv pour l'isolation.

…(voir les autres points dans les autres fichiers )

## Gestion de projet

Bien démarrer avec la sécurité by design et privacy by design :

Etape 0 : Choisir ses outils de travail ( ex : quel IDE, quel outil de planification, de suivi de tache, de messagerie instantanée à mettre en place)

Etape 1 : Partie management : Agile/Waterfall

Product backlog = définition de la feature a détaillée dans le sprint backlog

Etape 2 : Préparer son développement en toute sécurité

* Faire une AIPD
* Gardez la maitrise du système (comprendre les rouages de notre système de la partie développement a la partie production)
* Adopter un methodo agile intégrant la sécurité : <https://www.ssi.gouv.fr/uploads/2018/11/guide-securite-numerique-agile-anssi-pa-v1.pdf>
* Ne pas se reposer sur une seule défense
* Utiliser des normes de programmation incluant la sécurité ( Règles OASP : <https://owasp.org/www-project-top-ten/>)

Etape 3 : Plan

Etape 4 : Gestion du code source

* Penser à l’architecture de l’outil
* Gerer les paramètres de visibilité des dépots
* Gerer les permissions d’accès et de commit
* Ne pas enregistrer de données personnelles ni de « secrets » dans votre code sources
* Utiliser les fonctionnalités conçues pour facilité le dev

Etape 5 : Bibliothèques, SDK ou outils tiers : comment les intégrer dans vos applications ?

* Utilisez des sytèmes de gestion de dépendances
* Lisez les documentation et changer leur config par défaut
* Auditez vos bibliothèques et SDK
* Gerer les MAJ des dépendances

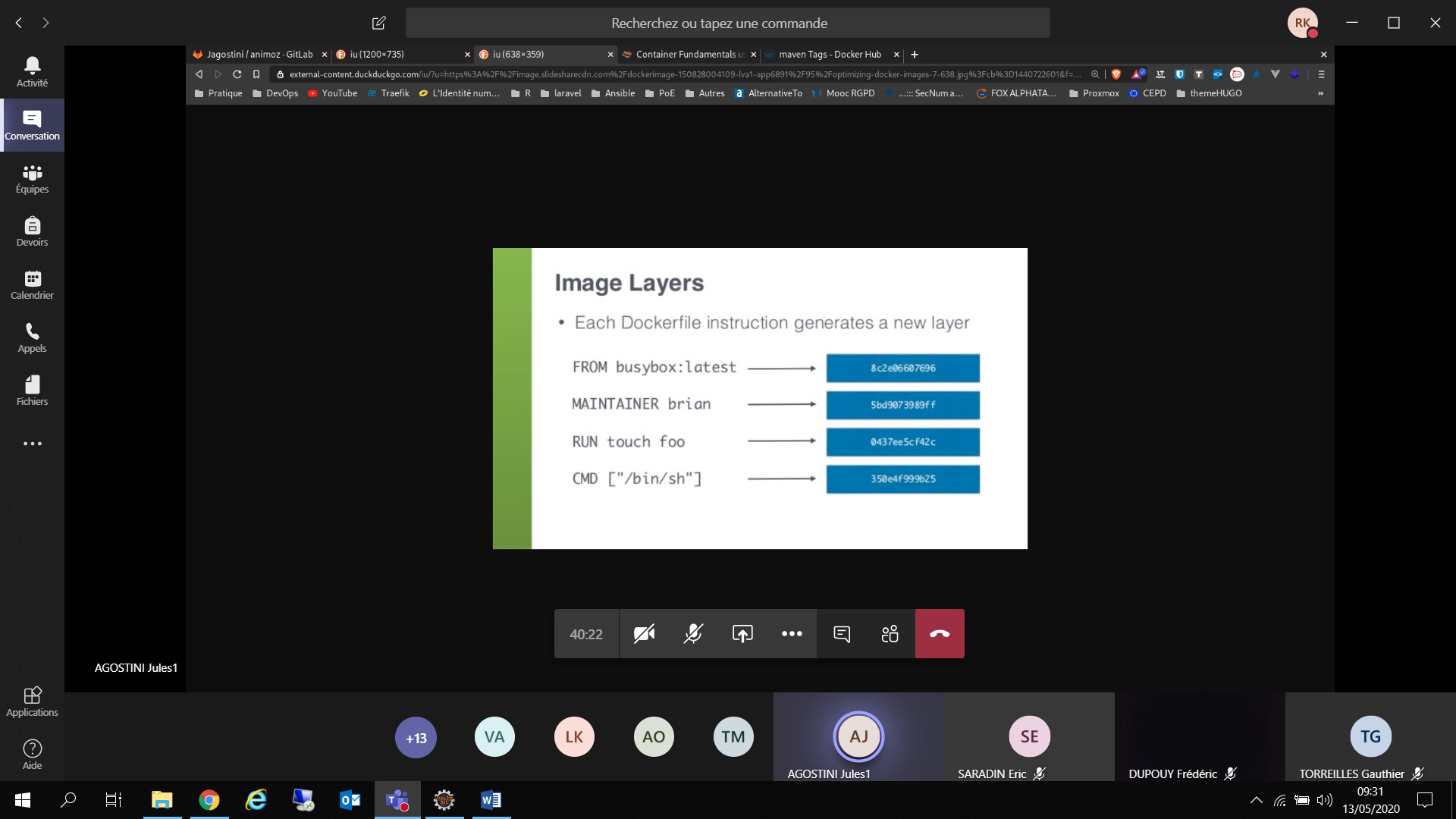
Swagger-PHP : générer de la documentation pour mon API

SPOF = Single Point of Failure



## Docker :

Dans le dockerfile chaque ligne de déclaration ajoute un nouveau layer, il faut donc penser a optimiser les declaration afin d’optimiser au mieux notre container



FROM : Nom de l’image docker de base utilisée pour créer le container

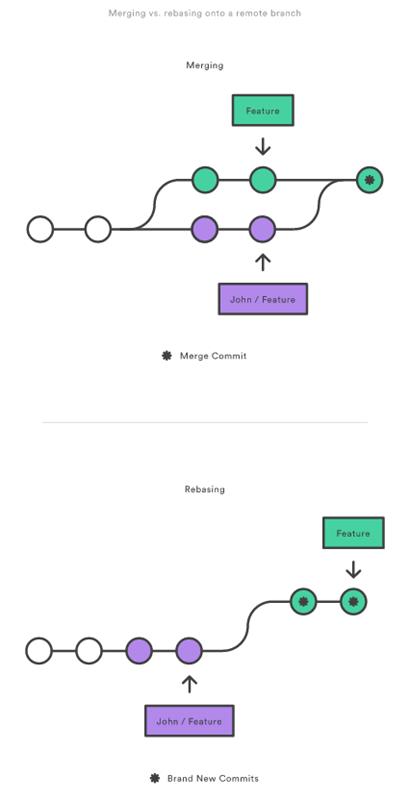
RUN : liste des actions préalable a lancer

CMD : première commande lancé au démarrage de l’image

Ensemble de cours/scenario pour Docker : <https://www.katacoda.com/loodse/courses/docker>

DockerCompose permet de définir un fichier de configuration permettant de spécifier les services dont on a besoin et de créer une suite de container les contenants : <https://docs.docker.com/compose/install/>

Puis faire le getting started



TP :

Objectif 1:

Ajouter une étape "dockerbuild", dans laquelle vous aurez un job qui :

- se connecte au registry

- construit l'image docker

- envoie l'image sur le registry

Objectif 2:

Variabliser le nom de votre image docker

- 1ere variable "CONTAINER\_IMAGE" pour le nom initial

- 2eme variable "CONTAINER\_IMAGE\_BUILT" pour reprendre le nom et ajouter un TAG (le tag devra etre le nom de la branche + le sha court du commit)

- 3eme variable "CONTAINER\_IMAGE\_LATEST" pour reprendre le nom et ajouter le tag "latest"

Modifier le job "dockerbuild" pour :

- construire l'image avec la variable "CONTAINER\_IMAGE\_BUILT"

- envoie l'image CONTAINER\_IMAGE\_BUILT sur le registry

Objectif 3:

Ajouter une étape "releases", dans laquelle vous aurez un job qui :

- se connecte au registry

- récupère l'image CONTAINER\_IMAGE\_BUILT

- rennomer l'image CONTAINER\_IMAGE\_BUILT en CONTAINER\_IMAGE\_LATEST

- envoie l'image CONTAINER\_IMAGE\_LATEST sur le registry