Automatisation Labos

REDS

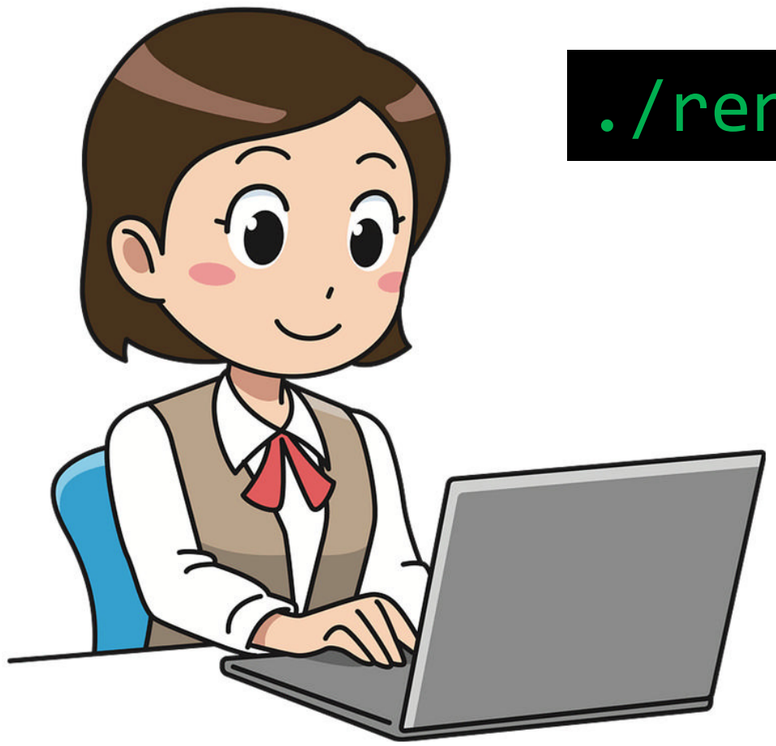
Buts

- Le plus simple et transparent pour les étudiants
- Le plus flexible pour les assistants (car labos très différents)
- Le plus automatisé possible

Rendus étudiants

```
./rendu.sh session
```

```
./rendu.sh fin
```



Archive à rendre sur Cyberlearn

```
./rendu.sh session
```

- **Commun :**
  - Crée une archive avec nom, timestamp et fichiers de labos (ou diff) et la chiffre
  - Dans /home/reds/rendus/COURS_LAB_...
  - Envoie une notification vers teams (utile ?)

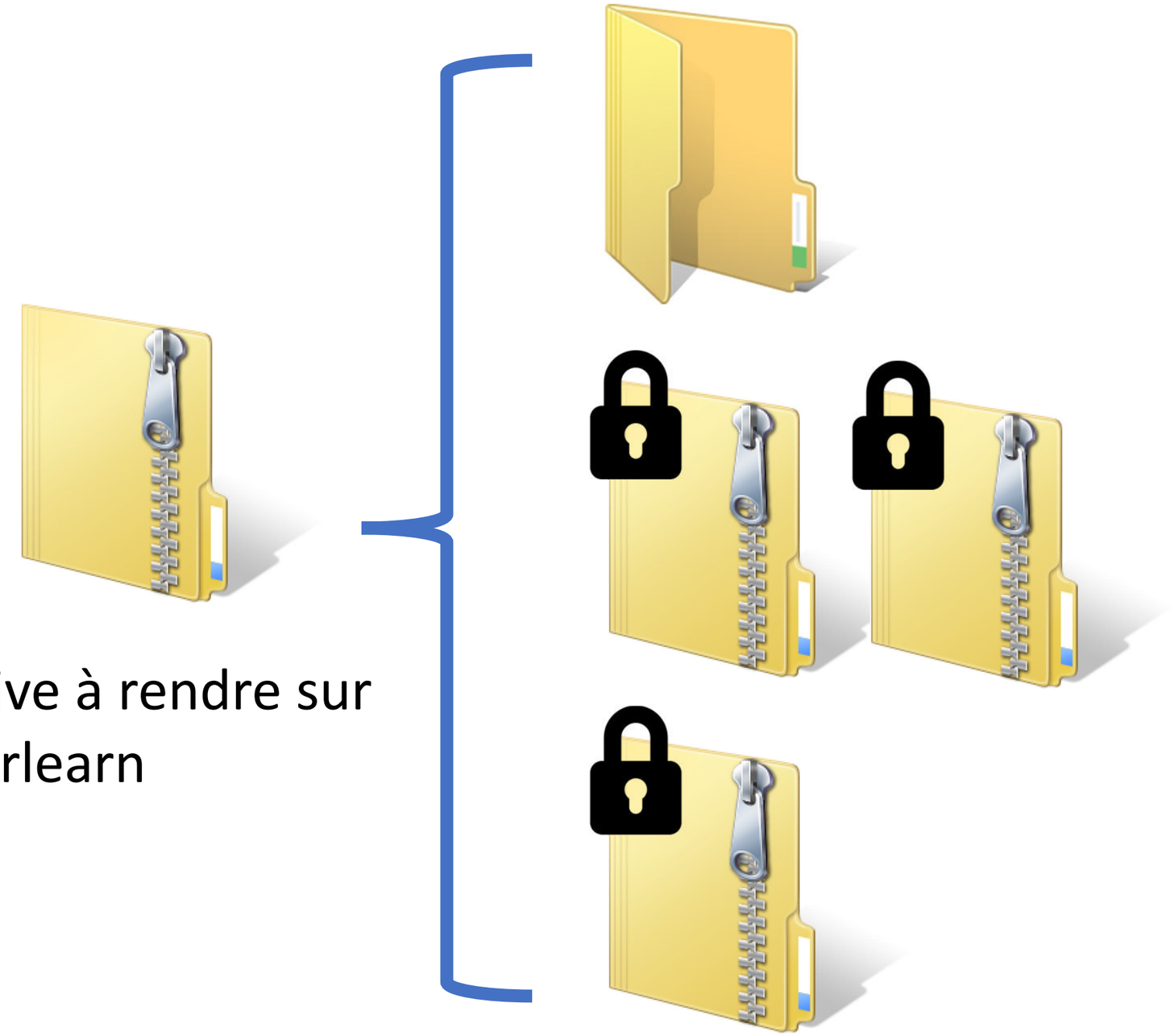
- **Spécifique à chaque labo** :
  - Le contenu (fichiers) de l'archive ci-dessus

Comme ça si ils ont trop mis le bazar dans le dossier du labo et qu'ils veulent le supprimmer pour recommencer les sessions restent ici

```
./rendu.sh fin
```

- **Commun :**
  - Crée une archive avec nom, timestamp et fichiers de labos (ou diff) et la chiffre
  - Dans /home/reds/rendus/COURS_LAB_...
  - Envoie une notification vers teams (utile ?)
  - **Crée une archive (pour rendu) avec les archives des sessions ainsi que l'archive de fin ainsi qu'un double de tous les fichiers rendus en clair**
    - **Afin que l'étudiant puisse vérifier son rendu**
    - **Nous pouvons vérifier l'avancement et contrôler les soupçons de plagiat avec les archives chiffrées**

- **Spécifique à chaque labo** :
  - Le contenu (fichiers) de l'archive ci-dessus



Fichiers finaux en clair

Sessions

Archive à rendre sur Cyberlearn

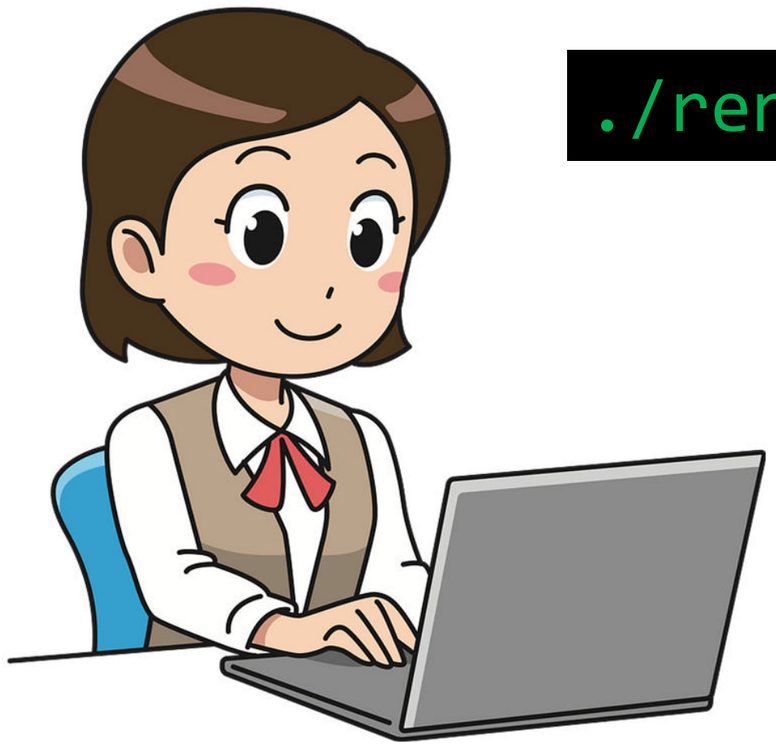
Double du rendu final

Clefs Publique (Students) / Privée (REDS)

Doit rester stable durant le semsestre !
Nos modifications / updates ne doivent pas changer les actions requises de l'étudiant !

```
./rendu.sh session
```

```
./rendu.sh fin
```



Archive à rendre sur Cyberlearn

```
./rendu.sh <session|fin>
```

- Le script rendu.sh
  - Valide les arguments
  - Fait un “checkout” de “exec_rendu”
    - Mise à jour (transparente et sans errreurs)
  - Lance “exec_rendu”

- ”exec_rendu”
  - Est un exécutable qui contient le script réel de rendu (chiffré)
  - Fait tout le travail effectif de rendu

Script rendu.sh

```bash
#!/bin/bash
script=${BASH_SOURCE[0]}
# Get the path of this script
SCRIPTPATH=$(realpath $(dirname "$script"))

LAB=lab02 # Lab specific

[ $# == 0 ] && { echo "usage : ./rendu.sh <session|fin>" ; exit 1 ; }
cd "${SCRIPTPATH}"

git checkout "origin/${LAB}" exec_rendu &> /dev/null
./exec_rendu $1
```

Script rendu.sh

```bash
#!/bin/bash
script=${BASH_SOURCE[0]}
# Get the path of this script
SCRIPTPATH=$(realpath $(dirname "$script"))

LAB=lab02 # Lab specific

[ $# == 0 ] && { echo "usage : ./rendu.sh <session|fin>" ; exit 1 ; }
cd "${SCRIPTPATH}"

git checkout "origin/${LAB}" exec_rendu &> /dev/null
./exec_rendu $1
```

Exécutable exec_rendu

- Script réel de rendu **rendu_real.sh**

  Dans notre git privé,
  Spécifique à chaque labo

  - ”chiffré” avec SHC (Shell Compiler)
    - Compile le script en exécutable
    - (Script chiffré avec RC4 dans un exécutable C)
    - Pas la manière la plus sécurisée de protéger le script alors ne rien mettre d’extrêmement sensible dedans (comme une clé privée par ex !)

- Tester le script “rendu_real.sh” avant les labos, ensuite déployer
  $ shc –f rendu_real.sh –o exec_rendu
  exec_rendu dans le git étudiant, il y a un autre script pour ça.

Attention si on teste sans passer par le git, si on remplace juste exec_rendu il sera écrasé par le checkout !

Va faire la collecte de ce qui est nécessaire pour la correction
- **Fichiers ou diffs <- spécifique**

Va créer les ”metadatas” (commun)
- **Nom**
- **Timestamp**
- **Session (archive chiffrée)**
- **Message teams**

```
./rendu.sh fin
```

Cyberlearn

Archive à rendre sur Cyberlearn

Corrections

Julian

étape de collecte

Cyberlearn

Archive de tous
Les rendus du labo

GAPS

Liste des étudiants
(avec e-mails)
(par classe)

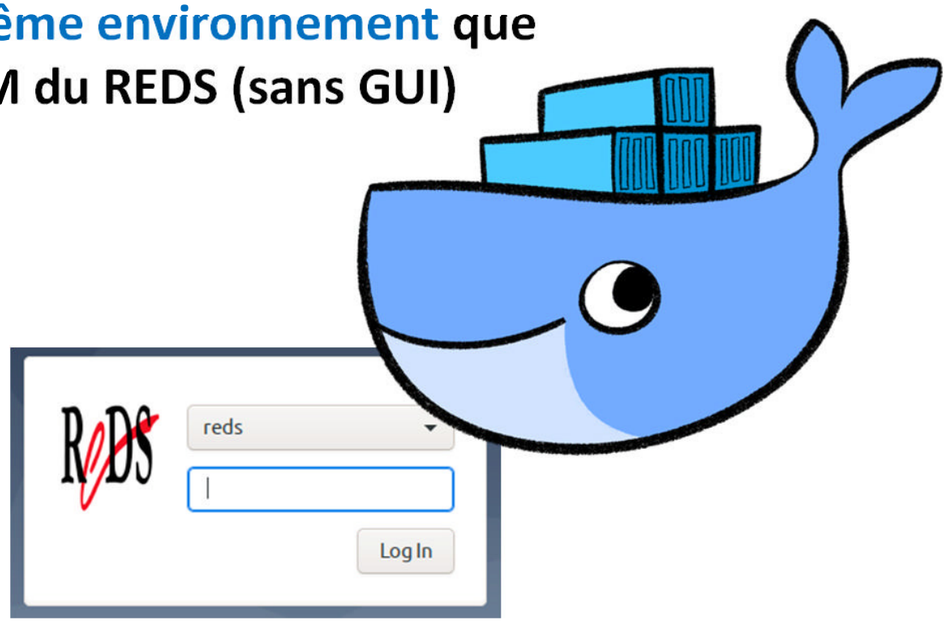
Décompression et association
dossiers / nom / e-mail

Artefacts :

- Dossier pour chaque étudiant
- Liste (excel ?) des étudiants qui ont rendu
- Liste des étudiants qui ont rendu à temps

Sur base des scripts
déjà existants de Julian

Image Docker qui contient
le même environnement que
la VM du REDS (sans GUI)



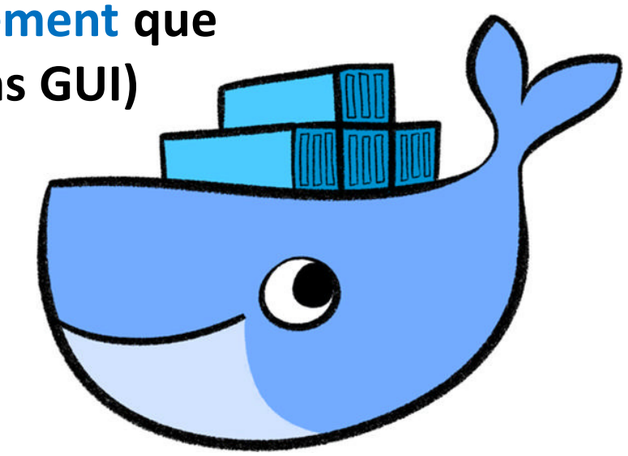
Lucas

étape de correction

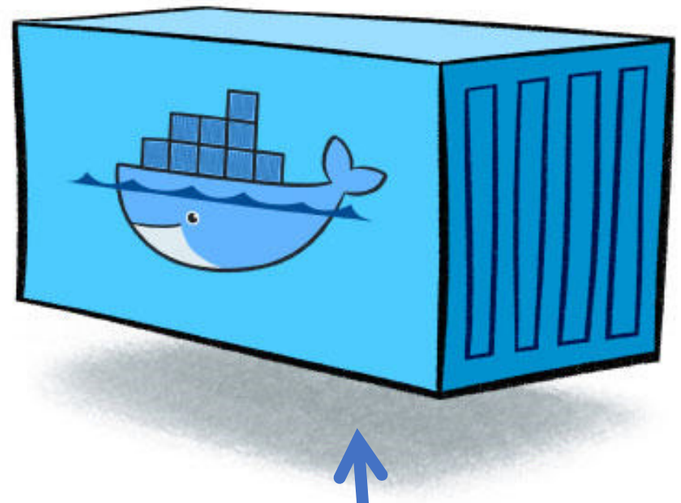
Automatisée, isolée, dans container Docker

Lucas

Image Docker qui contient le même environnement que la VM du REDS (sans GUI)

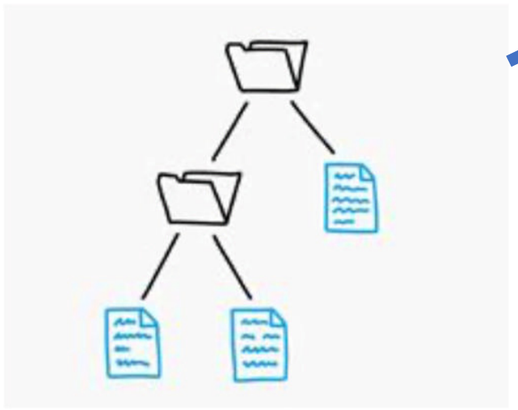


Container lancé pour chaque rendu

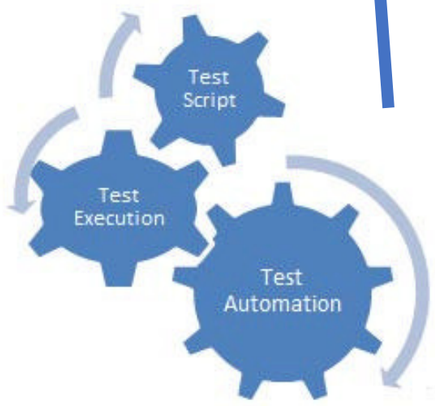


Artefacts :
Rendu corrigé ou amorce pour corr. manuelle

Dossiers de chaque étudiant
(artefact de l'étape de collecte)



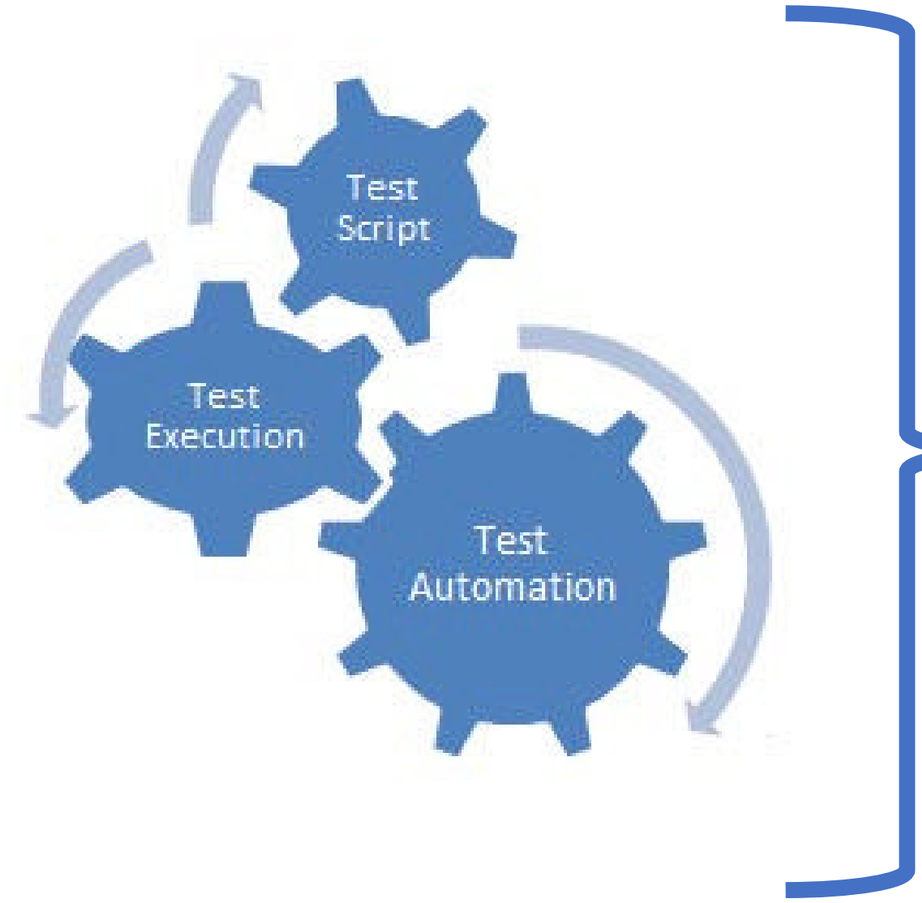
Dossier étudiant monté en volume



Environnement de test
Monté en volume aussi
(identique à tous les containers)

Spécifique au labo

Tous



Suite de tests :

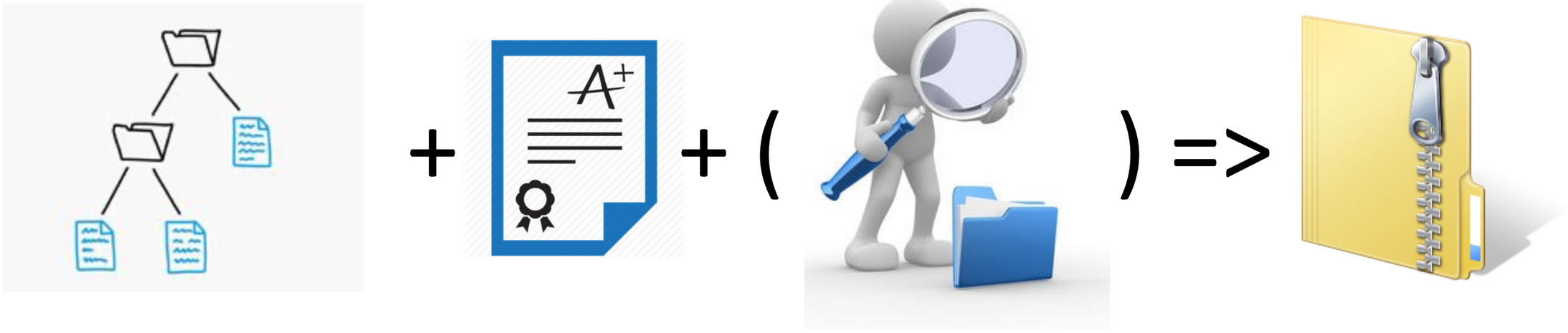
- **Spécifique à chaque labo** :
  - Environnement de test
  - Tests
    - Peuvent-être aussi simple que tester si les fichiers sont présents et compilent
  - Artefacts générés

- **Outils :**
  - Laissé libre afin que chacun puisse utiliser selon ses besoins et son expérience

- **Commun** :
  - Comment lancer la suite de tests
  - Comment récupérer les résultats (artefacts)
  - Report si elle s’est lancé / terminé correctement

Reflète le script “rendu_real.sh”

Pour chaque étudiant :
rendu + correction + (corr. Manuelle) => rendu corrigé



Tous

Retour aux étudiants par mail

Julian

Génération du mail
Génération de la note
Génération de la grille
(ou partie de la grille)

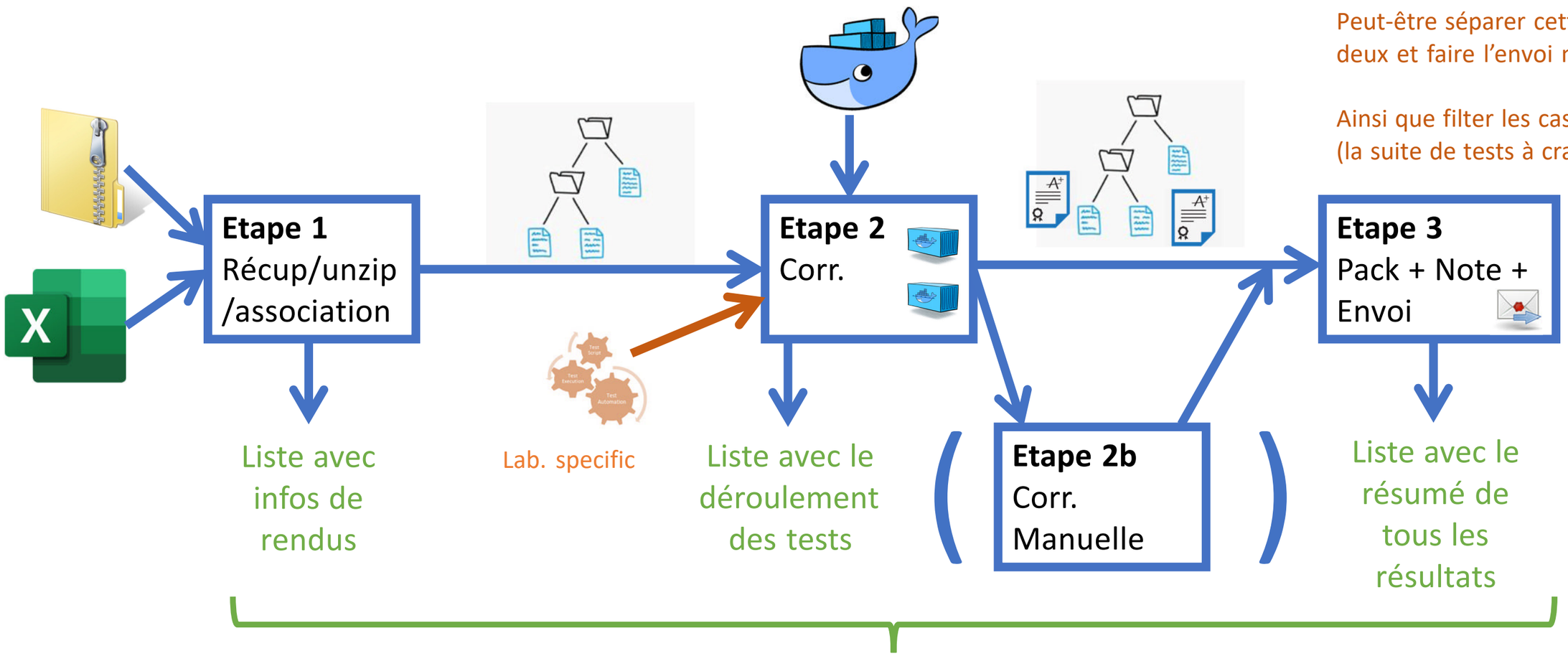
Sur base des scripts déjà existants de Julian

Pipeline de correction

Jenkins

Peut-être séparer cette étape en deux et faire l'envoi manuellement

Ainsi que filter les cas à problèmes (la suite de tests à crashé par ex.)



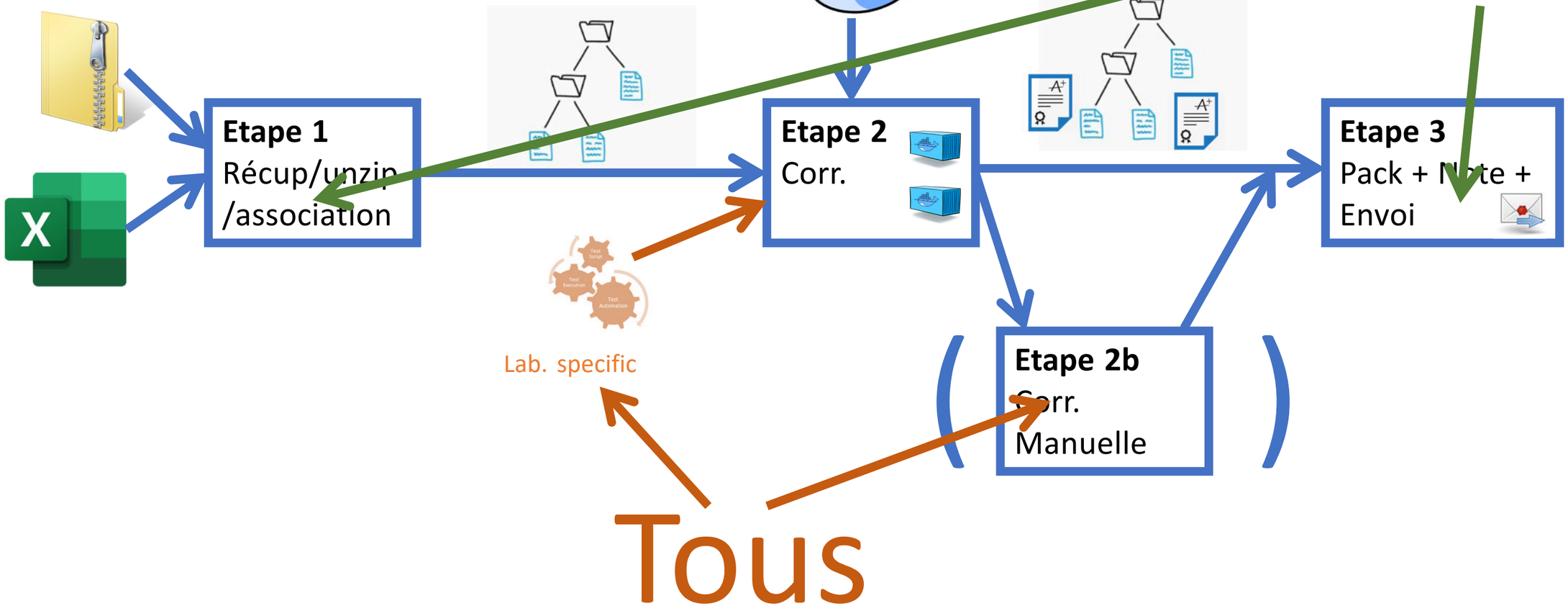
Lucas

Pipeline de correction

Lucas

Jenkins

Julian



Discussion

Validation de l’étape 1

- Tests depuis Cyberlearn et fichiers GAPS (classes)

Validation de la suite de tests (étape 2)

- Comment valider la suite de test (par labo)
- Comment valider qu'elle est ok pour le framework général

- Faire tourner le pipeline avec
  - Un labo vide
  - Le labo avec les solutions
- Valider les résultats (labo vide -> pas de points, solution -> tous les points)

Validation de l’étape 3

- Pack correct avec les infos nécessaires et rien de plus
- Qu’est-ce que l’on rend (par labo)
- Automatisation du packaging / annotations
- Interface commune mais actions spécifiques

TODO (Discussions 01.10.20)

- Délais de roadmap (à faire)

- Solution pour hors VM
  - -> Docker (à documenter, exemple avec visual studio code ou autres)
  - Dockerfile - Image - Dockerhub
- Gestion du plagiat -> Etape de collecte
- Correction manuelle -> Par labo
- Faciliter l'étape de test pour tests manuels !
- Labo groupe, comment gérer ?
  - -> Etape de collecte
- Documenter et faire un template pour les scripts

Futur

- Clé GPG (GNUPG) par étudiant ?
  - Envoi de mail à une adresse du REDS et cela génère leur clé on leur envoie en retour et enregistre la clé publique
  - Comme ça les labos peuvent être signés en plus de chiffrés
  - Certifie l'auteur du rendu
  - Pour les étudiants c'est
    - 1) Un email à envoyer en début de semestre
    - 2) Le fichier reçu en retour (ou script) à mettre dans la VM au début
- Archive finale signée par les étudiants, sessions (et fin) chiffrées avec clé publique du REDS

Futur

- Propositions ?
- Vue temps réels de sessions / rendus ?