|  |
| --- |
| Télécom SudParis |
| Manuel Labtainers |
| Projet de fin d’étude ASR |

|  |
| --- |
| Ilyes BENIGHIL, Jiawei Yang  25/01/2022 |

Une image contenant texte, bouteille, vaisseau

Description générée automatiquement

Table des matières

[Manuel 2](#_Toc94627354)

[Introduction à Labtainers 2](#_Toc94627355)

[Installation 2](#_Toc94627356)

[Création de TP 3](#_Toc94627357)

[Notation automatique 7](#_Toc94627358)

[Partie Infrastructure et développeur 8](#_Toc94627359)

[Script de création des VMs 8](#_Toc94627360)

[Github 9](#_Toc94627361)

[Le set up du projet 10](#_Toc94627362)

[Moodle 12](#_Toc94627363)

[Problème avec la vraie infrastructure de Moodle de Télécom SudParis 15](#_Toc94627364)

[Infrastructure 16](#_Toc94627365)

[Infrastructure locale 16](#_Toc94627366)

[Infrastructure cloud 16](#_Toc94627367)

# Manuel

## Introduction à Labtainers

## Labtainers est une solution qui permet de simplifier la notation et la réalisation d’un TP d’informatique. Cette solution permet de lancer un TP dans un conteneur Docker, avec toutes les dépendances nécessaires pour le bon déroulement du TP. A la fin du TP l’étudiant n’a qu’à effectuer une commande pour rendre son travail ( qui est généré automatiquement ) sur un dépôt Moodle.

## Pour les enseignants, cela permet de gagner beaucoup de temps non seulement lors de la notation des travaux rendus par les élèves, car avec juste une commande les enseignants peuvent noter tous les élèves d’un coup. Mais pour cela, il faut créer un TP adapté à Labtainers (cf partie Création de TP). Mais surtout, Labtainers permet aux enseignants d’avoir un environnement de travail commun à chacun des élèves. Donc plus de problèmes de mauvaise version des logiciels ou problème de compatibilité avec leur système d’exploitation, car tout le monde exécutera leur TP dans le conteneur Docker crée spécialement pour ce TP.

## Installation

Labtainers doit être installé sur une VM, et nous avons testé avec seulement VirtualBox donc il est certainement possible que cela ne fonctionne pas comme prévu avec d’autres logiciels de virtualisation. Dans un 1er temps télécharger l’image de la VM que nous avons préparé ici ( LIEN ), puis l’importer là dans VirtualBox. Il est impératif d’allouer au moins 20 GB, pour cette VM et vous pouvez changer le nombre de processus et la RAM selon les performances de votre machine Host.

Une fois votre VM (qui est un Ubuntu français à jours) lancé, vous avez juste à écrire ./update-lantaniers pour mettre à jour Labtainers qui est déjà installé sur cette VM avec toutes les dépendances. Vous pouvez installer sur cette VM tous les autres logiciels que vous voulez (comme un IDE par exemple ) mais faites toujours attention à l’espace disponible sur la VM.

Pour les personnes qui veulent installer Labtainers sur une machine host cela est beaucoup plus complexe. Dans un 1er temps, téléchargez l’archive sur du projet ici avec cette commande

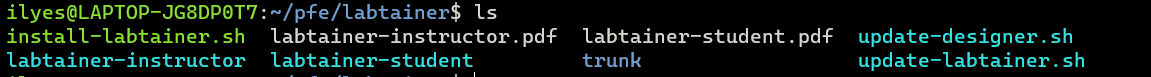
wget https://codeload.github.com/IlyesBenighil/LabtainersPFE/zip/refs/heads/

Puis dézipper le tout avec :

unzip master

Puis changer le nom du fichier dézippé par labtainer

Vous devriez avoir tous ces fichiers :



Pour la suite, vous devez avoir docker de lancer sur votre machine et avoir les droits administrateurs. Vous devez aussi au moins python 3 et pip pour l’installation et installer cela :

pip install python-dateutil

Puis vous pouvez lancer le script d’installation avec :

./install-labtainer.sh

Pour les étudiants, il faut aller dans le répertoire labtainer-student et pour les professeurs il faut aller dans labtainer-instructor.

Pour ces derniers, l’installation n’est pas finie, il faut aussi installer le pack designer qui permet de créer les TP. Pour cela, il suffit de lancer le script update-designer.sh

## Création de TP

Labtainers fournit une interface graphique pour créer des TP accessible uniquement après avoir lancé le script update-designer. Lancé la commande labedit dans un terminal pour accéder à cette interface

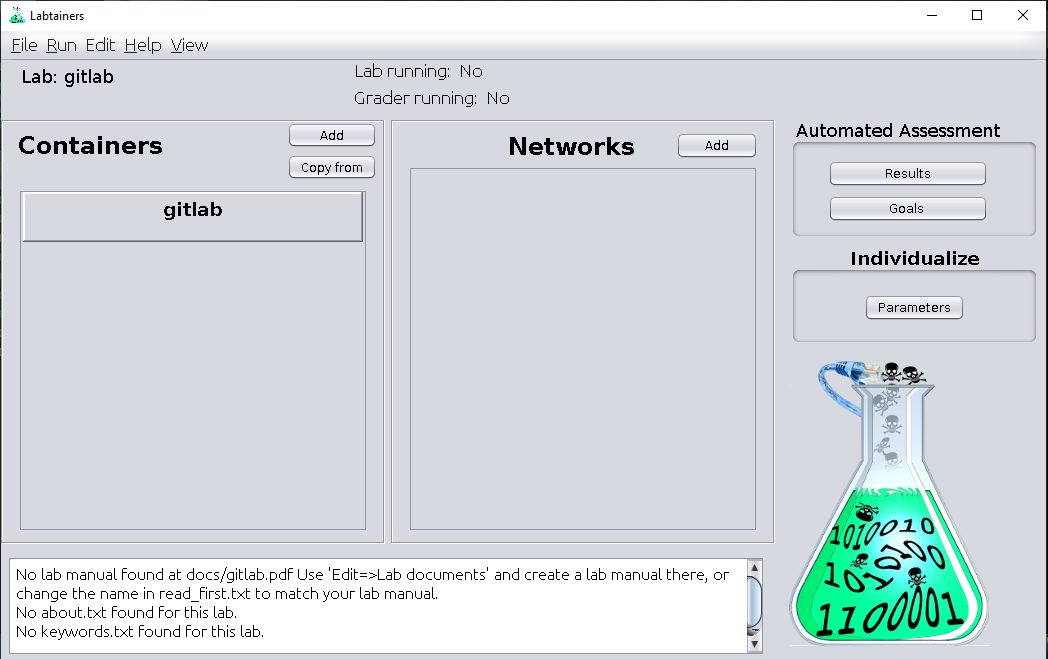




Figure 1 Interface graphique pour la création de TP

Pour créer un TP allez dans File => New Lab pour avoir cette interface :

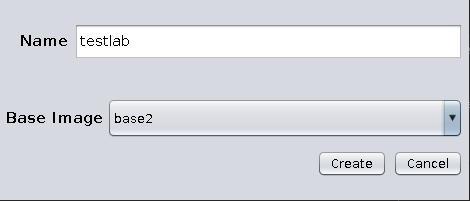
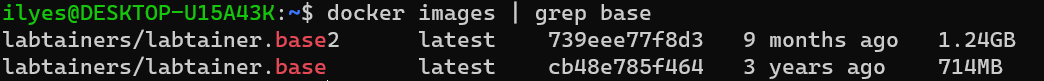


Figure 2 Fenêtre pour crée un TP

Les base images sont des images docker déjà installé sur votre machine si vous avez fait l’installation correctement. Par défaut, il y a deux images :



Tous les TP ont des dockerfiles, qui va genré une image quand l’étudiant lance un TP .

Pour votre TP, vous serez amené à avoir besoin de dépendance, pour cela, il ne faut pas modifier les images base. Il faut donc modifier les dockerfiles des TP. Pour cela, si vous savez utiliser VIM il suffit de cliquer sur le nom du TP puis Docker => edit dockerfile. Si vous voulez éditer le fichier avec votre éditeur de texte préférer, pour cela, il faut aller dans le répertoire labtainer/trunks/lab/nom\_du\_TP /dockerfile et modifier le fichier.

Vous serez également à amener à ajouter des documentations à lire pour l’apprenant au début de son TP comme l’énoncé du TP par exemple. Pour cela, il suffit de créer un répertoire /doc dans labtainer/trunks/lab/nom\_du\_TP/nom\_du\_conteneur/ et d’ajouter dedans tout ce que l’apprenant aura besoin pendant son TP.

Dans notre exemple de gitlab, on a ajouté un document description.txt dans lequel on décrit une addition que l’apprenant doit effectuer. ( Oui, ça n’a pas de rapport avec git, mais c’est un exemple simple pour décrire la fonction de randomisation plus tard.)

Une image contenant texte

Description générée automatiquement

Figure Structure des dossiers du TP

Pour finaliser le TP, il faut capturer les artéfacts et les règles dont on aura besoin pour faire la notation automatique du TP.

Pour récupérer les artéfacts, si c’est le stdin et le stdout d’un programme système, il faut ajouter le nom de votre programme dans le fichier treataslocal dans LABTAINER\_DIR/labs/[labname]/[container name]/\_bin/treataslocal.

Une image contenant texte

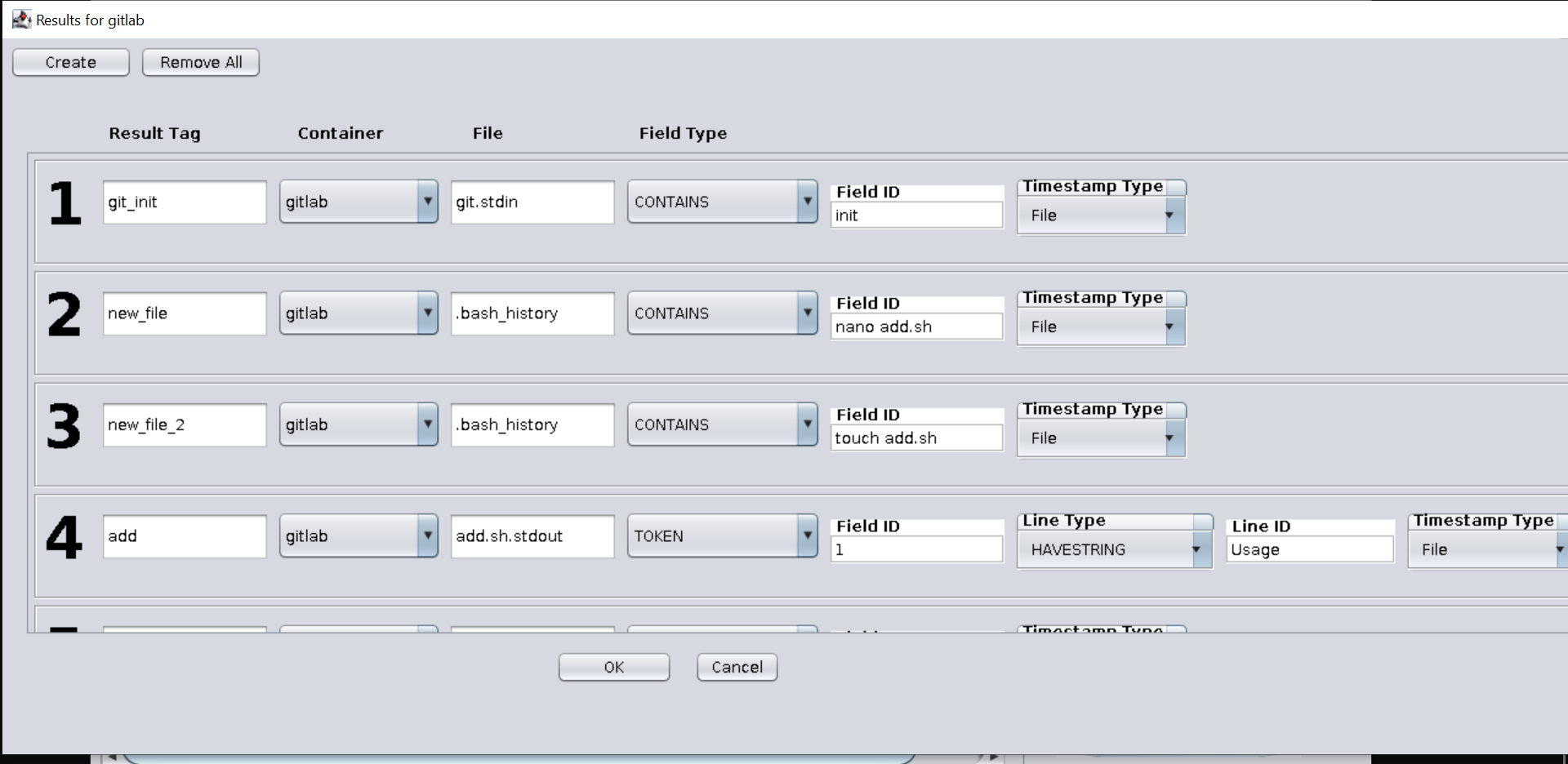
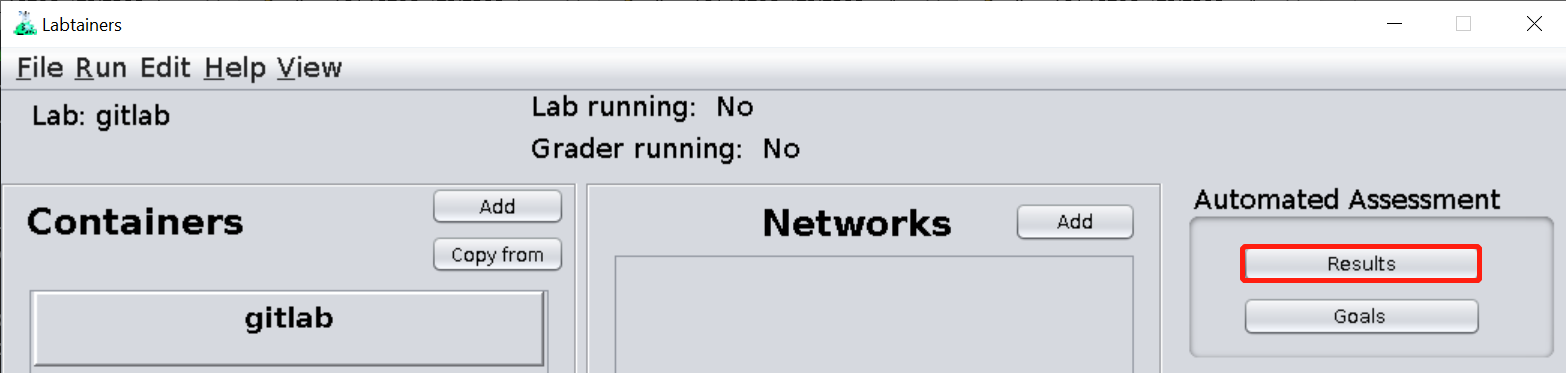
Description générée automatiquementPar exemple, pour le TP gitlab, on avait besoin des entrées et sorties de git, pour vérifier si les commandes de git ont bien été effectué, de ce fait, on a ajouté le nom du git dans le fichier. De plus, on peut aussi récupérer le stdin et le stdout d’un fichier de programme. Par exemple, dans un TP où on aura besoin d’écrire un script add.sh pour effectuer les additions en bash, on peut ajouter le nom du fichier add.sh dans le fichier treataslocal pour capturer tous les stdin et stdout du script bash.

Remarque importante : Le fichiers treataslocal DOIT avoir une fin de fichier LF et PAS CRLF.

Une image contenant table

Description générée automatiquementLes résultats de ces commandes sont donc stockés dans le dossier mail\_eleve.nom\_du\_TP= nom\_du\_TP. nom\_du\_TP.student.zip/.local/result

Une fois tous les artéfacts sont récupérés, on peut maintenant définir les résultats dont on veut récupérer pour évaluer le TP. Cette manipulation doit être fait avec l’interface de labedit.

 En effet, pour vérifier ce qu’on a fait dans le TP, on doit cliquer sur le bouton results qui nous dirige vers un nouvel onglet de récupération des résultats.

Ces critères sont définis de cette manière :

* Result Tag est le nom du test qu’on pourra utiliser dans la définition des Goals.
* Container est le nom du container dans lequel on veut récupérer les informations.
* File est le nom du fichier dans lequel on veut récupérer les informations.
* Filed Type est le type de comparaison qu’on veut mener, il en existe deux types. Le premier type sont des opérations mener sur le fichier en entier comme l’exemple en haut avec le CONTAINS. Un deuxième type est celle où on doit préciser la ligne où on veut mener l’opération. Exemple de add où on va récupérer le premier mot de la ligne (field id est le nombre de mot qu’on veut récupérer) des fichiers add.sh.st qui contient le mot (HAVESTRINGS) Usage.
* Field Id varie en fonction de Field Type, si Field Type est CONTAINS ou FILE\_REGEX alors le Field Id est le motif dont on veut chercher.
* Timestamp Type : Il n’est pas indiqué dans leur doc, mais avec File ça marche très bien.

Toutes ces descriptions comme vous pouvez le voir sont incomplètes, car l’utilité du Field Id peut varier avec Field Type donc il faut consulter leur doc pour plus de précision. Paragraphe 6.2.1 p28 du labdesigner.pdf

Une image contenant texte

Description générée automatiquementA partir de ces résultats récupérés, on peut maintenant définir le but du TP. C’est-à-dire résultat qu’on est censé avoir à la fin du TP. Par exemple dans ce TP de gitlab, on doit utiliser la commande git init ainsi que la commande git add. Donc, dans l’interface de définition de Goal, on peut définir un goal comme Use\_git\_command dans la figure ci-dessous :

Dans la définition de goal, il y a aussi plusieurs champs :

* Goal ID est le nom du goal.
* Goal Type est le type d’opération dont on veut effectuer sur les résultats selon les différents types on peut avoir des champs différents.

Pour le type matchany, matchlast ou matchacross on doit définir un opérateur classique du résultat qu’on a récupéré plus haut et de le comparer soit à une expression littérale qu’on écrit à la main comme dans le premier goal de la figure. On a récupéré le résultat add défini précédemment et on le compare au string Usage.

Pour le type count\_greater, on va définir une liste Subgoal List qui contient soit les goals id défini plus haut soit les results tag qu’on a défini dans l’onglet result et une valeur comprise entre 0 et la longueur de la liste moins un. L’opérateur va compter le nombre de True dans la liste, si ce nombre dépasse la valeur que nous avons définie, ce goal retournera True sinon elle sera False.

**Remarque** : la liste doit être séparée par des virgules + un espace. C’est-à-dire el\_1, el\_2, el\_3 et non pas el\_1,el\_2,el\_3! De plus, le Boolean Result Tags ne sert absolument à rien, il n’affecte pas du tout le résultat du goal et il est inutile de choisir un tag en particulier, car lorsqu’on réouvre l’onglet, le tag sera reset sur le premier tag.

Pour le type boolean, on doit mettre une expression booléenne dans Boolean et on peut ou non entourer l’expression par les parenthèses. De même, le Boolean Result Tags est inutile ici.

Il existe d’autre types dont nous n’avons pas eu le temps d’exploiter. Vous pouvez toujours vous référez au guide labdesigner.pdf section 6.3.1 page 31, 32.

[Partie sur la randomisation du TP]

De plus, Labtainers nous propose également la possibilité de randomiser des valeurs ou des hachages des fichiers dans l’énoncé du TP pour que les apprenants puissent avoir des sujets différents. Pour cela, dans l’interface de création de cours, dans la partie Individualize, en cliquant sur Parameters, on arrive sur un onglet de randomisation.

Comme vous pouvez le voir sur la figure ci-dessous, la randomisation doit contenir :

* Param ID qui est le nom de la randomisation
* Operator est le genre de remplacement on veut faire par exemple remplacer la variable par un entier ou un code de hachage etc.
* File name est le nom du fichier dans lequel se trouve la variable à remplacer (le bouton permet de choisir le fichier avec UI)
* Symbol est la variable dont on veut remplacer.
* Lower bound est la plus petite valeur que la variable peut prendre
* Upper bound est la plus grande valeur que la variable peut prendre
* Step est le pas à la quelle on peut avancer de la plus petite valeur à la plus grande.

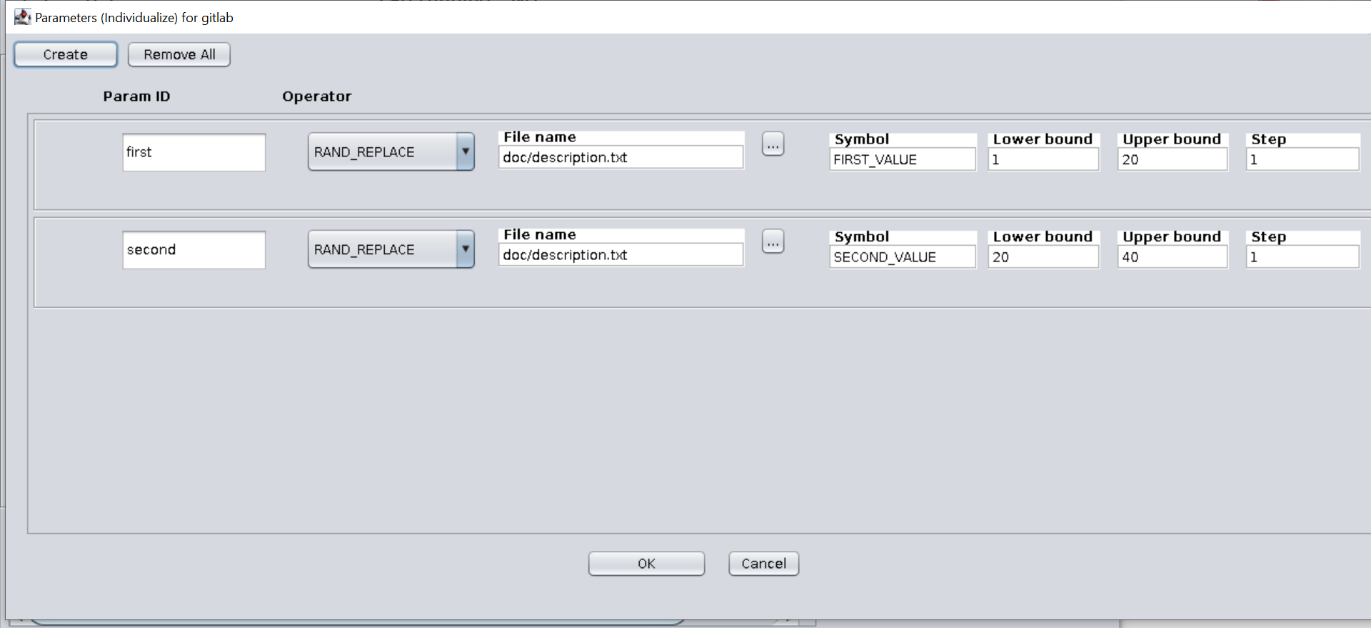
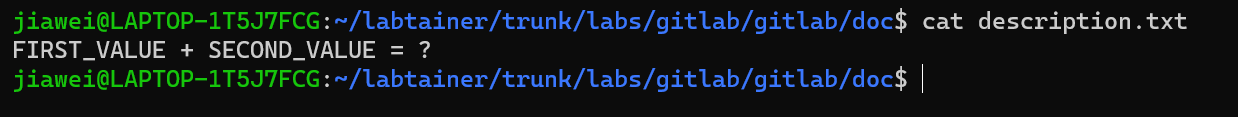


Figure L'onglet de randomisation de l'énoncé

Dans cet exemple, dans le fichier description.txt, on va remplacer la variable FIRST\_VALUE par un entier entre 1 et 20 et SECOND\_VALUE par un entier entre 20 et 40. Cette randomisation est générée au moment du build de l’image donc n’est exécuter qu’une fois au début lorsqu’on n’a pas l’image du TP.

Sur cet onglet, si on va tout à droite, on peut ajouter une description à chaque param.

Une image contenant texte

Description générée automatiquementCe qui donne :

## Notation automatique

Après la création du TP avec toute la démarche pour créer des résultats et les buts, on peut maintenant faire de l’évaluation automatique du TP. Pour cela, l’apprenant doit d’abord effectuer le TP puis restituer un dossier compressé dans le dossier labtainer\_xfer/nom\_du\_TP/ sous le nom du mail\_apprenant.nom\_TP.lab.

Une image contenant texte

Description générée automatiquementL’enseignant doit placer ce dossier compressé dans le même répertoire et puis lancé la ligne de commande suivant dans le /labtainer/labtainer\_instructor : gradelab nom\_TP

Cette commande va vous afficher le résultat à l’écran et générer un fichier json et txt dans le dossier labtainer\_xfer/nom\_du\_TP/ pour être exploité plus tard si besoin.

Dans le screen ci-dessus, on voit qu’Ilyes a juste utilisé les commandes git sans créer un nouveau fichier et n’a pas géré les fautes. Alors que Jiawei a fait tout le TP correctement selon les buts définis.

# Partie Infrastructure et développeur

## Script de création des VMs

Pour faciliter l’utilisation de Labtainers par un étudiant, nous mettons à disposition une image Virtual Box d’un Ubuntu en Français avec notre version de Labtainers. Le projet d’origine propose aussi une image similaire, mais nous ne savons pas comment cette VM est créé (surement à la main). Crée à la main une VM n’ai pas idéal, car pour la maintenabilité il est difficile de garder une trace et de comprendre un bug pour le corrigé.

C’est pour cette raison que nous utilisons [Packer](https://www.packer.io/) pour construire cette image. Packer est un outil disponible sur Windows et Linux (et autre) en ligne de commande (https://www.packer.io/downloads) . Pour construire cette VM il faut récupérer le code sur ce [lien](https://github.com/IlyesBenighil/LabtainersPFE/tree/vm) puis build l’image avec la commande :

packer build ubuntu.json

Packer va ensuite ouvrir Virtual Box et commencer l’installation de la VM. Cela peut prendre une vingtaine de minutes.

Nous sommes avons pris un Template d’un projet déjà existant ( <https://github.com/heizo/packer-ubuntu-20.04> ). Nous ne sommes pas parties d’une image ubuntu-desktop, mais bien d’une ubuntu-server dans laquelle nous avons installé une interface graphique (car il est plus simple d’interagir avec une VM sans GUI avec Packer).

Le fichier donne tous les arguments nécessaires à la création de cette VM comme le nombre de processus, la RAM ou bien la taille de l’image. Puis une fois l’installation de Ubuntu faites, Packer lance des scripts d’installation de package et de Labtainers (voir le fichier labtainers.sh). Pour enfin avoir un fichier .box (il faut dézipper ce fichier puis renommé le fichier avec l’extension .ova).

## Github

Une image contenant texte

Description générée automatiquementPour mieux gérer les TP et garantir la légèreté de Labtainers, nous avons choisi dans un premier temps d’utiliser Github comme interface de connexion entre les TP et le Labtainers. En effet, pour ne pas avoir à télécharger tous les TP de tous les cours à chaque installation de Labtainers, nous avons mis en place un projet Github [LabsPFE](https://github.com/Ironem/LabsPFE/) dans lequel on stocke tous les TP qui pourront être récupérés et utilisés par les élèves de manière simpliste avec une ligne de commande.

Le principe de fonctionnement est le suivant, un professeur va créer un TP à l’aide de l’interface graphique fourni par Labtainers décrit plus haut. Il récupérera par la suite le dossier généré et le compresse puis ajoute ce dossier compressé dans le répertoire du projet LabsPFE. Enfin, il lui reste plus qu’à push le TP avec un nouveau tag (comme la figure ci-dessous). Pour supprimer la release, il suffit de supprimer le tag avec la commande suivante : git push --delete origin tagname

Remarque : pour faire un push il faut donc que l'owner du projet ajouter ce professeur au préalable dans le projet Github. De plus, l'owner doit aussi ajouter un token au projet pour que tout fonctionne, le set up du projet est décrit plus bas.

Une image contenant texte

Description générée automatiquementLe push d’un tag permet de lancer une action de Github (qui a été défini dans upload\_zip.yml) pour créer une nouvelle release de tous les TP existant sur le projet avec le tag qui a été push. (ici on a le ssl.tgz qui a été ajouté)

Une image contenant texte

Description générée automatiquementLorsque la release générée, l’élève pourra récupérer le TP depuis son Labtainers avec le script get-lab.sh + le nom du TP en argument pour récupérer le TP. On peut aussi mettre la version du release qu’on veut pour le TP en deuxième argument.

Le contenu du TP sera automatique mis dans le bon répertoire et l’élève pourra ensuite effectuer la ligne de commande habituelle pour commencer son TP.

Remarque : Avec cette structure, il est plus difficile de gérer les TP et releases par module. Ce qui conduit à la réflexion d’une nouvelle structure (cf. Moodle et lti)

### Le set up du projet

Pour l'owner du projet, comme on va appeler les actions de Github, il faut ajouter dans le projet un token pour avoir les droits de créer une release à l’aide de l’action.

Une image contenant texte

Description générée automatiquementVoici les étapes à suivre : il faut tout d’abord générer ce token dans le Setting/Developer settings de votre compte personnel.

Une image contenant texte

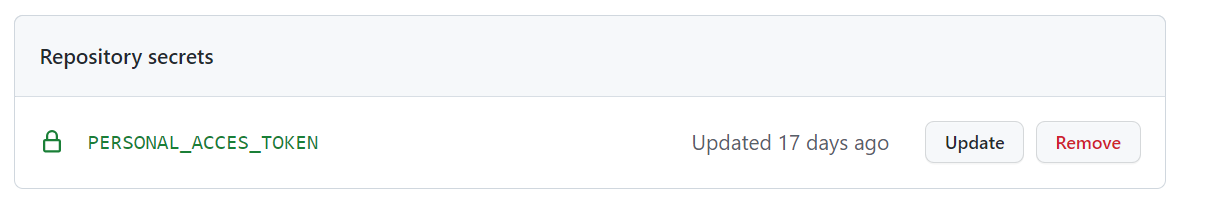
Description générée automatiquementPuis il faut cocher les 2 cases concernant le workflow comme le screen :

Ensuite cliquer sur Generate token tout en bas, ce qui va vous générer un token. Copier ce token puis allez sur la page du setting du projet LabsPFE, dans la section Secret puis Actions. Cliqué sur le New repository secret :

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquementPuis ajouter votre token et le nom du token est : PERSONAL\_ACCES\_TOKEN .

Enfin, vous devez obtenir cela sur la page de Secrets/Actions:

## Moodle

Pour que le rendu de chaque TP soit automatique, il faut connecter labtainers avec Moodle. Pour cela une api Moodle existe, cependant, elle est très difficile d’utilisation. Dans notre cas le script script.py permet d’uploader un fichier en demandant à l’utilisateur de se connecter. Pour réaliser cela il y a une grosse partie de configuration qui doit être faite manuellement sur l’interface web Moodle administrateur. Cependant, la documentation de Moodle est très mal faite pour expliquer comment faire.

Nous avons utilisé la version 3.11 de Moodle et tout tourné en local sur une machine Windows 10 20H2 build 19042.1466.

Dans un 1er temps, il faut installer Moodle sur votre machine (via ce [lien](https://download.moodle.org/windows/)) et crée un utilisateur administrateur.

Puis il faut créer un autre utilisateur (simulant un étudiant) avec ce lien http://127.0.0.1/user/editadvanced.php?id=-1 (127.0.0.1 est l’adresse du serveur web ou Moodle tourne)

Ensuite il faut activer les requêtes REST ici <http://127.0.0.1/admin/settings.php?section=webserviceprotocols>

Une image contenant texte

Description générée automatiquement

Figure Activation du protocole REST dans l’interface web

Puis il faut autoriser le groupe d’étudiant à utiliser le protocole REST

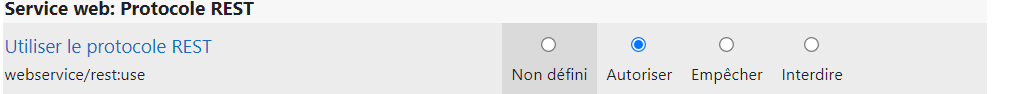
ICI Administration du site -> Utilisateurs -> Permissions ->Définition des rôles puis cliquer sur modifier pour descendre jusqu’à la partie Filtre et chercher « webservice/rest:use » et autoriser ce service 

Figure Autorisation d’utiliser le protocole REST pour le groupe ETUDIANT

Puis il faut donner la capacité à l’utilisateur de créer des tokens en leur autorisant dans cette même interface le service moodle/webservice:createtoken.

A ce stade-là, en théorie, un utilisateur peut faire des requêtes REST.

Il faut aussi activer l’api de Moodle, qui est appelé web service dans l’environnement Moodle, pour cela allez dans Tableau de bord -> Administration du site -> Serveur -> Services web -> Activer les services web et activer les services web.

A ce stade, nous voulons activer certains services pour rendre un rendu de devoir sur Moodle. En effet, il faut ajouter à un service, les fonctions. (donc les fonctionnalités) nécessaires. La liste de tous les noms des fonctions est disponible ici <https://docs.moodle.org/dev/Web_service_API_functions>)

Nous avons besoin de 5 fonctions :

* core\_webservice\_get\_site\_info : Pour récupérer userId de l’utilisateur
* core\_enrol\_get\_users\_courses : Pour avoir la liste des courseId de tous les cours d’un userId
* mod\_assign\_get\_assignments : Pour récupérer tous les rendus de devoir d’un courseID
* core\_files\_upload : Pour upload un fichier dans la zone personnelle de l’utilisateur
* mod\_assign\_save\_submission : Pour sauvegarder un nouveau fichier pour un devoir dans un cour

La documentation détaillée de ces services n’est pas disponible sur Internet, mais dans l’interface administration de Moodle ( <http://127.0.0.1/admin/webservice/documentation.php> ) et attention, il faut attendre quelques secondes avant que tout s’affiche !

Pour rendre un devoir, il faut d’abord passer par l’espace de stockage personnel Moodle de l’utilisateur. Puis une fois dans cet espace, Moodle attribue un id à ce fichier, donc pour rendre cette archive en devoir, il suiffe de préciser l’identifiant du fichier.

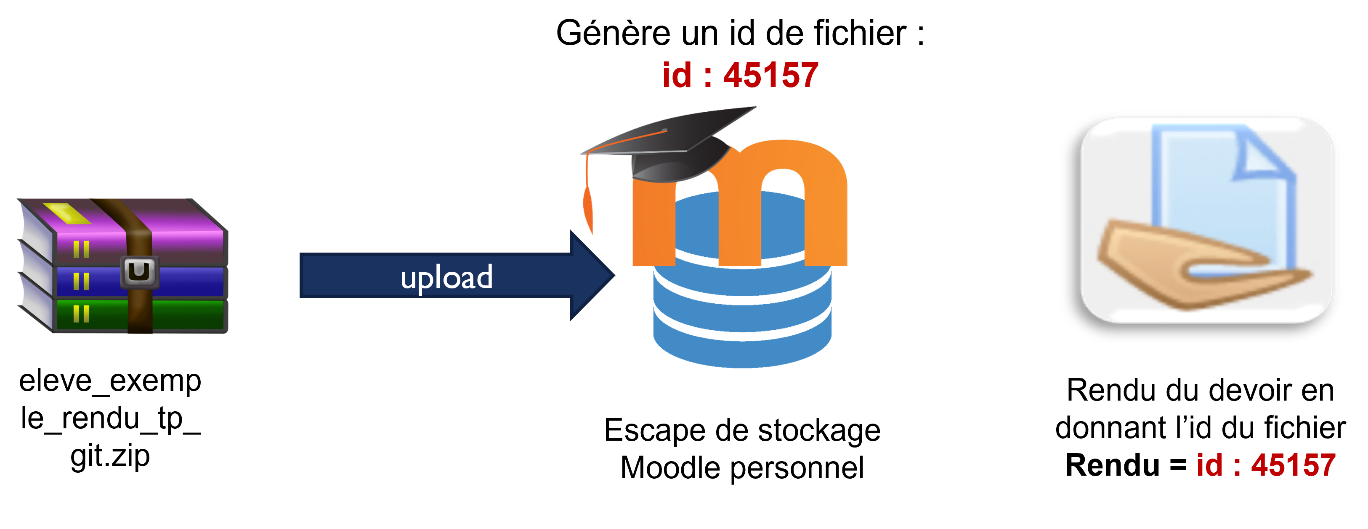


Figure Rendu de fichier

Il existe un service qui permet d’upolad un fichier dans l’espace personnel, mais ce service est inutilisable, car en argument, il faut donner le fichier encodé en base64 dans une requête http. Donc, naturellement, avec un fichier de plusieurs Mo la requête est trop longue pour être lue par le serveur. Il faut donc utiliser une petite astuce, en passant par une requête curl.

Pour utiliser, ces servies il faut créer un service Tableau de bord.

Administration du site -> Serveur -> Services web ->Services externes puis ajouter en donnant bien le droit de dépôt et de téléchargement (comme dans l’image ci-dessous)

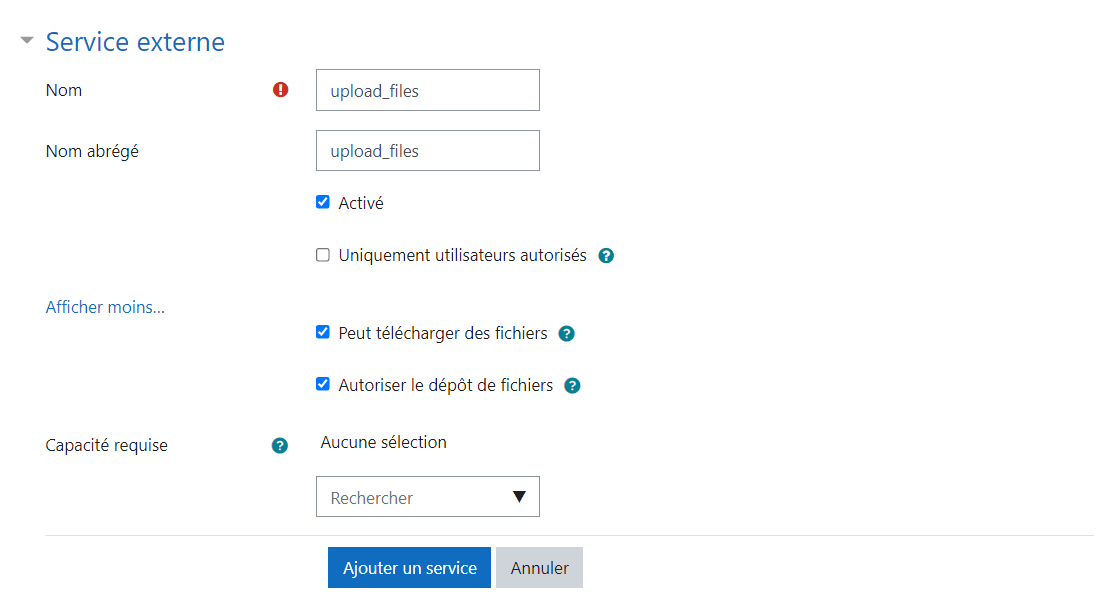


Figure Création de la fonction

Puis ajouter les fonctions lister plus haut

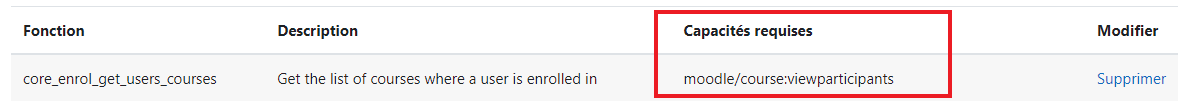
Une fois ajouter, il faut ajouter toutes les capacités aux groupes d’utilisateurs requises par exemple pour pouvoir utiliser la fonctions « core\_enrol\_get\_users\_courses » il faut que le groupe de cet utilisateur ait la capacité moodle/course:viewparticipants .

Figure Exemple de capacités requise

Après tout cela, vous pouvez enfin utiliser l’api de Moodle et utiliser script.py pour upload des fichiers (après avoir créé un cours et inscrit l’utilisateur à ce cours).

## Problème avec la vraie infrastructure de Moodle de Télécom SudParis

Tous les tests, on était réalisé avec un Moodle en local, cependant pour des questions de sécurité (comme ne pas donner les droits administrateurs à deux étudiants) nous n’avons pas pu faire de test en condition réel.

Dans un 1er temps le système authentification de <https://moodle.imtbs-tsp.eu/> pour les étudiants ne fonctionne pas avec un couple nom d’utilisateur moodle/ mot de passe moodle. En effet pour nous connecter nous utilisons un intermédiaire CAS (service central d’authentification) et donc le couple id/mdp de CAS n’ai pas le même que celui de moodle. Une synchronisation est faite entre ces deux bases de données régulièrement, mais tout cela est transparent pour un utilisateur et donc nous n’avons pas accès à ces identifiants pour obtenir un token.

Puis dans un 2nd temps en théorie, il faudrait créer un groupe pour les utilisateurs utilisant l’api REST et ajouter à la main tous les utilisateurs participants à des TP utilisant Labtainers. Mais tout cela est très long à faire à la main et sans avoir accès au droit administrateur (et donc à l’architecture interne du Moodle de TSP) nous ne pouvons pas développer de script pour automatiser cela (et nous ne savons pas si cela est possible.).

Il existe un protocole standardisé pour communiqué entre des LMS (comme Moodle) et des applications externes (dans notre cas Labtainers). Ce protocole est appelé LTI (Learning Tools Interoperability) .

Donc nous avons pensé à utiliser l’LTI de Moodle pour simplifier tout cela. Mais malheureusement nous n’avons pas eu le temps de d’étudier tout cela en détail, mais nous allons tout de même détailler le concept dans les parties suivantes.

## Infrastructure

### Infrastructure locale

Donc, finalement, notre infrastructure est la suivante :

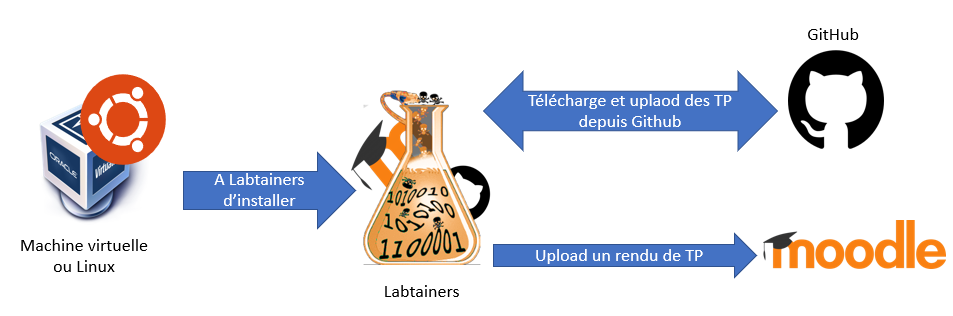


Figure Infrastructure locale

Nous avons soit une VM Linux, soit directement Linux avec Labtainers d’installé. Puis tous les TP sont stockés dans un GitHub sous forme de fichier léger (pas des images docker comme le projet d’origine). Labtainers peut télécharger les TP depuis GitHub et les enseignants peuvent en plus push des TP sur ce même GitHub. Et enfin, les étudiants peuvent rendre un rendu de TP directement en tapant une ligne de commande avec le script scirpt.py.

### Infrastructure cloud

Nous avons une infrastructure locale, cependant, il serait intéressant de tout déployer dans le cloud afin que ce soit le plus facile d’utilisation pour l’utilisateur.

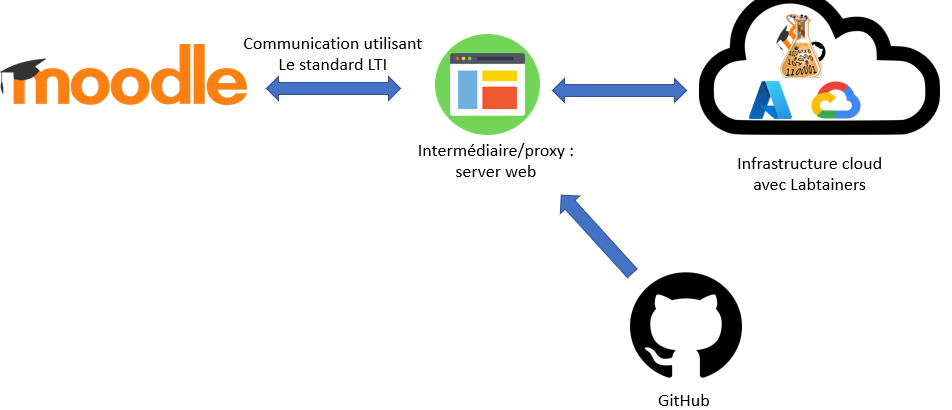


Figure 11 Infrastructure cloud

Cette infrastructure est un concept, nous n’avons pas de solution développée pour cela. Nous allons tout de même décrire cette infrastructure cloud.

L’idée est que l’étudiant puisse lancer son TP avec juste un lien sur la page du cours. Puis le TP se lancerait dans une interface web.

Pour cela tous les composant de cette infrastructure passerait par un intermédiaire/proxy/serveur web. Cet intermédiaire pourrait être utilisé pour enregistrer et rediriger toutes les communications afin par exemple que l’enseignant puisse suivre chaque élève sur son TP.

Un exemple : un étudiant se connecte sur Moodle pour faire un TP, il clique sur le lien du TP, ce lien est en réalité un lien pour faire passer les informations de l’utilisateur Moodle au serveur web, en utilisant le standard LTI. Avec ces informations le serveur web lance un conteneur docker pour ce TP dans le cloud (avec les services de google cloud platform ou bien Azure par exemple). Les TP pourront être toujours stoker dans GitHub et Labtainers ne ferait plus un appel direct à GitHub, mais à ce serveur web pour récupérer les TP. Il aussi envisageable de retourner sur la solution du projet d’origine et de payer un abonnement pour Docker Hub, car les de toute façon les services cloud sont payant.

Les informations de l’élève seront stockées pour pouvoir être visualisé dans un Dashboard enseignant. Dans ce Dashboard, il serait possible de voir si un élève fait le TP(et s’il a fini) et d’avoir la liste des étudiants qui ont fait le TP.

Une fois le TP fini l’élève ferme son navigateur, et l’envoie du rendu est automatique.