**choix de l’application terminal du linux:**

Le choix de l' application du terminal linux repose sur la grande importance qu'elle offre aux étudiants de 1ere classe préparatoire au sein de l'école où les étudiants seront amenés à faire les tps en présentiel et qui seront notés par la suite et ça permet de couvrir toutes les commandes nécessaire pour effectuer les tps et donc peuvent exploiter le système sans avoir à installer des machines virtuelles et les configurer.

En général, les étudiants en première année ont tendance à utiliser windows 10, windows 11 et galèrent généralement avec l’installation et la configuration d’une machine virtuelle à l’aide d’un logiciel de virtualisation comme VirtualBox ou VMware, donc leur donner accès à un outil rapidement utilisable et exploitable sans avoir à perdre beaucoup de temps et passer directement à la pratique et reproduction des scénarios des TPs, du coté des enseignants, ceci leur permettra aussi de mettre à jour des installations de commandes ou des LAB préparé (par exemple)

**Mise en place:**

Tout d'abord pour créer le conteneur du terminal linux on aura besoin du fichier docker (dockerfile) dans lequel on définit l’image de base à l’aide de l’instruction FROM dans ce cas là l'image de base est ubuntu:18.04.

On utilise la commande ENV DEBIAN\_FRONTEND non interactive pour ne pas interagir avec le système lors de l’installation via apt-get c'est-à -dire qu’on choisit les réponses par défaut sans avoir à interrompre l’installation ou les mises à jour du système. c’est un mode parfait pour les installations automatiques.

La deuxième étape consiste à utiliser la commande RUN pour exécuter les commandes et installer les dépendances nécessaires à la création de notre conteneur. la création des images se fait à l’aide des couche avec chaque commande RUN il faut créer une autre couche donc l’idéal est de ne pas utiliser plusieurs commandes de type RUN dans le fichier docker

Puis la dernière étape consiste à utiliser la commande CMD pour définir la commande par défaut pour l’exécution de notre conteneur .

| Commande | Rôle | Exemple |
| --- | --- | --- |
| ADD | Définit les fichiers à copier du système de fichiers hôte vers le conteneur | ADD ./local/config.file /etc/service/config.file |
| CMD | Il s'agit de la commande qui s'exécutera au démarrage du conteneur | CMD [“nginx”, “-g”, “daemon off;”] |
| ENTRYPOINT | Définit l'application par défaut utilisée chaque fois qu'un conteneur est créé à partir de l'image. S'il est utilisé conjointement avec CMD, vous pouvez supprimer l'application et simplement y définir les arguments | CMD Hello World!  ENTRYPOINT echo |
| ENV | Définissez/modifiez les variables d'environnement dans les conteneurs créés à partir de l'image. | ENV VERSION 1.0 |
| EXPOSE | Définir les ports de conteneur à exposer | EXPOSE 80 |
| FROM | Sélectionnez l'image de base pour créer la nouvelle image par-dessus | FROM ubuntu:latest |
| LABEL maintainer | Champ facultatif pour vous permettre de vous identifier en tant que mainteneur de cette image. Ceci n'est qu'une étiquette | LABEL maintainer=someone@xyz.xyz” |
| RUN | Spécifiez les commandes pour apporter des modifications à votre image, puis aux conteneurs démarrés à partir de cette image. Cela inclut la mise à jour des packages, l'installation de logiciels, l'ajout d'utilisateurs, la création d'une base de données initiale, la configuration de certificats, etc. Ce sont les commandes que vous exécuterez sur la ligne de commande pour installer et configurer votre application. C'est l'une des directives dockerfile les plus importantes. | RUN apt-get update && apt-get upgrade -y && apt-get install -y nginx && rm -rf /var/lib/apt/lists/\* |
| USER | Définissez l'utilisateur par défaut, toutes les commandes seront exécutées dans n'importe quel conteneur créé à partir de votre image. Il peut s'agir d'un UID ou d'un nom d'utilisateur | USER docker |
| VOLUME | Créer un point de montage dans le conteneur le reliant aux systèmes de fichiers accessibles par l'hôte Docker. Les nouveaux volumes sont remplis avec le contenu préexistant de l'emplacement spécifié dans l'image. Il est particulièrement pertinent de mentionner que la définition de volumes dans un Dockerfile peut entraîner des problèmes. | VOLUME /var/log |
| WORKDIR | Définir le répertoire de travail par défaut définie au niveau de ‘ENTRYPOINT’ ou ‘CMD’ | WORKDIR /home |

**configuration et test**

pour les commandes utilisées pour créer et exécuter le conteneur :

docker build -t bash/dockerized-bash . :pour la création de l’image chaque commande dans le dockerfile crée une couche ce qui réduit le temps de mis à jour car si on ajoute une commande une ajoute une couche sans avoir à modifier le reste, le point indique que le dockerfile est présent dans le répertoire courant

docker run -it --rm bash/dockerized-bash :commande pour le démarrage du conteneur.

**Exploiter l’image bash**

1. **Première méthode : partage des fichiers de configurations**

Les étudiants recevront des fichiers de configuration en particulier le Dockerfile.yaml par biais d’un dossier drive partagé ou mail, ensuite utiliser les commandes nécessaires au build de l’image. Par contre ceci nécessite de guider les étudiants en leur envoyant les commandes à taper.

1. **Deuxième méthode : partage des images directement grâce à Docker Hub**

Pour se ramener à une solution permettant l’étudiant d’exploiter directement une image docker sans soucier de la partie build, l’enseignant pourra concevoir l’image elle-même et la mettre au niveau de docker hub (le site permettant de partager des projets et des images), par la suite un étudiant n’aura plus qu'à pull cette dernière et la lancer dans un conteneur.

Commandes pour mettre une image sur docker-hub (enseignant):

* Vérifier les images : docker images
* S’authentifier sur docker hub (au niveau du terminal) : docker login
* taguer l’image avec un nom : docker tag image\_name:version repo\_name/name\_image:version //si version est laissée vide alors elle va être latest par défaut
* docker push name\_repo/image\_name

Commandes pour télécharger l’image depuis docker-hub (étudiant):

* docker pull repo\_name/image\_name
* docker run -t -d container\_name repo\_name/image\_name

Et c’est bon ! l’étudiant pourra ainsi travailler dans le conteneur.

Pour sauvegarder les changements et rendre l’image persistante, il suffit de faire un “docker commit”, comme indiqué dans la partie test et résultats du rapport.