# Packet tracer :

## Justification du choix de l’application :

Packet Tracer est un logiciel de CISCO, utilisé dans les TPs de 1CS et 2CS, permettant de construire un réseau physique virtuel et de simuler le comportement des différents protocoles sur ce réseau. L’étudiant construit sa topologie à l’aide d’équipements tels que les routeurs, les commutateurs ou des ordinateurs. Ces équipements doivent ensuite être reliés via des connexions (câbles divers, fibre optique). Une fois l’ensemble des équipements reliés, il est possible de configurer le réseau selon ce qui est demandé en TP.

Pendant l’année dernière et cette année, nous avons rencontré plusieurs problèmes, par exemple pour le projet réseau de l’année passée, les membres d’une même équipe peuvent avoir des versions différentes qui mènent à une perte de temps considérable. Aussi pendant le contrôle intermédiaire du module réseaux avancés de cette année, le même problème s’est reproduit pour quelques étudiants et par conséquent ils n’ont pas pu passer leur examen dans les meilleures conditions. De plus, les enseignants reçoivent les TPs par mail ce qui rend la procédure de correction plus difficile et ils peuvent même rater un livrable ou plus. D’où la nécessité de conteneuriser ce logiciel.

## Problèmes et défis rencontrés :

Pendant la conteneurisation de Packet Tracer nous avons rencontrés plusieurs difficultés :

* Les dépendances :

Le problème majeur que nous avons rencontré et qui nous a fait perdre énormément de temps est de trouver les packages nécessaires pour l’installation et démarrage corrects de Packet Tracer

* Les interactions lors de la création de l’image :

Pendant la création de l’image de packet tracer 8, le processus se bloque en attendant une entrée du clavier (la configuration du fuseau horaire et du clavier …) qui ne peut pas être faite pendant la création de l’image,donc l’image ne peut pas être créée.

## Étapes de la mise en place :

* Decomposition:

Packet Tracer est un logiciel complexe qui a de nombreuses dépendances. D’abord, nous devons vérifier les dépendances nécessaires et les installer. Pour notre image (Version 8.1.1 de Cisco Packet Tracer) la commande suivante doit être exécutée pour les récupérer :

**sudo dpkg-deb -I PacketTracer\_8.1.1\_amd64.deb**

* Sélection d’une image de base:

Il est inutile de réinventer la roue à chaque migration. [Docker nous donne la possibilité de commencer à partir d’une image de base, dans notre cas on a choisis une image de base Ubuntu 20.04](https://www.lemagit.fr/actualites/2240238846/Docker-Google-ajoute-un-registry-prive) qui est la version minimale.

* Ajout du code

Pour créer l’image, nous devons définir les étapes nécessaires à la construction de l’image dans un fichier Dockerfile. Nous commençons par définir l’image de base choisie (Ubuntu :20.04) avec la commande **FROM**. Ensuite, nous installons les dépendances avec les commandes que nous utiliserons lors de la création de notre image avec la commande **RUN**. Puis, nous créons l’utilisateur de notre conteneur (student). Par la suite, nous copions le fichier «.deb » de packet tracer dans l’image avec la commande **COPY** et nous installons Packet tracer. Finalement nous avons spécifié le nom d’utilisateur et le répertoire Home et la commande à exécuter lors du démarrage du conteneur.

Une fois l’image créée, nous pouvons l’ajouter au référentiel Docker Hub pour qu’elle puisse être utilisée par d’autres personnes.

* Configuration, test

Nous allons Configurer l’application conteneurisée : elle doit pouvoir se connecter automatiquement aux ressources externes telle que l’écran et les dossiers partagés, et pour cela nous utilisons le fichier docker-compose.yml pour partager la variable d'environnement DISPLAY de l'hôte avec le conteneur, partager le XServer de l'hôte avec le conteneur en créant un volume, Exécuter le conteneur avec le pilote de réseau hôte et pour les dossiers partagés nous créons un volume pour chacun d'entre eux.

Enfin, Le test de cette application se fait en exécutant un des TP du module réseaux avancés 2CS étudié dans 2CS (SIQ) pour garantir le bon fonctionnement des conteneurs pendant les séances TPs.

Pour qu'une application graphique fonctionne, nous devons disposer d'un XServer disponible dans chaque environnement de bureau Linux, mais dans un conteneur, nous n'avons aucun XServer, nous allons donc exécuter les commandes suivantes :

* Partager le XServer de l'hôte avec le conteneur en créant un volume

--volume="$HOME/.Xauthority:/root/.Xauthority:rw"

* Partager la variable d'environnement DISPLAY de l'hôte avec le conteneur

--env="AFFICHER"

* Exécuter le conteneur avec le pilote de réseau hôte avec

--net=hôte