# Résultats des tests:

Les tableaux suivants décrivent le scénario de test établi, avec les commandes utilisées et les résultats obtenus, qui vérifie le bon fonctionnement des applications conteneurisée:

1. Linux Terminal:

| Test | Result | Explication |
| --- | --- | --- |
| Créer l’image |  | On commence par la commande: docker build  qui crée l’image décrite par le dockerfile.  On utilise l’option: -t pour spécifier le nom de l’image, dans notre cas c’est: dockernized-bash.  Et on spécifie le chemin vers le dockerfile qui se trouve dans le répertoire courant représenté par le “dot” .  Le résultat de cette commande et l'exécution des commandes qui se trouve dans le dockerfile et la création de l’image. |
| Lister les images créées |  | Pour vérifier que l'image a été créée, on utilise la commande: docker images qui liste les images sur le système.  Lorsque l’image est bien créée elle apparaît dans la liste des images comme montré dans la capture. |
| Exécuter le conteneur |  | Pour démarrer le conteneur on utilise la commande:  docker run  On spécifie le nom d'image: dockernized-bash  Avec les options:  -t : pour allouer un pseudo-terminal.  -i: pour démarrer le conteneur en mode interactif qui permet d'accéder au terminal du conteneur en cours d'exécution.  Le résultat de cette commande qu’on on peut voir sur la capture ce qu’on ouvre un terminal sur le système de conteneur, et qu’on est sur le répertoire root “/”, maintenant on peut exécuter 'n'importe quelle commande.  on a exécuter apt-get update pour obtenir les informations sur les packages à installer ou mettre à jour. Et qui est recommandée pour pouvoir installer des des packages par la suite. |
| Tester la commande ping à l'intérieur du conteneur |  | Sur le conteneur on installer la commande: ping , qu’on l’utilise pour vérifier la connexion internet on pingant:  *google.com* |
| Executer un script à l'intérieur du conteneur |  | Sur le conteneur on crée un script appelé “*script*” qui exécute une commande echo.  On peut configurer les permission d'accès, et rendre le fichier exécutable, avec la commande: chmod  Le script s'exécute avec succès. |
| Afficher la liste des conteneurs créés |  | Pour lister les conteneurs créer on utilise la commande:  docker ps avec l’option:  -a: pour afficher tous les conteneurs (même qui ne sont pas en cours d'exécution) |

1. Packet Tracer:

| Test | Result | Explication |
| --- | --- | --- |
| création de l'image |  | On commence par la commande: docker-compose build  qui crée l’image en utilisant le fichier **docker-compose.yml**  Le résultat de cette commande est l'exécution des commandes du **dockerfile** qui se trouve dans le chemin indiqué par le champ build de **docker-compose**, et la création de l’image. |
| création du conteneur |  | On utilise la commande docker-compose up  pour créer le conteneur et démarrer les services définie dans le fichier **docker-compose.yml,** dans le mode attaché ou attaché mode (ajouter l’option -d pour le mode détache ou detached mode ) |
| démarrer packet tracer |  | Pour démarrer le conteneur on peut utiliser la commande:  docker start,  en spécifiant le nom de conteneur: packettracer\_app\_1  Le résultat de cette commande est le démarrage de l’application Packet Tracer comme on peut voir dans la capture. |
| Test TP |  | Sur l’application conteneurisé Packet Tracer on ouvre le fichier de TP (fichier .pkt) qui se trouve dans le volume:  **/home/student/Tps**  Le fichier de TP s'ouvre avec succès, et on peut voir que les liaisons entre les éléments de la topologie fonctionnent correctement |