**Introduction:**

Les tests sont une partie essentielle du développement de logiciels modernes. Les tests peuvent signifier beaucoup de choses pour différentes équipes de développement. Il existe des tests unitaires, des tests d'intégration et des tests de bout en bout. Dans notre cas, nous examinons comment exécuter des tests unitaires dans Docker.

Dans ce qui suit nous reprenons les commandes utilisées pour la mise en place des images tout en détaillant les scénarios et plans de test .

**Commandes de mise en place du DockerFile :**

Syntaxe générale d’une commande : “ command argument argument 1…"

Exemple:

# Print "Get Certified. Get Ahead"

Run echo "Get Certified. Get Ahead"

Avant de créer notre premier Dockerfile, il est important de comprendre ce qui compose le fichier. Dockerfile se compose de commandes spécifiques qui vous guident sur la façon de créer une image Docker spécifique.

Les commandes spécifiques que vous pouvez utiliser dans un dockerfile sont :

FROM, PULL, RUN et CMD

| **Commande** | **Effet** |
| --- | --- |
| FROM os:version | Créer une couche à partir de l’OS hote |
| PULL | Ajouter des fichiers à partir du repo Docker |
| RUN | Build le conteneur |
| CMD | Spécifier quelles commandes exécuter dans le conteneur |
| ENTRYPOINT application "arg, arg1". | Permet de spécifier une commande avec ses paramètres |
| ADD /[source]/[destination] | Permet de copier des données vers l’image Docker |
| ENV key value | Fournit les valeurs par défaut pour les variables dans le conteneur |
| MAINTAINER [name] | Déclarer l’auteur des images docker |
| docker build [OPTIONS] PATH | URL | - | Déclarer le chemin où stocker le DockerFile |
| docker build [-t] tag [location of your dockerfile] | Construire une image basique à partir d’un DockerFile, l’option -t permet de tagguer une image avec un nom |
| docker images | Vérifier la création des images docker |
| docker run --name nom\_conteneur nom\_image | Permet de créer un conteneur Docker à partir d’une image Docker |

