# TRAVAUX DIRIGÉS
# ORCHESTRATEUR - DOCKER SWARM -STACK

## I. INTRODUCTION

Dans le TD précédent, nous avons aborder la notion d'orchestrateur sur un unique service. À présent, nous allons utiliser l'orchestrateur pour gérer une stack docker, c'est à dire un ensemble de service oeuvrant ensemble pour déployer une application complète.

## II. ENVIRONNEMENT

Nous allons travailler sur le système Debian des salles du plateau technique. Vous sauvegardez donc tous vos fichiers sur un projet Github afin de pouvoir redéployer rapidement vos infrastructures si nous changeons de salles

Vous utiliserez vagrant pour déployer un ensemble de 3 VM, un master et 2 worker

## III. CRÉATION DE LA PREMIÈRE STACK

Cette première stack va utiliser déployer 2 services simples d'API django la première qui renvoie un texte disant "bonjour" suivi du nom passé à la suite de l'URI et la seconde qui renvoie un texte disant "au revoir" suivi du nom passé à la suite de l'URI Nous allons lancer

créer les Dockerfile permettant de concevoir ces 2 API. La création d'une stack passe simplement par l'écriture d'un fichier docker-compose décrivant les services.

```
version: "3.7"
services:
   api1:
      build : "./api1"
      deploy :
            replicas: 3
            ressources:
                limits:
                   memory: 50M
                   cpu:1
             restart-policy:
                condition: on-failure
      ports :
         - "8000:8000"
   api2:
      build : "./api2"
      deploy :
            replicas: 3
            ressources:
                limits:
                   memory: 50M
                   cpu:1
             restart-policy:
                condition: on-failure
      ports :
         - "8001:8000"
```

il faut ensuite exécuter la commande

```
docker stack deploy -c docker-compose.yml monapp
```

Les commandes de **docker stack** sont identiques à celle de **docker service**, vous avez accès au ps, ls, inspect,

## IV. utilisation des secrets dans une stack

Aprés avoir créer un secret *mysql_root_password*, dans le docker compose, quand vous positionnez des variables d'environnement pour les configurations de vos services, vous allez utiliser le *_FILE à ajouter à la fin de la variable d'environnement et vous indiquez comme valeur le fichier (avec son path, par défaut "/run/secrets/" dans le container) et en ajoutant dans le docker compose un item

```
environnement:
  MYSQL_PASSWORD_FILE : /run/secrets/#nom_secret
secrets:
    #nom_secret:
      external : true
```

## V. utilisation des configs dans une stack

Après avoir créer une config, vous pouvez l'utiliser dans le docker compose de votre stack.

```
config:
  source: #nom_config
  target: path/nom_fichier
  mode: 0440 #les permissions
```

## VI. exercices

VI.1. **créer une stack mettant en oeuvre un serveur nginx, un serveur de base de données.**

VI.2. **modifier votre stack pour utilise un secret pour le mot de passe root et le mot de passe utilisateur.**

VI.3. **créer une stack avec un serveur apache, une application django simple (mais 2 instances) et une base de données vous utiliserez les secrets pour les mots de passe.**

VI.4. **modifier la stack pour que le fichier de config django soit fourni au travers d'une configuration.**