# AI Frameworks. TP - Google Cloud.

Brendan Guillouet

29 Janvier 2018

## Contents



## Introduction

C

Au cours de ce TP nous allons voir les différentes étapes permettant d'exécuter un code python sur un serveur distant à l'aide de *Google Cloud* et de *Docker* afin d'utiliser des puissances de calcul supérieur.

Nous allons utiliser pour cela les données du TP *Cats Vs Dogs*.

Les différentes étapes :

- Créer un script fonctionnel en local
- Configurer le serveur
- Pousser le code et les données sur le serveur
- Exécuter le code
- Récupérer les résultats

Dans ce TP nous allons écrire deux scripts python permettant :

1. Envoyer les données de *CatsVsDogs* sur le serveur,
2. nnvoyer le code qui sera exécuté sur ces données.
3. Récupérer les résultats et les différentes observations.

# 1 Script *python.*

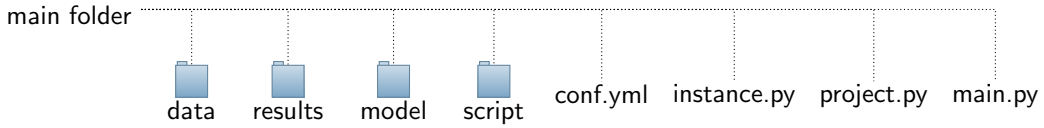
L'utilisation de serveur distant peut s'avérer coûteux. En effet les sociétés telles que *Google*, *Amazon* ou *Microsoft* facturent l'utilisation de leur machine à l'heure.

Il est alors très vivement déconseillé d'effectuer le travail d'exploration des données directement sur ces machines. Même si cela est possible (cf Section 6), l'utilisation d'outil tel que *Jupyter*, n'est pas adapté. Une bonne pratique vise donc à effectuer le travail d'exploration des données sur une machine en locale et sur un jeu de données réduit si cela est nécessaire. On écrit ensuite un script, toujours en local, permettant d'effectuer toutes les opérations désirées et de produire les résultats souhaitées. Enfin on pousse et on exécute ce script sur la machine distante.

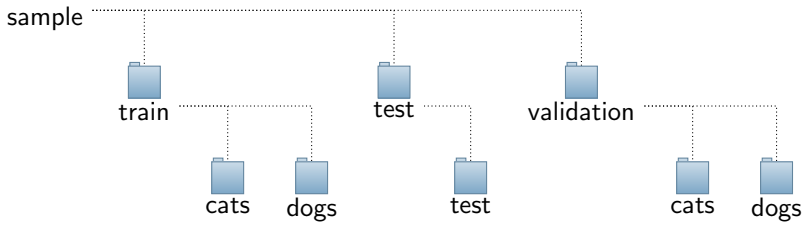
## 1.1 Créer l'arborescence

La première étape consiste à créer, en local, l'arborescence qui sera reproduite sur le serveur. Il est très important de définir rigoureusement cette organisation.

Voici celle-qui sera utilisée dans ce TP :




- **data** : Ce dossier va contenir les données sur lesquelles votre algorithme sera exécutée. De façon plus général, il doit contenir toutes les données nécessaire à l'exécution de votre code. Dans ce dossier data, on placera un dossier *sample*, qui contient les données *train*, *test* et *validation* organisées de la sorte :




- **model** : Les différents modèles seront stockés dans ce dossier. De façon plus général, il doit contenir toutes les données qui doivent être sauvegardées au moins temporairement, mais que l'on ne souhaite pas récupérer sur notre machine en local en fin d'éxécution.

- **results** : Ce dossier va contenir différentes informations et statistiques que votre script va générer. De façon plus général, il doit contenir toutes les données que vous souhaitez récupérer pour les explorer en local.
- **script** : Code python permettant d'effectuer l'apprentissage et de générer des prédictions.
- **conf.yml, instances.py, project.py, main.py** : Code python permettant d'utiliser, et d'éxécuter les codes du scripts sur le serveur (Section 3.2 et 4.5.)

Dans le dossier *GoogleCloud* du répertoire git *AI-Frameworks*, cette organisation est déjà partiellement recréer. Les dossiers *metadata*, *model* et *script* sont manquants. Ajoutez les.
Détailler l'organisation de data

## 1.2 Écriture des scripts

Nous allons maintenant écrire les deux scripts qui nous permettront d'atteindre nos objectifs :

- **learning.py** : Dans ce script nous allons définir et entraîner un nouveau modèle. Nous allons le sauvegarder dans le dossier *model* et sauvegarder les différents résultats que nous souhaitons étudier en fin d'exécutions dans le dossier *results*.
- **prediction.py** : Dans ce script nous allons télécharger le modèle entraîné précédemment et l'utiliser pour effectuer la prédiction sur les données tests et générer le fichier *.csv* contenant ces réponses.

Avant de commencer l'écriture de ses scripts, nous rappelons quelques bonne pratiques :

### 1.2.1 Argument en paramètre

L'exécution de vos scripts dans un terminal s'effectue de façon très simple et de la façon suivante :

```
python learning.py
```

Cependant votre script doit être assez souple pour prendre en compte différents paramètres au moment de l'éxecution. Ceux-ci peuvent-être classés en deux types :

- Paramètres du modèles (liste non exhaustive) : Nombre d'epochs, taille du batch, Nombre de neurones, architecture, etc...
- Paramètre de l'environnement : Ces paramètres changeront en fonction de la machine sur lequel le code va tourner: direction des données, du modèle, etc..

On utilisera la librairie **argparse** (*cf slide*) de python pour gérer ce type d'argument.

**NB:** Une autre possibilité consiste à boucler sur les paramètres à l'intérieur du script.

### 1.2.2 Résultats à sauvegarder

Il est très important de penser à l'avance quels serons les résultats et informations que vous souhaitez récupérer en fin d'exécution de votre code afin de pouvoir les analyser. Si vous avez oublié de sauvegarder certaines infos, celles-ci seront perdue en fin d'éxecution de scripts.

Ainsi avant d'envoyer votre script sur le cloud, veillez a ce que toutes les données et informations que vous aimeriez pouvoir étudier une fois l'apprentissage terminée soit sauvegarder dans le dossier results.

Vous pouvez sauvegarder ces informations dans un *dictionnaire* python à l'aide de la librairie **pickle** (*cf slide*).

### 1.2.3 Exercice

Dans le dossier *GoogleCloud/script*, ouvrez les fichiers *learning.py* et *prediction.py*. Complétez-les en suivant les consignes écrites dans chacun de ces fichiers afin qu'ils exécutent les objectifs énoncés section 1.2.

Assurez vous de pouvoir faire tourner ces deux scripts sur votre machines (sur l'échantillon sample) en ayant sauvegarder le modèle, la prédiction et les résultats dans les dossiers désirées.

**NB:** Une solutions à ces exercices sont disponibles dans les scripts *learning_solution.py* et *prediction_solution.py*. Il est vivement conseillé de ne pas regarder ces solutions dans un premier temps.

# 2 Configuration d'une machine *Google Cloud*

.

Aidez-vous des impressions écrans présentes dans les slides du cours pour plus de précisions.

## 2.1 Créer un projet

- Connectez-vous sur Google Cloud https://console.cloud.google.com/ à l'aide de votre adresse mail INSA. Vous arrivez sur la page d'accueil.
- Dans la barre supérieur, vous avez déjà un projet en cours. C'est le projet par défaut. (Vous pouvez utiliser directement ce projet par la suite.)
- Cliquez sur le nom de ce projet. Une fenêtre *Sélectionner un projet* s'ouvre.
- Dans cette fenêtre, cliquez sur *Nouveau Projet*.
- Rentrez un nom de projet puis choisissez le compte sur lequel il sera facturé. Le compte correspond au coupon que vous avez entrez précédemment. Cliquez sur *créer*
- Vous pouvez maintenant sélectionner votre projet dans la barre principale.

L'initialisation d'un projet peut prendre du temps. Vous pouvez donc continuer le TP sur le projet présent par défaut.

## 2.2 Configurer une machine

- Dans le menu principal (Cliquez sur l'*Hambuger*), dans le sous menu *Compute Engine* cliquez sur la section *instance de VM*.
- Cliquez alors sur l'onglet *Créer une instance*. Vous êtes alors sur le menu de configuration de votre machine :
  - Choisissez un nom et sélectionnez *europe-west1* comme région.
  - Dans le menu **Type de machine**, cliquez sur *Personnaliser*,sélectionner 4 coeurs CPU et 1 GPU de type Tesla K80. *Vous pourrez sélectionnez des cartes plus puissantes plus tard.*
  - Dans le menu **Disque de Démarrage** sélectionnez *Ubuntu 16.04 LTS*. Dans ce menu vous pouvez également augmenter la taille du disque de démarrage, configurez le à une taille de 50Go.
  - Dans le menu **Pare-feu**, cochez les cases *Autoriser le trafic HTTP* et *Autoriser le trafic HTTPS*.
  - Cliquez sur créer.

**Attention** : Votre VM démarre automatiquement à sa création.

### 2.3 Configuration de gcloud

*gcloud* est un outil de Google Cloud, permettant de gérer une instance depuis votre terminale. C'est à dire que vous pouvez envoyer des fichiers sur cette machine, des commandes, *etc.* directement depuis votre machine en local.

Un peu comme avec *git*, vous devez dans un premier temps, "initialisez" votre compte gcloud pour que celui-ci soit connecter avec votre compte Google Cloud.

- suivre les indications de https://cloud.google.com/sdk/docs/quickstart-macos de la partie *Initialize the SDK*
- Connectez-vous sur votre machine à l'aide de la commande suivante :

```
gcloud compute ssh NomDeVotreMachine
```

## 3 Exécuter le code

### 3.1 Installation de la machine

Vous êtes maintenant connectez sur votre machine. Celle-ci est presque entièrement vierge. Il vous faut donc installer les différents outils et logiciels nécessaire à l'exécution de votre code et notamment *Python3* et *Cuda* (pour installer les version GPU de Tensorflow).

Dans le dossier *tp_google_cloud/bash_util_on_gpu* vous trouverez différents scripts permettant l'installation de ces différents outils. Depuis votre terminal en local, utilisez la commande *gcloud compute scp* afin d'envoyer ce dossier sur votre machine distante.

```
gcloud compute scp --recurse --zone europe-west1-b bash_util_on_gpu/
instance-1:~ --ssh-key-file ACOMPLETER/.ssh/google_compute_engine
```

Une fois ce dossier envoyé, éxecuter ces différents scripts permettant l'installation de :

- Cuda,

```
sh bash_util_on_gpu/bash_cuda_install.sh
```

- Python3 et autre utilitaires.

```
sh bash_util_on_gpu/bash_python3.sh
```

**NB:** Ces scripts contiennent différentes commandes qui peuvent très facilement être retrouvés sur internet, sur les pages officiel de ces différents outils. N'hésitez pas à les parcourir.

Votre machine est maintenant prête a executer vos scripts.

### 3.2 Nuage Magix

Afin d'exécuter vos scripts sur votre machine distante vous devez effectuez les différentes actions suivantes : Envoyer la dernières version de votre code sur la machine, possiblement mettre à jour vos données, exécuter vos différents scripts avec les différents paramètres souhaités, ramener sur votre terminal, en local, les différents résultats à analyser, pensez à éteindre votres machine *etc...* Ces opérations ne sont pas compliquées, mais elles peuvent vite entraînez des erreurs de manipulations (oublie d'upate de vos codes, écrasement d'ancien fichier, oublie d'éteinte de machine, *etc...*), et représenter un travail fastidieux.

Afin d'éviter ce travail à chaque nouveau test, on va alors utiliser l'outil *Nuage Magix*. Cet outils est composé de quatre fichiers :

- **conf.yml**: Ce fichier fonctionne comme un dictionnaire et sert à stocker des variables globales tels que les directions des différents dossiers.
- **instances.py**: Ce script python définie une classe *InstanceManager* qui contient différentes fonction qui englobe des appels à la fonction gcloud. Par exemple la fonction *list()* de cette classe exécute implicitement la commande suivante sur le terminal :

  ```
  gcloud compute instances list
  ```

  Ce script permet donc de gérer votre machine distante directement depuis votre code python.
- **project.py** Ce script python définie une classe *ProjectManager* qui contient différentes fonction permettant de gérer votre projet.

  Par exemple, la fonction *update_data(self, zip_file)* va permettre, à l'aide des fonctions de la classe *InstanceManager*, d'envoyer un fichier *zip_file* depuis votre dossier *data* en local vers le dossier *data* sur votre machine distante (les directions de ces dossiers ayant été définie en amont dans le fichier *conf.yml*.
- **main.py** Ce script python va contenir les appels au différentes fonctions de la classe *ProjectManager* afin de mener à bien votre projet.

### 3.3 Exercice

- Compléter le fichier **conf.yml** en indiquant les directions des différents dossier dans votre arborescence en local, et en indiquant les directions des dossiers sur la machine distante telle que vous souhaitez les voir apparaître.
- Passez un peu de temps à explorer le fichier *main.py* et à comprendre l'utilité de chacune des fonctions appelées. Complétez les arguments indiqués *"ACOMPLETER"*
- Exécuter le script *main.py* et interpréter les différentes sorties du terminal.
- A l'aide de jupyter, en local, télécharger les résultats de votre exécution à l'aide de pickle.
- Affichez l'évolutions de la *loss* en fonction du nombre d'*epochs* à l'aide de *matplotlib*.

## 4 Docker

Dans cette partie nous allons voir comment exécuter notre code dans un container *Docker* sur notre machine distante.

### 4.1 Installation de la machine

Dans un premier temps il faut installer les logiciel *Docker* et *Nvidia-Docker* à l'aide des scripts bash contenu dans le dossier *bash_util_on_gpu* :

- docker

  ```
  sh bash_util_on_gpu/bash_docker_install.sh
  ```
- nvidia-docker

  ```
  sh bash_util_on_gpu/bash_nvidia_docker.sh
  ```

## 4.2 Dockerfile

Dans le dossier *tp_google_cloud/Docker* vous trouverez différents Dockerfile permettant de construire des images docker. Depuis votre terminal en local, utilisez la commande *gcloud compute scp* afin d'envoyer ce dossier sur votre machine distante :

```
gcloud compute scp --recurse --zone europe-west1-b utils/Docker/  instance-2:~
--ssh-key-file ACOMPLETER/.ssh/google_compute_engine
```

## 4.3 image

Pour info checker le docker file utilisé et le docker file de tensorflow officiel

```
sudo nvidia-docker build -t tfimg -f ~/Docker/Dockerfile ~/Docker/
```

```
sudo nvidia-docker image ls
```

COmbien y a t-il d'image?

## 4.4 container

### 4.4.1 container interactif

On se connecte de façon interactive grace a l'argument it

```
sudo nvidia-docker run -it --name container_it tfimg
```

Ouvrir un terminal python. Vérifier que tensorflow est installé et qu'il a accès au gpu.

```
from tensorflow.python.client import device_lib
MODE = "GPU" if "GPU" in [k.device_type for k in device_lib.list_local_devices()
```

Problème : Nous n'avons pas accès au donnée!
On quitte le container avec *ctrl+d*.
Lister les container à l'aide de la commande suivante :

```
sudo nvidia-docker container ls
```

Qu'observez vous? Réiterer la commande en y ajoutant l'option *-a*. En quittant un container, celui-ci se ferme automatiquement. Pour ré-utilser le container en mode interactif, vous devez le re-démarrer, puis l'attacher à l'aide des commande suivantes:

```
sudo nvidia-docker start container_it
sudo nvidia-docker attach container_it
```

Quitter de nouveau le container puis supprimer le définitivement.

```
sudo nvidia-docker rm container_it
```

#### 4.4.2 Mountained Volume

L'argument -v permet d'avoir accès au données

```
sudo nvidia-docker run -it --name container_it_v -v
ACOMPLETER/CatsVsDogs/:/root/CatsVsDogs tfimg
```

Un dossier CatsVsDogs est maintenant disponible dans votre container. Que contient-il? Créer un dossier "test" dans le dossier CatsVsDogs au sein de votre container. Quitter le container. Ouvre le dossier CatsVsDogs sur le serveur. Qu'observez vous?

#### 4.4.3 background mode

```
sudo nvidia-docker run -t -d --name container_background
-v ~/CatsVsDogs/:/root/CatsVsDogs tfimg
```

```
sudo nvidia-docker container ls
```

```
sudo nvidia-docker stop container_background
sudo nvidia-docker rm container_background
```

### 4.5 Nuage Magix

## 5 Bonus 1 : Jupyter sur un serveur distant

## 6 Bonus 2 : Google Colab

```
git config --global user.name USERNAME
git config --global user.email USERMAIL
```