AI Frameworks. TP - Google Cloud.

Brendan Guillouet

3 décembre 2018

Contents

Introduction

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 2 scripts python fonctionnel en local permettant d'apprendre un modède de DeepLearning et d'effectuer des prédictions à partir de ce modèle.
- Configurer une instance *Google Cloud*
- Envoyer les données de *CatsVsDogs* et les script pythons sur le serveur,
- Exécuter le code
- Récupérer les résultats et les différentes observations.
- Réiterer ce processur à l'aide de container *Docker*.

Tout au long de ce TP, nous ferons références au slide du cours. Assurez-vous de l'avoir sur votre machine.

1 Script *python.*

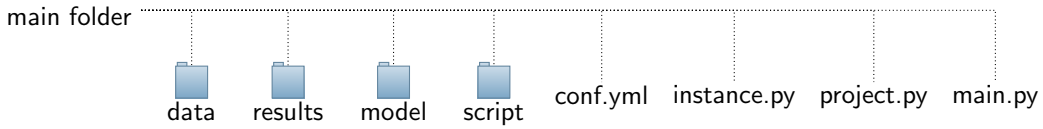
L'utilisation de serveur distant peut s'avérer coûteux. En effet les sociétés telles que *Google*, *Amazon* ou *Microsoft* facturent l'utilisation de leur machine à l'heure.

Il est alors très vivement déconseillé d'effectuer le travail d'exploration des données directement sur ces machines. Même si cela est possible, 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.

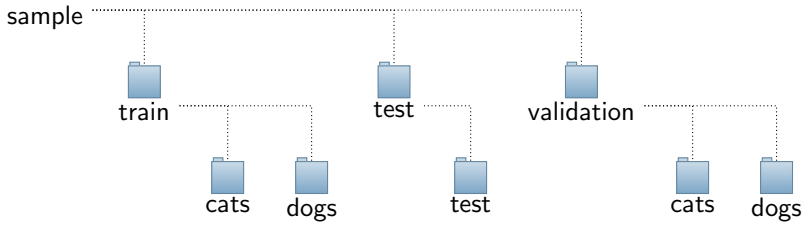
1.1 Créer l'arborescence

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.

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 machine (sur l'échantillon sample) en ayant sauvegarder le modèle, la prédiction et les résultats dans les dossiers désirés.

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 *GoogleCloud/utils/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 ACOMPLETER/bash_util_on_gpu/
instance-1:~ --ssh-key-file ACOMPLETER/.ssh/google_compute_engine
```

Une fois ce dossier envoyé, exécuter 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. Par exemple, lancer l'exécution d'un code python sur la machine distante se traduit par la commande suivante :

```
gcloud compute ssh nom_instance --zone europe-west-1 --ssh-key-file DirKeyFile
    --command 'python3 learning.py --data_dir DirData -- results_dir DirResults
                                    --epochs 10 --batch_size 128'
```

Afin d'éviter ce travail à chaque nouveau test, on va alors utiliser l'outil *Nuage Magix*.

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 outils *Docker* et *Nvidia-Docker* à l'aide des scripts bash contenu dans le dossier *bash_util_on_gpu*. Exécuter ces différents scripts de la manière suivante ;

- 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 *GoogleCloud/utils/Docker* vous trouverez différents Dockerfile permettant de construire des images docker.

Exercice: Ouvrez le fichier *Dockerfile* et décrivez très rapidement la fonction des trois lignes qui le compose.

Un fichier *Dockerfile.devel-gpu* est également présent dans le dossier. S'il ne correspond pas à l'image directement utilisé par notre *Dockerfile*, celui-ci est très similaire. Vous pouvez le parcourir pour avoir une idée des installations effectués.

Depuis votre terminal en local, utilisez la commande *gcloud compute scp* afin d'envoyer ce dossier sur votre machine distante afin de pouvoir les utiliser.

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

4.3 Image

Nous allons maintenant construire une image docker à partir du Dockerfile en notre possession. Pour cela, exécutez la commande suivante dans le terminal de la machine distance en complétant par les argument nécessaires.

```
sudo nvidia-docker build -t ACOMPLETER -f ACOMPLETER ACOMPLETER
```

Une fois l'image construite (cela peut prendre un moment la première fois!). Executez la commande suivante, qui vous permet de lister les images disponible sur votre machine.

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

Question Combien y a t-il d'image? A quoi correspondent elles?

4.4 Container

Vous avez maintenant construit votre propre *image*. La prochaine étape consiste donc à lancer un *container* à partir de cette image dans lequel vous allez pouvoir effectuez vos opération et calcul.

4.4.1 container interactif

Exercice: Complétez la commande suivante afin de lancer un container de façon interactive et sans volume.

```
sudo nvidia-docker run ACOMPLETER --name ACOMPLETER ACOMPLETER
```

Vous êtes maintenant à l'intérieur d'un container docker. Ouvrez une console python. Vérifier que *tensorflow* est installé et que c'est bien la version GPU qui est installée à l'aide de la commande suivante.

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

Quittez la console python. Tous nos outils sont en place excepté le fait que le container docker n'a, pour le moment, pas accès aux données du TP! Avant de régler ce problème (Section 4.4.2), nous allons voir comment quitter et relancer un container:

- Quittez le container avec la commande *ctrl+d*.
- Lister les container à l'aide de la commande suivante :

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

  Qu'observez vous? Réitérer la commande en y ajoutant l'option *-a*. Qu'observez-vous?

- En quittant un container, celui-ci se ferme automatiquement. Pour ré-utiliser le container en mode interactif, vous devez le re-démarrer

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

- Une fois redémarré, vous devez vous *attacher* à ce container :

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

- Quittez de nouveau le container puis supprimer le définitivement :

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

4.4.2 Mounted Volume

Dans cette partie nous allons voir comment "monter" un volume dans le container. C'est à dire rendre disponible dans le container, un dossier situé sur la machine. L'argument *-v* permet cette opération.

Exercice: Complétez la commande suivante afin de lancer un container, de façon interactive, qui aura accès au dossier *CatsVSDogs/sample*. On notera que, par défaut, le container s'ouvre dans le dossier *root*

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

Constatez que Les données sont maintenant disponibles dans votre container. Créez un dossier *tmp* dans le dossier *CatsVSDogs/sample* au sein de votre container. Quittez le container. Ouvrez le dossier *CatsVSDogs/sample* sur le serveur. Qu'observez vous?

Avec cette manipulation, vous êtes maintenant capable de lancer un container, de lancer des opérations au sein de ce container et de sauvegarder des résultats auxquels vous aurez accès sur votre serveur.

4.4.3 Background Mode

La dernière étape consiste a lancer un container en mode *background*. En effet, si le mode *interactif* est utile pour développer, comprendre le fonctionnement du container. Il est utile de pouvoir lancer des calculs sur le container sans avoir à garder un terminal ouvert.

Exercice: Complétez la commande suivante afin de lancer un container en background et qui aura accès au dossier *CatsVSDogs/sample*.

```
sudo nvidia-docker run ACOMPLETER --name ACOMPLETER
-v ACOMPLETER
```

Votre container est lancé en mode *background*. Vérifiez-le en exécutant la commande suivante :

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

Qu'observez-vous?

Maintenant que vous savez que le container est correctement lancé et en train de tourner. Vous pouvez lancer des exécutions au sein de ce container à l'aide de la commande *exec*. Pour commencer, listez les fichiers présents dans le dossier /root/ de votre container à l'aide de la commande suivante:

```
sudo nvidia-docker exec cn ls
```

Exercice: Dans votre container qui tourne en *background*, envoyez, à l'aide de la commande *exec*, une commande qui permettra de créer un dossier dans le volume que vous avez monté dans le container. Vérifiez sur votre machine que ce dossier est bien accessible. Une fois ces manipulations terminées, stoppez puis supprimez ce container, et vérifiez que la suppression est bien effective.

4.5 Nuage Magix

L'utilisation de container docker à l'aide de l'outil gcloud est très simple. En effet, au moment de l'appel à la fonction *gcloud compute ssh* il suffit d'ajouter l'option *–container* suivi du nom du container que l'on souhaite utilisez pour que le calcul soit lancé dans un container plutôt que directement sur la machine.

Pour reprendre l'exemple de la Section 3.2, la commande correspondante permettant de faire tourner le code dans le container *NameContainer* est la suivante :

```
gcloud compute ssh nom_instance --zone europe-west-1 --ssh-key-file DirKeyFile
    --container NameContainer
    --command 'python3 learning.py --data_dir DirData -- results_dir DirResults
                                    --epochs 10 --batch_size 128'
```

Ainsi pour adapter le fichier *main.py* de *NuageMagix* afin que celui-ci utilise le container il suffit d'ajouter les fonctions permettant les étapes suivantes :

- Lancer un container en background,
- Lancer les codes en ajoutant l'option *–container*,
- Stopper puis supprimer le container.

Le fichier *main_docker.py* a été écrit afin d'adapter ces différentes étapes.

4.6 Exercice

- Passez un peu de temps à explorer le fichier *main_docker.py* et à comprendre les différences avec le fichier *main.py* utilisé précédemment. Repérez les nouvelles fonctions et les nouveaux arguments utilisés. Complétez les arguments indiqués *"ACOMPLETER"*
- Exécuter le script *main_docker.py* et interpréter les différentes sorties du terminal.

5 Application

Appliquez ce que vous venez d'apprendre pour le défi IA.