



Pratique de Hadoop n°01

Start Hadoop

Table des matières

I. Objectif du TP	2
II. Hadoop et Docker.....	2
III. Installation et configuration de l'image Docker :	2
IV. Description de la configuration :	4
A. Avant-propos :.....	4
B. Configuration SSH.....	4
C. Fichiers de configuration hadoop :.....	6
1. Core-site.xml :.....	7
2. hdfs-site.xml :.....	8
3. mapred-site.xml :	8
D. Liens officiels des configurations des fichiers :.....	8
V. Interfaces web pour Hadoop.....	9

I. Objectif du TP

- Initiation au framework hadoop
- utilisation de docker
- Lancer un cluster hadoop de 3 noeuds.

II. Hadoop et Docker

Pour déployer le Framework Hadoop, nous allons utiliser des conteneurs Docker. L'utilisation des conteneurs va garantir la consistance entre les environnements de développement et permettra de réduire considérablement la complexité de configuration des machines (dans le cas d'un accès natif) ainsi que la lourdeur d'exécution (si on opte pour l'utilisation d'une machine virtuelle).

III. Installation et configuration de l'image Docker :

Nous allons utiliser tout au long de ce TP trois conteneurs représentant respectivement :

- un noeud maître (Namenode)
- deux noeuds esclaves (Datanodes)

Vous devez pour cela avoir installé docker sur votre machine, et l'avoir correctement configuré.

Ouvrir la ligne de commande, et taper les instructions suivantes:

1. Télécharger l'image docker uploadée sur dockerhub:
docker pull liliastaxi/spark-hadoop:hv-2.7.2
2. Créer les trois conteneurs à partir de l'image téléchargée. Pour cela: 2.1.
Créer un réseau qui permettra de relier les trois conteneurs:
docker network create --driver=bridge hadoop

4. Créer et lancer les trois conteneurs (les instructions -p permettent de faire un mapping entre les ports de la machine hôte et ceux du conteneur):

```
docker run -itd --net=hadoop -p 9070:50070 -p 8088:8088 -p 7077:7077 \  
-p 16010:16010 \  
--name hadoop-master --hostname hadoop-master \  
liliasfaxi/spark-hadoop:hv-2.7.2
```

```
docker run -itd -p 8040:8042 --net=hadoop \  
--name hadoop-slave1 --hostname hadoop-slave1 \  
liliasfaxi/spark-hadoop:hv-2.7.2
```

```
docker run -itd -p 8041:8042 --net=hadoop \  
--name hadoop-slave2 --hostname hadoop-slave2 \  
liliasfaxi/spark-hadoop:hv-2.7.2
```

5. Entrer dans le conteneur master pour commencer à l'utiliser.

```
docker exec -it hadoop-master bash
```

Le résultat de cette exécution sera le suivant:

```
root@hadoop-master:~#
```

Vous vous retrouverez dans le shell du namenode, et vous pourrez ainsi manipuler le cluster à votre guise. La première chose à faire, une fois dans le conteneur, est de lancer Hadoop et Yarn. Un script est fourni pour cela, appelé start-hadoop.sh.

Lancer ce script.

./start-hadoop.sh

IV. Description de la configuration :

A. Avant-propos :

Ce chapitre est une explication très technique (avec des commandes linux).

C'est pour votre connaissance informatique.

B. Configuration SSH

Hadoop nécessite un accès SSH pour gérer les différents nœuds. Bien que nous soyons dans une configuration simple nœud, nous avons besoin de configurer l'accès vers localhost pour l'utilisateur **hduser** que nous venons de créer précédemment.



Avant tout, nous devons générer une clé SSH pour l'utilisateur hduser.

```
$ ssh-keygen -t rsa -P ""
```

Generating public/private rsa key pair.

Enter file in which to save the key (/home/hduser/.ssh/id_rsa):

Created directory '/home/hduser/.ssh'.

Your identification has been saved in /home/hduser/.ssh/id_rsa.

Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.

The key fingerprint is:

80:36:b8:8d:04:32:d2:d8:ef:02:ff:01:a4:c5:63 hduser@precise64

The key's randomart image is:

```
+--[ RSA 2048 ]-----+
```

```
|=*          |
```

```
|=oE. .     |
```

```
| *oo+ .    |
```

```
|o..=.. .   |
```

```
|ooo. S     |
```

```
| o o       |
```

```
| o .       |
```

```
| .         |
```

```
|           |
```

```
+-----+
```

Cette commande va créer une clé **RSA** avec un mot de passe vide. Dans notre cas de virtualisation, l'absence de mot de passe n'a pas d'importance. Assurez-vous d'en fixer un si votre serveur est accessible depuis l'extérieur.

Vous devez ensuite autoriser l'accès au SSH de la machine avec cette nouvelle clé fraîchement créée.

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

La dernière chose à réaliser est de tester la connexion SSH à partir de l'utilisateur `hduser`.

```
$ ssh localhost
```

```
The authenticity of host 'localhost (127.0.0.1)' can't be established.
```

```
ECDSA key fingerprint is 11:5d:55:29:8a:77:d8:08:b4:00:9b:a3:61:93:fe:e5.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
```

```
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)
```

```
* Documentation: https://help.ubuntu.com/
```

```
Welcome to your Vagrant-built virtual machine.
```

```
Last login: Sat Dec 14 20:19:33 2013 from 127.0.0.1
```

```
hduser@precise64:~$
```

C. Fichiers de configuration hadoop :

Tout d'abord :

Tous les fichiers de configuration d'Hadoop sont disponibles dans le répertoire `/etc/hadoop/conf`.

Les fichiers de configuration d'Hadoop fonctionnent sur le principe de clé/valeur : la clé correspondant au nom du paramètre et valeur à la valeur assignée à ce paramètre. Ces fichiers de configuration utilisent le format XML. Les nouveaux paramètres sont à ajouter entre la balise `<configuration> ... </configuration>`.

Je ne peux être exhaustif sur les modifications à apporter sur ces fichiers de configuration. Je me limiterai donc aux paramètres de base pour exécuter un cluster Hadoop d'un nœud. Pour plus d'informations sur les paramètres autorisés, je vous invite à consulter les liens que je donnerai pour chaque fichier modifié.

1. Core-site.xml :

Exemple :

Depuis le fichier `/etc/hadoop/conf/core-site.xml` modifier le contenu afin d'obtenir le résultat ci-dessous :

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
    <description>The name of the default file
system.</description>
  </property>
</configuration>
```

La propriété **fs.defaultFS** (avant : `fs.default.name`) permet de spécifier le nom du système de fichier. Ainsi tous les répertoires et fichiers HDFS sont préfixés par **hdfs://localhost:9000**.

Précision :

La propriété `hadoop.tmp.dir` pointe par défaut sur le répertoire `/tmp/hadoop-{user.name}`. Vous retrouvez donc dans ce répertoire (`/tmp/hadoop-{user.name}`) tous les sous-répertoires nécessaires au stockage des données pour **Hadoop (Namenode, Datanode...)**.

2. **hdfs-site.xml** :

Le fichier **/etc/hadoop/conf/hdfs-site.xml** contient les paramètres spécifiques au système de fichiers HDFS.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
  </property>
</configuration>
```

Le paramètre **dfs.replication** permet de préciser le nombre de réplication d'un block. La valeur sera 1 puisque notre cluster ne se compose que d'un nœud. Finalement le paramètre **dfs.webhdfs.enabled** permet d'activer le service Web REST HDFS. ([cf les interfaces web Hadoop](#))

3. **mapred-site.xml** :

Le fichier **/etc/hadoop/conf/mapred-site.xml** contient les paramètres spécifiques à MapReduce.

Depuis la version 2.x d'Hadoop avec l'arrivée de **Yarn**, ce fichier de configuration est épaulé par **yarn-site.xml**. Ainsi, si vous souhaitez utiliser **Yarn** comme implémentation de MapReduce, il faudra configurer le fichier **mapred-site.xml** comme présenté ci-dessous.

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

D. Liens officiels des configurations des fichiers :

[core-site.xml](#)
[hdfs-default.xml](#)
[mapred-site.xml](#)
[yarn-default.xml](#)

V. Interfaces web pour Hadoop

Comme vous le savez : Hadoop offre plusieurs interfaces web pour pouvoir observer le comportement de ses différentes composantes. Vous pouvez afficher ces pages en local sur votre machine grâce à l'option `-p` de la commande **docker run**. En effet, cette option permet de publier un port du conteneur sur la machine hôte. Pour pouvoir publier tous les ports exposés, vous pouvez lancer votre conteneur en utilisant l'option `-P`.

En regardant le contenu du fichier **start-container.sh** fourni dans le projet, vous verrez que deux ports de la machine maître ont été exposés:

- Le port 9070: qui permet d'afficher les informations de votre namenode.
- Le port 8088: qui permet d'afficher les informations du Resource Manager de Yarn et visualiser le comportement des différents jobs.

Une fois votre cluster lancé et prêt à l'emploi, vous pouvez, sur votre navigateur préféré de votre machine hôte, aller à : `http://localhost:9070`. Vous obtiendrez le résultat suivant:

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Overview 'hadoop-master:9000' (active)

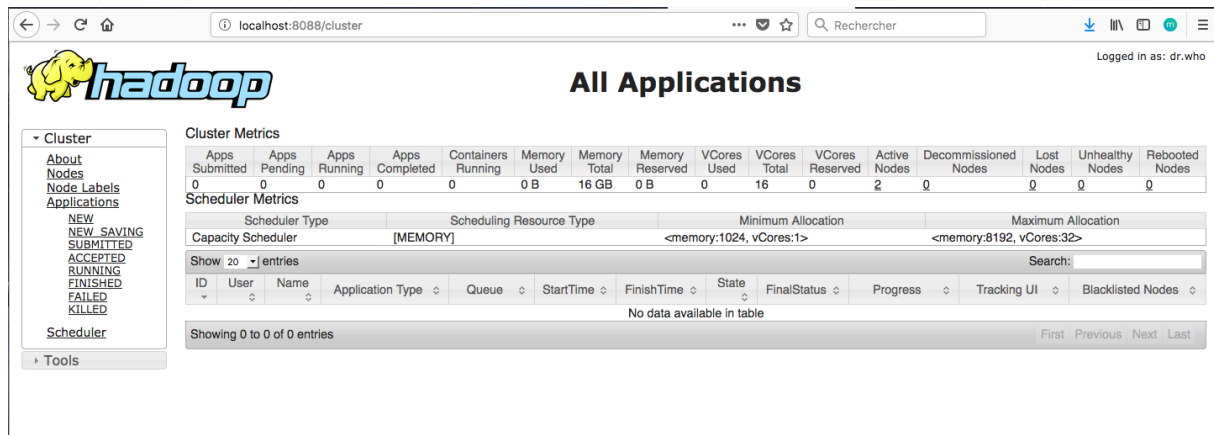
Started:	Fri Jan 26 11:47:09 UTC 2018
Version:	2.7.2, rUnknown
Compiled:	2016-05-27T18:05Z by root from Unknown
Cluster ID:	CID-3c662456-d44e-4301-bc39-28e479c4dc88
Block Pool ID:	BP-431089505-172.17.0.2-1465730089024

Summary

Security is off.
Safemode is off.
20 files and directories, 8 blocks = 28 total filesystem object(s).
Heap Memory used 85.33 MB of 165.5 MB Heap Memory. Max Heap Memory is 889 MB.
Non Heap Memory used 37.42 MB of 38.44 MB Committed Non Heap Memory. Max Non Heap Memory is 214 MB.

Configured Capacity:	125.49 GB
DFS Used:	406.85 MB (0.32%)
Non DFS Used:	34.1 GB

Vous pouvez également visualiser l'avancement et les résultats de vos Jobs (Map Reduce ou autre) en allant à l'adresse: <http://localhost:8088>



The screenshot shows the Hadoop YARN web interface at localhost:8088/cluster. The page title is "All Applications". On the left, there is a sidebar with a "Cluster" menu containing links for "About", "Nodes", "Node Labels", "Applications", and "Scheduler". The "Applications" link is selected. The main content area displays "Cluster Metrics" and "Scheduler Metrics".

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
0	0	0	0	0	0 B	16 GB	0 B	0	16	0	2	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:32>

Below the scheduler metrics, there is a table for applications. The table has columns: ID, User, Name, Application Type, Queue, StartTime, FinishTime, State, FinalStatus, Progress, Tracking UI, and Blacklisted Nodes. The table is currently empty, showing "No data available in table".