Atelier : HDFS avec Python

L'objectif :

- Se familiariser avec le stockage des données dans HDFS
- Manipuler, Copier, transférer, supprimer les données dans HDFS

Emplacement du fichier : /formation/ateliers/hdfs/

Réalisation : Vous allez copier, supprimer plusieurs fichiers de données dans HDFS

Chapitre correspondant : HadoopDistributed File System (HDFS)

1- Installer hdfs3 :
   a- Cd /home/cloudera/miniconda3/bin
   b- ./conda install hdfs3

```
[cloudera@quickstart bin]$ ./conda install hdfs3
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/cloudera/miniconda3

  added / updated specs:
    - hdfs3


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    bzip2-1.0.8                |       h7b6447c_0          78 KB
    hdfs3-0.3.1                |           py37_0          42 KB
    icu-58.2                   |       h9c2bf20_1        10.3 MB
    libboost-1.67.0            |       h46d08c1_4        13.0 MB
    libcurl-7.65.3             |       h20c2e04_0         431 KB
    libgcrypt-1.8.4            |       h7b6447c_0         550 KB
    libgpg-error-1.32          |       hf484d3e_0         219 KB
    libgsasl-1.8.0             |       h7b6447c_3         119 KB
    libhdfs3-2.3.0             |       h2fca0e8_1         6.7 MB
    libprotobuf-3.6.0          |       hdbcaa40_0         2.5 MB
    libssh2-1.8.2              |       h1ba5d50_0         226 KB
    libuuid-1.0.3              |       h1bed415_2          15 KB
    libxml2-2.9.9              |       hea5a465_1         1.6 MB
    ------------------------------------------------------------
                                           Total:        35.6 MB

The following NEW packages will be INSTALLED:
```

2- Utilisation de hdfs3 :

```
>>> from hdfs3 import HDFileSystem
>>> hdfs = HDFileSystem(host='localhost', port=8020)
>>> hdfs.ls('/')
>>> hdfs.put('toto.txt', '/user/data/toto1.txt')
>>> hdfs.cp('/repertoire1/toto1.txt', '/repertoire2/toto2.txt')
```

```
>>> with hdfs.open('/repertoire1/toto.txt') as f:
...     data = f.read(4)

>>> with hdfs.open('/user/data/file.csv.gz') as f:
...     df = pandas.read_csv(f, compression='gzip', nrows=1000)
```

```
with hdfs.open('/tmp/myfile.txt', 'wb') as f:
...     f.write(b'Hello, world!')
```

3- Count word

a. Dézziper le contenu du zip HDFS_data.zip et transférer le contenu dans hadoop : /data_in par exemple.
b. Écrivez un petit programme en python qui compte le nombre de mot présent dans tous les fichiers
c. Afficher le nombre de mot rejetés
d. Afficher les 10 mots les plus répétés .

```
[cloudera@quickstart data]$ ./word_count.py
Le nombre de mot qui se repete : 465
Le top 10 des mots les plus repetes
le mot  et  est repete  544  fois
le mot  sit  est repete  514  fois
le mot  ac  est repete  504  fois
le mot  in  est repete  477  fois
le mot  sed  est repete  452  fois
le mot  id  est repete  425  fois
le mot  eget  est repete  419  fois
le mot  ut  est repete  415  fois
le mot  quis  est repete  415  fois
le mot  vel  est repete  413  fois
```

4- API :

HDFileSystem([host, port, connect, …])	Connection to an HDFS namenode
HDFileSystem.cat(path)	Return contents of file
HDFileSystem.chmod(path, mode)	Change access control of given path
HDFileSystem.chown(path, owner, group)	Change owner/group
HDFileSystem.df()	Used/free disc space on the HDFS system
HDFileSystem.du(path[, total, deep])	Returns file sizes on a path.
HDFileSystem.exists(path)	Is there an entry at path?
HDFileSystem.get(hdfs_path, local_path[, …])	Copy HDFS file to local
HDFileSystem.getmerge(path, filename[, …])	Concat all files in path (a directory) to local output file
HDFileSystem.get_block_locations(path[, …])	Fetch physical locations of blocks
HDFileSystem.glob(path)	Get list of paths mathing glob-like pattern (i.e., with “*”s).
HDFileSystem.info(path)	File information (as a dict)
HDFileSystem.ls(path[, detail])	List files at path
HDFileSystem.mkdir(path)	Make directory at path
HDFileSystem.mv(path1, path2)	Move file at path1 to path2
HDFileSystem.open(path[, mode, replication, …])	Open a file for reading or writing
HDFileSystem.put(filename, path[, chunk, …])	Copy local file to path in HDFS
HDFileSystem.read_block(fn, offset, length)	Read a block of bytes from an HDFS file
HDFileSystem.rm(path[, recursive])	Use recursive for rm -r, i.e., delete directory and contents
HDFileSystem.set_replication(path, replication)	Instruct HDFS to set the replication for the given file.
HDFileSystem.tail(path[, size])	Return last bytes of file
HDFileSystem.touch(path)	Create zero-length file

HDFile(fs, path, mode[, replication, buff, …])	File on HDFS
HDFile.close()	Flush and close file, ensuring the data is readable
HDFile.flush()	Send buffer to the data-node; actual write may happen later
HDFile.info()	Filesystem metadata about this file
HDFile.read([length])	Read bytes from open file
HDFile.readlines()	Return all lines in a file as a list
HDFile.seek(offset[, from_what])	Set file read position.
HDFile.tell()	Get current byte location in a file
HDFile.write(data)	Write bytes to open file (which must be in w or a mode)