

HDFS

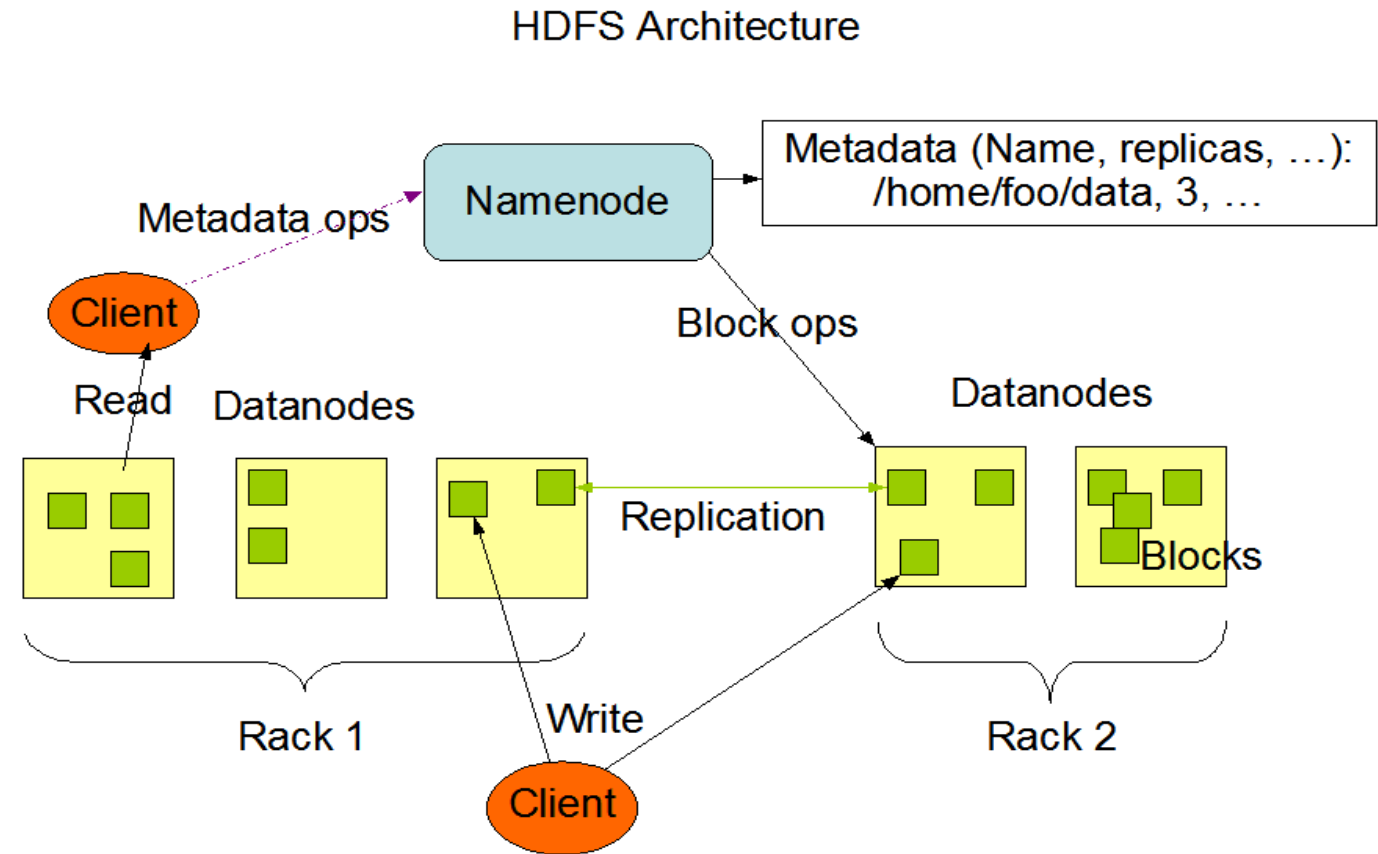
Fonctionnement
Outils
Utilisation
Programmation

Hadoop Distributed Filesystem: HDFS

- **Débit** : HDFS (Hadoop distributed file system) est un système de fichiers distribué tolérant aux pannes conçu pour s'exécuter sur du matériel de milieu de gamme. Il a été conçu spécialement pour la lecture par flux sur des fichiers volumineux. Le débit des I/O est favorisé au détriment de la latence.
- **Résilience** : HDFS est basé sur un système de duplication de bloc pour dupliquer les données sur les nœuds du cluster. La taille des blocs par défaut est de 64M (128 sur h2). Par défaut chaque bloc est copier trois fois sur le cluster.
- **Scalabilité** : HDFS a été spécialement conçu pour mettre l'ajout de machine à chaud ou le retrait de machine. Il permet ainsi de faire évoluer les moyens de stockage graduellement en fonction des demandes réelles.

HDFS: Architecture

- **Bloc** : chaque fichier est découper en blocs de tailles de fixes dupliquer n (3) fois.
- **DataNode** : Les datanodes stocke physiquement les blocs. La localisation des blocs est optimiser pour limiter les transferts réseaux. Machine, rack, datancenter.
- **Namenode** : Le namenode index tous les blocs (single point of failure). Il faut donc demander au namenode les accès en I/O sur les blocs.
- **Heartbeat** : Les datanode envoie à intervalle régulier un message au namenode. Si un datanode tombe le namenode réadapte la configuration des blocs pour garantir un accès à 3 blocs.



Quand utiliser HDFS :

- Très grands fichiers : 100 méga, giga, or tera, Il y a des clusters qui gèrent des Petabytes. Par exemple les images générées par les télescopes en très haute résolution qui ne tiennent pas sur une machine.
- Streaming data-access : L'idée derrière hadoop est que dans le domaine du « big data » les données sont écrites une seule fois et lues de nombreuses fois. De plus pour faire l'analyse il faut généralement relire l'intégralité des données. Le plus important est donc le débit et non la latence.
- Matériel milieu de gamme : Hadoop a été conçu pour fonctionner sur du matériel de milieu de gamme. C'est la multiplication des machine qui donnera la puissance du cluster non la puissance de chaque machine (a relativiser évidemment).

Quand ne pas utiliser HDFS :

- Faible latence, les applications qui ont besoin d'une faible latence (millisecondes), ne peuvent pas utiliser HDFS qui est beaucoup trop lent. Il faut se tourner vers d'autre système : par exemple Cassandra, ou HBASE.
- Plein de petits fichiers : HDFS n'est pas fait pour stocker beaucoup de petits fichiers. Le problème vient du namenode qui stocke toutes les meta-informations sur les fichiers. Chaque bloc utilise 150 octets, pour fonctionner correctement le namenode doit monter tout cela en mémoire, du coup il n'est pas possible d'avoir un nombre de fichiers proche du giga ($1500 * 1\text{giga} = 150\text{Go}$ de RAM).
- Ecritures multiples : Dans HDFS une seule écriture sur un fichiers est possible car on ne peut que ajouter des données à la fin d'un fichier. On ne peut pas modifier un fichier à un offset particulier.

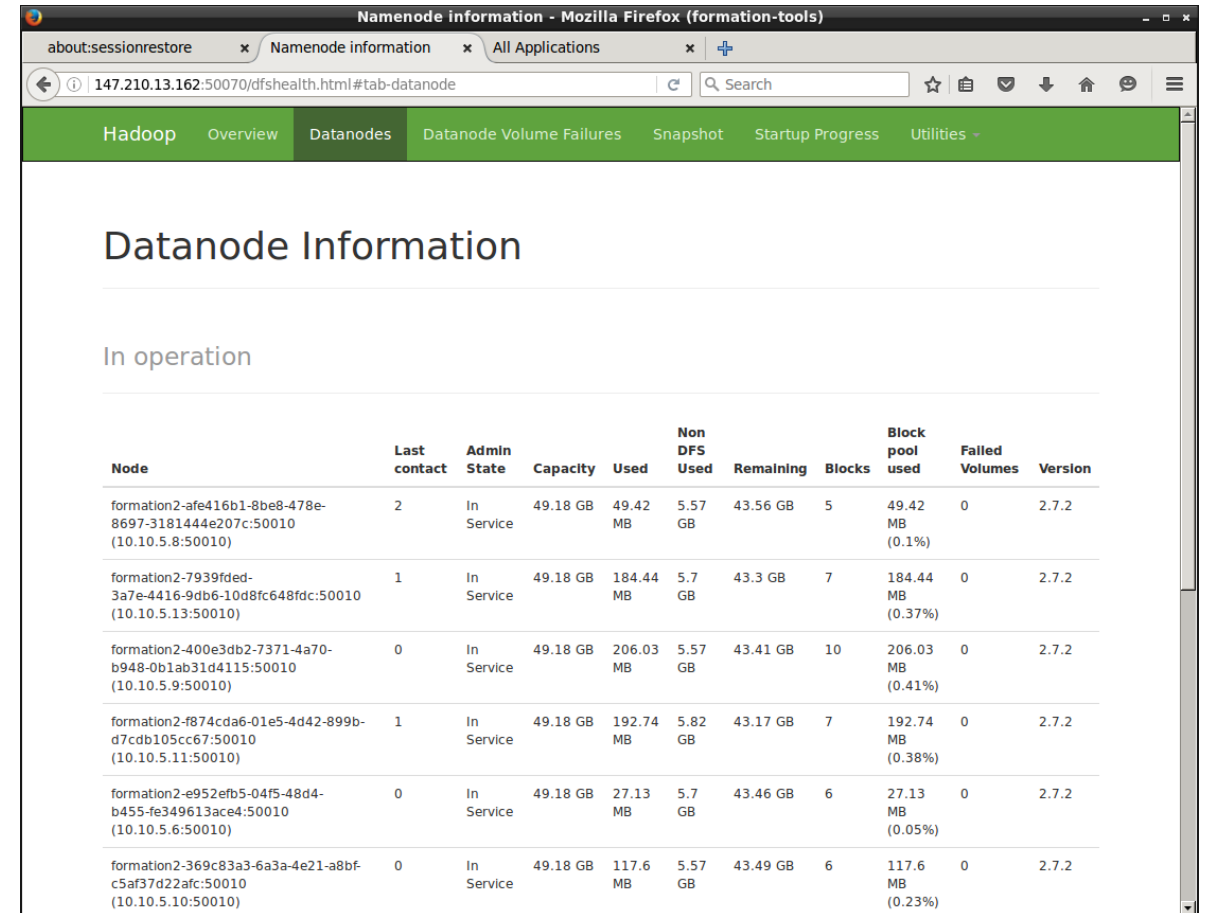
HDFS en pratique: WebUI

HDFS offre une interface web pour consulter l'ensemble des ressources disponibles et effectuer un minimum de monitoring

- Machine en marche/arrêt
- Nombre de blocs
- Espace disponible
- Snapshot
- Heartbeat

L'interface permet aussi d'explorer l'arborescence des fichiers stockés dans HDFS, de voir l'emplacement des blocs et de télécharger des fichiers.

WebUI 147.210.13.62:50070



The screenshot shows the Hadoop NameNode WebUI in a Mozilla Firefox browser. The page title is "Namenode information - Mozilla Firefox (formation-tools)". The browser address bar shows "147.210.13.62:50070/dfshealth.html#tab-datanode". The page has a green navigation bar with tabs: "Hadoop", "Overview", "Datanodes", "Datanode Volume Failures", "Snapshot", "Startup Progress", and "Utilities". The "Datanodes" tab is selected. The main content area is titled "Datanode Information" and shows "In operation". Below this is a table with 11 columns: Node, Last contact, Admin State, Capacity, Used, Non DFS Used, Remaining, Blocks, Block pool used, Failed Volumes, and Version. There are six rows of data, each representing a datanode in the cluster.

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
formation2-afe416b1-8be8-478e-8697-3181444e207c:50010 (10.10.5.8:50010)	2	In Service	49.18 GB	49.42 MB	5.57 GB	43.56 GB	5	49.42 MB (0.1%)	0	2.7.2
formation2-7939fded-3a7e-4416-9db6-10d8fc648fdc:50010 (10.10.5.13:50010)	1	In Service	49.18 GB	184.44 MB	5.7 GB	43.3 GB	7	184.44 MB (0.37%)	0	2.7.2
formation2-400e3db2-7371-4a70-b948-0b1ab31d4115:50010 (10.10.5.9:50010)	0	In Service	49.18 GB	206.03 MB	5.57 GB	43.41 GB	10	206.03 MB (0.41%)	0	2.7.2
formation2-f874cda6-01e5-4d42-899b-d7cdb105cc67:50010 (10.10.5.11:50010)	1	In Service	49.18 GB	192.74 MB	5.82 GB	43.17 GB	7	192.74 MB (0.38%)	0	2.7.2
formation2-e952efb5-04f5-48d4-b455-fe349613ace4:50010 (10.10.5.6:50010)	0	In Service	49.18 GB	27.13 MB	5.7 GB	43.46 GB	6	27.13 MB (0.05%)	0	2.7.2
formation2-369c83a3-6a3a-4e21-a8bf-c5af37d22afc:50010 (10.10.5.10:50010)	0	In Service	49.18 GB	117.6 MB	5.57 GB	43.49 GB	6	117.6 MB (0.23%)	0	2.7.2

HDFS en pratique: Ligne de commande

Usage: `hadoop fs [generic options]`

- `[-appendToFile <localsrc> ... <dst>]`
- `[-cat [-ignoreCrc] <src> ...]`
- `[-checksum <src> ...]`
- `[-chgrp [-R] GROUP PATH...]`
- `[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]`
- `[-chown [-R] [OWNER][:[GROUP]] PATH...]`
- `[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]`
- `[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]`
- `[-count [-q] [-h] <path> ...]`
- `[-cp [-f] [-p | -p[topax]] <src> ... <dst>]`
- `[-createSnapshot <snapshotDir> [<snapshotName>]]`
- `[-deleteSnapshot <snapshotDir> <snapshotName>]`
- `[-df [-h] [<path> ...]]`
- `[-du [-s] [-h] <path> ...]`
- `[-expunge]`
- `[-find <path> ... <expression> ...]`
- `[-getfacl [-R] <path>]`
- `[-getfattr [-R] {-n name | -d} [-e en] <path>]`
- `[-getmerge [-nl] <src> <localdst>]`

- `[-help [cmd ...]]`
- `[-ls [-d] [-h] [-R] [<path> ...]]`
- `[-mkdir [-p] <path> ...]`
- `[-moveFromLocal <localsrc> ... <dst>]`
- `[-moveToLocal <src> <localdst>]`
- `[-mv <src> ... <dst>]`
- `[-put [-f] [-p] [-l] <localsrc> ... <dst>]`
- `[-renameSnapshot <snapshotDir> <oldName> <newName>]`
- `[-rm [-f] [-r] [-R] [-skipTrash] <src> ...]`
- `[-rmdir [--ignore-fail-on-non-empty] <dir> ...]`
- `[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>}] [--set`
- `[-setfattr {-n name [-v value] | -x name} <path>]`
- `[-setrep [-R] [-w] <rep> <path> ...]`
- `[-stat [format] <path> ...]`
- `[-tail [-f] <file>]`
- `[-test -[defsz] <path>]`
- `[-text [-ignoreCrc] <src> ...]`
- `[-touchz <path> ...]`
- `[-truncate [-w] <length> <path> ...]`
- `[-usage [cmd ...]]`

HDFS en pratique: Ligne de commande

Usage: `hadoop fs [generic options]`

`[-appendToFile <localsrc> ... <dst>]`
`[-cat [-ignoreCrc] <src> ...]`
`[-checksum <src> ...]`
`[-chgrp [-R] GROUP PATH...]`
`[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]`
`[-chown [-R] [OWNER][:[GROUP]] PATH...]`
`[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]`
`[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]`
`[-count [-q] [-h] <path> ...]`
`[-cp [-f] [-p | -p[topax]] <src> ... <dst>]`
`[-createSnapshot <snapshotDir> [<snapshotName>]]`
`[-deleteSnapshot <snapshotDir> <snapshotName>]`
`[-df [-h] [<path> ...]]`
`[-du [-s] [-h] <path> ...]`
`[-expunge]`
`[-find <path> ... <expression> ...]`
`[-getfacl [-R] <path>]`
`[-getfattr [-R] {-n name | -d} [-e en] <path>]`
`[-getmerge [-nl] <src> <localdst>]`

`[-help [cmd ...]]`
`[-ls [-d] [-h] [-R] [<path> ...]]`
`[-mkdir [-p] <path> ...]`
`[-moveFromLocal <localsrc> ... <dst>]`
`[-moveToLocal <src> <localdst>]`
`[-mv <src> ... <dst>]`
`[-put [-f] [-p] [-l] <localsrc> ... <dst>]`
`[-renameSnapshot <snapshotDir> <oldName> <newName>]`
`[-rm [-f] [-r] [-R] [-skipTrash] <src> ...]`
`[-rmdir [--ignore-fail-on-non-empty] <dir> ...]`
`[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>}]`
`[-setfattr {-n name [-v value] | -x name} <path>]`
`[-setrep [-R] [-w] <rep> <path> ...]`
`[-stat [format] <path> ...]`
`[-tail [-f] <file>]`
`[-test -[defsz] <path>]`
`[-text [-ignoreCrc] <src> ...]`
`[-touchz <path> ...]`
`[-truncate [-w] <length> <path> ...]`
`[-usage [cmd ...]]`

HDFS en pratique: Ligne de commande

Usage: `hadoop fs [generic options]`

- `[-appendToFile <localsrc> ... <dst>]`
- `[-cat [-ignoreCrc] <src> ...]`
- `[-checksum <src> ...]`
- `[-chgrp [-R] GROUP PATH...]`
- `[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]`
- `[-chown [-R] [OWNER][:[GROUP]] PATH...]`
- `[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]`
- `[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]`
- `[-count [-q] [-h] <path> ...]`
- `[-cp [-f] [-p | -p[topax]] <src> ... <dst>]`
- `[-createSnapshot <snapshotDir> [<snapshotName>]]`
- `[-deleteSnapshot <snapshotDir> <snapshotName>]`
- `[-df [-h] [<path> ...]]`
- `[-du [-s] [-h] <path> ...]`
- `[-expunge]`
- `[-find <path> ... <expression> ...]`
- `[-getfacl [-R] <path>]`
- `[-getfattr [-R] {-n name | -d} [-e en] <path>]`
- `[-getmerge [-nl] <src> <localdst>]`

- `[-help [cmd ...]]`
- `[-ls [-d] [-h] [-R] [<path> ...]]`
- `[-mkdir [-p] <path> ...]`
- `[-moveFromLocal <localsrc> ... <dst>]`
- `[-moveToLocal <src> <localdst>]`
- `[-mv <src> ... <dst>]`
- `[-put [-f] [-p] [-l] <localsrc> ... <dst>]`
- `[-renameSnapshot <snapshotDir> <oldName> <newName>]`
- `[-rm [-f] [-r] [-R] [-skipTrash] <src> ...]`
- `[-rmdir [--ignore-fail-on-non-empty] <dir> ...]`
- `[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>}]`
- `[-setfattr {-n name [-v value] | -x name} <path>]`
- `[-setrep [-R] [-w] <rep> <path> ...]`
- `[-stat [format] <path> ...]`
- `[-tail [-f] <file>]`
- `[-test -[defsz] <path>]`
- `[-text [-ignoreCrc] <src> ...]`
- `[-touchz <path> ...]`
- `[-truncate [-w] <length> <path> ...]`
- `[-usage [cmd ...]]`

HDFS en pratique: Ligne de commande

Usage: `hadoop fs [generic options]`

- `[-appendToFile <localsrc> ... <dst>]`
- `[-cat [-ignoreCrc] <src> ...]`
- `[-checksum <src> ...]`
- `[-chgrp [-R] GROUP PATH...]`
- `[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]`
- `[-chown [-R] [OWNER][:[GROUP]] PATH...]`
- `[-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]`
- `[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]`
- `[-count [-q] [-h] <path> ...]`
- `[-cp [-f] [-p | -p[topax]] <src> ... <dst>]`
- `[-createSnapshot <snapshotDir> [<snapshotName>]]`
- `[-deleteSnapshot <snapshotDir> <snapshotName>]`
- `[-df [-h] [<path> ...]]`
- `[-du [-s] [-h] <path> ...]`
- `[-expunge]`
- `[-find <path> ... <expression> ...]`
- `[-getfacl [-R] <path>]`
- `[-getfattr [-R] {-n name | -d} [-e en] <path>]`
- `[-getmerge [-nl] <src> <localdst>]`

- `[-help [cmd ...]]`
- `[-ls [-d] [-h] [-R] [<path> ...]]`
- `[-mkdir [-p] <path> ...]`
- `[-moveFromLocal <localsrc> ... <dst>]`
- `[-moveToLocal <src> <localdst>]`
- `[-mv <src> ... <dst>]`
- `[-put [-f] [-p] [-l] <localsrc> ... <dst>]`
- `[-renameSnapshot <snapshotDir> <oldName> <newName>]`
- `[-rm [-f] [-r] [-R] [-skipTrash] <src> ...]`
- `[-rmdir [--ignore-fail-on-non-empty] <dir> ...]`
- `[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>}] [--set`
- `[-setfattr {-n name [-v value] | -x name} <path>]`
- `[-setrep [-R] [-w] <rep> <path> ...]`
- `[-stat [format] <path> ...]`
- `[-tail [-f] <file>]`
- `[-test -[defsz] <path>]`
- `[-text [-ignoreCrc] <src> ...]`
- `[-touchz <path> ...]`
- `[-truncate [-w] <length> <path> ...]`
- `[-usage [cmd ...]]`

HDFS en pratique: API Java

Hadoop propose une API complète pour utiliser le système HDFS à partir d'un programme JAVA. Vous trouverez tous la documentation dans le répertoire \$CLUSTER_INSTALL_DIR/hadoop/share/doc/

Le principe est très similaire aux fichier classiques. La différence est qu'il faut paramétrer l'API correctement pour qu'elle puisse faire les connections aux serveurs distants. Les packages à utiliser sont tous dans le package org.apahe.hadoop. L'utilisation d'un ToolRunner permet de récupérer la configuration d'hadoop pour pouvoir créer la connexion vers le namenode.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
public class MonApplication {
    public static class MonProg extends Configured implements Tool {
        public int run(String[] args) throws Exception {
            //CODE DE VOTRE PROGRAMME ICI
            return 0;
        }
    }
    public static void main( String[] args ) throws Exception {
        ToolRunner.run(new MonApplication.MonProg(), args);
    }
}
```

HDFS en pratique: Compilation/Exécution

Pour executer votre « driver » il faut d'abord générer un « jar » java contenant toutes les librairies nécessaire. Si les librairies java ont été déployées dans le CLASS_PATH des machines du cluster vous n'êtes pas obligé de les mettre. Attention le transfert du jar sur toutes les machines peut prendre un certain temps et du coup augmenter les temps d'exécution de vos programmes.

Compilation:

Si vous compilez directement avec eclipse, il faut ajouter les bibliothèques externes de hadoop pour pouvoir compiler. Ces bibliothèques sont dans \$CLUSTER_INSTALL_DIR/hadoop/share/hadoop. Attention, il n'est pas toujours évident de trouver quels sont les jar qu'il faut inclure.

Il est préférable d'utiliser Maven pour compiler et générer vos jar hadoop. Sur eclipse vous pouvez créer un projet Maven et ajouter la dépendance hadoop-common à votre projet. Utilisez le « goals » package pour générer les jar.

Exécution:

Lancer votre programme en utilisant la commande: `hadoop jar votrejar.jar`

HDFS en pratique:

Exemple: copie de fichier

```
public static class CopyFromLocal extends Configured implements Tool {
    public int run(String[] args) throws Exception {
        String localInputPath = args[0];
        URI uri = new URI(args[1]);
        uri = uri.normalize();
        Configuration conf = getConf();
        FileSystem fs = FileSystem.get(uri, conf, "hadoop");
        Path outputPath = new Path(uri.getPath());
        OutputStream os = fs.create(outputPath);
        InputStream is = new BufferedInputStream(new FileInputStream(localInputPath));
        IOUtils.copyBytes(is, os, conf);
        os.close();
        is.close();
        return 0;
    }
}
```

HDFS en pratique:

Exemple: Merge de fichiers

```
public static class MergeFromLocal extends Configured implements Tool {
    public int run(String[] args) throws Exception {
        URI uri = new URI(args[args.length - 1]);
        uri = uri.normalize();
        Configuration conf = getConf();
        FileSystem fs = FileSystem.get(uri, conf, "hadoop");
        Path outputPath = new Path(uri.getPath());
        OutputStream os = fs.create(outputPath);
        for (int i=0; i < args.length - 1; ++i) {
            InputStream is = new BufferedInputStream(new FileInputStream(args[i]));
            IOUtils.copyBytes(is, os, conf, false);
            is.close();
        }
        os.close();
        return 0;
    }
}
```