# Hive

## Conexion avec JDBC à travers Knox

La chaine de conexion est la suivante

jdbc:hive2://inbdfkx01.nor.fr.ftgroup:8443/;transportMode=http;httpPath=gateway/default/hive;ssl=true

Il faut normalement importer le certificat dans le truststore de Java

keytool.exe -importcert -alias "hd-knox-ecm.si.francetelecom.fr" -file "C:\Program Files (x86)\Java\jre7\lib\security\hd-knox-ecm.si.francetelecom.fr.crt" -keystore "C:\Program Files (x86)\Java\jre7\lib\security\cacerts"



where rank = 1  // pour conserver uniquement le premier enregistrement

## Transport mode

HiveServer2 has two transport modes:

* binary - this is the default mode and by default will listen on port 10000 (thrift port)
* HTTP - This is the non-default mode and by default will listen on port 10001 (hive.server2.thrift.http.port). Hive http transport mode is slower than binary.

Beeline peut se connecter sur les 2 types de transport mode.

Knox requires HTTP, and clients only support Binary mode, such as the Hive Ambari View

Hue only works with HS2 in 'binary' mode. (3.9 works with http)

Also Hive View doesn't work in HTTP transport mode. (Should work with Ambari 2.2)

## Formats de compression

ORC gère deux algos de compressions :

* ZLIB
* SNAPPY

En temps normal :

* ZLIB permet une bonne compression (supérieure à SNAPPY), propose une bonne performance IO (inférieure à SNAPPY), nécessite davantage de CPU que SNAPPY
* SNAPPY permet une bonne compression (inférieure à ZLIB), propose une bonne performance IO (supérieure à ZLIB), nécessite moins de CPU que ZLIB

Avec ORC, chaque type de colonne dispose d’un algorithme ZLIB spécifiquement adapté au type de données de la colonne.

Du coup, la combinaison ORC et ZLIB ne présente plus les faiblesses historiques de cet algo de compression.

On obtient ainsi une meilleure compression que Snappy, une plus grande rapidité pour lire les données que Snappy, et un ralentissement de 10% en écriture par rapport à SNAPPY.

Du coup, je vous conseillerai de partir sur des tables ORC avec comme algo de compression ZLIB.

Pour passer à un autre format (de TEXT à ORC ZLIB ou de ORC SNAPPY à ORC ZLIB), le plus simple est de recréer les tables au bon format et d’insérer les données depuis l’ancienne table vers la nouvelle table.

USE <my\_db>;

CREATE TABLE <mytable\_orc>

(

<myfied1> <datatype1>

, <myfield2> <datatype2>

)

STORED AS ORC

TBLPROPERTIES

(

"orc.compress"="ZLIB"

);

AS SELECT \* FROM mytable;

## Bugs

### hive.fetch.task.conversion (Table ORC)

Cette opération consiste en une modification de configuration dans hive

La modification est la suivante :

SET hive.fetch.task.conversion=none;

Cela correspond à la résolution du ticket https://www.forge.orange-labs.fr/plugins/tracker/?aid=294239 concernant des problèmes de droits d’accès sur des tables au format ORC en passant par HiveServer2.

# Beeline

## Se connecter Beeline

**http**

beeline -u "jdbc:hive2://opbdf1217.rouen.francetelecom.fr:10001/;principal=hive/opbdf1217.rouen.francetelecom.fr@BIGDATAFRANCE;transportMode=http;httpPath=cliservice" -e "show databases;"

**binary**

beeline -u "jdbc:hive2://opbdf0620.rouen.francetelecom.fr:10000/;principal=hive/opbdf0620.rouen.francetelecom.fr@BIGDATAFRANCE;transporttMode=binary;httpPath=cliservice" -e "show databases;"

## Erreur 1

Si on détache beeline du terminal (par exemple avec nohup), beeline se chope un signal SIGTTOU (<https://issues.apache.org/jira/browse/HIVE-6758>)

Pour éviter ça :

export HADOOP\_CLIENT\_OPTS="-Djline.terminal=jline.UnsupportedTerminal"

beeline ……

Si on met beeline en mode silencieux (pour simplifier le parsing de la sortie) et avec un fichier sql en paramètre, la première ligne contient de la merde !

La même commande avec –e au lieu de –f fonctionne correctement.

Ex:

[root@uabdfkx01 transfert]# cat req.sql

use z\_app\_bds\_hive\_socle;show tables;

[root@uabdfkx01 transfert]# beeline -u 'jdbc:hive2://uabdfot02.rouen.francetelecom.fr:10000/default;principal=hive/uabdfot02.rouen.francetelecom.fr@BDFPOCP' --silent=true --showHeader=false --outputformat=tsv2 -f req.sql 2>/dev/null | cat -A

**^Mnull                                                   ^Mnull**anytable$

fai\_bds\_twitter\_followers$

fai\_bds\_twitter\_panel\_accounts$

fai\_bds\_twitter\_perf$

fai\_bds\_twitter\_tweet\_hashtags$

fai\_bds\_twitter\_tweet\_media$

fai\_bds\_twitter\_tweet\_places$

fai\_bds\_twitter\_tweet\_symbols$

fai\_bds\_twitter\_tweet\_urls$

fai\_bds\_twitter\_tweet\_user\_mentions$

fai\_bds\_twitter\_tweets$

fai\_bds\_twitter\_user\_descriptions$

fai\_bds\_twitter\_user\_urls$

fai\_bds\_twitter\_users$

ref\_bds\_twitter\_perf\_accounts$

$

[root@uabdfkx01 transfert]#

# TEZ

## Quelques paramètres intéressants

SET hive.execution.engine=tez;

SET hive.tez.container.size=8192;

SET hive.tez.java.opts=-Xmx6560m;

SET tez.runtime.io.sort.mb=5120;

SET tez.runtime.unordered.output.buffer.size-mb=820

## Swimlane

Petite note : Il faut se kiniter avec le principal du user d’exécution pour récupérer les logs

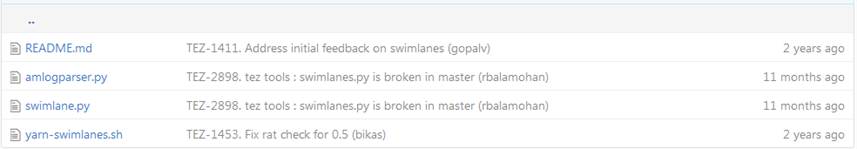
1. La première étape consiste à récupérer la log de l’application.
   1. Tu as normalement l’application ID via l’IHM de Yarn de G2PROD
   2. Une fois cette information en ta possession, connecte-toi sur le DA de G2PROD et lance la commande suivante :

yarn logs -applicationId <TON APPLICATION ID> > <TON FICHIER DE LOG>

* 1. Tu as maintenant le fichier avec toutes les infos pour ton job. Cela te sera utile plus tard pour mieux analyser ton job.

1. Récupère trois fichiers depuis le GitHub  de Swimlane :

<https://github.com/apache/tez/tree/master/tez-tools/swimlanes>



Mets les dans ta homedir sur la DA, dans un dossier swimlane :

Lances la commande suivante :

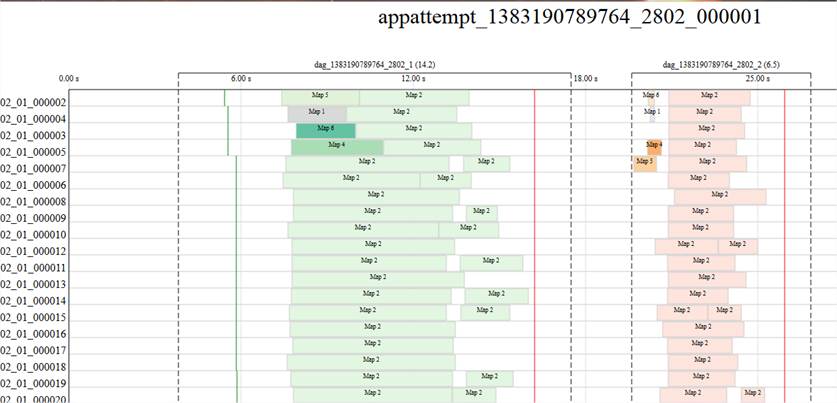
./yarn-swimlanes.sh <TON APPLICATION ID>

Ca va te créer un fichier SVG

Récupère sur ton poste ton fichier de log et ce fichier SVG.

Le fichier SVG est à ouvrir via un navigateur web.

On obtient quelque chose de ce genre quand on ouvre un SVG :



* Le premier trait vertical à gauche (en vert) pour chaque ligne correspond à la création du conteneur dans yarn. (Ligne dans le fichier de log « Launching container » me semble)
* Le second trait vertical à gauche (au tout début du rectangle Map5 ou Map1 ou Map4 etc…) correspond au lancement de la JVM du conteneur. (Recherche dans le fichier de log : PID)

Cela va te permettre de voir s’il y a un conteneur qui retarde les autres.

En tez, tu as des étapes de traitement ou vertices (MAP1, MAP2, REDUCE1 etc…). Ces étapes correspondent au lancement d’un certain nombre de conteneurs (on va avoir une patate de mappers pour MAP1, une autre patate pour MAP2, un petite patate de reducer pour REDUCE1 etc… C’est généralement grâce à ces étapes que l’on peut voir si un conteneur bloque l’avancée globale du job ou non.

Note que nous avons remarqué de temps en temps des problèmes de lenteur pour le spawn de la JVM des conteneurs.

Ces problèmes de lenteur pourraient être liés à l’étape de localisation des données dans Yarn (pour plus d’infos là-dessus, google « yarn localization »)

Malheureusement, ces infos pour la localisation ne sont disponibles que dans la log du nodemanager itself et donc il faut aller directement sur un worker pour avoir cette info locale.

# Hadoop

## Installation de la stack manuelement :

/usr/bin/yum -d 0 -e 0 -y install 'hadoop\_2\_3\_\*'

## HDP-SELECT

Le hdp-select permet de switcher (avec le lien symbolique current) un composant d’une version à une autre.

hdp-select set kafka-broker 2.2.4.0-2633

hdp-select set all 2.2.4.0-2633

# Ajouter un worker

1. Installer les prérequis (dont ambari-agent)
2. Ajouter le serveur à partir d’ambari et installer les cients + NodeManager + DataNode + RegionServer
3. Copier les keytabs utilisateurs depuis un autre worker. Appliquer les chown et chmod.
4. Créer les pricipal de service (http,dn, hbase, nm, spnego) pour le nouveau host et récupérer les keytabs. Appliquer les chown et chmod.
5. Récupérer les fichiers keystore et truststore d’un autre worker.

#Depuis le nouveau WK

mkdir /etc/security/serverKeys

chown yarn:hadoop /etc/security/serverKeys

#Depuis le serveur AM

scp /etc/security/serverKeys/keystore.jks inbdf0401:/etc/security/serverKeys/

scp /etc/security/serverKeys/truststore.jks inbdf0401:/etc/security/serverKeys/

#Depuis le serveur WK

chown yarn:hadoop /etc/security/serverKeys/\*

# HDFS

## Blocks vs Splits

**Conclusion:**

1. Split is a logical division of the input data while block is a physical division of data.
2. HDFS default block size is default split size if input split is not specified.
3. Split is user defined and user can control split size in his Map/Reduce program.
4. One split can be mapping to multiple blocks and there can be multiple split of one block.
5. The number of map tasks (Mapper) are equal to the number of splits.

src : <http://stackoverflow.com/questions/30549261/split-size-vs-block-size-in-hadoop>

## Ajouter un journal node

src : <http://zdatainc.com/2015/09/add-journalnode-to-ambari-managed-hadoop-cluster/>

Assign the role of JournalNode using the following command:

curl -u u\_qkdp3997\_adm -H "X-Requested-By: Ambari" -X POST http://localhost:8080/api/v1/clusters/BDFPOCHP/hosts/inbdf0301.rouen.francetelecom.fr/host\_components/JOURNALNODE

Now go ahead and install the JournalNode.

curl -u u\_qkdp3997\_adm -i -H "X-Requested-By: ambari" -X PUT -d '{"RequestInfo": {"context": "Install JournalNode"}, "Body":{"HostRoles": {"state": "INSTALLED"}}}' http://localhost:8080/api/v1/clusters/BDFPOCHP/hosts/inbdf0301.rouen.francetelecom.fr/host\_components/JOURNALNODE

Après cette étape, le journal node était bien installé mais il ne démarrait pas à cause de l’erreur suivante :

-bash: line 0: ulimit: core file size: cannot modify limit: Operation not permitted

qui est générée par

ambari-sudo.sh su hdfs -l -s /bin/bash -c 'ulimit -c unlimited

Du coup j’ai du éditer le fichier /etc/security.limits.conf et rajouter la ligne suivante:

hdfs hard core unlimited

Ensuite j’ai dû ajouter la keytab du journalnode.

**Note de Gwen :**

*Bonjour !*

*Je vous contacte car il m’est arrivé un truc sur G2POCP.*

*J’avais installé un nouveau journalnode, mais je ne l’avais pas synchronisé avec les autres, et donc le nn en stand-by plantait.*

*En fait, il y a plus qu’une installation à faire dans le cas d’un journalnode :*

*- Assigner le role journalnode à un serveur via une requête POST de l'API REST d'Ambari (on ne peut pas le faire par l'IHM)*

*- Installer le composant journalnode sur le serveur via une requête PUT de l'API REST d'Ambari (on ne peut pas le faire par l'IHM)*

*- Ajouter le serveur dans la propriété dfs.namenode.shared.edits.dir*

*- Créer le dossier des "Shared Edits" sur le nouveau serveur journalnode (/var/opt/data/flat/hadoop/journal/BDFPOCPNN)*

*- Stopper le service HDFS*

*- Copier depuis un autre journalnode le dossier current de ce dossier et le coller sur le nouveau journalnode*

*- Mettre les bons droits sur cette arborescence /var/opt/data/flat/hadoop/journal/BDFPOCPNN (hdfs:hadoop 755 sur tous les dossiers et 644 sur les fichiers)*

*- Ne pas oublier les keytabs HTTP et journalnode*

*- Demarrer le service HDFS*

## Starter un nodemanaer

ulimit -c unlimited; export HADOOP\_LIBEXEC\_DIR=/usr/hdp/current/hadoop-client/libexec && /usr/hdp/current/hadoop-yarn-nodemanager/sbin/yarn-daemon.sh --config /usr/hdp/current/hadoop-client/conf start nodemanager

## ProxyUser

Quand on ajoute un groupe dans hadoop.proxyuser.hive.groups, cela veut dire que l’utilisateur hive pourra s’impersonnifier en tant qu’un utilisateur appartenant à ce groupe dans l’écosystème Hadoop.

Les user-proxy sont parfois à repréciser pour certain service (ex webcat.proxyuser.hue.groups)

## HeapSize

Pour lancer une action hadoop avec un heap spécifique :

HADOOP\_CLIENT\_OPTS= "-Xmx4g" hdfs dfs –ls –R /

## Pour retrouver les blocks d’un fichier réparti sur les datanodes

**Lancement de job avec facteur de réplication = 3**

hadoop jar /usr/hdp/2.3.2.0-2950/hadoop-mapreduce/hadoop-streaming.^Cr -D mapred.job.queue.name=q\_datalab -mapper cat -reducer cat -input /tmp/greg\_in -output /tmp/greg\_out\_3

hdfs fsck /tmp/greg\_out\_3  -files -blocks -locations |  awk -F ":1019"  '/BP-/ {for (i=1;i<NF;i++) {sub(".\*[[]", "", $i); N[$i]++}} END{for (i in N) print N[i],i}' | sort –n

|  |  |
| --- | --- |
| Nb Blocks | Server |
| 11 | 10.79.28.8 |
| 15 | 10.79.28.27 |
| 16 | 10.79.28.26 |
| 17 | 10.79.28.10 |
| 18 | 10.79.28.29 |
| 18 | 10.79.28.6 |
| 18 | 10.79.28.9 |
| 24 | 10.79.28.30 |
| 25 | 10.79.28.7 |
| 81 | 10.79.28.28 |

**Lancement de job avec facteur de réplication = 1**

hadoop jar /usr/hdp/2.3.2.0-2950/hadoop-mapreduce/hadoop-streaming.jar -D mapred.job.queue.name=q\_datalab -D dfs.replication=1 -mapper cat -reducer cat -input /tmp/greg\_in -output /tmp/greg\_out

-sh-4.1$ hdfs fsck /tmp/greg\_out  -files -blocks -locations |  awk -F ":1019"  '/BP-/ {for (i=1;i<NF;i++) {sub(".\*[[]", "", $i); N[$i]++}} END{for (i in N) print N[i],i}' | sort –n

|  |  |
| --- | --- |
| Nb Blocks | Server |
| 81 | 10.79.28.29 |

## Pour activer et créer un snapshot HDFS, voici les commandes à utilisées (ex RJDM) :

1. Ajout des droits de création de snapshot sur un path HDFS:

hdfs dfsadmin -allowSnapshot /user/a\_app\_bdf\_hdfs/socle\_rjdm

2. Création d’un snapshot sur le path :

hdfs dfs -createSnapshot /user/a\_app\_bdf\_hdfs/socle\_rjdm 20160126

3. accès au snapshot :

hdfs dfs -ls /user/a\_app\_bdf\_hdfs/socle\_rjdm/.snapshot/20160126

Found 4 items

drwxr-x--- - a\_app\_bdf\_alim r\_app\_bdf\_socle\_rjdm\_reader 0 2015-11-12 12:39 /user/a\_app\_bdf\_hdfs/socle\_rjdm/.snapshot/20160126/20150930

drwxr-x--- - a\_app\_bdf\_alim r\_app\_bdf\_socle\_rjdm\_reader 0 2015-11-12 13:04 /user/a\_app\_bdf\_hdfs/socle\_rjdm/.snapshot/20160126/20151005

Pour information, les blocs des fichiers ne sont pas physiquement dupliqués sur disque, seul les métadonnées le sont ce qui permet de se prémunir d’un rm sauvage.

## Pour auditer les actions faites sur le filesystem HDFS

Alexis a développé un job talend qui récupère quotidiennement les logs depuis les namenode et qui les enregistre dans une table externe HIVE :

select \* from z\_app\_alh\_hive\_socle.hdfs\_audit where src RLIKE '/Test\_03'  ;

## Rafraichissement des relations des user/group dans HDFS

-sh-4.1$ hdfs dfsadmin -refreshUserToGroupsMappings

## Chargement de gros fichier à partir du poste local

Bonjour Marc.

Pour faire suite à notre conversation téléphonique + communicator, j’arrive également à reproduire ce problème.

Voici la solution de dépannage que je te propose : utiliser le client http curl pour uploader directement tes fichiers dans hdfs au travers de la passerelle knox.

1. Télécharger le client curl pour windows :
   1. Aller sur <http://winampplugins.co.uk/curl/curl_7_49_1_openssl_nghttp2_x64.7z> (je sais l’adresse est chelou mais bon…)
2. Décompresser l’archive dans un dossier appelé « curl »
3. Déplacer ce dossier dans C:\My Program Files sur ta station de travail.
4. Récupérer le script putfile.bat en PJ (ignore le message d’avertissement, c’est moi qui l’ai développé, donc pas de soucis) et le placer dans le dossier C:\My Program Files\curl
   1. Dans ce script, à la ligne 9, il faut que tu mettes ton mot de passe à la place de la chaîne de caractère <TON PASSWORD ICI>
5. Lancer une fenêtre de commande MS-DOS (démarrer -> exécuter -> taper cmd -> entrée)
   1. Une fois dans la fenêtre de commande MS-DOS, taper la commande suivante

"C:\My Program Files\curl\putfile.bat" <DOSSIER EN ENTREE SUR TA STATION> <TON COMPTE SUR HUBDATA> <DOSSIER EN SORTIE SUR LE CLUSTER HUBDATA>

Le script putfile.bat est dans le dossier script dans U de mon poste..

Le script liste les fichiers dans le dossier <DOSSIER EN ENTREE> sur ta station de travail (il faut que tu mettes le path complet).

Le script utilise ton compte hubdata pour se connecter à la passerelle knox.

Le script dépose les fichiers dans le dossier <DOSSIER EN SORTIE> sur le cluster hubdata.

De plus, il te faut faire une GENERGY pour mes collègues de la DESI, afin de leur signaler qu’il y a un problème sur Hue et que cela demande des investigations.

# WEBHDFS et WEBHCAT

## WEBHCAT

**G2INT1 : templeton\_url=http://inbdfot11.rouen.francetelecom.fr:50111/templeton/v1/**

# this will execute a hive query and save result to hdfs file in your home directory called output

curl -s -d execute="select+\*+from+sample\_08;" -d statusdir="output" 'http://localhost:50111/templeton/v1/hive?user.name=root'

# if you ls on the directory, it will have two files, stderr and stdout

hdfs dfs -ls output

# if the job succeeded, you can cat the stdout file and view the results

hdfs dfs -cat output/stdout

#this will get a describe of your target table :

curl --negotiate -u : 'http://inbdfot21.rouen.francetelecom.fr:50111/templeton/v1/ddl/database/default/table/dual'

#this will get the webcat status

curl --negotiate -u : 'http://inbdfot21.rouen.francetelecom.fr:50111/templeton/v1/status'

## WEBHDFS

**G2INT1 :** http://inbdfot11.rouen.francetelecom.fr:14000/webhdfs/v1/

# list the output directory, notice the webhdfs port

curl -i "http://sandbox.hortonworks.com:50070/webhdfs/v1/user/root/output/?op=LISTSTATUS"

# read the output file

curl -i -L "http://sandbox.hortonworks.com:50070/webhdfs/v1/user/root/output/stdout?op=OPEN"

# rename a file, if you get dr. who error, add &user.name=root or any other user in the context

curl -i -X PUT "sandbox.hortonworks.com:50070/webhdfs/v1/user/root/output/stdout?op=RENAME&user.name=root&destination=/user/root/newname"

# read the output of the new file

curl -i -L "http://sandbox.hortonworks.com:50070/webhdfs/v1/user/root/newname?op=OPEN"

## HTTPFS

Using WebHDFS

http://<active-namenode-server>:<namenode-port>/webhdfs/v1/<file-path>?op=OPEN

Using HttpFs

http://<hadoop-httpfs-server>:<httpfs-port>/webhdfs/v1/<file-path>?op=OPEN

src : <https://community.hortonworks.com/articles/18075/comparison-of-httpfs-and-webhdfs.html>

# POSTGRES

## Connexion

Se connecter au serveur DB

Taper

su - postgres

Puis psql

. pgqenv

Il faut ensuite faire son choix parmis les environnements proposés

#========================================================================#

# Pour se positionner sur un environnement PostgreSQL : faire '. pgqenv' #

# To set postgresql environment : execute '. pgqenv' #

#========================================================================#

[dvbdfdb21.rouen.francetelecom.fr:postgres:null]$. pgqenv

----------------------------------------------------------------------------------

1 <> PostgreSQL : 9.3.5 - PostgreSQL Server : pghue - PostgreSQL Port : 5435

PGDATA /pgqdata/pghue/data

PGHOME /opt/pgsql/na/9.3.5

2 <> PostgreSQL : 9.3.5 - PostgreSQL Server : pgambari - PostgreSQL Port : 5432

PGDATA /pgqdata/pgambari/data

PGHOME /opt/pgsql/na/9.3.5

3 <> PostgreSQL : 9.3.5 - PostgreSQL Server : pghive - PostgreSQL Port : 5433

PGDATA /pgqdata/pghive/data

PGHOME /opt/pgsql/na/9.3.5

4 <> PostgreSQL : 9.3.5 - PostgreSQL Server : pgoozie - PostgreSQL Port : 5434

PGDATA /pgqdata/pgoozie/data

PGHOME /opt/pgsql/na/9.3.5

5 <> PostgreSQL : 9.3.5 - PostgreSQL Server : pgsqoop - PostgreSQL Port : 5436

PGDATA /pgqdata/pgsqoop/data

PGHOME /opt/pgsql/na/9.3.5

6 <> PostgreSQL : 9.3.5 - PostgreSQL Server : pghueexp - PostgreSQL Port : 5437

PGDATA /pgqdata/pghueexp/data

PGHOME /opt/pgsql/na/9.3.5

Puis taper psql pour se connecter à psql et enfin taper \l pour avoir la liste des bases de données.

Taper \connect [BDD] pour se connecter à une BDD et puis \dt pour lister les tables.

## Quelques commandes

Lister les schemas

select schema\_name from information\_schema.schema

Lister les BDD

\l

Lister les tables dans la BDD courante

\dt

To switch databases:

\connect database\_name

Lister toutes les tables

\dt \*.\*

Lister les tables d’un schema

\dt public.\*

Lister les tables d’un schema

SELECT \* FROM information\_schema.tables WHERE table\_schema = 'public'

## Configuration

cat /pgqdata/pgoozie/data/pg\_hba.conf

## Requête taille des tables

SELECT nspname || '.' || relname AS "relation",

pg\_size\_pretty(pg\_total\_relation\_size(C.oid)) AS "total\_size"

FROM pg\_class C

LEFT JOIN pg\_namespace N ON (N.oid = C.relnamespace)

WHERE nspname NOT IN ('pg\_catalog', 'information\_schema')

AND C.relkind <> 'i'

AND nspname !~ '^pg\_toast'

ORDER BY pg\_total\_relation\_size(C.oid) DESC

LIMIT 20;

This will report size information for all tables, in both raw bytes and "pretty" form.

SELECT \*, pg\_size\_pretty(total\_bytes) AS total

, pg\_size\_pretty(index\_bytes) AS INDEX

, pg\_size\_pretty(toast\_bytes) AS toast

, pg\_size\_pretty(table\_bytes) AS TABLE

FROM (

SELECT \*, total\_bytes-index\_bytes-COALESCE(toast\_bytes,0) AS table\_bytes FROM (

SELECT c.oid,nspname AS table\_schema, relname AS TABLE\_NAME

, c.reltuples AS row\_estimate

, pg\_total\_relation\_size(c.oid) AS total\_bytes

, pg\_indexes\_size(c.oid) AS index\_bytes

, pg\_total\_relation\_size(reltoastrelid) AS toast\_bytes

FROM pg\_class c

LEFT JOIN pg\_namespace n ON n.oid = c.relnamespace

WHERE relkind = 'r'

) a

) a;

# KERBEROS/IPA

## Accéder service via kerberos

se kiniter puis :

curl -u : --negotiate http://inbdfot01.rouen.francetelecom.fr:9080/analyticserver/admin

## Modifier emplacement keytabs pour la session

export KRB5CCNAME=/tmp/truc

## Désactiver/Activer la recherche anonyme

D'abord lancer la commande suivante :

ldapmodify -x -D "cn=Directory Manager" -W -h <SERVER IPA> -p 389

Un prompt demande le mdp du directory manager apparaît. Entrer le mdp du user, puis entrer les lignes suivantes :

dn: cn=config  
changetype: modify  
replace: nsslapd-allow-anonymous-access  
nsslapd-allow-anonymous-access: off

Et appuyer deux fois sur entrée. Le message suivant s'affiche quand la modification est effectuée.

modifying entry "cn=config"

## Faire une recherche dans le ldap

ldapsearch -x -D "uid=ldap\_browser,cn=users,cn=accounts,dc=bdfpochp" -W -H ldap://localhost -b "cn=users,cn=accounts,dc=bdfpochp" -s sub 'uid=u\_qkdp3997\_adm'

## Restart de pki-ca ne fonctionne pas

Bonjour,

En copie le fichier de sauvegarde qui semblait gêner le restart de pki-ca et donc globalement de IPA.

Sur l’appel de « service ipa «  on avait ce message en WARNING :

/../

Stopping pki-ca:                                           [  OK  ]

Starting pki-ca:                                           [  OK  ]

WARNING:  Since the file '/var/lib/pki-ca/conf/CS.cfg.bak.saved' exists, a previous backup attempt has failed!  Backups will be discontinued until this issue has been resolved!

Failed to restart CA Service

Shutting down

Stopping Kerberos 5 KDC:                                   [  OK  ]

/…/

Donc pour débloquer cette situtation :

1. Arret IPA
2. Copie de /var/lib/pki-ca/conf/CS.cfg.bak.saved en /var/lib/pki-ca/conf/CS.cfg.bak.saved.20160712

cp -p /var/lib/pki-ca/conf/CS.cfg.bak.saved /var/lib/pki-ca/conf/CS.cfg.bak.saved.20160712

rm -f /var/lib/pki-ca/conf/CS.cfg.bak.saved

1. Restart IPA => **OK**

# LINUX

## Récupérer tous les host d’ambari dans /etc/pdsh/machines

#!/bin/bash

curl -u $1 -i -X GET http://localhost:8080/api/v1/clusters/BDFINT1/hosts/ | grep -ire host\_name | awk -F '"' '{ print $4 }' | awk -F '.' '{print $1}' > /etc/pdsh/machines

## PDSH spécifier le fichier host

pdsh -w ^/etc/pdsh/kafka\_hosts 'chmod 440 /etc/security/keytabs/kafka.service.keytab'

## Lancer une commande en //

pdsh –w inbdfka11,inbdfkx12 "ls ./"

## Lancer une commande avec for

for srv in inbdf0106 inbdf0107 inbdf0108 inbdf0109 inbdf0110 inbdf0206 inbdf0207 inbdf0208 inbdf0209 inbdf0210; do ssh $srv "ls -ld /var/opt/data/flat/data008/hadoop/yarn/local/usercache/u\_sbfs0057\_dev"; done

OU

for p in $(cat /etc/pdsh/machines)

do

scp ./ambari\_sudoer $p:/etc/sudoers.d/

done

OU

for p in $(cat /etc/pdsh/machines)

do

cat /root/.ssh/id\_rsa.pub | ssh $p 'cat >> /root/.ssh/authorized\_keys'

done

OU

for i in {1..5}

do

echo "Welcome $i times"

done

## Lister les disques dur rattachés à la machine

lsblk | grep disk

ou encore

mount -ls

## Fournir une liste de fichier à grep

locate oozie-site.xml | xargs grep -A 5 'oozie.service.PurgeService'

## Modification du fichier sshd

*En attendant, afin d’éviter les reconfigurations multipliées des serveurs (livrés conformes) par les experts systèmes, merci de les contacter par mail ou par SWAN à AQZT15 avant de modifier tout fichier système (Mail de Pernot Florian)*

## Synchroniser ntp

Le service ntpd était KO sur es12.

Du coup, j’ai refait une synchro sur ce serveur es12 et j’ai relancé le service :

service ntpd stop; ntpdate ad.francetelecom.fr; service ntpd start

Sur les serveurs hors prod de la G1, on utilisait ad.francetelecom.fr comme serveur ntpd.

# cat /etc/ntp.conf

server ad.francetelecom.fr

Par contre, maintenant que ça c’est fait, j’ai quelque chose d’un peu bizarre avec mon test :

###

# rm -f /root/test /var/opt/data/flat/data/test; date; echo "creation fichier local"; touch /root/test; date; echo "creation fichier filer"; touch /var/opt/data/flat/data/test; date; echo "Listing"; ls -l /root/test /var/opt/data/flat/data/test;

Thu Mar  3 10:27:44 CET 2016

creation fichier local

Thu Mar  3 10:27:44 CET 2016

creation fichier filer

Thu Mar  3 10:27:44 CET 2016

Listing

-rw-r--r-- 1 root root 0 Mar  3 10:27 /root/test

-rw-r--r-- 1 root root 0 Mar  3  2016 /var/opt/data/flat/data/test

###

Gwenael Le Barzic

## Dépôt – Repositories

Le fichier /etc/yum.conf contient les paramètres généraux de tous les dépôts

Ensuite, il y a un fichier par déport dans /etc/yum.repos.d/

Pour ajouter un dépôt, voir le site source ci-dessous.

Pour avoir la liste des dépôts du serveur, taper

yum repolist

src : <https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/sec-Configuring_Yum_and_Yum_Repositories.html>

Ouverture de port

## Générer un fichier aléatoire avec awk

awk 'BEGIN{for (i;i<=6000000;i++) print ENVIRON["PATH"]}' > ./testflume.txt

## Obtenir une version d’un logiciel

rpm -qa | grep hive-odbc

## Vérifier si un logiciel est déjà installé

/usr/bin/yum list installed | grep ipa-server | wc -l

## Faire un recherche sur tout le serveur

find / -name '\*.yml' -exec grep -n "svc" {} +

## IPTABLES

Il y a 2 types de configuration

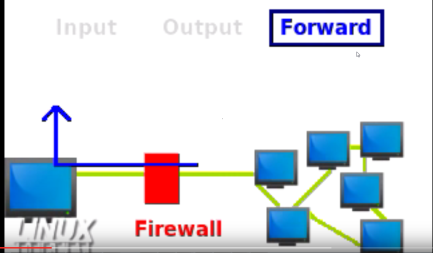
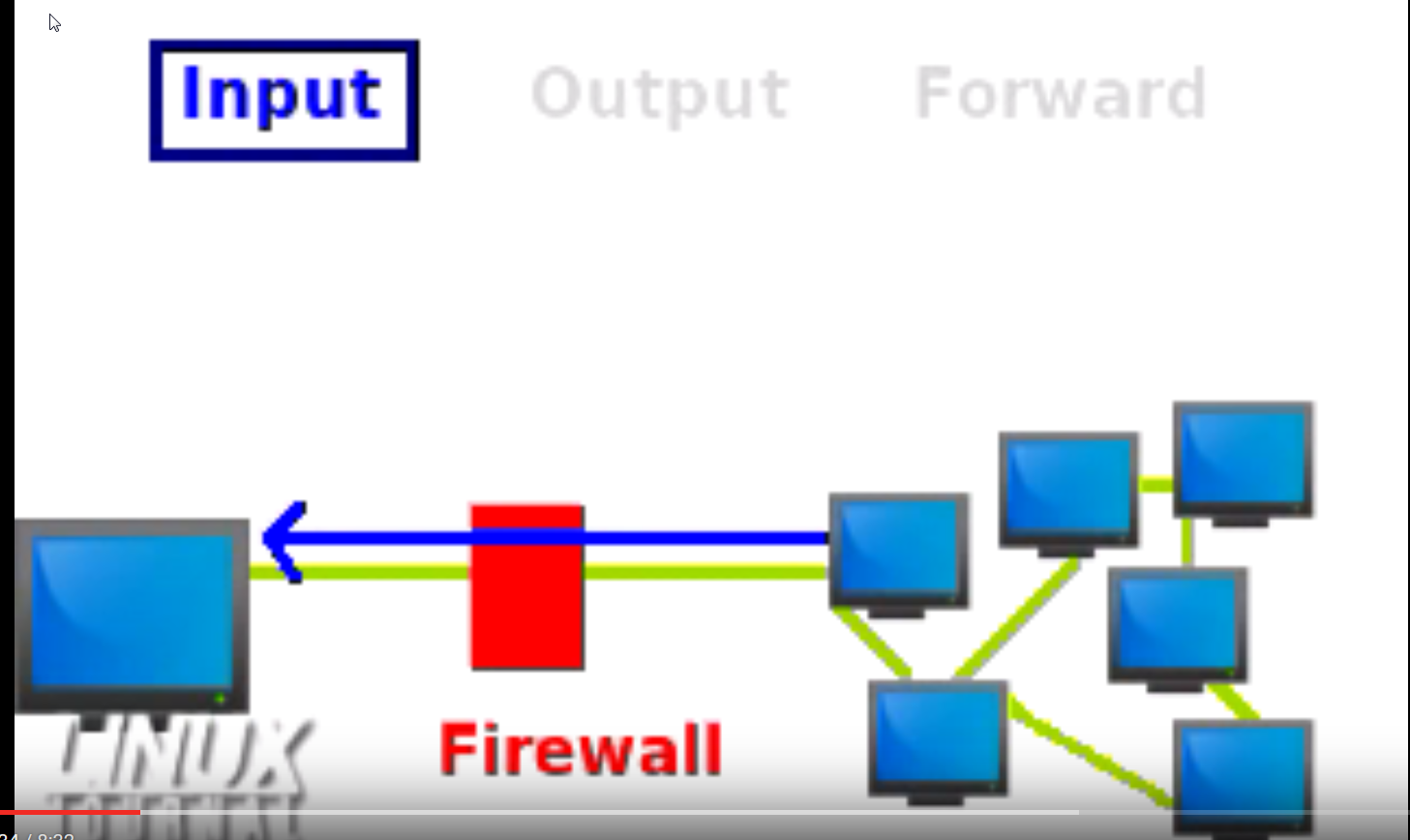
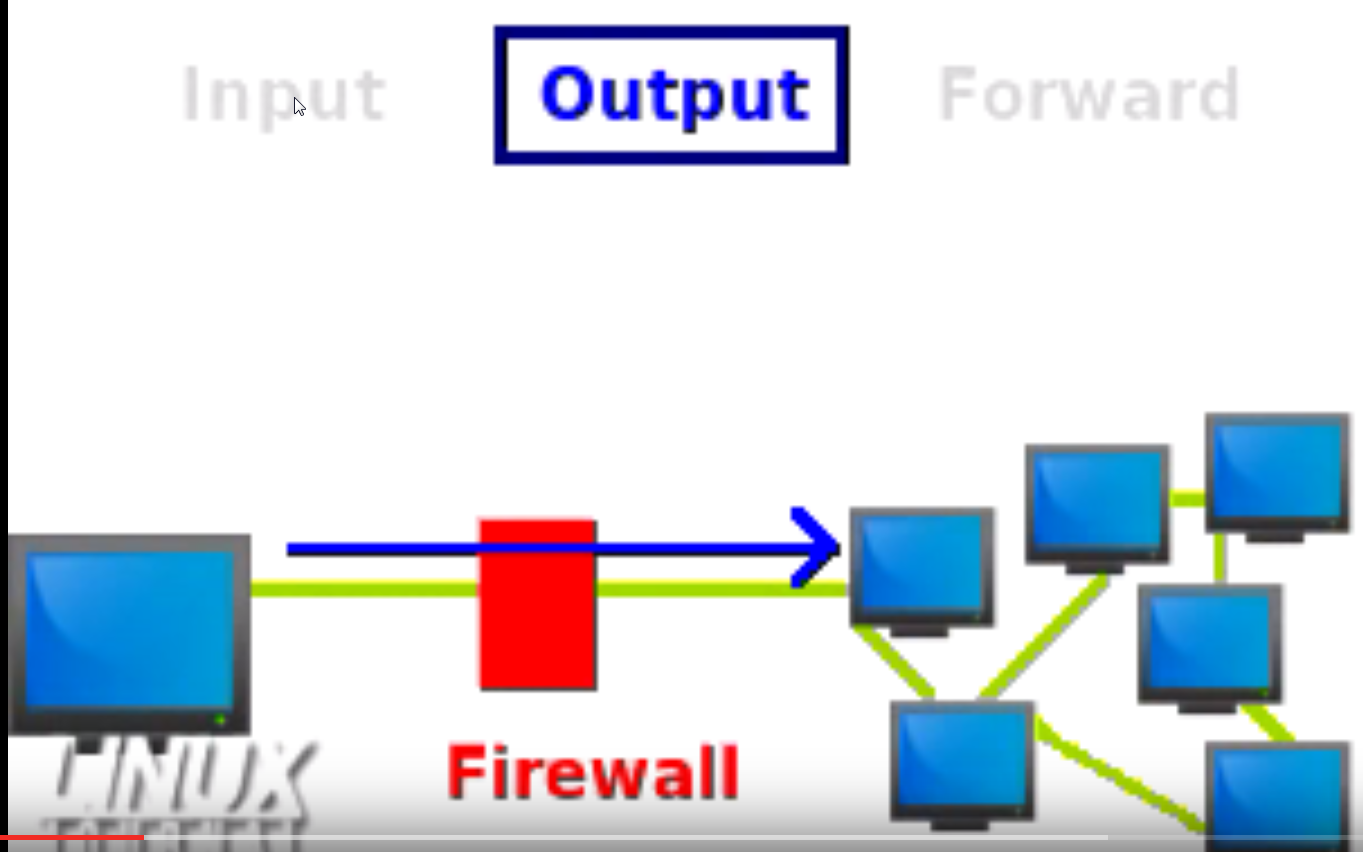
* FILTER (Filtrage)
* NAT (Network Address Translation => redirection)

**Il faut comprendre que la liite des paramètres précisées (mis à part -j, -t, -A/I) sont en fait les conditions qui doivent être respectées (matching) afin que l’action -j soit appliqué.**

### Filtres

#### Définition

* Les chains sont de type INPUT OUTPUT FORWARD

* **chaine** : INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING,[USER DEFINED CHAIN]
* **target** : ACCEPT, DROP, RETURN, QUEUE, REJECT (DROP lâche le paquet de manière silencieuse contrairement à REJECT qui renvoie un message au client)

#### Commandes

* **iptables -flush [chaine]** // supprime toutes les règles en mémoire dans iptables
* **iptables -A INPUT -i lo -j ACCEPT**
  + -A : append => ajoute la règle à la chaine
  + -i : --in-interface => l’interface d’entrée concernée (n’a de sens que pour les chaines INPUT, FORWARD, PREROUTING
  + -j : --jump => Quelle action faire si la requête matche la règle ACCEPT, [chaine], DROP
* **iptables -A OUTPUT -o lo -j ACCEPT**
  + -o : --out-interface => l’interface de sortie concernée (n’a de sens que pour les chaines OUTPUT, FORWARD, POSTROUTING
* **iptables -P [chaine] [TARGET] (ex : iptables -P OUTPUT ACCEPT]**
  + -P : --policy => permet de définir l’action par défaut pour une chaine donnée
* **iptables -I INPUT 1 -i eth0 -p tcp -s 0/0 -d 192.168.0.03 --dport 21 -j DROP (cette règle refuse toutes les connexions sur le serveur FTP)**
  + -I : --insert => permet d’insérer la règle dans un emplacement précis de la chaine
  + -p : --protocol => permet de préciser le protocole
  + -s : --source => permet de définir la source (0/0 => toutes les sources)
  + -d : --destination => permet de définir la destination
  + --dport => permet de définir le port de destination
* **Autoriser le ping sur une machine :** 
  + **iptables -I INPUT -i eth0 -p icmp --icmp-type 8 -s 0/0 -d 192.168.0.03 -d 0/0 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT**
    - --icmp-type 8 : le type 8 du protocole ICMP représente le ping
    - -m : permet de dire que cette règle ne sera matchée que pour certains états de la connexion (précisée avec --state)
    - --state :
      * NEW : Toute nouvelle connexion
      * ESTABLISHED,RELATED : connexion déjà établie
  + **iptables -I OUTPUT -i eth0 -p icmp --icmp-type 0 -s 192.168.0.03 -d 0/0 -m state --state ESTABLISHED,RELATED -j ACCEPT**
    - **--icmp-type 0 :** le type 0 du protocole ICMP représente la réponse à un type 8 (ping)

src : <https://www.youtube.com/watch?v=XKfhOQWrUVw#t=604.4285>

#### Pour avoir la liste des règles en cours

iptables -L -v

### NAT

#### Définition

* Les chain sont de type : PREROUTING, POSTROUTING et OUTPUT
* On ne peut modifier que la **destination** (DNAT) dans un chaine de type PREROUTING
* On ne peut modifier que la **source** (SNAT) dans une chaine de type POSTROUTING

La phase de **prerouting** permet de décider si le paquet sera traité en local sur le serveur ou s’il devra être redirigé vers un autre serveur. Elle intervient lorsque le paquet arrive au niveau de la carte réseau.

La phase de **postrouting** ne sert que lorsqu’il y a une redirection du paquet vers un autre host et elle sert en fait à remplacer la source du paquet avec l’adresse du mandataire (routeur) afin que le client le serveur de destination réponde au mandataire et non pas au serveur qui a initié la commande. Elle interveinjt lorsque le paquet est en train de quitter la carte réseau.

#### Commandes

iptables -t nat -A PREROUTING -p tcp --dport 8080 -j DNAT --to-destination 10.79.28.73:8080

* -t nat => Pour dire qu‘on fait du NAT
* -A PREROUTING => On append à la chaîne PREROUTING
* -p tcp => ne s’applique qu’au TCP
* --dport 8080 : ne s’applique que si le port de destination est le 8080
* -j DNAT : On souhaite modifier la destination
* --to-destination 10.79.28.73:8080 : nouvelle destination

iptables -t nat -A POSTROUTING -p tcp --dst 10.79.28.73 --dport 8080 -j SNAT --to-source 10.79.30.8

* -t nat => Pour dire qu‘on fait du NAT
* -A POSTROUTING => On append à la chaîne POSTROUTING
* -p tcp => ne s’applique qu’au TCP
* --dst (-d) 10.79.28.73 => ne s’applique que si la destination est 10.79.28.73
* --dport 8080 => ne s’applique que si le port de destination est le 8080
* -j SNAT => On souhaite modifier la source
* --to-source 10.79.30.8 => Nouvelle source

Src : <http://www.linux-france.org/prj/edu/archinet/systeme/ch62s03.html>

Ex : pour une redirection des requêtes sur le port 8080 vers la machine Ambari

// Les IP à utiliser ci-dessous sont celles de l'interface eth1 du serveur AM (serveur distant) et de eth4 pour le serveur AX (serveur local)

echo "1" > /proc/sys/net/ipv4/ip\_forward

iptables -t nat -A PREROUTING -p tcp --dport 8080 -j DNAT --to-destination 10.79.28.73:8080

iptables -t nat -A POSTROUTING -p tcp -d 10.79.28.73 --dport 8080 -j SNAT --to-source 10.79.30.8

service iptables save

service iptables restart

#### Pour avoir la liste des règles en cours :

iptables -t nat -L -v

#### Pour supprimer toutes les règles en cours :

iptables -F -t nat

#### Pour sauvegarder la conf :

service iptables save

**A noter** :

* que lorsqu’on fait un « service iptables save », les règles en cours sont prises en compte et sauvegardées dans /etc/sysconfig/iptables et elles seront rechargées au redémarrage de la machine.
* les nouvelles règles sont prises en compte directement après que la commande iptables soit exécutée (pas besoin de save ou restart)
* en revanche la suppression de règles ne sera effective qu’après le redémarrage du service (service iptables restart)

## SSL et certificats

La mise en place de SSL se fait pour 2 raisons :

* Cryptage des échanges entre 2 serveurs
* Identification : s’assurer que le client communique avec le bon serveur

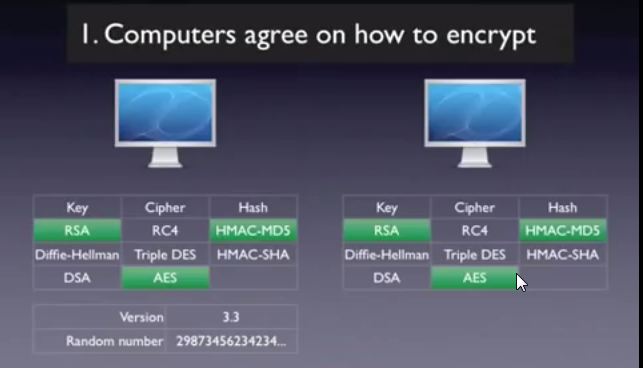
### Cryptage

Le cryptage des données commencent avec le **handshake** entre les 2 hosts. Ce handshake se déroule en 5 étapes :

1. Les hosts se mettent d’accord sur la méthode de cryptage et le client envoie un premier message Hello qui contient les informations suivantes :

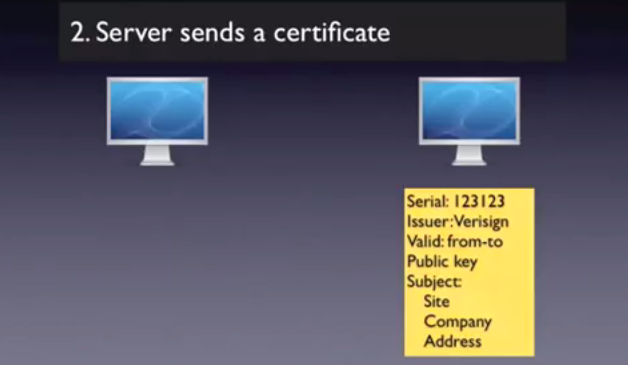
* Key (RSA, DSA, Diffie-Helman) : Key exchange method
* Cipher (RC4, Triple DES, AES) : Way of encrypting the data
* Hash (HMAC-MD5, HMAC-SHA) : Uses to generate a message authentication code to ensure the integrity of the messages
* La version de SSL (3.3 => TLS)
* A random number that is used to generate the Master secret which is then used to calculate the encryption keys

Le serveur répond à son tour en sélectionnant le key, Cipher et Hash qui vont être utilisés pour les échanges. Au final, le client a fait des propositions et le serveur a tranché.



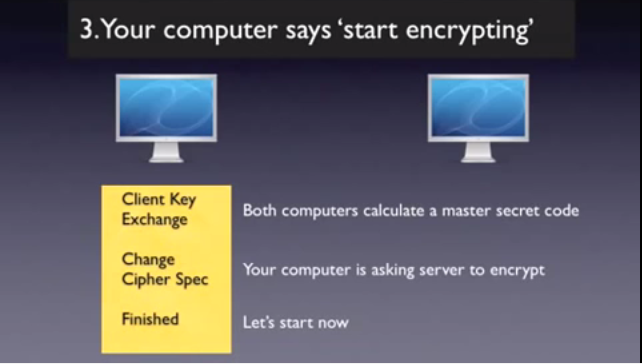
2. Ensuite le serveur va envoyer un certificat au client. Ce certificat contient :

* Des informations concernant le serveur (site, company…)
* Sa validité
* Et la clé publique du serveur



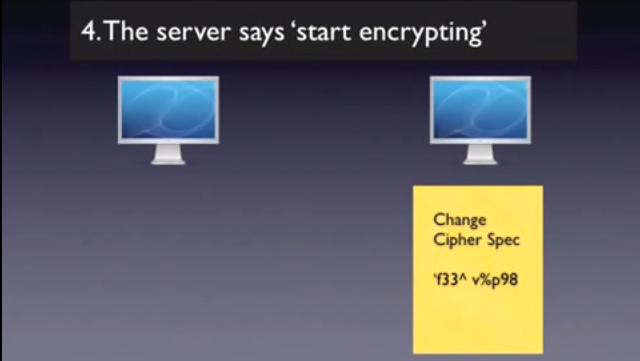
3. A réception de ce certificat, le client va commencer à crypter les échanges. Cela se fait en 3 étapes :

* Client key exchange : Les 2 hosts génèrent le Master Secret Code. Ce code permettra de crypter les échanges.
* Cipher spec : Le client décide de commencer à crypter suivant la méthode sélectionnée précédemment et l’annonce au serveur
* Message de Fin



4. A réception de ce message, le serveur va aussi :

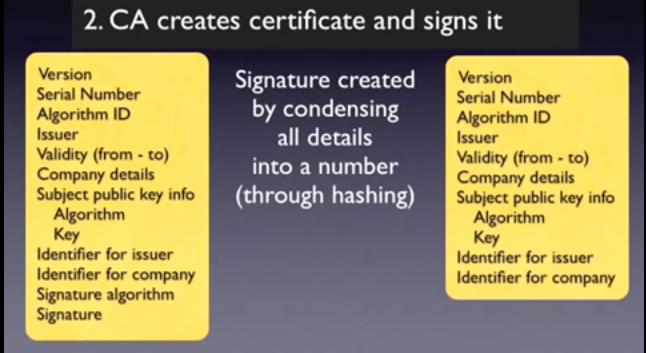
* Décider de commencer à crypter les messages avec la méthode présélectionnée.
* Et envoyer le premier crypté au client



### Identification

L’identification permet au client de s’assurer que le serveur auquel il s’adresse est bien le serveur qu’il prétend être. Voici les étapes :

1. La société propriétaire du serveur doit demander un certificat a une Autorité de Certification (Verisign). (nom de la société, son emplacement,…)
2. Le CA va alors vérifier ces informations et puis créer un certificat et le signer et puis crypter toutes les données de ce certificat dans un nombre en utilisant une clé privée. Donc quiconque qui détient la clé publique correspondante pourra décrypter le certificat.



1. La compagnie va alors installer ce certificat sur son serveur (apache, tomcat, weblogic…) et le serveur doit être configuré pour utiliser ce certificat.
2. Tous les navigateurs sont livrés avec des Root Certificates qui sont des certificats associés à des CA reconnus mondialement.
3. Chaque certificat livré dans le navigateur possède la clè publique de ces certificats. C’est ce qui permet au client (navigateur) de décrytper le certificat envoyé par le serveur et de s’assurer qu’il est valide.

### En pratique

Sur Hadoop avec ssl activé, il faut :

**1. Générer les clés publique/privées et le certificat de chaque serveur**

$ keytool -keystore {keystore} -alias localhost -validity {validity} -genkey

You need to specify two parameters in the above command:

* keystore: the keystore file that stores the certificate. The keystore file contains the private key of the certificate; therefore, it needs to be kept safely.
* validity: the valid time of the certificate in days.

Ensure that common name (CN) matches exactly with the fully qualified domain name (FQDN) of the server.

**2. Se créer sa propre autorité de certification CA :**

openssl req –new – x509 –keyout <ca-key> -out <ca-cert> -days <validity>

The generated CA is simply a public-private key pair and certificate, and it is intended to sign other certificates.

**3. Il faut ensuite ajouter le certificat de ce CA à tous les nœuds** afin que les nœuds puissent valider les certificats signés par cette autorité.

$ keytool -keystore {truststore} -alias CARoot -import -file {ca-cert}

Le trustore d’une machine contient tous les CA auxquels la machine peut faire confiance. On peut par exemple signer tous les certificats de tous les nœuds avec le même CA et partager ce même CA sur tous serveurs pour que tous les serveurs utilisent le même CA.

**4. Signer les certificats**

Il faut ensuite signer tous les certificats des serveurs avec le CA généré.

First, you need to export the certificate from the keystore of each server:

$ keytool -keystore -alias localhost -certreq -file {cert-file}

Then sign it with the CA:

$ openssl x509 -req -CA {ca-cert} -CAkey {ca-key} -in {cert-file} -out {cert-signed} -days {validity} -CAcreateserial -passin pass:{ca-password}

Finally, you need to import both the certificate of the CA and the signed certificate into the keystore:

$ keytool -keystore -alias CARoot -import -file {ca-cert}

$ keytool -keystore -alias localhost -import -file {cert-signed}

The definitions of the parameters are the following:

* **keystore:** the location of the keystore
* **ca-cert:** the certificate of the CA
* **ca-key:** the private key of the CA
* **ca-password:** the passphrase of the CA
* **cert-file:** the exported, unsigned certificate of the server
* **cert-signed:** the signed certificate of the server

src : <http://hortonworks.com/blog/deploying-https-hdfs/>

La procedure ci-dessous n’a pas fonctionné; Il faut revenir dessus. Pour résoudre on problème avec mon serveur 0301, j’ai juste copié le trustore et le keystore depuis un autre nœud.

# Fait sur le serveur 0301

mkdir /etc/security/serverKeys

chmod 755 /etc/security/serverKeys

chown yarn:hadoop /etc/security/serverKeys

cd /etc/security/serverKeys/

keytool -keystore keystore.jks -alias localhost -validity 365 –genkey

# J’ai mis le même mot de passe qie le compte root du serveur 0301

<https://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.3.6/bk_Security_Guide/content/ch_wire-ssl-httpfactory.html>

**Résumé :**

1. On génère la clé privée et le certificat sur chaque nœud. Ils sont ajoutés dans le fichier keystore.jks
2. On crée un CA auto-sign et on le déploie sur tous les nœuds dans le fichier trustore.jks
3. Sur chaque nœud, on exporte le certificat du serveur et on le signe avec le certificat du CA
4. Sur chaque nœud, on importe dans le fichier keystore.jks le certificat du serveur signé et le certificat du CA

=> Le fait d’avoir un CA lambda dans le fichier trustore d’un serveur lui permet de décrypter le certificat qui sera envoyé par le serveur distant car le CA contient la clé publique du certificat signé du serveur distant. Et c’est ce qui permet au serveur de « faire confiance » au serveur distant.

=> Le fait d’avoir signé tous les certificats des serveurs avec le même CA permet d’importer un seul et unique certificat CA sur les serveurs.

* Le fichier keystore.jks contient donc au final le certificat auto-signé du serveur (qu’il présentera aux serveurs distants pour initier un échange)
* Le fichier trustore.jks contient les certificats des CA auquel le serveur peut faire confiance (car il contient la clé publique du CA qui permettra au serveur de déchiffrer les certificats des serveurs distants)

### Génération des certificats

#### Certificat Mono-CN autosigné

keytool -genkey -keyalg RSA -alias ssl -keystore keystore.jks -validity 360 -storepass password -keypass password -dname "cn=uabdfkx01.rouen.francetelecom.fr, ou=test, o=test, c=XX"  
java HTTPSServer

#### Certificat Multi-CN autosigné

keytool -genkey -keyalg RSA -alias ssl -keystore keystore.jks -validity 360 -storepass password -keypass password -dname "cn=uabdfkx01.rouen.francetelecom.fr, ou=test, o=test, c=XX" -ext SAN=dns:uabdfkx01.rouen.francetelecom.fr,dns:uabdfkx01.nor.fr.ftgroup,dns:localhost

#### Certificat Multi-CN avec wildcard autosigné

**Les wildcards ne sont pas supportés par keytool, il faut donc utiliser openssl**

#Génère une clé privée

openssl genrsa -passout pass:password -aes256 -out server.key 2048

#Génère un certificat Multi-CN avec wildcard auto-signé qui expire dans 10 ans

openssl req -passin pass:password -x509 -sha256 -days 3652 -new -key server.key -out server.crt -subj "/C=XX/ST=test/L=test/O=test/OU=test/CN=uabdfkx01.rouen.francetelecom.fr" -config <(cat /etc/pki/tls/openssl.cnf; printf "[v3\_ca]\nsubjectAltName=@SAN\n[SAN]\nDNS.1=uabdfkx\*.rouen.francetelecom.fr\nDNS.2=uabdfkx\*.nor.fr.ftgroup\nDNS.3=localhost\n")

# On peut vérifier avec :

openssl x509 -in server.csr -text -noout

#Crée un keystore de type PKCS12 keystore à partir de la clée privée et du certificat  
openssl pkcs12 -passin pass:password -passout pass:password -export -name ssl -in server.crt -inkey server.key -out keystore.p12

#converti le PKCS12 en JKS  
keytool -importkeystore -destkeystore keystore.jks -srckeystore keystore.p12 -srcstoretype pkcs12 -alias ssl -storepass password -keypass password -srcstorepass password

#### Certificat Multi-CN avec wildcard signé par CA (non testé)

**Les wildcards ne sont pas supportés par keytool, il faut donc utiliser openssl**

#Génère une clé privée

openssl genrsa -passout pass:password -aes256 -out server.key 2048

#Génère une demande de certificat Multi-CN avec wildcard

openssl req -passin pass:password -sha256 -new -key server.key -out server.csr -subj "/C=XX/ST=test/L=test/O=test/OU=test/CN=uabdfkx01.rouen.francetelecom.fr" -config <(cat /etc/pki/tls/openssl.cnf; printf "[req]\nreq\_extensions=v3\_req\n[v3\_req]\nsubjectAltName=@SAN\n[SAN]\nDNS.1=uabdfkx\*.rouen.francetelecom.fr\nDNS.2=uabdfkx\*.nor.fr.ftgroup\nDNS.3=localhost\n")

# On peut vérifier avec :

openssl req -in server.csr -text -noout

#Crée un keystore de type PKCS12 keystore à partir de la clée privée et du certificat signé

openssl pkcs12 -passin pass:password -passout pass:password -export -name ssl -in certsigned.crt -inkey server.key -out keystore.p12

#converti le PKCS12 en JKS

keytool -importkeystore -destkeystore keystore.jks -srckeystore keystore.p12 -srcstoretype pkcs12 -alias ssl -storepass password -keypass password -srcstorepass password

PS: explication de :

-config <(cat /etc/pki/tls/openssl.cnf; printf "[req]\nreq\_extensions=v3\_req\n[v3\_req]\nsubjectAltName=@SAN\n[SAN]\nDNS.1=uabdfkx\*.rouen.francetelecom.fr\nDNS.2=uabdfkx\*.nor.fr.ftgroup\nDNS.3=localhost\n")

ça crée un fichier "temporaire" qui contient le contenu de /etc/pki/tls/openssl.cnf avec en plus ces quelques lignes à la fin du fichier :  
[req]  
req\_extensions=v3\_req  
[v3\_req]  
subjectAltName=@SAN  
[SAN]  
DNS.1=uabdfkx\*.rouen.francetelecom.fr  
DNS.2=uabdfkx\*.nor.fr.ftgroup  
DNS.3=localhost

Le nom du fichier temporaire est passé en paramètre à l'option config

#### Pour tester :



Générer un keystore.jks avec mdp = password contenant un certificat avec le même mdp.

Compiler/Lancer le programme java en pièce jointe :  
[root@uabdfkx01 cert\_java]# java HTTPSServer  
SSL server started

**Dans une autre session :**

Récupèrer le certficat :  
openssl s\_client -showcerts -connect uabdfkx01.rouen.francetelecom.fr:9999  </dev/null > cert.pem

Lancer différentes commandes curl pour vérifier que ça fonctionne comme prévu :

[root@uabdfkx01 cert\_java]# curl --cacert ./cert.pem https://uabdfkx01.rouen.francetelecom.fr:9999  
Hello  
[root@uabdfkx01 cert\_java]# curl --cacert ./cert.pem https://uabdfkx01.nor.fr.ftgroup:9999  
Hello  
[root@uabdfkx01 cert\_java]# curl --cacert ./cert.pem https://localhost:9999  
Hello  
[root@uabdfkx01 cert\_java]# curl --cacert ./cert.pem https://uabdfkx01-adm.rouen.francetelecom.fr:9999  
Hello  
[root@uabdfkx01 cert\_java]# curl --cacert ./cert.pem https://uabdfkx01:9999  
curl: (51) SSL: certificate subject name 'uabdfkx01.rouen.francetelecom.fr' does not match target host name 'uabdfkx01'

### Explication certificat et clé publique clé privée

**Clé privée et publique :**

Pour une communication chiffrée entre 2 serveurs, on peut chiffrer soit de manière **symétrique** (une seule clé) soit **asymétrique** (une clé publique et privée). Le **problème avec le chiffrement symétrique** c'est qu'il faut **envoyer une clé différente pour chaque interlocuteur** afin que les différents interlocuteurs ne puissent pas déchiffrer des messages qui ne leur sont pas attribués.

**Du coup on passe par un système asymétrique**, c’est-à-dire par une **clé privée et une publique**. Un message chiffré avec la clé privée peut être déchiffrée que par les détenteurs de la clé public et vice versa.

Le **chiffrement asymétrique** pose le problème de l**a transmission de la clé publique qui pourrait être captée par l'homme du milieu** => Il se ferait alors passer par le détenteur de la clé privée et transmettra sa propre clé publique au destinataire. Du coup l'homme du milieu peut déchiffrer les messages envoyés par le détenteur de la clé privée et peut surtout déchiffrer les messages envoyés par le détenteur de la clé publique.

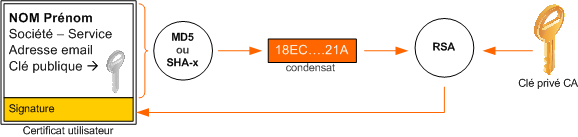
**Du coup la solution serait de passer par des certificats :**

Un certificat est un ensemble de données contenant :

- au moins une clé publique

- des informations d'identification : nom, localisation, adresse électronique…

- au moins une signature (clé privée)

[](https://commons.wikimedia.org/wiki/File:Certificat_utilisateur_contenu.png?uselang=fr)

Le certificat peut être au format X.509 ou OpenPGP

Les certificats électroniques servent à garantir :

- la **non-répudiatio**n et **l'intégrité** des données avec la signature numérique ;

- la **confidentialité** des données grâce au chiffrement des données ;

- **l'authentification** ou l'authentification forte d'un individu ou d'une identité numérique.

Les autorités de certifications diffusent leur propre clé publique. Les navigateurs incluent nativement ces clés publiques (appelés clés **publiques racines** ou **certificats racines**), elle sont utilisées pour identifier les clés publiques d'autres organismes

**Donc l'échange est le suivant :**

1. Host A initie une connexion sécurisée avec Host B pour lui envoyer sa clé publique
2. Host B envoie son propre certificat (signé avec la clé privée de l'autorité) et qui contient la clé publique du Host B
3. Host A arrive à déchiffrer le certificat avec la clé publique de l'autorité signataire, ce qui garantit que le Host B n'est pas l'homme du milieu (car ce dernier ne peut pas avoir le certificat signé de B qui a été signé par la clé privé du CA)
4. Host A chiffre sa propre clé publique avec la clé publique du host B et l'envoie au Host B
5. Host B récupère la clé publique du Host A en déchiffrant le paquet avec sa propre clé privée
6. A ce stade le Host B a la clé publique du Host A et le Host A n'a jamais envoyé la clé publique en clair.

### Commandes

#### Voir les certificats qu’un serveur présente

openssl s\_client -showcerts -connect inbdfkx01.rouen.francetelecom.fr:8443

#### Avoir des détails d’une connexion SSL à un serveur

openssl s\_client -connect 10.110.162.85:8443 -msg

#### Récupérer la clé privée d’un JKS

keytool -importkeystore -srckeystore existing-store.jks -destkeystore keystore.p12 -deststoretype PKCS12

#### Export certificate using openssl or keytool :

openssl pkcs12 -in keystore.p12 -nokeys -out cert.pem

keytool -export -keystore examplestore -alias signFiles -file Example.ce -rfc

#### Export unencrypted private key:

openssl pkcs12 -in keystore.p12 -nodes -nocerts -out key.pem

#### Avoir le detail d’un certificate

openssl x509 -in ./../allkxcertificates/inbdfkx01.nor.fr.ftgroup\_G-IGCGROUPE-G2-SERVEURS-AUTH-CERT\_pem\_chain.cer -text -noout

#### Vérifier la validité d’un certificat et des CA associés

Le CAFile contient la concaténation des certificats des CA

openssl verify -CAfile ./nouveaucertif.cer inbdfkx01.nor.fr.ftgroup\_pem.cer

OU

Le CApath contient les certificats des CA

openssl verify -CApath ./nouveaucertif.cer inbdfkx01.nor.fr.ftgroup\_pem.cer

## ULIMITS

When a process is launched, it is launched within a shell, and that shell is launched as a particular user.  This means the process inherits the ulimit values for the user who launched the shell.  If you want a process to have larger ulimits than you'd give a standard user, launch that process under a separate user account and change the ulimits for that particular account.  
  
So, you may ask, how can I view the ulimit settings for a particular process?  *ulimits -a* shows you the current shell, which isn't what a process may be using.  So a different approach is needed.  
  
Each running process has a Process ID or pid.  Typically you find the pid for a process in the /var/run directories.  Here's how to display the NTP daemon's pid:

cat /var/run/ntpd.pid

1515

Some processes have subdirectories under /var/run, or they locate their pid files in different places, so you may have to hunt to find them.  You can run a search on files with an extension of ".pid" to locate them.  
  
Once you know the pid, you can display the current running settings for that Process ID.  Here's how to show the current ulimit settings for the NTP daemon once the pid is known:

cat /proc/1515/limits

Limit                     Soft Limit           Hard Limit           Units

Max cpu time              unlimited            unlimited            seconds

Max file size             unlimited            unlimited            bytes

Max data size             unlimited            unlimited            bytes

Max stack size            10485760             unlimited            bytes

Max core file size        0                    unlimited            bytes

Max resident set          unlimited            unlimited            bytes

Max processes             39443                39443                processes

Max open files            1024                 4096                 files

Max locked memory         65536                65536                bytes

Max address space         unlimited            unlimited            bytes

Max file locks            unlimited            unlimited            locks

Max pending signals       39443                39443                signals

Max msgqueue size         819200               819200               bytes

Max nice priority         0                    0

Max realtime priority     0                    0

Max realtime timeout      unlimited            unlimited            us

You can also get a little fancier with the command and do both at once:

cat /proc/`cat /var/run/ntpd.pid`/limits

If you paste the command above into a terminal window and it doesn't work, the back tick symbols may need to be retyped.

### SYSTEM MAX LIMITS

Is there a maximum cap on all of this?  Why yes, there is.  The Linux OS can only go so high with resource consumption.  Below are some helpful commands to show these limits -- these were run on a single node test cluster, so these values will be low   
  
To display the absolute max ceiling on open files:

cat /proc/sys/fs/file-max

1002977

To display all currently open files by all processes:

lsof | wc -l

4236

To display the absolute max ceiling on open processes:

cat /proc/sys/kernel/pid\_max

32768

### ULIMIT SETTINGS AND HDP

So now that you know how the Linux OS handles all of this, what about HDP specifically?  
  
In the case of HDP processes, we are concerned mostly with the soft limit values for max open files and max processes.  We are also mainly concerned with the heavy hitting processes, such as HDFS, MapReduce, YARN and Hive.  Lighter weight processes such as Ranger or Knox can get by with the system defaults defined by the kernel or limits.conf.  
  
Be aware if you make changes to the limits.conf file, the settings can be overridden by files in the /etc/security/limits.d directory.

### AMBARI AND ULIMITS

Ambari actually gets involved in all this too, and creates files in the /etc/security/limits.d directory for certain HDP processes.  The behavior and defaults differ in various versions of Ambari, see below for some details.

#### Ambari 2.0.2

In earlier versions, such as Ambari 2.0.2, Ambari only creates ulimit files for HDFS, MapReduce and YARN.  Hive isn't touched, and inherits from the limits.conf / kernel settings which are probably way too low.  The values for the ulimits are included in template files, and are not exposed to the Ambari GUI or saved in the Ambari metadata, so you have to go behind the scenes to edit these template files.  More on this later.  
  
**Here's the ulimit config flow for Ambari 2.0.2:**

1. Ambari server templates contain the actual values and live at /var/lib/ambari-server/resources/common-services/<service>/<version>/configuration/<service>-env.xml
2. These push to Ambari Agent templates cached on each node at /var/lib/ambari-agent/cache/common-services/<service>/<version>/package/templates/<service>.conf.j2
3. Ambari Agents use these cached templates to generate the OS files under /etc/security/limits.d

#### Ambari 2.2.1.0

In later versions, such as Ambari 2.2.1.0, Ambari also created a limits.d file for Hive.  The settings for the ulimits are all exposed in the Ambari UI, but they are not searchable by "ulimit" so you have to know what to look for.    
  
**Here's the ulimit config flow for Ambari 2.2.1.0:**

1. Settings are editable in Ambari UI / saved in metastore database
2. Ambari server templates live at /var/lib/ambari-server/resources/common-services/<service>/<version>/configuration/<service>-env.xml -- you don't need to modify this file in this version
3. These push to Ambari Agent templates cached on each node at /var/lib/ambari-agent/cache/common-services/<service>/<version>/package/templates/<service>.conf.j2
4. Ambari Agents use these cached templates and data to generate the OS files under /etc/security/limits.d

### ULIMIT SETTINGS FOR HDFS

#### Check Existing Values for HDFS

We must check the values for namenodes and datanodes separately.  Please note these commands will fail if the process is not installed or not actively running on a node:

# HDFS Namenode

cat /proc/`cat /var/run/hadoop/hdfs/hadoop-hdfs-namenode.pid`/limits | grep 'open files\|processes'

Max processes             65536                65536                processes

Max open files            128000               128000               files

# HDFS Datanode

cat /proc/`cat /var/run/hadoop/hdfs/hadoop-hdfs-datanode.pid`/limits | grep 'open files\|processes'

Max processes             65536                65536                processes

Max open files            128000               128000               files

If you have trouble with the commands, retype the back tick symbols.

#### Default Values in Ambari 2.2.1.0 for HDFS

Below is a screenshot of the default values:

#### Editing the Settings in Ambari 2.2.1.0 for HDFS

1. Log into Ambari
2. Navigate to Services / HDFS
3. Search on "hdfs\_user\_" or on "limit" to locate "hdfs\_user\_nofile\_limit" and "hdfs\_user\_nproc\_limit" under Advanced hadoop-env
4. Edit the settings as needed
5. Save and restart HDFS services
6. Verify the new settings are in effect by running the "Check Existing Values for HDFS" commands listed above

#### Default Values in Ambari 2.0.2 for HDFS

The HDFS defaults are set in the Ambari server's template file as shown below:

cat /var/lib/ambari-server/resources/common-services/HDFS/2.1.0.2.0/package/templates/hdfs.conf.j2 | grep "nofile\|nproc"

{{hdfs\_user}}   - nofile 32768

{{hdfs\_user}}   - nproc  65536

#### Editing the Settings in Ambari 2.0.2 for HDFS

To make changes:

1. vi /var/lib/ambari-server/resources/common-services/HDFS/2.1.0.2.0/package/templates/hdfs.conf.j2
2. Edit the settings as needed
3. Save the file
4. Restart HDFS services from Ambari
5. Verify the new settings are in effect by running the "Check Existing Values for HDFS" commands listed above

#### Datanodes Running as Root

If you are running a Kerberized cluster, there are some special considerations for Datanodes.  When Kerberized, the datanodes inherit settings from the root user (rather than hdfs user) which may be way too low.  To fix this, there are a couple of choices: (1) increase the ulimits for the root user, which can effect many things in addition to the datanodes (not advisable), or (2) follow the steps below to run Datanodes as non-root users so they proper inherit the hdfs user's ulimit settings:

* HDP 2.4.0: <http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.4.0/bk_hdfs_admin_tools/content/ch09.html>
* HDP 2.2.4.2: [http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.2.4/bk\_hdfs\_admin\_tools/content/ch09.html](http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.2.4/bk_hdfs_admin_tools/content/ch09.html%C2%A0)
* For other versions of HDP, looks for "Running DataNodes as Non-Root" in the HDFS admin guides

### ULIMIT SETTINGS FOR MAPREDUCE

#### Check Existing Values for MapReduce

Please note this command will fail if the process is not installed or actively running on a node:

# MapReduce History Server

cat /proc/`cat /var/run/hadoop-mapreduce/mapred/mapred-mapred-historyserver.pid`/limits | grep 'open files\|processes'

Max processes             65536                65536                processes

Max open files            32768                32768                files

If you have trouble with the commands, retype the back tick symbols.

#### Default Values in Ambari 2.2.1.0 for MapReduce

Below is a screenshot of the default values:

#### Editing the Settings in Ambari 2.2.1.0 for MapReduce

1. Log into Ambari
2. Navigate to Services / MapReduce2
3. Search on "mapred\_user\_" or "limit" to locate "mapred\_user\_nofile\_limit" and "mapred\_user\_nproc\_limit" under Advanced mapred-env
4. Edit the settings as needed
5. Save and restart MapReduce2 services
6. Verify the new settings are in effect by running the "Check Existing Values for MapReduce" commands listed above

#### Default Values in Ambari 2.0.2 for MapReduce

The MapReduce defaults are set in the Ambari server's template file as shown below:

cat /var/lib/ambari-server/resources/common-services/YARN/2.1.0.2.0/package/templates/mapreduce.conf.j2 | grep "nofile\|nproc"

{{mapred\_user}}   - nofile 32768

{{mapred\_user}}   - nproc  65536

#### Editing the Settings in Ambari 2.0.2 for MapReduce

To make changes:

1. vi /var/lib/ambari-server/resources/common-services/YARN/2.1.0.2.0/package/templates/mapreduce.conf.j2
2. Edit the settings as needed
3. Save the file
4. Restart MapReduce2 services from Ambari
5. Verify the new settings are in effect by running the "Check Existing Values for MapReduce" commands listed above

### ULIMIT SETTINGS FOR YARN

#### Check Existing Values for YARN

We must check the values for Node Manager, Resource Manager and Timeline Server separately.  Please note these commands will fail if the process is not installed or actively running on a node:

# YARN Node Manager

cat /proc/`cat /var/run/hadoop-yarn/yarn/yarn-yarn-nodemanager.pid`/limits | grep 'open files\|processes'

Max processes             65536                65536                processes

Max open files            32768                32768                files

# YARN Resource Manager

cat /proc/`cat /var/run/hadoop-yarn/yarn/yarn-yarn-resourcemanager.pid`/limits | grep 'open files\|processes'

Max processes             65536                65536                processes

Max open files            32768                32768                files

# YARN Timeline Server

cat /proc/`cat /var/run/hadoop-yarn/yarn/yarn-yarn-timelineserver.pid`/limits | grep 'open files\|processes'

Max processes             65536                65536                processes

Max open files            32768                32768                files

If you have trouble with the commands, retype the back tick symbols.

#### Default Values in Ambari 2.2.1.0 for YARN

Below is a screenshot of the default values:

#### Editing the Settings in Ambari 2.2.1.0 for YARN

1. Log into Ambari
2. Navigate to Services / YARN
3. Search on "yarn\_user\_" or "limit" o locate "yarn\_user\_nofile\_limit" and "yarn\_user\_nproc\_limit" under Advanced yarn-env
4. Edit the settings as needed
5. Save and restart YARN services
6. Verify the new settings are in effect by running the "Check Existing Values for YARN" commands listed above

#### Default Values in Ambari 2.0.2 for YARN

The YARN defaults are set in the Ambari server's template file as shown below:

cat /var/lib/ambari-server/resources/common-services/YARN/2.1.0.2.0/package/templates/yarn.conf.j2 | grep "nofile\|nproc"

{{yarn\_user}}   - nofile 32768

{{yarn\_user}}   - nproc  65536

#### Editing the Settings in Ambari 2.0.2 for YARN

To make changes:

1. vi /var/lib/ambari-server/resources/common-services/YARN/2.1.0.2.0/package/templates/yarn.conf.j2
2. Edit the settings as needed
3. Save the file
4. Restart YARN services from Ambari
5. Verify the new settings are in effect by running the "Check Existing Values for YARN" commands listed above

### ULIMIT SETTINGS FOR HIVE

#### Check Existing Values for Hive

We must check the values for Hive Server 2 and Hive Metastore service separately.  Please note these commands will fail if the process is not installed or actively running on a node:

# Hive Server 2

cat /proc/`cat /var/run/hive/hive-server.pid`/limits | grep 'open files\|processes'

Max processes             1024                 39443                processes

Max open files            4096                 4096                 files

# Hive Metastore Service

cat /proc/`cat /var/run/hive/hive.pid`/limits | grep 'open files\|processes'

Max processes             1024                 39443                processes

Max open files            4096                 4096                 files

If you have trouble with the commands, retype the back tick symbols.

#### Default Values in Ambari 2.2.1.0 for Hive

Below is a screenshot of the default values:

#### Editing the Settings in Ambari 2.2.1.0 for Hive

1. Log into Ambari
2. Navigate to Services / Hive
3. Search on "hive\_user\_" or "limit" to locate "hive\_user\_nofile\_limit" and "hive\_user\_nproc\_limit" under Advanced hive-env
4. Edit the settings as needed
5. Save and restart Hive services
6. Verify the new settings are in effect by running the "Check Existing Values for Hive" commands listed above

#### Default Values in Ambari 2.0.2 for Hive

Ambari doesn't manage the ulimit settings for Hive in Ambari 2.0.2, so they inherit the standard settings for a general user:

ulimit -a

core file size          (blocks, -c) 0

data seg size           (kbytes, -d) unlimited

scheduling priority             (-e) 0

file size               (blocks, -f) unlimited

pending signals                 (-i) 7389

max locked memory       (kbytes, -l) 64

max memory size         (kbytes, -m) unlimited

open files                      (-n) 1024

pipe size            (512 bytes, -p) 8

POSIX message queues     (bytes, -q) 819200

real-time priority              (-r) 0

stack size              (kbytes, -s) 10240

cpu time               (seconds, -t) unlimited

max user processes              (-u) 7389

virtual memory          (kbytes, -v) unlimited

file locks                      (-x) unlimited

#### Editing the Settings in Ambari 2.0.2 for Hive

Ambari doesn't manage the ulimit settings for Hive in Ambari 2.0.2, so a different approach is needed to change these settings.  A file with the appropriate settings must be created under the /etc/security/limits.d directory:

1. mv /etc/security/limits.d/hive.conf /etc/security/limits.d/hive.conf.bak
2. cat "hive - nofile 64000" > /etc/security/limits.d/hive.conf
3. cat "hive - nproc 32000" >> /etc/security/limits.d/hive.conf
4. Restart Hive services from Ambari
5. Verify the new settings are in effect by running the "Check Existing Values for Hive" commands listed above

## Installation CURL en complément avec prise en compte SSL

Déposer et décompresser le dossier joint sur le serveur cible dans /usr/lib

**Prérequis :**

* Un compilateur C doit être présent sur le serveur.
  + yum install gcc
* Et  openssl-deve doit également être installé
  + yum install openssl-devel.x86\_64

Et lancer les commandes suivantes :

cd /usr/lib/curl-7.51.0/  
chmod +x  ./configure  
./configure --prefix=/root/curl --with-ssl=/usr/lib64/python2.6/site-packages/OpenSSL

Bien vérifier que la ligne suivante s'affiche avant de lancer les commandes suivantes

**SSL support:      enabled (OpenSSL)**

Commandes suivantes :

make  
make install

 Attention, ceci ne remplace pas le curl par défaut du serveur mais il vient en complément. Pour l’exécuter il faut donc passer par la commande suivante :

/usr/lib/curl/bin/curl

# GESTION UTILISATEURS

## Régénrer mot de passe utilisateur local + le faire expirer

passwd <user local> (login = mdp)  
chage -d 0 <user local>

## Récupération des emails des utilisateurs nomnatifs d’un environnement

ipa user-find --sizelimit=500 u\_ | egrep "Email address." | cut -d ':' -f 2 | sed -e 's/\ //' | sort | awk '!a[$0]++' | awk '{printf("%s;", $0)}'

ipa user-find --sizelimit=10000 u\_ | awk -F ': ' '/Email/ && ! /rouen.francetelecom.fr/ {print($2)}' | sort | awk '!a[$0]++ {printf("%s;", $0)}'

## Ajouter un utilisateur à un groupe

Se connecter au serveur ADM

Faire un kinit kinit -kt /etc/security/keytabs/a\_app\_bdfadm.headless.keytab a\_app\_bdfadm pour pouvoir être admin ipa

Lancer la commande suivante : ipa group-add-member r\_app\_bdf\_batch --users=flume

## Modification de mot de passe

* ipa passwd <login> (mettre un mot de passe temporaire) => modifie le mot de passe dans ldap et dans la BDD KDC de Kerberos (mais le mot de passe est à usage unique). C’est-à-dire qu’on peut se connecter à Hue (ldap ?) ou via le kinit <login> (KDC ?) avec ce mot de passe.
* kpasswd <login> (mettre le bon mot de passe) => met à jour le mot de passe dans le ldap et la BDD du KDC. Mot de passe à multiple utilisation.

En revanche si je fais un ipa-getkeytab sans préciser de mot de passe => Ça met à jour le mot de passe dans la BDD KDC mais pas dans le LDAP. Il n’est donc plus possible de se connecter via un kinit avec le mot de passe initial. Pour garder le mot de passe initial, il faut ajouter le paramètre -P lors du getkeytab et repréciser le mot de passe.

**/!\** Il faut re-générer les keytabs suite à une modification de mot de passe.

## Comment connaitre les groupes d'un utilisateur ?

* Faire un ipa show-user
* Sur le serveur d'ambari qui lui seul possède le client admin ipa

## Je dois pouvoir me connecter avec mon compte ccbih sur n'importe quel serveur de n'importe quel environnement ?

Non uniquement sur ADM03 (serveur de déploiement) ou le Datalab (si je suis datalabeur....)

Actuellement je n'arrive pas à me connecter en ssh avec ccbih car je ne suis pas un daipa talaber et le serveur de déploiement n'existe pas.

## Réinitialiser les mots de passe des utilisateurs

Le script suivant permet de réinitialiser le mot de passe des utilisateurs et leur envoi un mot de passe

/opt/application/bdfadm/current/bdfadmin/bdfadmin.py reinit-passwd -l <login du user>

Le script suivant permet de faire la réinitialisation des mots de passe des utilisateurs de BDFAlim qui ont la particularité d’embarquer leur mot de passe dans la keytabl de l’utilisateur applicatif bdfalim

PL\_BDFEXP\deployKeyTab\deploykeytab

## Too many failed logins - Server is unwilling to perform

ldapsearch -x -LLL -D "uid=u\_qkdp3997\_ccbih,cn=users,cn=accounts,dc=bdfpocp" -W

# OOZIE

## Mise à jour du war

Pour information j’ai tenté la commande fournie par Hortonworks…

Si jamais il y a un pb en INT, j’ai sauvegarder le war précédent dans /root de inbdfot21

Ci-dessous la log :

./bin/oozie-setup.sh prepare-war

  setting OOZIE\_CONFIG=${OOZIE\_CONFIG:-/usr/hdp/current/oozie-server/conf}

  setting CATALINA\_BASE=${CATALINA\_BASE:-/usr/hdp/current/oozie-server/oozie-server}

  setting CATALINA\_TMPDIR=${CATALINA\_TMPDIR:-/var/tmp/oozie}

  setting OOZIE\_CATALINA\_HOME=/usr/lib/bigtop-tomcat

  setting JAVA\_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86\_64

  setting JRE\_HOME=${JAVA\_HOME}

  setting CATALINA\_OPTS="$CATALINA\_OPTS -Xmx2048m -XX:MaxPermSize=256m"

  setting OOZIE\_LOG=/opt/hdp/log/oozie

  setting CATALINA\_PID=/var/run/oozie/oozie.pid

  setting OOZIE\_DATA=/var/opt/data/flat/hadoop/oozie/data

  setting OOZIE\_HTTP\_PORT=11000

  setting OOZIE\_ADMIN\_PORT=11001

  setting JAVA\_LIBRARY\_PATH=/usr/hdp/current/hadoop-client/lib/native/Linux-amd64-64

  setting OOZIE\_CLIENT\_OPTS="${OOZIE\_CLIENT\_OPTS} -Doozie.connection.retry.count=5 "

  setting OOZIE\_CONFIG=${OOZIE\_CONFIG:-/usr/hdp/current/oozie-server/conf}

  setting CATALINA\_BASE=${CATALINA\_BASE:-/usr/hdp/current/oozie-server/oozie-server}

  setting CATALINA\_TMPDIR=${CATALINA\_TMPDIR:-/var/tmp/oozie}

  setting OOZIE\_CATALINA\_HOME=/usr/lib/bigtop-tomcat

  setting JAVA\_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86\_64

  setting JRE\_HOME=${JAVA\_HOME}

  setting CATALINA\_OPTS="$CATALINA\_OPTS -Xmx2048m -XX:MaxPermSize=256m"

  setting OOZIE\_LOG=/opt/hdp/log/oozie

  setting CATALINA\_PID=/var/run/oozie/oozie.pid

  setting OOZIE\_DATA=/var/opt/data/flat/hadoop/oozie/data

  setting OOZIE\_HTTP\_PORT=11000

  setting OOZIE\_ADMIN\_PORT=11001

  setting JAVA\_LIBRARY\_PATH=/usr/hdp/current/hadoop-client/lib/native/Linux-amd64-64

  setting OOZIE\_CLIENT\_OPTS="${OOZIE\_CLIENT\_OPTS} -Doozie.connection.retry.count=5 "

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/hive-common-1.2.1.2.3.2.0-2950.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/hive-exec-1.2.1.2.3.2.0-2950.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/hive-hcatalog-core-1.2.1.2.3.2.0-2950.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/hive-metastore-1.2.1.2.3.2.0-2950.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/hive-serde-1.2.1.2.3.2.0-2950.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/hive-shims-1.2.1.2.3.2.0-2950.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/hive-webhcat-java-client-1.2.1.2.3.2.0-2950.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/jackson-core-asl-1.9.13.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/jackson-mapper-asl-1.9.13.jar

INFO: Adding extension: /usr/hdp/2.3.2.0-2950/oozie/libext/libfb303-0.9.2.jar

New Oozie WAR file with added 'ExtJS library, JARs' at /usr/hdp/current/oozie-server/oozie-server/webapps/oozie.war

INFO: Oozie is ready to be started

## Example de bundle oozie (sur G1INT2)

hdfs dfs -cat :user/a\_app\_aqs\_atv\_hdfs/project/pa\_app\_aqs\_atv\_G01R02C04/oozie/bundle\_app\_aqs\_atv/coord\_app\_aqs\_atv\_kpi/wf\_app\_aqs\_atv\_kpi/workflow.xml

## Sharelibs

### Fonctionnement

Il y a une FEX dans le dossier Oozie (U:) qui trace les étapes pour la création d’une sharelib oozie pour le cluster Orange.

Les sharelibs se trouvent sur HDFS /user/oozie/share/lib

Principales commandes :

oozie-setup sharelib create -fs FS\_URI [-locallib SHARED\_LIBRARY]

oozie admin –shareliblist

oozie admin -shareliblist pig

oozie admin –sharelibupdate

source : <http://blog.cloudera.com/blog/2014/05/how-to-use-the-sharelib-in-apache-oozie-cdh-5/>

### Utiliser une lib d’une brique X par une action Y

Il faut spécifiser dans le workflow :

oozie.action.sharelib.for.#ACTIONTYPE#=#SHARELIBNAME#

Pour une action java :

oozie.action.sharelib.for.java=log4j2.5

## Fichiers de conf sur HDFS

Pour information/rappel lorsque l’on modifie la configuration des services TEZ, HIVE ou HBASE, il est nécessaire de pousser les fichiers de configuration dans HDFS afin que les traitements Oozie puissent y accéder.

kinit -kt /etc/security/keytabs/hdfs.headless.keytab hdfs

ll /etc/tez/conf/tez-site.xml

hdfs dfs -ls /apps/tez/conf

hdfs dfs -rm -skipTrash /apps/tez/conf/tez-site.xml

hdfs dfs -put /etc/tez/conf/tez-site.xml /apps/tez/conf/tez-site.xml

hdfs dfs -chown tez:q\_default /apps/tez/conf/tez-site.xml

hdfs dfs -chmod 750 /apps/tez/conf/tez-site.xml

ll /etc/hive/conf/hive-site.xml

hdfs dfs -ls /apps/hive/conf/

hdfs dfs -rm -skipTrash /apps/hive/conf/hive-site.xml

hdfs dfs -put /etc/hive/conf/hive-site.xml /apps/hive/conf/hive-site.xml

hdfs dfs -chown hive:q\_default /apps/hive/conf/hive-site.xml

hdfs dfs -chmod 750 /apps/hive/conf/hive-site.xml

ll /etc/hbase/conf/hbase-site.xml

hdfs dfs -ls /apps/hbase/conf/hbase-site.xml

hdfs dfs -rm -skipTrash /apps/hbase/conf/hbase-site.xml

hdfs dfs -put /etc/hbase/conf/hbase-site.xml /apps/hbase/conf/hbase-site.xml

hdfs dfs -chown hbase:q\_default /apps/hbase/conf/hbase-site.xml

hdfs dfs -chmod 750 /apps/hbase/conf/hbase-site.xml

## Démarrage des bundles wf et cordinator

Exporter l’URL d’oozie pour éviter de la retaper à chaque fois

export OOZIE\_URL=http://inbdfot11.rouen.francetelecom.fr:11000/oozie

Lister les jobs

oozie jobs -len 1000 | grep

oozie jobs -len 1000 -filter status=RUNNING -localtime // On peut filtrer sur user=<U>;name=<N>;group=<G>;status=<S> )

oozie jobs -len 1000 -jobtype bundle -localtime

INFO sur UN JOB

oozie job -info 0000011-160608183435155-oozie-oozi-W

LANCER UNE JOB

oozie job -config oozie/bundle\_app\_too\_myBundle/job.properties –run

oozie job -oozie http://inbdfot11.rouen.francetelecom.fr:11000/oozie -config ./job.properties -D jobName="bundle\_app\_too\_ucb" -run -debug

LOG d'un JOB

oozie job -log 0000010-160608183435155-oozie-oozi-W

Killer un job (bundle, wf ou coord)

oozie job -kill 0000010-160608183435155-oozie-oozi-W

## Un job de test

# ------------------------------------------------------------------------------

# Environment

# ------------------------------------------------------------------------------

nameNode=hdfs://BDFPOCPNN

jobTracker=opbdf0316.rouen.francetelecom.fr:8050

kerberosRealm=BDFPOCP

queueName=q\_datalab

# ------------------------------------------------------------------------------

# Application

# ------------------------------------------------------------------------------

appRoot=${nameNode}/tmp/test

oozie.wf.application.path=${appRoot}/shell.xml

# ------------------------------------------------------------------------------

# Oozie

# ------------------------------------------------------------------------------

oozie.use.system.libpath=true

oozie.wf.rerun.failnodes=true

Creer un fichier shell.xml

<workflow-app xmlns="uri:oozie:workflow:0.5" name="TEST\_SH\_ACTION">

       <start to="init-sh" />

             <action name="init-sh">

                    <shell xmlns="uri:oozie:shell-action:0.1">

                           <job-tracker>${jobTracker}</job-tracker>

                           <name-node>${nameNode}</name-node>

                           <configuration>

                                  <property>

                                        <name>mapreduce.job.queuename</name>

                                        <value>${queueName}</value>

                                  </property>

                           </configuration>

                           <exec>shell.sh</exec>

                           <file>shell.sh#shell.sh</file>

                    </shell>

                    <ok to="end"/>

                    <error to="fail"/>

             </action>

       <kill name="fail">

             <message>Script failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>

       </kill>

       <end name="end"/>

</workflow-app>

Créer un fichier shell.sh avec comme contenu :

hdfs dfs -rm -r -skipTrash /tmp/hourra

hdfs dfs -mkdir /tmp/hourra

Commandes à passer pour lancer le workflow :

kinit –kt /etc/security/keytabs/hdfs.headless.keytab hdfs

hdfs dfs –rm –f -skipTrash /tmp/test

kdestroy -A

kinit u\_uglo8383\_adm

hdfs dfs -mkdir -p /tmp/test

hdfs dfs -put -f shell\* /tmp/test

oozie job -oozie <http://uabdfot01.rouen.francetelecom.fr:11000/oozie> -config shell.properties -run

# Hue

## Dossier temporaire par défaut lors du chargement de fichier

# Modifier le paramètre suivant

FILE\_UPLOAD\_TEMP\_DIR = None

#par cette valeur

FILE\_UPLOAD\_TEMP\_DIR = "/var/opt/data/flat/hue"

#Rédemarrer Hue

service hue restart

# Tez

## SWIMLANES, un parser python des logs TEZ :

<https://github.com/apache/tez/tree/master/tez-tools/swimlanes>

## Counter

Si le nombre de counter estr dépassé alors que nous en avons suffisamment configuré alors il se peut que ce soit un problème de mémoire. du coup il faut l’augmenter tez.task.resource.memory.mb=2048

# Yarn

## Fonctionnement

### Différences entre MR1 et Yarn (MR2)

Many will draw parallels between YARN and the existing Hadoop MapReduce system (MR1 in Apache Hadoop 1.x). However, the key difference is the new concept of an **ApplicationMaster**.

The **ApplicationMaster** is responsible for negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the containers and their resource consumption. It has the responsibility of **negotiating** appropriate **resource** **containers** from the **ResourceManager**, **tracking** their **status** and **monitoring** **progress**

We shifted that (that = provide **fault-toleranc**e for resources) to become a primary responsibility of the **ApplicationMaster** instance. Furthermore, since there is an instance of an ApplicationMaster per application, the ApplicationMaster itself **isn’t** a common **bottleneck** in the cluster.

Moving all application framework specific code into the **ApplicationMaster** generalizes the system so that we can now **support multiple frameworks** such as MapReduce, MPI and Graph Processing.

### Resource Model

An application (via the ApplicationMaster) can request resources with highly specific requirements such as:

* Resource-name (hostname, rackname – we are in the process of generalizing this further to support more complex network topologies with [YARN-18](https://issues.apache.org/jira/browse/YARN-18)).
* Memory (in MB)
* CPU (cores, for now)
* In future, expect us to add more resource-types such as disk/network I/O, GPUs etc.

### ResourceRequest and Container

**YARN** is designed to allow individual applications (via the ApplicationMaster) to utilize cluster **resources in a shared, secure and multi-tenant** manner. Also, it remains aware of cluster topology in order to efficiently schedule and **optimize data access i.e. reduce data motion for applications** to the extent possible.

Essentially an application can ask for specific resource requests via the ApplicationMaster to satisfy its resource needs. The **Scheduler responds to a resource request by granting a container**, which satisfies the requirements laid out by the ApplicationMaster in the initial ResourceRequest.

Let’s look at the ResourceRequest – it has the following form:

<resource-name, priority, resource-requirement, number-of-containers>

Let’s walk through each component of the **ResourceRequest** to understand this better.

* **resource-name** is either **hostname**, **rackname** or \* to indicate no preference. In future, we expect to support even more complex topologies for virtual machines on a host, more complex networks etc.
* **priority** is intra-application priority for this request (to stress, this isn’t across multiple applications).
* resource-requirement is required capabilities such as **memory**, **cpu** etc. (at the time of writing YARN only supports memory and cpu).
* **number-of-containers** is just a multiple of such containers.

Essentially, the Container is the resource allocation, which is the successful result of the ResourceManager granting a specific ResourceRequest. A **Container** **grants** **rights** to an **application** to **use** a **specific** **amount** of **resources** (memory, cpu etc.) on a **specific** **host**.

### Container Specification during Container Launch

While a Container, as described above, is merely a right to use a specified amount of resources on a specific machine (NodeManager) in the cluster, the **ApplicationMaster** has to **provide considerably** more **information** to the **NodeManager** to actually launch the container.

YARN allows applications to launch any process and, unlike existing Hadoop MapReduce in hadoop-1.x (aka MR1), **it isn’t limited to Java applications** alone.

The YARN Container launch specification API is platform agnostic and contains:

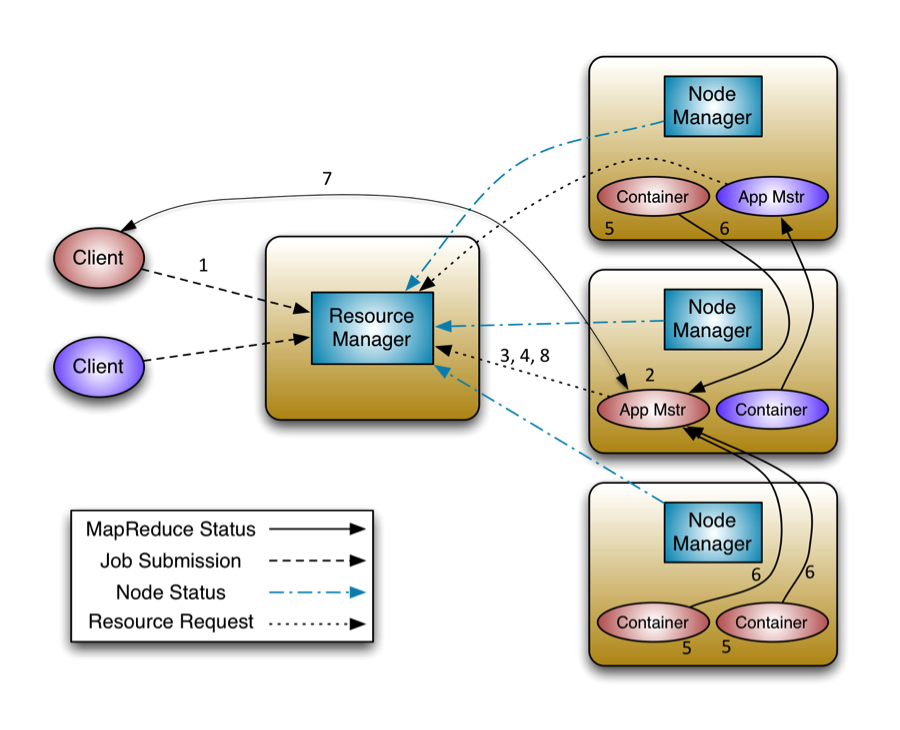
* Command line to launch the process within the container.
* Environment variables.
* Local resources necessary on the machine prior to launch, such as jars, shared-objects, auxiliary data files etc.
* Security-related tokens.

This allows the ApplicationMaster to work with the NodeManager to launch containers ranging from **simple shell scripts to C/Java/Python processes on Unix/Windows** to full-fledged **virtual machines** (e.g. KVMs).

### Résumé

Let’s walk through an application execution sequence (steps are illustrated in the diagram):

1. A **client** program ***submits*** the application, including the necessary specifications to *launch the application-specific ApplicationMaster* itself.
2. The **ResourceManager** assumes the responsibility to **negotiate** a **specified** **container** in which to start the **ApplicationMaster** and then *launches* the ApplicationMaster.
3. The **ApplicationMaster**, on boot-up, ***registers*** with the **ResourceManager** – the registration allows the **client** **program** to **query** the **ResourceManager** for details, which allow it to **directly communicate with its own ApplicationMaster**.
4. During normal operation the **ApplicationMaster** **negotiates** appropriate **resource** **containers** via the **resource-request** protocol.
5. On successful container allocations, the **ApplicationMaster** **launches** the **container** by providing the container **launch specification** to the **NodeManager**. The launch specification, typically, includes the necessary information to **allow** the **container** to **communicate** with the **ApplicationMaster** itself.
6. The application code executing within the container then **provides necessary information (progress, status etc.)** to its **ApplicationMaster** via an *application-specific protocol*.
7. During the application execution, the **client** that submitted the program communicates directly with the **ApplicationMaster** to get **status**, **progress** **updates** etc. via an application-specific protocol.
8. Once the **application is complete**, and all necessary work has been finished, the **ApplicationMaster** **deregisters** with the **ResourceManager** and shuts down, allowing its **own container to be repurposed**



## Elasticité

Bonjour à tous,

Suite au problème remonté lors de la dernière réunion de suivi migration G2 concernant l’élasticité de la queue IPI, j’ai effectué plusieurs tests de configuration afin de comprendre les relations entre l’élasticité d’une file yarn  enfant et l’élasticité de sa file Yarn parent.

Définition des propriétés utilisées  :

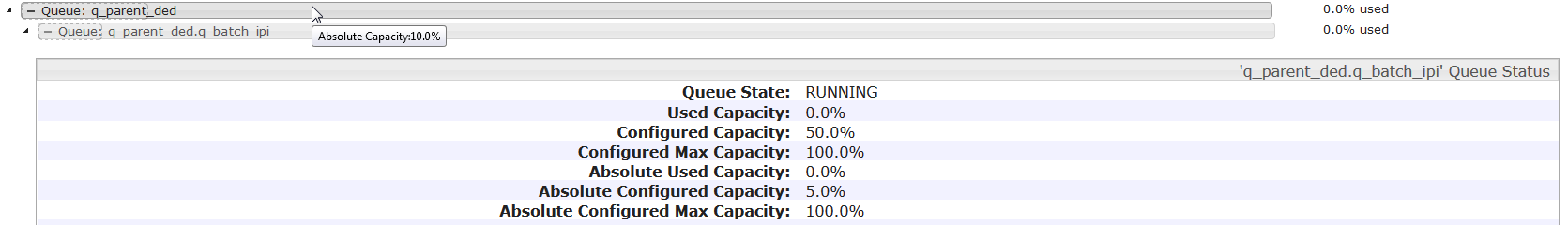
* Configured Capacity : Pourcentage de capacité mémoire que la file yarn peut réserver sur le cluster relativement à sa file yarn parent. Cette valeur est définie par l’administrateur
* Configured Max Capacitty : Pourcentage max de capacité mémoire que la file yarn peut réserver sur le cluster relativement à sa file yarn parent. Cette valeur est définie par l’administrateur
* Facteur d’élasticité configuré : Facteur entre les propriétés Configured Capacity et Configured Max Capacitty
* Absolute Capacity : Pourcentage de capacité que peut réserver la file yarn sur le cluster. Cette valeur est calculée par Yarn en fonction de la propriété Configured Capacity de la file enfant et parent)
* Absolute Max Capacitty : Pourcentage max de capacité que peut réserver la file yarn sur le cluster. Cette valeur est calculée par Yarn en fonction de la propriété Configured Max Capacity de la file enfant et parent)
* Facteur d’élasticité absolu : Facteur entre les propriétés Absolute Capacity et Absolute Max Capacitty.

Voici les tests réalisés :

* **Configuration 1 - file enfant élastique et file parent élastique**
  + file parent (enfant de la file root):
    - Configured Capacity = 10%
    - Configured Max Capacitty = 100%
      * Facteur d’élasticité Configuré => 10
    - Absolute Capacity => 10%
    - Absolute Max Capacitty => 100%
      * Facteur d’élasticité absolu => 10
  + file enfant :
    - Configured Capacity = 50%
    - Configured Max Capacitty = 100%
      * Facteur d’élasticité Configuré => 2
    - Absolute Capacity => 5%
    - Absolute Max Capacitty => 100%
      * Facteur d’élasticité absolu => 20

**Conclusion, le facteur d’élasticité absolu de la file enfant est proportionnelle à la file parent**.

Facteur d’élasticité absolu file enfant **=** Facteur d’élasticité Configuré file enfant **X** Facteur d’élasticité Configuré file parent **= 2 X 10 = 20**



* **Configuration 2 - file enfant non élastique et file parent élastique (configuration actuelle de la file q\_batch\_ipi en G2PROD)**
  + file parent (enfant de la file root):
    - Configured Capacity = 10%
    - Configured Max Capacitty = 100%
      * Facteur d’élasticité Configuré => 10
    - Absolute Capacity => 10%
    - Absolute Max Capacitty => 100%
      * Facteur d’élasticité absolu => 10
  + file enfant :
    - Configured Capacity = 50%
    - Configured Max Capacitty = 50%
      * Facteur d’élasticité Configuré => 1
    - Absolute Capacity => 5%
    - Absolute Max Capacitty => 50%
      * Facteur d’élasticité absolu => 10

**Conclusion, le facteur d’élasticité absolu de la file enfant reste proportionnelle à la file parent bien qu’aucune élasticité n’a été configuré sur la file enfant.**

Facteur d’élasticité absolu file enfant **=** Facteur d’élasticité Configuré file enfant **X** Facteur d’élasticité Configuré file parent **= 1 X 10 = 10**



* **Configuration 3 - file enfant non élastique et file parent non élastique**
  + file parent (enfant de la file root):
    - Configured Capacity = 10%
    - Configured Max Capacitty = 10%
      * Facteur d’élasticité Configuré => 1
    - Absolute Capacity => 10%
    - Absolute Max Capacitty => 100%
      * Facteur d’élasticité absolu => 1
  + file enfant :
    - Configured Capacity = 50%
    - Configured Max Capacitty = 50%
      * Facteur d’élasticité Configuré => 1
    - Absolute Capacity => 5%
    - Absolute Max Capacitty => 5%
      * Facteur d’élasticité absolu => 1

**Conclusion, si l’on souhaite aucune élasticité sur la file enfant, il faut configurer aucune élasticité sur la file enfant et la file parent.**

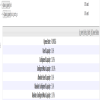
Facteur d’élasticité absolu file enfant **=** Facteur d’élasticité Configuré file enfant **X** Facteur d’élasticité Configuré file parent **= 1 X 1 = 1**



* **Configuration 4 - file enfant élastique et file parent non élastique**
  + file parent (enfant de la file root):
    - Configured Capacity = 10%
    - Configured Max Capacitty = 10%
      * Facteur d’élasticité Configuré => 1
    - Absolute Capacity => 10%
    - Absolute Max Capacitty => 10%
      * Facteur d’élasticité absolu => 1
  + file enfant :
    - Configured Capacity = 50%
    - Configured Max Capacitty = 100%
      * Facteur d’élasticité Configuré => 2
    - Absolute Capacity => 5%
    - Absolute Max Capacitty => 10%
      * Facteur d’élasticité absolu => 2

**Conclusion, le facteur d’élasticité absolu de la file enfant reste proportionnelle à la file parent bien qu’aucune élasticité n’a été configuré sur la file parent.**

Facteur d’élasticité absolu file enfant **=** Facteur d’élasticité Configuré file enfant **X** Facteur d’élasticité Configuré file parent **= 1 X 2 = 2**



## Refresh Yarn Scheduler

yarn rmadmin -refreshQueues

## Get Queeu status

yarn queue -status <QueueName>

## Afficher les droits d’un utilisateur d’une file de traitement

hadoop queue -showacls u\_qddp3997\_cbih

## Delay Scheduling and locality requirement

All the YARN schedulers try to honor locality requests. On a busy cluster, if an application requests a particular node, there is a good chance that other containers are running on it at the time of the request. The obvious course of action is to immediately loosen the locality requirement and allocate a container on the same rack. However, it has been observed in practice that waiting a short time (no more than a few seconds) can dramatically increase the chances of being allocated a container on the requested node, and therefore increase the efficiency of the cluster. This feature is called delay scheduling, and it is supported by both the Capacity Scheduler and the Fair Scheduler.

Every node manager in a YARN cluster periodically sends a heartbeat request to the resource manager—by default, one per second. Heartbeats carry information about the node manager’s running containers and the resources available for new containers, so each heartbeat is a potential scheduling opportunity for an application to run a container.

When using delay scheduling, the scheduler doesn’t simply use the first scheduling opportunity it receives, but waits for up to a given maximum number of scheduling opportunities to occur before loosening the locality constraint and taking the next scheduling opportunity.

For the Capacity Scheduler, delay scheduling is configured by setting yarn.scheduler.capacity.node-locality-delay to a positive integer representing the number of scheduling opportunities that it is prepared to miss before loosening the node constraint to match any node in the same rack.

## Gestion des utilisateurs et administrateurs d’une file de traitement

* yarn.scheduler.capacity.root.q\_default.acl\_administer\_jobs=admins
* yarn.scheduler.capacity.root.q\_default.acl\_submit\_applications=q\_default q\_default

## Definition des files d’exécution par defaut

* Il est possible de modifier le paramètre global définissant la file par défaut dans le fichier mapred-site.xml : mapreduce.job.queuename default
* Pour les jobs tez, l’ajout de la propriété tez.queue.name dans le tez-site.xml et dans le hive-site.xml permet également de définir la file par défaut.

## Containers distants

Rechercher dans les logs YARN les termes « RMContainerAllocator », « HostLocal » et « RackLocal » qui permettent d’identifier s’il y a des containers distants (container qui manipulent des données récupérées sur le réseau)

# Ambari

## Créer des alertes custom

<https://github.com/monolive/ambari-custom-alerts>

## Les versions des compostants affichés dans l’IHM d’ambari

Se trouvent dans ce fichier qu’il faut modifier et redmérrer ambari server

/var/lib/ambari-server/resources/stacks/HDP/2.3/services/KAFKA/metainfo.xml

## Déclarer un nouveau host dans Ambari

L’ajout d’un nouvel host dans Ambari se fait en 2 étapes :

1. La partie registration : Qui est faite automatiquement à partir du moment où le host a un service ambari-agent qui communique bien avec le serveur Ambari
2. L’ajout du serveur : Se fait via l’IHM Ambari ou via l’API REST.

Il faut ensuite passer par « hosts > add host » et sélectionner les composants à installer sur ce nouveau host.

**Note :** On a du commenter le « set-user » dans le hook d’Ambari car il cherchait l’utilisateur en local (et pas dans l’ipa) (/var/lib/ambari-agent/cache/stacks/HDP/2.0.6/hooks/before-ANY/scripts/hook.py)

Autre alternative via api REST : <https://cwiki.apache.org/confluence/display/AMBARI/Add+a+host+and+deploy+components+using+APIs>

curl --user u\_qkdp3997\_adm -i -X POST -H "X-Requested-By: ambari" http://localhost:8080/api/v1/clusters/BDFINT1/hosts/inbdfax13.rouen.francetelecom.fr

## Supprimer un host

Il faut stopper tous les servies sur le host (à faire à partir d’Ambari)

Puis supprimer les composants

curl -u u\_qkdp3997\_adm:'0112358Fibo!' -H "X-Requested-By: ambari" -X DELETE <http://inbdfam01.rouen.francetelecom.fr:8080/api/v1/clusters/BDFPOCHP/hosts/inbdfka01.rouen.francetelecom.fr/host_components/METRICS_MONITOR>

Puis supprimer le host

curl -u u\_qkdp3997\_adm:'0112358Fibo!' -H "X-Requested-By: ambari" -X DELETE http://inbdfam01.rouen.francetelecom.fr:8080/api/v1/clusters/BDFPOCHP/hosts/inbdfka01.rouen.francetelecom.fr

## Installation de service/composant

### Add service to cluster

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H 'X-Requested-By: ambari' -X POST '{"ServiceInfo":{"service\_name":"FLUME"}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/FLUME

### Add component to the service

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H "X-Requested-By: ambari" -X POST http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/FLUME/components/FLUME\_HANDLER

### Create configuration

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H 'X-Requested-By:admin' -X POST -d '{ "type": "flume-env", "tag": "INITIAL1", "properties" : { "content" : "default", "flume\_conf\_dir" : "/etc/flume/conf", "flume\_log\_dir" : ”/opt/hdp/log/flume", "flume\_run\_dir" : "/var/run/flume", "flume\_user" : "flume" }}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/configurations

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H 'X-Requested-By:admin' -X POST -d '{ "type": "flume-conf", "tag": "INITIAL1", "properties" : { "content" : "default"}}'

http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/configurations

### Apply configuration

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H 'X-Requested-By:admin' -X PUT -d '{"Clusters": {"desired\_configs": { "type": "flume-env", "tag" :"INITIAL" }}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H 'X-Requested-By:admin' -X PUT -d '{"Clusters": {"desired\_configs": { "type": "flume-conf", "tag" :"INITIAL" }}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2

### Create host components

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H 'X-Requested-By: ambari' -X POST http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/hosts/dvbdfax21.rouen.francetelecom.fr/host\_components/FLUME\_HANDLER

### Install component to host

Assign the role of the component using the following command:

curl -u u\_qkdp3997\_adm -H "X-Requested-By: Ambari" -X POST http://localhost:8080/api/v1/clusters/BDFPOCHP/hosts/inbdf0301.rouen.francetelecom.fr/host\_components/JOURNALNODE

Install the service.

curl -u admin:kz\_vNW4fF6JVWd9gbalU -i -H "X-Requested-By: ambari" -X PUT -d '{"RequestInfo": {"context": "Install Flume"}, "Body":{"HostRoles": {"state": "INSTALLED"}}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/hosts/dvbdfax21.rouen.francetelecom.fr/host\_components/FLUME\_HANDLER

## Désinstallation de service/composant

### Delete component from host

curl -u u\_qkdp3997\_adm -i -H 'X-Requested-By: ambari' -X DELETE http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/hosts/dvbdfax21.rouen.francetelecom.fr/host\_components/FLUME\_HANDLER

### Delete component from the service

curl -u u\_qkdp3997\_adm -i -H "X-Requested-By: ambari" -X DELETE http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/FLUME/components/FLUME\_HANDLER

### Delete service from cluster

curl -u u\_qkdp3997\_adm -i -H "X-Requested-By: ambari" -X DELETE http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/FLUME

## Other commands

### Pour récupérer les status des alertes

Pour retrouver le nomù des alertes :

<http://uabdfam01-adm.rouen.francetelecom.fr:8080/api/v1/clusters/BDFPOCP/alert_definitions>

curl -u u\_qkdp3997\_adm -H "X-Requested-By: ambari" -X GET "**http://inbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFINT2/services/KAFKA/alerts?Alert/definition\_name=kafka\_service\_check&fields=Alert/definition\_name,Alert/state**"

{

  "href" : "http://inbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFINT2/services/KAFKA/alerts?Alert/definition\_name=kafka\_service\_check&fields=Alert/definition\_name,Alert/state",

  "items" : [

    {

      "href" : "http://inbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFINT2/services/KAFKA/alerts/601",

      "Alert" : {

        "cluster\_name" : "BDFINT2",

        "definition\_id" : 301,

        "definition\_name" : "kafka\_service\_check",

        "host\_name" : "inbdfka21.rouen.francetelecom.fr",

        "id" : 601,

        "service\_name" : "KAFKA",

        "state" : "OK"

      }

    },

    {

      "href" : "http://inbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFINT2/services/KAFKA/alerts/602",

      "Alert" : {

        "cluster\_name" : "BDFINT2",

        "definition\_id" : 301,

        "definition\_name" : "kafka\_service\_check",

        "host\_name" : "inbdfka22.rouen.francetelecom.fr",

        "id" : 602,

        "service\_name" : "KAFKA",

        "state" : "OK"

      }

    }

  ]

OU

curl -u u\_qkdp3997\_adm -H "X-Requested-By: ambari" -X GET "**http://inbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFINT2/services/KAFKA/alerts?Alert/definition\_name=kafka\_service\_check&format=summary**"

{

  "href" : "http://inbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFINT2/services/KAFKA/alerts?Alert/definition\_name=kafka\_service\_check&format=summary",

  "alerts\_summary" : {

    "OK" : {

      "count" : 2,

      "original\_timestamp" : 1473178380633,

      "maintenance\_count" : 0

    },

    "WARNING" : {

      "count" : 0,

      "original\_timestamp" : 0,

      "maintenance\_count" : 0

    },

    "CRITICAL" : {

      "count" : 0,

      "original\_timestamp" : 0,

      "maintenance\_count" : 0

    },

    "UNKNOWN" : {

      "count" : 0,

      "original\_timestamp" : 0,

      "maintenance\_count" : 0

    }

  }

Plus d’infos : <https://github.com/apache/ambari/blob/trunk/ambari-server/docs/api/v1/alerts.md>

### Pour récupérer les config des composants

curl -i -u u\_qkdp3997\_adm -H 'X-Requested-By:admin' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/configurations/service\_config\_versions?is\_current=true

### Install service on cluster

curl -u u\_qkdp3997\_adm -i -H "X-Requested-By: ambari" -X PUT -d '{"ServiceInfo": {"state" : "INSTALLED"}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/FLUME

### Récupérer la liste des composants d’un host

curl -k -u u\_qkdp3997\_adm -H "X-Requested-By:ambari" -i -X GET http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/hosts/dvbdfka21.rouen.francetelecom.fr

### Récupere le statut d’un composant

curl -k -u u\_qkdp3997\_adm -H "X-Requested-By:ambari" -i -X GET http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/hosts/dvbdfka21.rouen.francetelecom.fr/host\_components/YARN\_CLIENT

### Récupérer la liste des services

curl -u u\_qkdp3997\_adm -H "X-Requested-by:ambari" -i -k -X GET http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/KNOX/

### Arret d’un service

curl -i -k -u u\_qkdp3997\_adm -H "X-Requested-By: ambari" -X PUT -d '{"RequestInfo":{"context":"Stop Service"},"Body":{"ServiceInfo":{"state":"INSTALLED"}}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/KAFKA

### Arrêt d’un composant sur un host spécifique

curl -i -u u\_qkdp3997\_adm -H "X-Requested-By: ambari" -X PUT -d '{"RequestInfo":{"context":"Stop Kafka Component"},"Body":{"HostRoles":{"state":"INSTALLED"}}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/hosts/dvbdfka21.rouen.francetelecom.fr/host\_components/KAFKA\_BROKER

### Arrêt de tous les composants d’un service

curl -i -u u\_qkdp3997\_adm -H "X-Requested-By: ambari" -X PUT -d '{"RequestInfo":{"context":"Stop All Components"},"Body":{"ServiceComponentInfo":{"state":"INSTALLED"}}}' http://dvbdfam21.rouen.francetelecom.fr:8080/api/v1/clusters/BDFDEV2/services/KAFKA/components/KAFKA\_BROKER

### Récupérer la liste des composants associés à un service

curl –i -u admin:admin -H "X-Requested-By: ambari" -X GET  http://AMBARI\_SERVER\_HOST:8080/api/v1/clusters/c1/services/SERVICENAME

## Dossier d’installation de hdp (problème du lost+found)

Le problème du dossier hdp (qui est le dossier d’installation de HDP) est que quand on monte un espace disque sur un point de montage, alors celui-ci contient un dossier nommé lost+found. Or ce dossier pose problème à Hortonworks.

Donc la solution serait de monter le volume logique qui va contenir le dossier hdp dans un dossier hdp\_mount puis de faire pointer le dossier /usr/hdp sur ce dossier hdp\_mount. De cette manière Ambari ne verra pas le dossier lost+found dans /usr/hdp.

La manipulation est la suivante :

1. On crée un volume logique qui va contenir le dossier hdp puis on le formate au format ext4
2. On crée le dossier /usr/hdp\_mount/hdp
3. On monte le volume logique créé sur le point de montage /usr/hdp\_mount
4. On ajoute le montage dans le fichier fstab
5. Et on ajoute le bind qui permet de faire pointer le dossier /usr/hdp sur /usr/hdp\_mount/hdp
6. On applique le montage

lvcreate -L 3G -n hadoop\_lv infravg

mkfs.ext4 /dev/infravg/hadoop\_lv

mkdir -p /usr/hdp\_mount/

mount /dev/infravg/hadoop\_lv /usr/hdp\_mount

mkdir -p /usr/hdp\_mount/hdp

vim /etc/fstab

/dev/infravg/hadoop\_lv /usr/hdp\_mount ext4 defaults 1 2

/usr/hdp\_mount/hdp /usr/hdp none bind 0 0

mount -a

**Note pour le bind** : Si l’on veut déplacer le contenu du dossier que l’on souhaite binder, il faut d’abord déplacer son contenu dans un dossier temp, puis binder le dossier et enfin redéplacer le contenu dans le dossier bindé.

## Mise à jour Ambari

Les binaires sont téléchargées dans ce dossier sur le serveur transverse de déploiement (Ansible).

/var/opt/data/flat/bdfops/files/repository/ambary

Ensuite, il est fait référence à ce dépôt, dans la conf du dépôt sur le serveur Ambari par exemple

[root@inbdfam01 yum.repos.d]# cat ambari.repo

[ambari-2.1.2.1]

name=Ambari 2.1.2.1

baseurl=http://inbdfam11.rouen.francetelecom.fr/ambari/centos6/2.x/updates/2.1.2.1

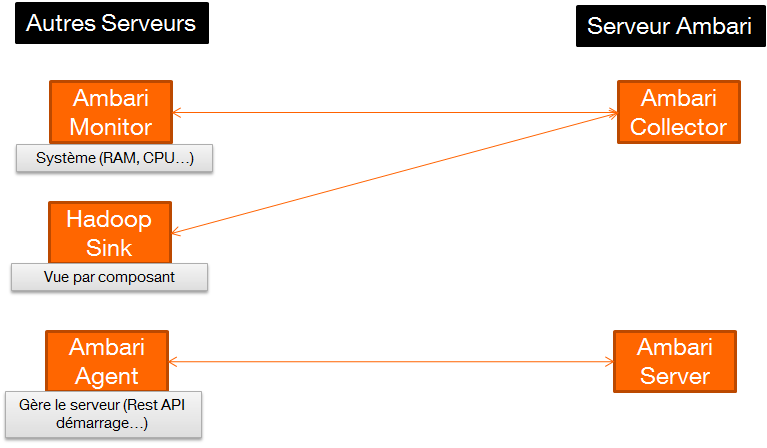
gpgcheck=0

gpgkey=http://bdf-mgmt.si.francetelecom.fr/ambari/centos6/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins

enabled=1

priority=1

## Metrics



## Récupérer le bluprint d’un cluster

Se connecter à Ambari puis taper l’url suivante : <http://xxx.xxx.xxx.xxx:8080/api/v1/clusters/BDFDEV1?format=blueprint>

## Librairie

/var/lib/ambari-server/resources/common-services/

## Versions des services sur l’IHM

Après avoir installé Kafka sur la préproduction PP2 de la G2, la version qui est proposée par Ambari 2.2 n’est pas celle qui a été installée.

En effet, sur Ambari la version proposée est la 0.9.0.2.3 alors que la version réellement installée est la 0.8.2.2.3.

Modifier le fichier "**/var/lib/ambari-server/resources/stacks/HDP/2.3/services/KAFKA/metainfo.xml**" sur le serveur Ambari :

<metainfo>

  <schemaVersion>2.0</schemaVersion>

  <services>

    <service>

      <name>KAFKA</name>

      <version>0.9.0.2.3</version>

[...]

    </service>

 </services>

</metainfo>

par

<metainfo>

  <schemaVersion>2.0</schemaVersion>

  <services>

    <service>

      <name>KAFKA</name>

      <version>0.8.2.2.3</version>

[...]

    </service>

 </services>

</metainfo>

# Kafka

## Vider un topic

Diminuer la rétention

/usr/hdp/2.3.2.0-2950/kafka/bin/kafka-topics.sh --zookeeper opbdf0316.rouen.francetelecom.fr:2181 --alter --topic z\_app\_2ip\_kafka\_socle --config retention.ms=1000

Verifier la suppression

/usr/hdp/2.3.2.0-2950/kafka/bin/kafka-console-consumer.sh --zookeeper opbdf0316.rouen.francetelecom.fr:2181 --topic z\_app\_2ip\_kafka\_socle --from-beginning --security-protocol PLAINTEXTSASL

Puis remettre la rétention

/usr/hdp/2.3.2.0-2950/kafka/bin/kafka-topics.sh --zookeeper opbdf0316.rouen.francetelecom.fr:2181 --alter --topic z\_app\_2ip\_kafka\_socle --config retention.ms=36000

## SASL, Kerberos et Jaas

Les différents méthodes de communication de Kerberos (avec les clients producer ou consommateurs) sont les suivantes :

* PLAINTEXT : N auth N Chiffrement
* PLAINTEXT SASL : Y auth Kerberos N Chiffrement
* SSL : N auth Y Chiffrement
* SASL\_SSL : Y auth Kerberos Y Chiffrement
* ssl.client.auth (boolean) : auth par certificat

A noter que :

* Kafka 0.8 Officiel
  + Ancienne API : N SSL N Kerberos
* Kafka 0.8.2 (Hortonworks)
  + Ancienne API (patchée) : N SSL Y Kerberos (=> C’est notre contexte HDP 2.3.2)
* Kafka 0.9 (Hortonworks)
  + Ancienne API (patchée) : N SSL Y Kerberos (=> C’est notre contexte HDP 2.3.4)
  + Nouvelle API  : N SSL Y Kerberos
* Kafka 0.9 Officiel
  + Ancienne API: N SSL Y Kerberos

**/!\** Défaut Ambari : Le protocole de communication écrit dans le fichier de conf de Kafka (advertised.listeners=) est écrasé par le protocole défini dans le paramètre security.inter.broker.protocol

Le fichier jaas de Flume (en tant que producer) permet à Kafka (à travers la librairie Kafka dans Flume) de retrouver le protocole de communication définit sur Kafka.

Le fichier jaas de Flume (en tant que consumer) permet à Zooeeper (à travers la librairie Zookeeper dans Flume) de retrouver le protocole de communication définit sur Zookeper (ex : Kerberos).

## Superviser

### Interface graphique

J'ai installé sur la preprod G2 un petit outil, [KafkaOffsetMonitor](https://github.com/laurentedel/KafkaOffsetMonitor), qui permet de surveiller la progression des consumers et les lags entre consumers et brokers.

On peut voir les consumer groups, les topics de chacun, etc ce qui permet de se faire une bonne idée de la façon dont on consomme la queue et à quelle vitesse elle grossit.

J'ai forké le projet pour que les ressources soient en local (pas d'accès internet nécessaire) et recompilé un assembly pour avoir un jar auto-porteur, et mis dans le projet le jar pour qu'il n'y ait même pas besoin de le compiler :)

<https://github.com/laurentedel/KafkaOffsetMonitor/blob/master/KafkaOffsetMonitor-assembly-0.3.0-SNAPSHOT.jar>

L'utilisation est très simple puisqu'il suffit de lancer le jar avec quelques paramètres :

[root@opbdf0810 ~]# java -cp KafkaOffsetMonitor-assembly-0.3.0-SNAPSHOT.jar \

     com.quantifind.kafka.offsetapp.OffsetGetterWeb \

     --zk opbdf0417.rouen.francetelecom.fr,opbdf0617.rouen.francetelecom.fr,opbdf0819.rouen.francetelecom.fr \

     --port 8080 \

     --refresh 10.seconds \

     --retain 2.days

Le jar contient un petit serveur web sur le port 8080, il suffit donc d'aller directement dessus (via am02 puisqu'on n'a pas d'accès à partir de notre zone réseau)

Laurent Edel

### Lignes de commande

# Flume

## Quelques affirmations

* Il faut un channel distinct par agent
* Plusieurs sources et sink peuvent utiliser le même channel MAIs il faut ajouter un interceptor dans le cas où l'on souhaite envoyer les données sources dans 2 sinks différents

With a large batch size, you get much better throughput, but increased latency, and in the case of a transaction failure, the number of possible duplicates increase

# Hbase:

## Gestion des droits sur les tables

La table qui contient la lsite des acl sur Hbase est hbase:acl

revoke 'a\_app\_qos\_care', 'tb\_hb\_reqgi\_orange'

grant 'r\_writer\_qos', 'RWXCA','tb\_hb\_histo\_cellules\_msisdn'

# Ansible

## Ansible ne fonctionne plus après un ctrl+c

Dans ce cas il faut tuer les process qui tournent :

ps aux | grep ansible

## Commande pour lister les facts d’un serveur

ansible -i ./env\_g2pochp -m setup inbdfam01

## Messages Alexis

16:40TRINQUET Alexis Ext DTSI/DSI

commande pour lancer une commande sur tout ou une partie du cluster

tout le cluster

ansible -i env\_g2int2 -a "hostname" all

la commande lancée est hostname

16:43TRINQUET Alexis Ext DTSI/DSI

lancement du playbook sur tout les serveurs :

ansible-playbook -i env\_g2int2 hdp\_cluster\_prerequisite.yml

lancement du playbook sur un serveur :

ansible-playbook -i env\_g2int2 hdp\_cluster\_prerequisite.yml -l inbdfax23

-

Lancement playbook en démarrant à partir d'une étape :

sible-playbook -i env\_g2int2 hdp\_cluster\_prerequisite.yml -l inbdfax23 --start-at-task="ipa\_client | Install ipa-client package" -vvv

et la commande pour crypter le le fichier yml de mot de passe :

ansible-vault edit env\_g2int2/group\_vars/all/secret.yml

voilà, je ne vois pas d'autres commandes utilent pour le moment

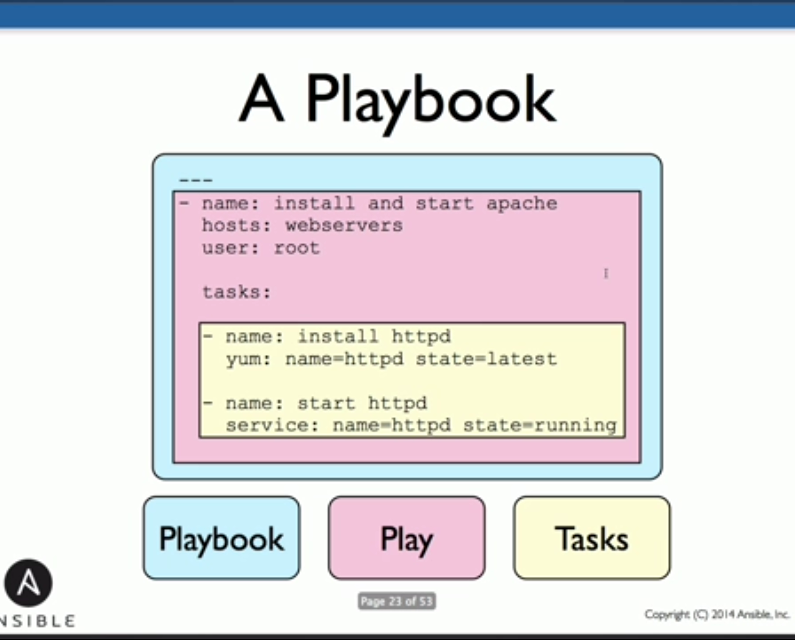
11:54TRINQUET Alexis Ext DTSI/DSI

Jade, voici le bouquin sur lequel nous avons dicsuté dernièrement

<http://www.amazon.fr/Big-Data-Kenneth-Cukier/dp/2221140044/ref=sr_1_6?s=books&ie=UTF8&qid=1457693592&sr=1-6&keywords=big+data>

## Oraganisation

* Playbook contains plays
* Plays contains Tasks
* Tasks contain Modules



## Variables

1. Les variables peuvent être déclarées au sein du module

- name: install a set of packages

yum : name={{ item }} state=present

with\_items:

- httpd

- php

- git

1. Ou dans le play

IMAGE

1. Ou dans des fichiers séparés (group vars, host vars)
2. ou en ligne de commande
3. ou à partir des discovered variables (facts) récupérées du host cible

# KNOX

## Logs LDAP et KDC

KDC : /opt/ipa/log/kerberos/krb5kdc.log

Ldap : /opt/ipa/log/dirsrv/slapd-BDFINT2/audit errors access

## Augmenter le buffer

Je vous envoie ce courriel au sujet du ticket suivant déposé sur OrangeForge : Migration G2 : Limitation à 800 items dans le filtre de la requête Hive

Voici le lien : <https://www.forge.orange-labs.fr/plugins/tracker/?aid=306600>

Les utilisateurs ont un problème avec une requête Hive longue, par Knox (et également par Hue, mais ce n’est pas le sujet principal de mon mail).

Afin d’autoriser des requêtes hive un peu plus longue via Knox, il faudrait appliquer la modification suivante dans le paragraphe « Advanced topology » de la partie Knox de l’IHM Ambari :

      <service>

            <role>HIVE</role>

            <url>[http://inbdfot21.rouen.francetelecom.fr:10001/cliservice</url](http://inbdfot21.rouen.francetelecom.fr:10001/cliservice%3c/url)>

            <param>

                  <name>replayBufferSize</name>

                  <value>16</value>

            </param>

      </service>

Cette propriété est exprimé en kilo-octets et a comme valeur par défaut 8 ko.

L’idée de cette modification est de passer à 16 Ko.

## Fonctionnement – ajout d’un service

Lorsque Knox récupère une URL, il voit si elle matche avec un des path défini dans le fichier service. Le cas échéant il applique la règle d’écriture définie dans le fichier rewrite.

**Topolgy** : On déclare dans topolgy le nom du service et la base URL vers laquelle Knox redirigera les requêtes. Knox complétera cette URL en fonction des règles de réécriture définies.

     <service>

          <role>RMBHINT</role>

          <url>[http://inbdfapi22.rouen.francetelecom.fr:8080/rmbhint/api</url](http://inbdfapi22.rouen.francetelecom.fr:8080/rmbhint/api%3c/url)>

     </service>

On déclare dans **rewrite**.xml des règles de réécritures. Le *pattern* contient le pattern de **l’URL d’appel sur le serveur Knox** (cad l’appel depuis l’extérieur de la VRF) et permet d’extraire des éléments qui seront donc utilisés dans la règle de réécriture. (ex : {\*\*})

<rules>

<rule dir="IN" name="RMBHINT/inbound" pattern="\*://\*:\*/\*\*/rmbhint/api/{\*\*}">

<rewrite template="{$serviceUrl[RMBHINT]}/{\*\*}"/>

</rule>

<rule dir="IN" name="RMBHINT/inbound/query" pattern="\*://\*:\*/\*\*/rmbhint/api/{\*\*}?{\*\*}">

<rewrite template="{$serviceUrl[RMBHINT]}/{\*\*}?{\*\*}"/>

</rule>

<rule dir="IN" name="RMBHINT/inbound/root" pattern="\*://\*:\*/\*\*/rmbhint/api">

<rewrite template="{$serviceUrl[RMBHINT]}"/>

</rule>

</rules>

On déclare dans **service**.xml, quelle sont les path à matcher avec les URLs Knox (mais sans la partie /gateway/default). Et dans le cas d’un match on applique la règle de réécriture désignée (apply) sur la ressource désignée (to). En effet on peut réécrire d’autres éléments comme le header par exemple.

<service role="RMBHINT" name="rmbhint" version="1.0.0">

<routes>

<route path="/rmbhint/api/\*\*?\*\*">

<rewrite apply="RMBHINT/inbound/query" to="request.url"/>

</route>

<route path="/rmbhint/api/\*\*">

<rewrite apply="RMBHINT/inbound" to="request.url"/>

</route>

<route path="/rmbhint/api">

<rewrite apply="RMBHINT/inbound/root" to="request.url"/>

</route>

</routes>

</service>

## Quelques définitions – Keystore et credentials

**identity store** : gateway.jks. The identity store contains the certificate and private key used to represent the identity of the server for SSL connections and signature creation.

**credential store** : \_\_gateway-credentials.jceks. This credential store is used to store secrets/passwords that are used by the gateway. For instance, this is where the passphrase for accessing the gateway-identity certificate is kept. Il contient le keypass (mot de passe) du certificate gateway-identity. L’accès au credential store est sécurisé par la masterkey

**credential store for the topology** : default-credentials.jceks. This topology specific credential store is used for storing secrets/passwords that are used for encrypting sensitive data with topology specific keys. L’accès au credential store for topology est sécurisé par la masterkey

En fait ces 3 stores sont sécurisés par le master key de knox.

* identity store contient les certificats
* credential store contient le mot de passe du certificat gateway-identity
* credentiel store for the topology : contient la passphrase de crytpage des données sensibles de la topologie

## Mise en cached de l’authentification pour améliorer les perfs

Comments: Hi Thibault

Could you try adding the following lines to your topology:

<param>

<name>main.cacheManager</name>

<value>org.apache.shiro.cache.ehcache.EhCacheManager</value>

</param>

<param>

<name>main.securityManager.cacheManager</name>

<value>$cacheManager</value>

</param>

<param>

<name>main.ldapRealm.authenticationCachingEnabled</name>

<value>true</value>

</param>

Add them in the Shiro configuration section of the topology just below the lines that read

<param>

<name>main.ldapRealm</name>

<value>org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm</value>

</param>

Restart Knox and then re-try the JDBC connection through Knox. This configuration turns on caching for the LDAP authentication performed by Knox and should speed up JDBC connections which do a lot of authentications.

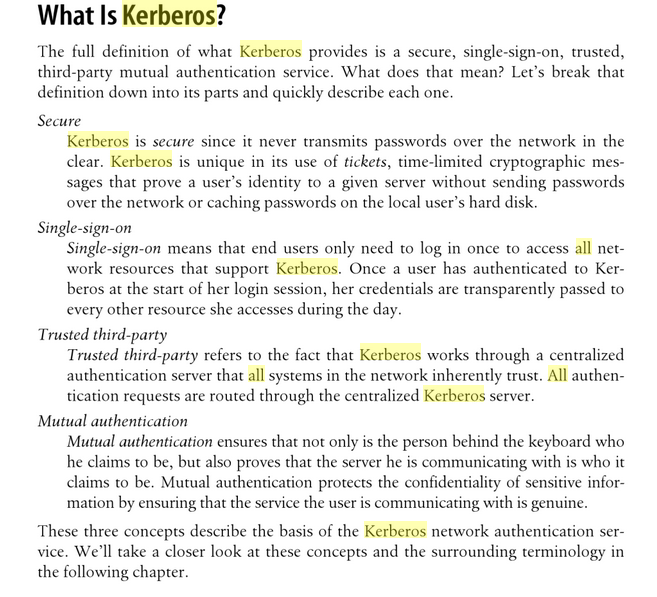
# Kerberos

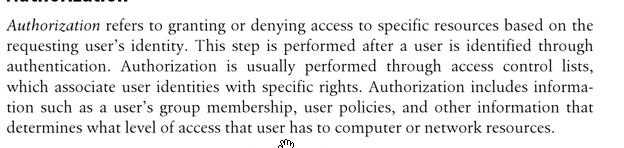
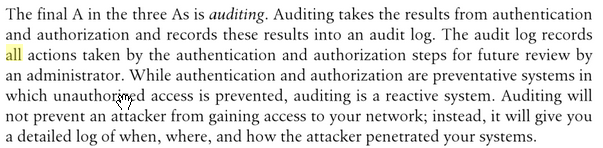
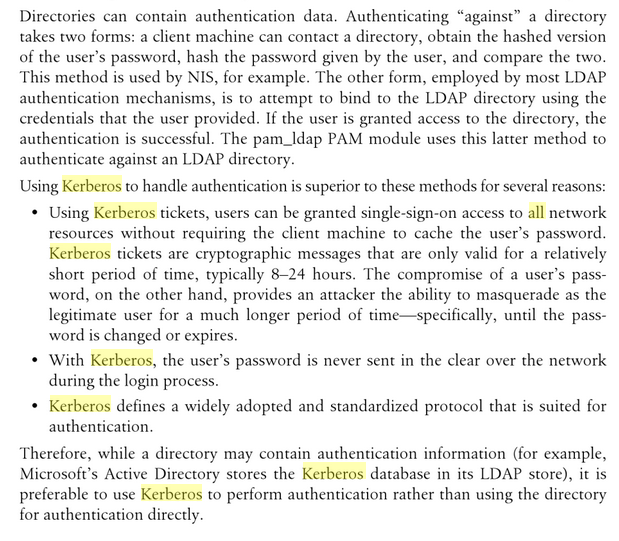
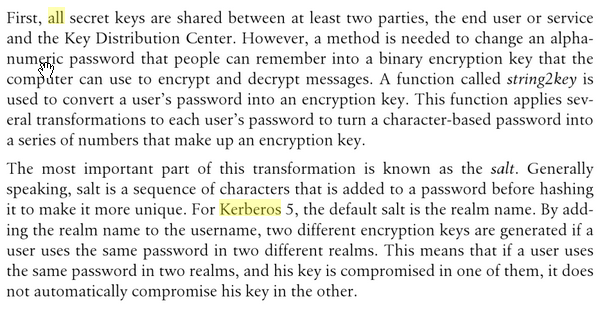
## exitRôle

L’objectif de Kerberos est de :

* Permettre aux utilisateurs de s’authentifier sur plusieurs host/service avec un unique mot de passe(SSO)
* Et de sécuriser la transmission des données sensibles (mot de passe)

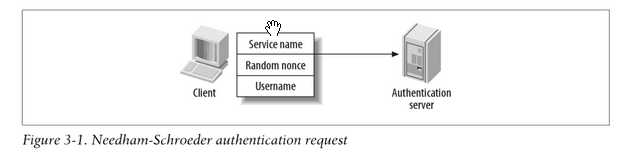
## Définition



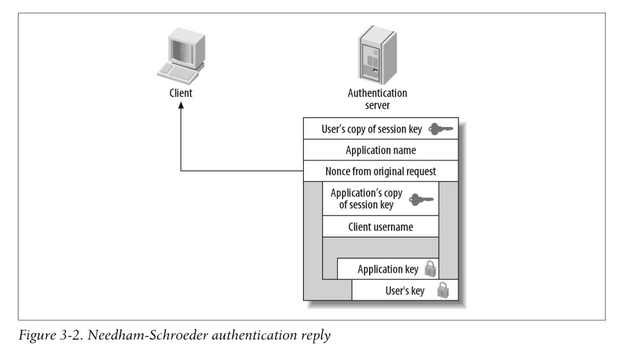
* **KDC** (Key Distribution Server) : contains a database of usernames and passwords for both users and Kerberos-enabled services.
* **The 3 As :** 
  + Authentication :
    - What you know (password)
    - What you have (Secure ID Token, smart cards..)
    - Who you are (Biometrics)
  + Authorization
    - 
  + Auditing
    - 
* Difference between Directory (LDAP) and Kerberos :
  + The directory (LDAP) contains information about you
  + Kerberos : Kerberos is the protocol which lets you authenticate yourself against that Directory
  + 
* **Message integrity** : Méthode qui permet de s’assurer que le message n’a pas été altéré en route. On applique une fonction mathématiques de hachage (ex : somme des bits du message) avant envoi et à réception et on s’assure de leur égalité. Etant donné, que plusieurs messages en entrée pourraient avoir la même sortie, on peut utiliser les nombres premiers pour s’assurer qu’il n’y aura pas de collision.
* **Principal**: Chaque entité dans une installation Kerberos (utilisateur, machine, service) ont un principal qui leur est associé. Chaque principal est associé avec un « long term key » qui peut être un mot de passe ou une passphrase. Un principal est censé être unique, pour ce, il est composé par une structure hiérarchique.
  + Chaque principal commence avec un nom d’utilisateur ou un nom de service suivi par une instance optionnelle
    - Cette instance optionnelle sert au principal de service. En fait chaque serveur et service du Royaume se voit attribuer un principal. Le nom d’utilisateur d’un service sera le nom du service alors que celui d’une machine sera le terme « host ». Pour distinguer un même service tournant sur différentes machines, l’instance optionnelle d’un principal de service est le hostname de la machine concernée. Les services qui utilisent l’authentification Kerberos sont dits Kerbérisés.
    - Ou pour créer des principal spéciaux, ie pour une utilisation d’administration (un admin pourrait avoir un compte pour les utilisations standards et un autre principal pour avoir les droits d’administration élevée à utilisation plus rare)
    - Le couple nom d’utilisateur / instance forme une identité unique dans le royaume Kerberos.
* **GSSAPI** : The Generic Security Service Application Program Interface (GSSAPI, also GSS-API) is an application programming interface for programs to access security services. The GSSAPI, by itself, does not provide any security. Instead, security-service vendors provide GSSAPI implementations - usually in the form of libraries installed with their security software. These libraries present a GSSAPI-compatible interface to application writers who can write their application to use only the vendor-independent GSSAPI. If the security implementation ever needs replacing, the application need not be rewritten.
* **SPENEGO** : Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO), often pronounced "spen-go", is a GSSAPI "pseudo mechanism" used by client-server software to negotiate the choice of security technology. SPNEGO is used when a client application wants to authenticate to a remote server, but neither end is sure what authentication protocols the other supports. The pseudo-mechanism uses a protocol to determine what common GSSAPI mechanisms are available, selects one and then dispatches all further security operations to it. This can help organizations deploy new security mechanisms in a phased manner.
* **Royaume** : Chaque installation de Kerberos possède un unique royaume qui prend généralement la forme du nom de domaine en lettre capitales.
* **Key, salts and passwords**
  + 
* **Key Distribution Center (KDC) :** Le KDC est un composant intégré à Kerberos, il est composé de 3 composants logiques :
  + une base de données de tous les principaux et leur encryption key (mot de passe/passpharse)
  + Le serveur d’authentification (AuthenticationServer)
    - Ce serveur génère un Ticket-granting ticket (TGT), au client qui souhaite s’authentifier dans le Royaume Kerberos. Le client n’a pas besoin de s’authentifier sur le KDC, en revanche le TGT renvoyé est crypté avec le mot de passe du client. Du coup, lors du process d’authentification (géré par le client de Kerberos) essaie de décrytper le TGT en utilisant le mot de passe fourni par le client (l’utilisateur). Si le décryptage échoue alors le client est invité à retaper son mot de passe. Une fois ce TGT décrypté par le client, il peut être utilisé pour faire la demande de tickets de service. C’est le TGT qui offre l’avantage de ne pas retaper son mot de passe à chaque accès à un service.
  + Le ticket granting server (TGS)
    - Fourni au client les tickets de service à la demande. Il prend en entrée 2 éléments
      * Le ticket request qui contient le nom du principal de service auquel le client souhaite se connecter
      * Le TGT
      * Le TGS vérifie que le TGT est valide en vérifiant qu’il a bien été encrypté avec la *clé TGT du serveur Kerberos* et renvoie au client le ticket de service demandé.
  + Un Royaume Kerberos peut contenir plusieurs KDC. Or les BDD de ces KDC doivent être synchronisées. Chaque éditeur a développé sa propose solution de synchro entre les différentes BDD. Kerberos ne propose pas de solution de synchro.
* **Ticket :** 
  + Un ticket kerberos est une structure cryptée issue par le KDC qui contient :
    - une clé de cryptage partagée unique à chaque session
    - et des flags qui indiquent par exemple si le ticket peut être transféré à un autre service.
  + Les tickets Kerberos, ont 2 objectifs :
    - confirmer l’identité des participants finaux
    - provisionner une clé de cryptage temporaire qui permettra de crypter les échanges entre le client et le service. (session key)
  + Un ticket est comme un permis de conduire. Il contient des informations concernant le client, une date de validité et certaines restrictions. Les principaux champs utilisés par Kerberos dans un ticket Kerberos sont :
    - le principal du client
    - le principal du service
    - Date de validité du ticket (début et fin)
    - Une liste d’adresse IP à partir desquelles le ticket peut être utilisé
    - une clé de cryptage partagée.
    - Certains de ces champs sont remplis par le KDC (date de validité, clé de cryptage) tandis que les autres sont remplis par le client. Quand un ticket est généré par le KDC, il est crypté pour éviter qu’un hacker modifie par exemple sa date de validité.
* **Ticket cache :** Les tickets sont sauvegardés par défaut dans un fichier sur le filesystem. Mais des systèmes de cache plus sécurisés utilisent la mémoire RAM pour s’assure de la destruction des tickets en fin de sessions. Le fichier ci-dessous contient une liste de tickets, le premier étant le TGT.
* ****

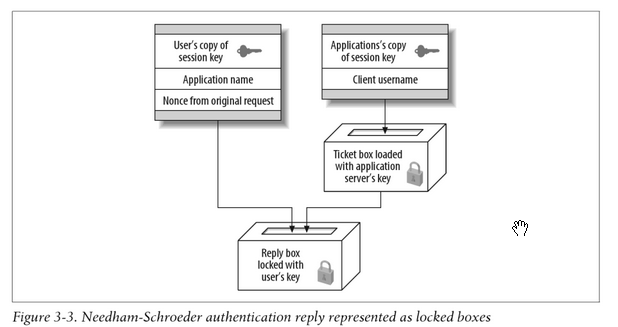
## Protocole de Needham-Schroeder

* Le client contacte le serveur d’authentification et envoie 3 éléments :
  + sa propre identité
  + l’identité du service auquel il souhaite accéder
  + une valeur aléatoire

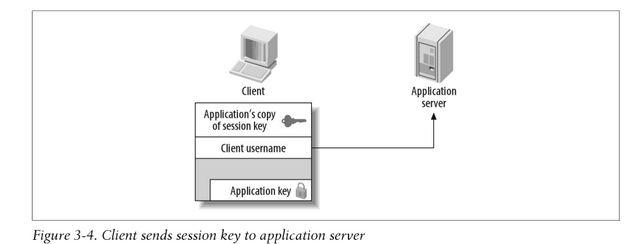


* Le serveur d’authentification (du KDC pour Kerberos) renvoie au client
  + une clé de session
  + les identités vérifiées du client et du service
  + 2 questions se posent à ce niveau :
    - Comment ce message renvoyé peut-il être sécurisé d’un hacker qu’il l’intercepterait ?
    - Et comment lorsque le client va envoyer la clé de session au service, ce dernier pourra être sûr de l’authenticité du message ?
    - Les réponses :
      * Un premier message est construit par le serveur d’authentification et qui ne pourra être lu que par le serveur d’application (le service). Ce message contient le nom de l’utilisateur et la clé de session. Ce message est crypté avec le mot de passe/passphrase du service. Etant donné que seuls le serveur d’auth et le serveur d’application ont connaissance du mot de passe/passphrase du service, si le message est intercepté par un tiers, alors il ne pourra pas récupérer la clé de session ou modifier le message. Dans le langage Kerberos, ce message crypté est appelé « **ticket** ».
      * Ce message est encapsulé dans un autre message à destination cette fois du client. Ce message contient le nom du service, une copie de la clé de session et une copie de la valeur aléatoire envoyée initialement par le client. Tout le message, le premier et le second), sont cryptés avec le mot de passe de l’utilisateur. Ce gros message est alors envoyé au client.





* Réception du paquet par le client : A noter que le serveur d’auth renvoie ce paquet à quiconque qui le demande sans être sûr que le client est vraiment la personne qu’il prétend être. Mais vu que le gros paquet est crypté avec le mot de passe de l’utilisateur, celui lui sera demandé lors de la réception du gros paquet. Et si le mot de passe fournir par le client ne permet pas de décrypter le paquet alors l’authentification échoue.
* Envoi du **ticket** au service par le client : A ce stade le client, n’a plus qu’à renvoyer le sous paquet (le ticket) au service qui ne pourra pas être lu ni modifié par un intercepteur car il est crypté avec le mot de passe du service.



* A ce stade :
  + La communication entre le service et le client est crypté via la clé de session
  + Le client est sûr de l’authenticité du service car ce dernier a réussi à décrypter le ticket (qui était crypté avec son propre mot de passe)
  + Le service est sûr de l’authenticité du client car ce dernier a réussi à décrypter le gros paquet.
  + Mais il reste néanmoins une attaque possible : Supposons qu’un intercepteur intercepte le ticket envoyé au service par le client et décide de le lui renvoyer plus tard. Si le service n’utilise pas la clé de session pour valider l’authentification, alors il authentifiera cet intercepteur qui lui aurait envoyé un ticket valide (ça voudrait dire qu’il a réussi à décrypter le gros paquet initial et qu’il possède le mot de passe du client). Il faut donc s’assurer que le client possède bien la clé de session. Pour ce, il y a un échange supplémentaire qui consiste à ce que le service envoie un nombre aléatoire crypté avec la clé de session au client qui doit le lui renvoyer après l’avoir décrypté, y avoir appliqué un calcul (+1 par ex) et recrypté. Du coup le service est sûr que le client possède bien la clé de session.

## Analogie entre Kerberos et le protocole de Needham-Schroeder

Les 3 principales différences sont :

* Les 2 derniers échanges du protocole de NS sont remplacés par l’envoi de l’heure (du coup synchronisés entre les différents serveurs) qui est crypté via la clé de session => Ceci implique de synchroniser l’heure ente les serveurs mais réduit le nombre d’échange
* Le serveur d’authentification et le client d’authentification du protocole de NS sont assimilés au serveur KDC dans Kerberos. C’est ce qui crée le concept de TGT qui permettra donc au client de s’authentifier à plusieurs services mais en tapant son mot de passe une seule fois.
* Kerberos intègre la notion de temps de validé d’un ticket pour améliorer la sécurité.

Le protocole Kerberos est donc composé de 2 composants logiques :

* Le serveur d’authentification Kerberos (KDC) (à ne pa confondre avec le serveur d’authentification du protocole de NS qui fait plus de choses que le KDC)
* Le Ticket Granting Server (TGS)
* Ces 2 composants sont implémentés dans un programme unique mais sont bien 2 process distincts.

## Le serveur d’authentification de Kerberos (AS) et TGS - Fonctionnement

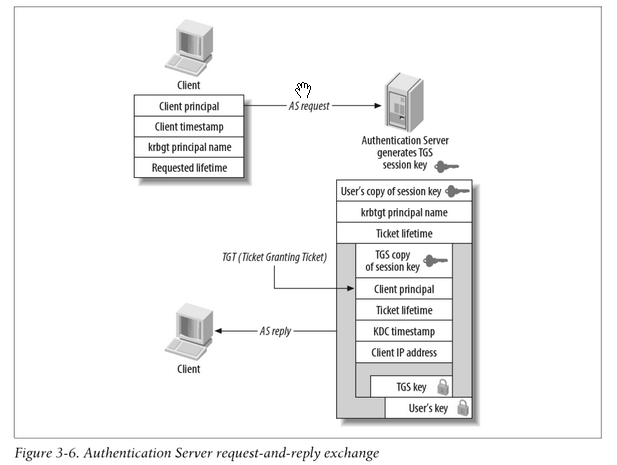
L’AS a un seul objectif, c’est de recevoir une requête de la part du client (AS\_REQ) qui contient l’identité du client, l’heure locale du client, et le principal du TGS principal (nommé krbtgt par convention)

Lorsque le KDC reçoit l’AS\_REQ (qui est envoyé en clair), il vérifie que le principal du client existe et que l’heure locale est proche de la sienne (~5min). Si la vérification échoue alors le client n’est pas authentifié.

L’AS génère une clé de session, cette clé de session sera partagée entre le client et le TGS. Cette clé de session permettra de crypter les demandes faites par le client au TGS lors des demandes de tickets de service.

Le KDC répond avec un Authentication Server Reply Message (AS\_REP) qui contient tous les éléments ci-dessous. Le plus important c’est  :

* la copie de la clé de session et l’identité vérifiée du client, tout cela crypté avec la la TGS key.
* une autre copie de la clé de session mais cette fois-ci cryptée avec le mot de passe de l’utilisateur.
* **La connaissance de cette clé de session et la possession de la TGT permettront de faire des demandes de tickets de servie sans retaper son mot de passe.**



**A ce stade,** le client possède un gros paquet crypté avec son propre mot de passe qui contient lui-même la clé de session et un sous-paquet crypté avec la clé du TGS (c’est le TGT). Le client décrypte le gros paquet avec son mot de passe et sauvegarde dans le ticket-cache la clé de session et le TGT.

Lorsque l’utilisateur souhaite accéder à un service, il doit initier un échange avec le TGS. Il lui envoie un paquet qui contient trois éléments :

* une requête TGS (qui contient notamment le service auquel on souhaite accéder)
* une copie du TGT
* et un authenticator (qui permet d’éviter la faille de sécurité site « replay » qui consiste à intercepter le ticket envoyé par le client au service et de le réutiliser plus tard. Pour rappel, dans le protocole de NS, cette faille a été palliée par l’échange supplémentaire qui consistait à effectuer une opération mathématique sur une variable aléatoire envoyée par le service et puis renvoyée par le client avec encryptage/décryptage via la clé de session. Ceci permettait de s’assurer que le client possède bien la clé de session.
  + Ici l’authenticator consiste est un timestamp cryptée avec la clé de session. Cet authenticator assure l’unicité de chaque demande de ticket au TGS et permet aussi de garantir que le client a connaissance de la clé de session.
  + Sans cet authenticator, un intercepteur qui intercepterait un TGT ou ticket de service pourrait les réutiliser et les renvoyer respectivement au TGS ou au service.

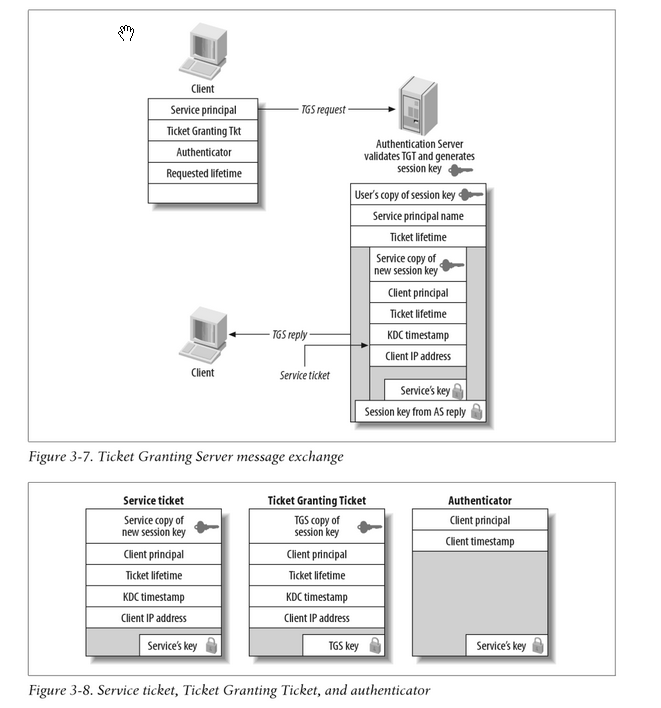
A réception de ce paquet, le KDC renvoie un message qui inclue :

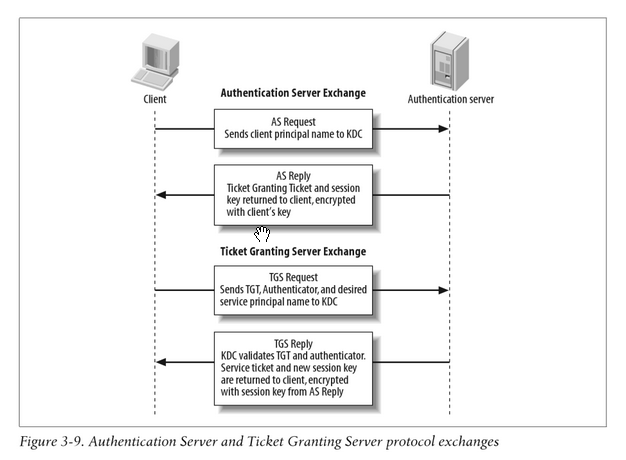
* une clé de session qui servira aux communications entre le service et le client
* la copie du client de la nouvelle clé de session est cryptée avec la clé de session initiale (générée lors du premier échange avec l’AS). Ceci garantit que seul le client peut récupérer la nouvelle clé de session pour les échanges avec le service.
* La copie du service de la nouvelle clé de session est embarquée dans un ticket de service qui est cryptée via le mot de passe du service. Le client ajoute ce ticket de service crypté et la nouvelle clé de session dans son ticket-cache.
* Lorsque l’utilisateur voudra accéder au service, la possession du ticket de service et de la nouvelle clé de session garantit au service l’authenticité de l’utilisateur.

Les 3 principaux messages cryptés dans ces échanges sont donc :

* Le TGT
* Le ticket de service
* L’authenticator

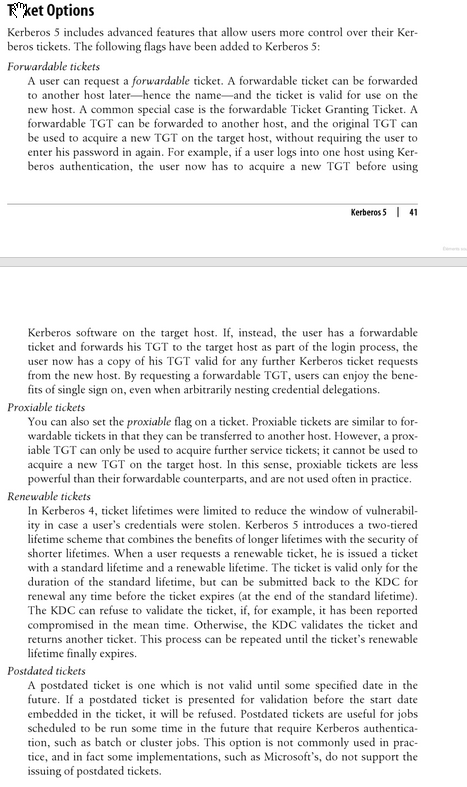
/!\ La possession d’un ticket de service ne garantit pas que l’utilisateur aura le droit d’accéder au service. Le service se chargera d’autoriser l’utilisateur ou pas suivant les autorisations. La possession du ticket de service et de la clé de session ne serve qu’à garantir que l’utilisateur qui demande l’accès au service est la personne qu’il prétend être.





## Différence de Kerberos 5 avec Kerberos 4

Ce qui a été présenté jusque-là c’est Kerberos 4. Voici quelques-unes des différences entre les 2 :

* KB5 utilise le protocle ASN.1 qui lui permet d’être plus extensible
* KB5 concatène les messages cryptés au lieu de les encapsuler. => Ca améliore les performances.
* KB5 propose des fonctionnalités des options sur les tickets de service :
* 

# Spark

## SparkThirft server

### Pour démarrer le spark thrift server

[root@inbdfot01 ~]# su hive

bash-4.1$ /usr/hdp/current/spark-thriftserver/sbin/start-thriftserver.sh --driver-memory 1g --verbose --master yarn --executor-memory 1g --num-executors 1 --executor-cores 1 --conf spark.hadoop.yarn.timeline-service.enabled=false --conf spark.yarn.queue=q\_batch --conf spark.yarn.am.memoryOverhead=384 --conf spark.driver.maxResultSize=384m --conf spark.sql.crossJoin.enabled=true --conf spark.rpc.askTimeout=700 --conf spark.sql.broadcastTimeout=800 --hiveconf hive.server2.thrift.http.port=10015 --hiveconf hive.server2.transport.mode=http --hiveconf hive.server2.http.endpoint=cliservice --hiveconf hive.server2.thrift.http.cookie.is.secure=false

### Pour se connecter au spark thrift server

[root@inbdfot01 ~]# kinit u\_qkdp3997\_adm

Password for u\_qkdp3997\_adm@BDFPOCHP:

[root@inbdfot01 ~]# beeline

WARNING: Use "yarn jar" to launch YARN applications.

Beeline version 1.2.1.2.3.4.7-4 by Apache Hive

beeline> !connect jdbc:hive2://inbdfot01.rouen.francetelecom.fr:10015/default;transportMode=http;httpPath=cliservice;principal=hive/inbdfot01.rouen.francetelecom.fr@BDFPOCHP

Connecting to jdbc:hive2://inbdfot01.rouen.francetelecom.fr:10015/default;transportMode=http;httpPath=cliservice;principal=hive/inbdfot01.rouen.francetelecom.fr@BDFPOCHP

Enter username for jdbc:hive2://inbdfot01.rouen.francetelecom.fr:10015/default;transportMode=http;httpPath=cliservice;principal=hive/inbdfot01.rouen.francetelecom.fr@BDFPOCHP: u\_qkdp3997\_adm

Enter password for jdbc:hive2://inbdfot01.rouen.francetelecom.fr:10015/default;transportMode=http;httpPath=cliservice;principal=hive/inbdfot01.rouen.francetelecom.fr@BDFPOCHP: \*\*\*\*\*\*\*\*

Connected to: Spark SQL (version 1.5.2)

Driver: Hive JDBC (version 1.2.1.2.3.4.7-4)

Transaction isolation: TRANSACTION\_REPEATABLE\_READ

0: jdbc:hive2://inbdfot01.rouen.francetelecom>

# Cassandra

## Transfert données de Hive vers Cassandra.

**1.      Utilisation de tables externes Cassandra dans Hive**

**a.      Librairie Tuplejump/cash**

Lien vers le Githoub (dédicace :D) :

<https://github.com/tuplejump/cash>

Librairie non-livrée dans Hadoop, il faut la packager avec l’option -DskipTests .

Cette solution utilise de vielles librairies Hive/Hadoop/Cassandra, donc à voir si cela fonctionne.

Cette solution nécessite une désactivation de l’authentification Cassandra, i.e. Cassandra devient OpenBar.

C’est la librairie que souhaite tester ALIM dans son ticket :

<https://www.forge.orange-labs.fr/plugins/tracker/?aid=309051>

Pour cette librairie, aucun support ne sera demandée au CC.

**b.      Libraire DFY**

Lien vers le Githoub :

<https://github.com/Orange-OpenSource/cassandra_handler_for_hive>

Librairie mise à jour par DFY (dernière modification date quand même de deux ans).

Soit disant prend en compte l’authentification, pas testé.

**2.      Spark**

La préconisation du CC, utilisation de spark pour lire les tables internes hive, reformater les données et les envoyer dans Cassandra.

Cette solution implique des devs Spark. ALIM travaille dessus mais préfère la solution court terme 1.a pour le moment.

**Conclusion :**

On va demander à ALIM de packager eux-mêmes la librairie (problème d’accès à internet côté CC + problème installation openJDK sur windows + urgence des tests ALIM)

ALIM va ensuite attacher cette lib au ticket et il faudra la déployer sur les différents clusters, dans le dossier pour les libs auxiliaires de hive, a priori uniquement sur les serveurs Edge outils (là où se trouvent les composants Hive).

# SVN

## Pour remonter les fichiers non commités.

Se positionner dans le dossier et taper :

svn st -u

Checkout

svn checkout --username qkdp3997 https://www.forge.orange-labs.fr/svnroot/BDF

svn checkout --username qkdp3997 http://inbdfax01.rouen.francetelecom.fr:443/svnroot/BDF

# Yum

## Retrouver le repo depuis lequel est installé un package

yum list [package]

Renvoie un résultat du type :



Ce qui veut dire que la paquet ambari-agent provient du repo qui est référence dans le repo yum nommé ambari-2.2.1.0-161. Pour rappel les repo yum sont spécifiés dans /etc/yum.repo.d

## Créer un nouveau repo

1. Installer createrepo s’il n’est pas encore installé

yum install createrepo

1. Créer un dossier et y mettre tous les RPM souhaités
2. Se placer dans le répertoire et éxecuter

createrepo --database [repertoire courant]

## Installer apache

J’ai dû réactiver le repo standard sur le serveur ops

vi /etc/yum.repos.d/common-rhel6.repo

Et puis taper

yum clean expire-cache

Et enfin installer httpd

yum install httpd

Démarrer le service

service httpd start

On peut essayer de faire appel à l’URL du serveur à partir d’un autre serveur pour vérifie que le service fonctionne

curl http://uabdfops01.rouen.francetelecom.fr/

## Mise en place d’un repo

Pour ajoute un repo sur le serveur ops1 de prod, j’ai dû :

* Activer le repo yum « standard »
* Installer httpd
* Lancer httpd
* Copier les rpm dans le dossier /var/opt….
* Créer un lien symbolique depuis /var/www/html vre le dossier dans /var/opt…

# Java

## Retrouver dans quelle le jar d’un classe

## Pour trouver le jar

for i in $(locate "\*hcatalog\*.jar");

do echo $i; jar -tvf $i | grep -Hsi HCatLoader;

done

## Pour aficheir el contenu du jar

jar -tvf /usr/hdp/2.3.4.7-4/hive-hcatalog/share/hcatalog/hive-hcatalog-pig-adapter-1.2.1.2.3.4.7-4.jar

## Manipulation de jar

Décompresser un jar

jar xf myBundle-G00R04C00BETA.jar

Compresser un jar

jar cf jar-file input-file

## Lancer le garbage colector :

jcmd [PID] GC.run

ex : jcmd 91504 GC.run

## Accéder au JMX dun process HDFS

Suite à quelques problèmes de heap sur des datanodes de la G2, il est intéressant de voir comment mesurer un certain nombre de métriques, en l'occurrence… la heap !

Globalement dans le monde Java, on utilise un système appelé JMX (Java Management Extensions) vers qui les différentes classes au sein d'une JVM vont remonter un certain nombre de paramètres.

Pour Hadoop, on peut accéder aux JMX d'un process HDFS via son interface web suivie de /jmx

# Par exemple sur la G2, la datanode UI est sur le port 1022, donc

[root@opbdfam01 ~]# curl -i http://opbdf0804.rouen.francetelecom.fr:1022/jmx

{

"beans" : [ {

"name" : "java.lang:type=Memory",

"modelerType" : "sun.management.MemoryImpl",

...

On peut s'attacher à une partie particulière identifiée par son champ name en le passant en paramètre, ce qui permet une lecture un peu plus aisée (le jmx complet est assez verbose) : ici on va vouloir regarder les paramètres MemHeapUsedM et MemHeapMaxM pour arriver à un ratio de 18% de heap occupée (191/1024), et on peut voir qu'il y a eu 54 garbage collection depuis le lancement du process et que la dernière gc a duré 723ms.

[root@opbdfam01 ~]# curl -i http://opbdf0804.rouen.francetelecom.fr:1022/jmx?qry=Hadoop:service=DataNode,name=JvmMetrics

{

"beans" : [ {

"name" : "Hadoop:service=DataNode,name=JvmMetrics",

"modelerType" : "JvmMetrics",

"tag.Context" : "jvm",

"tag.ProcessName" : "DataNode",

# Nginx

## Faire une redirection à partir d’un mandataire

Aller sur le mandataire

Créer un fichier de conf dans sites-availables. On peut nommer le fichier avec le service et le host cible.

touch /etc/nginx/site-availables/Ambari.opbdfkx01.nor.fr.ftgroup.conf

Créer un lien symbolique dans sites-enabled

ln -s /etc/nginx/site-availables/Ambari.opbdfkx01.nor.fr.ftgroup.conf /etc/nginx/sites-enabled/Ambari.opbdfkx01.nor.fr.ftgroup.conf

Editer le fichier de conf :

server {

listen 8096; //post sur lequel le mandataire vaécouter les requetes; ce port ne doit pas être utilisé par un autre service. On peut vérifier si le port est utilisé avec « netstat -plunt | grep 80 »

location / {

proxy\_pass http://opbdfkx01.nor.fr.ftgroup:8080 ; // c’est l’url vers laquelle seront redirigées les requêtes.

}

}

Redémarrer nginx.

service nginx restart

# Python

## Modules

Liste des modules

python

>>> help('modules')

Tester l’import d’un paquet

python

>>> import [module]

Path de python

python

>>> import sys

>>> print sys.path

# Suivi applicatif

Le suivi applicatif écrit dans elastic qui est installé sur le serveur db03

On peut lancer elastic via

/opt/application/bdfess/products/elasticsearch/current/operating/bin/OperateElasticsearchAll.sh start

On peut lancer l’application “suivi applicatif” depuis le serveur ws. Cette application écoute sur le port 8080 (REST API)

/opt/application/bdfsap/PA-BDF-SUIVIAPPLICATIF-G01R01C01/operating/bin/OperateSuiviapplicatifAll.sh start

On peut tester l’API sur le serveur ws

curl -i http://localhost:8080/api/v1/rjdm/events

Ou depuis le serveur Knox en passant par Knox

curl -k -v https://dvbdfkx11.nor.fr.ftgroup:8443/gateway/default/suiviapplicatif/api/v1/rjdm/events

Base URL du suivi applicatif depuis l’intérieur du cluster :

* <http://dvbdfws11.rouen.francetelecom.fr:8080> pour dev1
* <http://inbdfws11.rouen.francetelecom:8080> pour int1
* <http://inbdfws21.rouen.francetelecom.fr:8080> pour int2.

Base URL (et IHM) depuis l’extérieur en passant par Knox :

* <https://dvbdfkx11.nor.fr.ftgroup/gateway/default/suiviapplicatif> pour dev1
* <https://inbdfkx11.nor.fr.ftgroup/gateway/default/suiviapplicatif> pour int1
* <https://inbdfkx21.nor.fr.ftgroup/gateway/default/suiviapplicatif> pour int2.

# Maven

src : <https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html>

1. Installation de Maven (apt, yum…)
2. Création d'un dossier projet

Exécuter la commande suivante pour créer l'architecture de l'application :

mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

mvn test

* génère le dossier target avec les fichiers compilés (le contenu de scr/main est compilé dans un dossier classes et ceux du dossier test dans un dossier test-classes)
* exécute les tests du dossier test
* génère des rapports de l'application et des tests unitaires dans un dossier target/surefire-reports/
* mvn install
* génère un fichier maven-archiver/pom.properties qui contient la version du projet et des infos de l'artifact
* génère le jar et ajoute dans le dossier les fichiers pom.xml

Le jar contient

* Les classes du dossier src/main compilés
* un doosier META-INF qui contient
  + pom.properties
  + pom.xml
  + MANIFEST.MF

**Exécuter la classe dans le jar :**

java -cp ./my-app-1.0-SNAPSHOT.jar com.jjcorp.app.App

**Maven Phases**

Although hardly a comprehensive list, these are the most common default lifecycle phases executed.

* validate: validate the project is correct and all necessary information is available
* compile: compile the source code of the project
* test: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* package: take the compiled code and package it in its distributable format, such as a JAR.
* integration-test: process and deploy the package if necessary into an environment where integration tests can be run
* verify: run any checks to verify the package is valid and meets quality criteria
* install: install the package into the local repository, for use as a dependency in other projects locally
* deploy: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

There are two other Maven lifecycles of note beyond the default list above. They are

* clean: cleans up artifacts created by prior builds
* site: generates site documentation for this project

Phases are actually mapped to underlying goals. The specific goals executed per phase is dependant upon the packaging type of the project. For example, package executes jar:jar if the project type is a JAR, and war:war if the project type is - you guessed it - a WAR.

An interesting thing to note is that phases and goals may be executed in sequence.

mvn clean dependency:copy-dependencies package

This command will clean the project, copy dependencies, and package the project (executing all phases up to package, of course).

**Generating the Site**

mvn site

This phase generates a site based upon information on the project's pom. You can look at the documentation generated under target/site.

**Notes :** Quand on exécute mvn package par exemple, cela veu dire 2 choses :

Maven va se charger d'exécuter toutes les étapes du cycle avant d'exécuter la tâche appelée.

Maven va faire correspondre au terme « pakage » la paire module:goal suivante : jar:jar car maven a detecté que mon appli est un jar. Si celui-ci était un war alors Maven aurait automatiquement asssocié le terme package à war:war.

# Réseau

Les serveurs ont plusieurs interfaces réseaux. Par exemple pour le **serveur dvbdfax21.rouen.francetelecom.fr (mandataire)**

|  |  |  |  |
| --- | --- | --- | --- |
| Fichier hosts | IP | eth | Accès |
| dvbdfax21-adm | 10.105.192.21 | eth0 | TDIMG |
|  | 10.117.56.28 | eth1 | Mon poste (domaine : rouen.francetelecom.fr) |
| dvbdfax21-sav | 10.71.130.21 | eth2 |  |
| dvbdfax21-stk | 10.197.194.19 | eth3 |  |
| dvbdfax21-svc  dvbdfax21.rouen.francetelecom.fr dvbdfax21  dvbdfax21-prd | 10.79.30.8 | eth4 | Mon poste (domaine : nor.ft.ftgroup) |

**Server db (dans la zone VRF)**

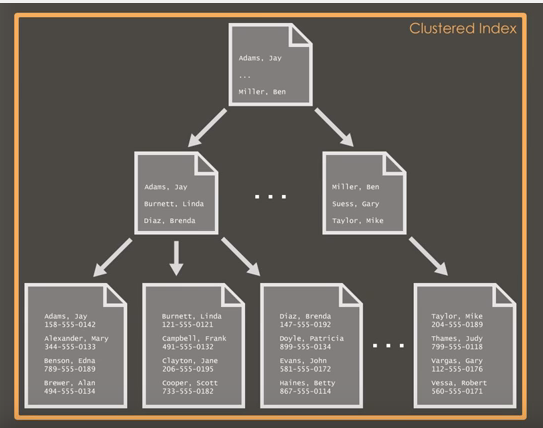
|  |  |  |  |
| --- | --- | --- | --- |
| Fichier hosts | IP | eth | Accès |
| dvbdfdb21-adm | 10.109.178.68 | eth0 | TDIMG |
| dvbdfdb21.rouen.francetelecom.fr dvbdfdb21  dvbdfdb21-prd | 10.79.28.68 | eth1 | Mon poste (domaine : rouen.francetelecom.fr) |
| dvbdfdb21-sav | 10.79.24.29 | eth2 |  |
| dvbdfdb21-stk | 10.198.121.29 | eth3 |  |

Donc les serveurs de la VRF n’ont pas de domaine en nor.ft.ftgroup car ils ne sont pas accessibles depuis l’extérieur de la zone.

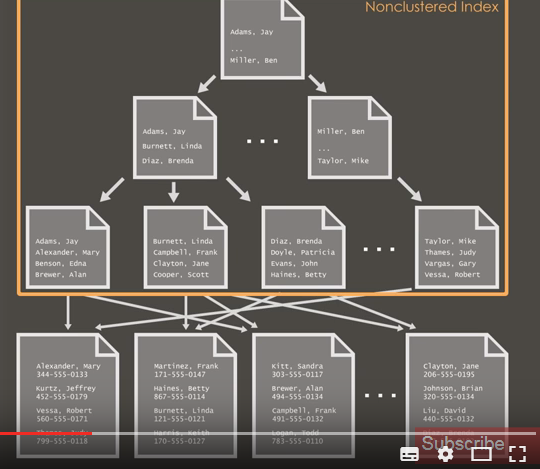
# Base de donénes

## Les types d’indexes

Le **clustered index** est une structure en arbre dont les feuilles sont les pages mêmes de la table. Un peu comme un annuaire téléphonique où l’on aurait en haut de chaque page le premier et dernier nom de la page. Du coup on ne peut avoir qu’une seule indexation clustered par table. Attention, à chaque mise à jour de la clé indexée, il faut déplacer la ligne (cad un delete et un insert à faire => couteux)



L'**index composite (index non clustered)** est aussi une structure en arbre mais dont les feuilles sont des pointeurs vers les lignes recherchés de la table. On peut donc en avoir plusieurs pour une même table. On peut également avoir plusieurs clés d’indexes comme « nom,prenom » pour un annuaire. On peut également inclure dans l’index des champs en plus des clés indexés dans le cas où ces champs sont souvent sélectionnés dans le select. Et ce, pour éviter de devoir accéder à la table de données vu que les données à récupérer sont dans l'index.



L**'index primaire** est un index dont la clé est unique

L'**index secondaire** est un index où la clé peut être doublonné (ex indexer un annuaire par rapport au nom de famille ou prénom)

# Création des espaces applicatifs

J’ai sorti les scripts de création des Use-Case de BDFADM et les ai placé dans un nouveau produit logiciel BDFUC (Pour Use-Case).

Il y a donc une nouvelle arborescence dans le SVN, qui est la suivante :

<https://www.forge.orange-labs.fr/svnroot/BDF/trunk/BDFUC>

Il faudra se baser sur cette URL maintenant pour les scripts de création des Use-Case.

Voici également un modop pour mettre à jour les 4 environnements G2 Hors-Prod  :

**# Si on livre le PL BDFUC sur un serveur ambari où le PL n’a jamais été déployé, il faut realiser les actions 1 et 2. Sinon l’action 3 suffit.**

**# 1. Creation du dossier BDFUC sur le serveur ambari**

mkdir -p /opt/application/bdfuc

**# 2. Checkout du PL BDFUC dans ce dossier /opt/application/bdfuc**

/usr/bin/svn checkout --username **<CODE ALLIANCE USER>** --no-auth-cache <http://inbdfax01.rouen.francetelecom.fr:443/svnroot/BDF/trunk/PL_BDFUC> /opt/application/bdfuc

**# 3. Mise a jour du dossier /opt/application/bdfuc quand on veut rapatrier les modifs du serveur SVN**

/usr/bin/svn update --username **<CODE ALLIANCE USER>** --no-auth-cache /opt/application/bdfuc

J’ai également supprimé le dossier create\_applicative\_space du PL BDFADM.

J’ai réalisé cette modification pour la raison suivante :

Lors des livraisons du PL BDFADM, j’embarquais des versions données des scripts de création des Use-Case.

Et parfois, nos collègues de DESI exécutaient les versions des scripts embarqués dans BDFADM, au lieu d’exécuter les dernières versions de ces scripts.

Cela créait des incompréhensions et posait des problèmes de droits en PROD.

# Procédures

## Dossier /tmp plein sur worker

Voici le modop pour modifier l’endroit où Hive stocke des fichiers temporaires.

### Pré-requis : créer un dossier pour recevoir les fichiers temporaires, dossier qui répond aux normes CB

La solution finale sera de stocker les fichiers temporaires de hive dans un LV du type /var/opt/data/flat/hadoop/hive/tmp.

Le LV /var/opt/data/flat/hadoop :

* existe sur les Workers
* n’existe pas sur les Edge Datalab
* n’existe pas sur les Edge Outils

Obtenir ces LV hadoop\_data demande une modification des plans de nommage Edge Datalab et Edge Outils.

Le LV /opt/hdp/log :

* existe sur les Edge Datalab
* existe sur les Edge Outils

La solution temporaire pour les Edge Datalab et les Edge Outils est la suivante :

1. Créer un dossier /opt/hdp/log/hive s’il n’existe pas déjà

mkdir -p /opt/hdp/log/hive

chown hive:hadoop /opt/hdp/log/hive

chmod 755 /opt/hdp/log/hive

1. Créer un sous-dossier tmp dans ce dossier hive :

mkdir -p /opt/hdp/log/hive/tmp

chown hive:hadoop /opt/hdp/log/hive/tmp

chmod 1777 /opt/hdp/log/hive/tmp

1. Le dossier /var/opt/data/flat/hadoop devrait déjà exister sur tous les serveurs, même si ce n’est un point de montage que pour les workers (à valider)
2. Créer un dossier hive dans /var/opt/data/flat/hadoop/

mkdir -p /var/opt/data/flat/hadoop/hive

chown hive:hadoop /var/opt/data/flat/hadoop/hive

chmod 755 /var/opt/data/flat/hadoop/hive

1. Créer un lien symbolique tmp pointant sur le sous-dossier /opt/hdp/log/hive/tmp.

ln -s /opt/hdp/log/hive/tmp /var/opt/data/flat/hadoop/hive/tmp

### Modification dans la configuration hive

Modification #1– Modifier la destination des logs que hive stocke dans /tmp pour les requêtes des users

Advanced hive-log4j

Remplacement de

hive.log.dir=${java.io.tmpdir}/${user.name}

Par

hive.log.dir=/var/opt/data/flat/hadoop/hive/tmp/${user.name}

Modification #2 – Modifier la destination des logs que hive stocke dans /tmp pour les requêtes des users (bis)

Advanced hive-exec-log4j

Remplacement de

hive.log.dir=${java.io.tmpdir}/${user.name}

Par

hive.log.dir=/var/opt/data/flat/hadoop/hive/tmp/${user.name}

Modification #3 – Modifier la destination des logs d’opération que le hvieserver2 stocke dans /tmp

Advanced hive-site

Remplacement de

hive.server2.logging.operation.log.location=/tmp/hive/operation\_logs

Par

hive.server2.logging.operation.log.location=/var/opt/data/flat/hadoop/hive/tmp/operation\_logs

Modification #4 – Modifier la destination des données temporaires des requêtes des utilisateurs

Custom hive-site

Ajout de

hive.exec.local.scratchdir=/var/opt/data/flat/hadoop/hive/tmp/${user.name}

Note :

Normalement, seule la modification #4 est utile pour résoudre le problème qu’il y a eu en production sur le /tmp.

Les modifications #1 à #3 consistent en une action en prévoyance de problèmes pouvant survenir sur /tmp.

Redémarrer le service hive.

### Notes

Dans mes investigations sur ces problèmes du dossier /tmp, j’ai trouvé un dernier problème connu sur les serveurs hébergeant un hiveserver2 (HS2).

Autrement dit, les serveurs concernés sont les serveurs Edge Outils.

Le HS2 crée des fichiers .pipeout de 0 octets dans le dossier /tmp/hive/ et il ne les purge jamais.

C’est un bug connu du composant hive qui devrait être résolu avec la HDP 2.5.

**Impact :**

Si trop de fichiers pipeout sont créés, cela peut consommer la totalité des inodes du serveur (65536).

Pour voir le nombre d’inodes actuellement utilisés dans /tmp :

df -i /tmp

Pour voir l’espace actuellement utilisé dans /tmp :

df -h /tmp

Solution en attendant la HDP 2.5 :

Purger les vieux fichiers pipeouts de /tmp/hive.

## Cloud

Sharepoint -->  (Hadoop Working / Manuels / Utilisateurs / Demande d'environnements FAST IT.docx )  
--> Hadoopworking / Industrialisation / Recap Cld.docx

## Utilisation outils ticketing

N’oubliez pas quand vous prenez un ticket :

* D’ajouter votre nom en Porteur CCBI HADOOP
* D’ajouter le nom d’une autre personne du support dans le champ Porteur Backup CCBI HADOOP

N’oubliez pas quand vous clôturez un ticket :

* De renseigner le champ durée de résolution ticket (en jour)

## Activité de support

Les personnes officiellement membre du support sont :

* Gwenael Le Barzic
* Michel Barbe
* Thibault Storai
* Alexis Trinquet
* Stephen Clairville
* Jade Jaber

Pour les personnes du support, la priorité d’activité est la suivante :

* P1. Activité sur un problème de production
* P2. Activité sur un ticket de hors production (objectif : la durée de vie des tickets ne doit pas dépasser 15 jours)
* P3. Activité sur un chantier

## Opérations de maintenance en hors-production

Un chtit rappel pas que pour les membres du support pour le coup, quand vous réalisez une opération de maintenance en hors-production (G1 ou G2)

1. Prévenir les utilisateurs du cluster impacté par mail
   1. Vous pouvez récupérer les adresses mails via la commande suivante (avec un ticket admin)

ipa user-find --sizelimit=10000 u\_ | awk -F ': ' '/Email/ && ! /rouen.francetelecom.fr/ {print($2)}' | sort | awk '!a[$0]++ {printf("%s;", $0)}'

1. Inscrire l’opération de maintenance dans le fichier excel de disponibilité des environnements hors prod.

<http://shp.itn.ftgroup/sites/CCDWH/Hadoop%20%20Public/disponibilite_environnements_horsprd_v2.xlsm>

1. Renvoyer un mail aux utilisateurs quand l’opération de maintenance est terminée.

Enfin, si l’opération de maintenance consiste en une modification de configuration, et si cette modification de configuration résout un problème, alors il faut créer un chantier pour que cette modification soit prise en compte par l’industrialisation, à savoir BDFOPS.

## Océane

CF MODOP envoyé par Fabrice (dans U:)

-       Je joins ci-dessous, une liste des EDS (identifiant OCEANE) avec lesquels nous aurons le plus d‘échanges.  Le nôtre étant JVOL60

* AQZT15 : DESI DIXSI SYS PROD SOUTIEN - Florian et Jean-Christophe
* AQZS02 : Equipe VM Ware pour les snapshot de VM.
* AQZC22 : DESI DDC HADOOP
* JVOL60 : S2A Big Data France - Centre de compétence
* JVOL10 : S3A CCBI Data Mining - Centre de compétence
* JVOL27 : S3A MOE BDF Alim
* JVOY51 : S3A BDF DSM

JABER Jade Ext DTSI/DSI

Bonjour Frédérique

Merci pour ton retour

mais je reste perdu

pour un redémarrage d'un serveur en hors production, je dois solliciter le service RUN ?

je pensais qu'ils ne se chargeaient que de la production ?

09:18MONCET Frédérique DTSI/DESI

Un serveur connait 2 phases la mep et le run

la mep des serveurs de Prod et de HProd est assuréee par la meme quipe

equipe, cad moi

le run est assuré par 2 equipes distinctes

09:20JABER Jade Ext DTSI/DSI

ok

quelles sont ces équipes ?

09:28MONCET Frédérique DTSI/DESI

RUn Prod : guillain Binel AQZS04

Run Hprod : Alain Dameme AQZI05

attention : le poc Prod est de la prod

09:28JABER Jade Ext DTSI/DSI

ok . Et votre code à vous ?

09:29MONCET Frédérique DTSI/DESI

aqzt10 pour les experts syst

et les coordinateurs

aqzs08 pour les coordinateurs en 2017

09:30JABER Jade Ext DTSI/DSI

merci Frédérique, c'est plus clair

Pour tout problème d’exploitation HP, vous pouvez contacter le département PFUNIX :

DESI / PPE/DEV :

* Océane (EDS : AQZI05)
* mail ([snpi.pfunix@orange.com](mailto:snpi.pfunix@orange.com))
* hotline (05 56 53 10 02)

Voici la liste des contacts de la supervision réseau de la DIF

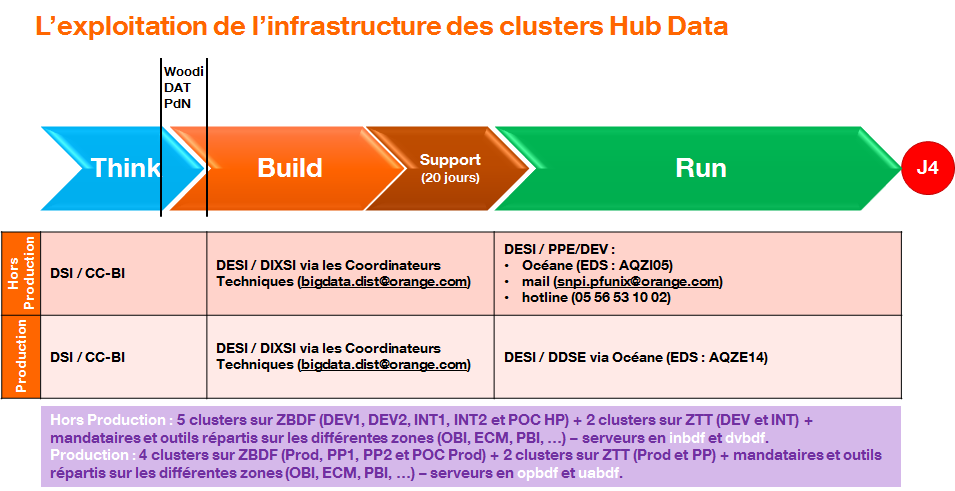
Contact équipe exploitant réseau :

* N° de groupement : 03 90 00 08 10
* Code EDS : FQCRMH

Contact SPOC : 02 28 56 12 12

Frederique MONCET

## Exploitation de l’infrastructure Hub Data



## Demande de création d’espace applicatif

La procédure pour demander un espace applicatif est la suivante :

1. Remplir le formulaire de demande d’espace applicatif situé sur le sharepoint public du CCBI Hadoop

<http://shp.itn.ftgroup/sites/CCDWH/Hadoop%20%20Public/Manuels/modele_demande_espace.xlsx>

1. Créer un ticket sur le site Orange Forge avec les informations suivantes :
   1. champ « sujet principal du ticket » : creation espace
   2. Attacher le fichier excel précédemment rempli.
2. Le CCBI Hadoop génère un script de création de l’espace à partir de ce fichier excel, puis va exécuter ce script sur les différents environnements Hors-Prod.
3. Le CCBI Hadoop « commit » ensuite le script de création de l’espace dans la gestion de configuration du use-case.
4. Lors du passage en production, l’équipe MOE du use-case doit embarquer le script ainsi livré à l’exploitant.

###

Issu du livret d’accueil disponible à l’adresse suivante ::

<http://shp.itn.ftgroup/sites/CCDWH/Hadoop%20%20Public/Livrets%20d%27accueil/Livret%20d%27accueil.docx>

Du coup, il faudrait donner un accès aux membres du CCBI qui le demandent.

Pour l’instant, Michel (ambo6211) et moi (uglo8383).

Jean-Nicolas, en attendant que je finalise un MOI que chaque Use-Case devra ajouter à sa procédure d’installation, voici comment faire une fois le nouveau colis reçu.

1. Le fichier create\_lbf.sh se trouve dans le colis applicatif du Use Case
2. Il faut transférer ce script create\_lbf.sh sur le serveur ADM01 du cluster sur lequel le UC installe son colis.
3. Vérifier l’existence du dossier suivant :
   1. /opt/application/bdfuc
   2. S’il n’existe pas, le créer avec les droits root:root et 700
4. Déposer le script create\_lbf.sh dans le dossier /opt/application/bdfuc
5. Vérifier que le script a les bons droits :
   1. root:root
   2. 700
6. Exécuter le script create\_lbf.sh via la commande :
   1. /opt/application/bdfuc/create\_lbf.sh G2PP2

J’ai créé le produit logiciel BDFUC afin de dissocier les scripts de créations livrés par les Use-Case à la DESI du script BDFADM livré par CCBIH à la DESI.

## Accèder aux IHM sans déclaration Cactus

Hello,

Pour info, on peut se connecter aux IHM de la VRF même si celles-ci ne sont pas déclarés dans Cactus.

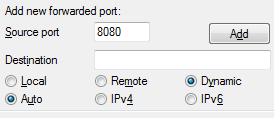
Evidemment, le plus simple est de les ajouter à Cactus, mais ça peut dépanner dans le cas où ce n’est pas fait, ou dans le cas où l’IHM fait des redirections sans utiliser le « -adm ». (ex : YARN UI)

++

Greg.

Ex : accéder à l’ihm IPA de prod.

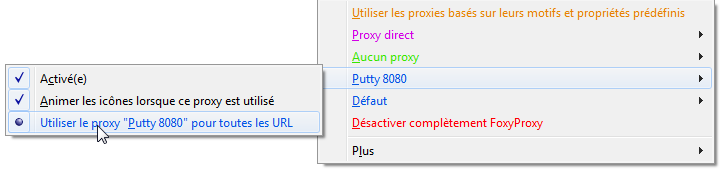
Il suffit de se connecter vi putty/cactus à un serveur (opbdfda01), et de faire un tunnel **dynamique** (sans oublier de cliquer sur Add)



Puis dans foxyproxy, déclarer un proxy socks v5 :



Et de forcer l’utilisation quand on a besoin de passer via ce proxy (ou via des motifs) :



TADA !

## Process réinstallation à l'identique des serveurs et modification unitaire des serveurs

Bonjour à tous,

Suite à la réunion sur la mise à jour du PDN Worker de cette semaine, j’ai contacté Axel Nabet pour convenir d’un process concernant les mises à jours impliquant une coordination entre les exploitants applicatifs et systèmes pour les serveurs de Hors-Production.

Dans le cas de modifications manuelles à réaliser sur plusieurs serveurs et nécessitant un traitement unitaire/par groupe (sortie du cluster, modification des paramètres, réintégration au cluster), l’équipe PFUNIX préfèrera une gestion avec **un ticket par serveur ou par groupe de serveur modifiable**.

Dans le cas où il serait plus rapide de faire une réinstallation « à l’identique » (même fonction/environnement mais configuration système additionnelle appliquée), l’opération serait traitée par les experts Prod dédiés BigData mais **il faudra préalablement notifier l’équipe exploitation système HProd** afin qu’il ne prennent pas en compte les alarmes qui pourraient remonter (indispo serveur). Il suffira de leur préciser une plage de jour pendant laquelle se déroulera l’opération ainsi que les serveurs impactés.

Pour ce cas, je propose le process suivant :

- CCBI demande à l’équipe Big Data DIXSI la réinstallation du/des serveurs

- TDM DIXSI BigData fait un mail à l’équipe PFUNIX (toutes les acteurs en copie de ce mail)

- Un expert système DIXSI procède à la réinstallation du/des serveurs en synchronisation avec CCBI

Est-ce que cela on viendrait à tout le monde ?

**@ Yann** : si validé, à ajouter au doc process stp ;)

Je reste à disposition si besoin.

Cordialement,

[**Florian Pernot**](http://annuaire.sso.infra.ftgroup/persons?searchType=PERSON_SIMPLE&_personCriteria.usePhonetic=on&personCriteria.sn=pernot&personCriteria.givenName=florian)