Storage Hadoop in 15 minutes

Preparing Hadoop

OpenCGA uses dependencies from Hortonworks HDP-2.5.0 internally. It has not been tested with other flavours of Hadoop.

You can use one of the many available hadoop sandboxes provided by Hortonworks or Cloudera or downloading and installing manually the required Hadoop components: Hadoop, Spark, HBase, Phoenix

Preparing OpenCGA

Download or pull the version you want to try.

```
git clone https://github.com/opencb/opencga.git
cd opencga
```

You can build the application from sources executing:

```
mvn clean install -DskipTests -Dstorage-hadoop -Popencga-storage-hadoop-
deps
```

You can customize some configuration parameters adding them to the compilation with `-D<param>=<value>`. Some interesting params are:

- `OPENCGA.INSTALLATION.DIR` for changing the installation directory.
- `OPENCGA.CLIENT.REST.HOST` This parameter indicates the address of the REST server. For this tutorial we are going to use a embedded REST server.
- `OPENCGA.CELLBASE.REST.HOST` to specify the cellbase installation.
- `OPENCGA.CELLBASE.VERSION` to specify the cellbase version.
- `OPENCGA.STORAGE.DEFAULT_ENGINE` to specify the default storage engine. By default is "mongodb", so we will need to add `--storage-engine hadoop` to each command. Compile with `-DOPENCGA.STORAGE.DEFAULT_ENGINE=hadoop` to avoid that.

To see the rest of the configurable parameters, check the **default-config** profile at the main pom.xml.

For example, to change the default engine and the rest host, execute:

```
mvn clean install -DskipTests -Dstorage-hadoop -Popencga-storage-hadoop-
deps -DOPENCGA.STORAGE.DEFAULT_ENGINE=hadoop  -DOPENCGA.CLIENT.REST.
HOST=http://localhost:9090/opencga
```

Then copy the application (the content of build folder) into the installation directory, by default and in this tutorial this is ***/opt/opencga***.

```
mkdir /opt/opencga
cp ./build/* /opt/opencga
```

- See Download and Installation for more information.

Needless to say, the computer where opencga is installed **must** have access to the Hadoop cluster.

Hadoop version

OpenCGA supports multiple versions of Hadoop. More specifically, it can be compiled against specific distributions of Hadoop.

To select a specific version (see table above) add **-P<version_key>** to the maven command line.

```
mvn clean install -DskipTests -Dstorage-hadoop -Popencga-storage-hadoop-
deps -DOPENCGA.STORAGE.DEFAULT_ENGINE=hadoop -Phdp-2.6.0
```

Supported Hadoop Distributions

Key	Name	Hadoop	HBase	Status	Since
hdp-2.5.0	Hortonworks HDP v2.5.0	`2.7.1`	1.1.2	Default	1.3.0
hdp-2.6.0	Hortonworks HDP v2.6.0	`2.7.3`	1.1.2		1.3.0
hdp-2.6.5	Hortonworks HDP v2.6.5	`2.7.3`	1.1.2		1.4.0
hdp-3.0.1	Hortonworks HDP v3.0.1	3.1.1	2.0.0	Experimental	1.4.0
emr-5.3.0	Amazon EMR Release 5.3.0	2.7.1	1.1.12		1.3.0
emr-5.8.0	Amazon EMR Release 5.8.0	2.7.3	1.3.1		1.3.0
cdh-5.13.0	Cloudera CDH v5.13.0	2.6.0	1.2.0		1.3.0

Hadoop configuration

In order to interact with Hadoop, we need to include the Hadoop configuration files to the classpath of the program. This can be automatic, or manual, depending on the way of accessing to Hadoop.

a) Read hadoop configuration from the server

This mode is for hadoop client nodes (or local installations, or hadoop nodes) where the commands 'hadoop', 'yarn' and 'hbase' are installed, and the client configuration updated. The script `bin/opencga-env.sh` will add the configuration files to the java classpath. Nothing else is needed.

In this scenario, you should be able to execute this commands:

```
hadoop classpath
hbase classpath
```

note: This mode does not work for executing OpenCGA within a Tomcat server. The configuration needs to be provided manually.

b) Provide hadoop configuration manually

In case of not having the hadoop binaries installed in the node, we need to provide the hadoop **client configuration files** manually. Take the configuration files from the cluster hadoop, and copy them into the folder `conf/hadoop` in the installation directory. This folder is automatically added to the classpath.

The hadoop client configuration files are usually distributed across multiple folders. Make sure that you get at least the general configuration files from hadoop (core-site.xml, hdfs-site.xml, yarn-site.xml, ...) and the specific configuration files from hbase (hbase-site.xml, ...).

You can also easily download the client configuration files from **Ambari** (Download Client Configuration Files) or from **Cloudera Manager** (Downloading Client Configuration Files)

The installation directory should look like this:

```
/opt/opencga/
 bin
 libs
 opencga-1.4.0.war
 opencga-storage-hadoop-core-1.4.0-jar-with-dependencies.jar
 README.md
 LICENSE
 conf
     configuration.yml
     storage-configuration.yml
     client-configuration.yml
     log4j.properties
     hadoop
         core-site.xml
         hbase-site.xml
         hdfs-size.xml
         hadoop-metrics2.properties
         yarn-site.xml
         ....
```

Configure Apache-Tomcat to connect to Hadoop

To provide the configuration files to the opencga server through Apache-Tomcat, we have to include them as a resource in the context.

Simply, place all the configuration files in one single folder, and add this configuration file to tomcat, replacing the "base" property to the location of the hadoop configuration files. The name of the file must match with the name of the opencga war.

tomcat8/conf/Catalina/localhost/opencga.xml

```
<Context>
    <Resources>
        <PostResources className="org.apache.catalina.webresources.
DirResourceSet"
                       webAppMount="/WEB-INF/classes"
base="${OPENCGA_INSTALLATION_DIR}/conf/hadoop" />
    </Resources>
</Context>
```

Initialising OpenCGA

To simplify the installation, we are going to use the embedded server for the REST API.

```
CATALOG_ADMIN_PASSWORD=admin_P@ssword
USER_PASSWORD=user_P@ssword

# Set up opencga
## Install OpenCGA
cd /opt/opencga
bin/opencga-admin.sh catalog install --secret-key not_so_secret_key <<<
$CATALOG_ADMIN_PASSWORD

## Start the daemon
mkdir -p logs
bin/opencga-admin.sh catalog daemon --start -p <<< $CATALOG_ADMIN_PASSWORD
$>> logs/daemon.log &

## Create our first user
bin/opencga-admin.sh users create -u platinum --user-email
platinum@illumina.com \
                --user-name Platinum \
                --user-organization Illumina \
                --user-password $USER_PASSWORD \
                --password <<< $CATALOG_ADMIN_PASSWORD
```

- See Getting started in 5 min for more info.

Using OpenCGA embedded REST server

In case of not having a Tomcat available, or to simplify test environment installation, we can use the embedded server.

```
## Start servers
mkdir -p logs/
bin/opencga-admin.sh server rest --start -p <<< $CATALOG_ADMIN_PASSWORD
$>> logs/server.log &
```

Then, we have to make sure that the `client-configuration.yml` points to the embedded server, instead of the tomcat server. By default, this server runs on the port 9090.

client-configuration.yml

```
---
## REST client configuration options
rest:
  host: "http://localhost:9090/opencga"
```

Indexing a VCF file

For this testing area, we are going to use a sample VCF data from the Platinum genomes. You can use any other file, but all the examples below use the VCF file platinum-genomes-vcf-NA12877_S1.genome.vcf.gz

You can find other files to load in this link: http://swdev.bioinfo.hpc.cam.ac.uk/downloads/datasets/vcf/platinum_genomes/gz/

Once OpenCGA is installed and running, we need to create a new project and study in catalog, and register our VCF file. You can also download all the files from that link

```
# Create study and folder structure
./opencga.sh users login -u platinum -p <<< $USER_PASSWORD
./opencga.sh projects create --alias platinum --name Platinum --organism-
scientific-name "homo sapiens" --organism-assembly GRCh37
./opencga.sh studies create --project platinum --alias platinum --name
Platinum
./opencga.sh files create-folder -s platinum --path 10_input
./opencga.sh files create-folder -s platinum --path 20_transformed
./opencga.sh files create-folder -s platinum --path 30_load
./opencga.sh files create-folder -s platinum --path 40_annotation
./opencga.sh files tree -s platinum --folder .

# Link files
wget 'http://swdev.bioinfo.hpc.cam.ac.uk/downloads/datasets/vcf
/platinum_genomes/gz/platinum-genomes-vcf-NA12877_S1.genome.vcf.gz'
./opencga.sh files link -s platinum -i /path/to/platinum/vcf/* --path
10_input
```

Once everything is set up, just need to load the files. This command line will create an internal job that will be executed by the catalog daemon.

```
# Index asynchronously via Daemon all the files in the folder 10_input
./opencga.sh files index --file 10_input --outdir 20_transformed --load
```

Optionally, we can use the opencga-analysis.sh command line for a synchronous execution:

```
# Index synchronously all the files in the folder 10_input
mkdir /tmp/opencga_job
./opencga-analysis.sh variants index --file 10_input --outdir /tmp
/opencga_job --path 20_transformed
```

Without Catalog

For testing purposes, it may be interesting to have an standalone installation of OpenCGA-Storage.You can find another build folder at `opencga/opencga-storage/build/` that contains only the binaries for storage

A simple indexation can be done executing the next command:

```
./opencga-storage.sh variant index --storage-engine hadoop --study-id 1 --
study-name platinum --gvcf --database opencga_platinum -i /path/to/platinum
/vcf/*
```

Annotate the variants database.

At this point, the last but not least, is annotate the variants. Despite this can be done at the same time than indexing variant files, it may be more clear in separated executions:

```
./opencga-analysis.sh variant annotate -o . --study platinum
```

This will annotate all the variants without annotation at the database, skipping the already annotated variants.

Querying variants

And we are done! At this point we will be ready to query variants. Here are some examples commands:

- Count number of variants

```
./opencga.sh variant query --study platinum --count
```

- Get the first 10 variants from the Chromosome 8

```
./opencga.sh variant query --study platinum --region 8 --limit 10 --sort
```

- Count variants in gene BRCA2

```
./opencga.sh variant query --study platinum --gene BRCA2 --count
```

You can find the full list of options at the help:

```
./opencga.sh variant query --help
```

You can find other query examples at this other tutorial: Querying Variants with the Command Line