Server Configuration

Required Dependencies

Java

OpenCGA needs **Java 8** to compile and run, we recommend to use **Oracle JDK 1.8.0_60+** since is the one we use and therefore it is fully supported. There are two main ways of installing Oracle JDK in Linux: *via package manager* and *manually*.

Installing ***via package manager*** can be more or less easy to do depending on the Linux distribution you use, for example for **Ubuntu 16.04** you can execute the following commands:

Shell

```
## You need root permissions
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

To install Java ***manually*** you can download the latest Java SDK for Linux x64 file (*linux-x64.tar.gz*) from Oracle at **http://www.oracle.com/technetwork/es/java/javase/downloads/index.html**

Then you have to uncompress it , move it to an installation folder such as */opt* and create a symbolic link:

Shell

```
## You can uncompress from the Download directory
tar -zxvf jdk-8u91-linux-x64.tar.gz

## You need root permissions for the following
sudo mv jdk1.8.0_91 /opt
sudo ln -s /opt/jdk1.8.0_91 java

## Then you should have something like this:
lrwxrwxrwx  1 root    root      11 Jun 13  2016 java -> jdk1.8.0_91/
drwxr-xr-x  8 imedina imedina 4.0K Jan  6 13:31 jdk1.8.0_91/
```

and finally you must set the *JAVA_HOME* variable in the system or user *bashrc* file:

Shell

```
## Add this to ~/.bashrc or /etc/bash.bashrc
export JAVA_HOME="/opt/java"
PATH="$JAVA_HOME/bin:$M2:$PATH"
```

Independently of the installation procedure followed above you can check is Java properly installed by executing *java -version*, you should get something like this:

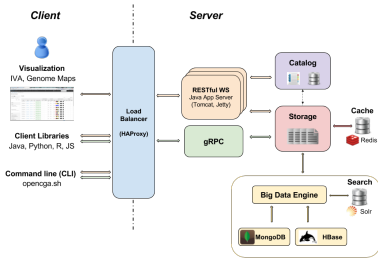
java -version

```
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
```

Tomcat

Table of Contents:

Remember OpenCGA Architecture:



OpenCGA needs **Tomcat 8** to run RESTful web services, we recommend to use **Apache Tomcat 8.x** since is the one we use in production but we have also used **Jetty** with any problem. You can install Apache Tomcat in Linux either *via package manager* or *manually*.

Installing ***via package manager*** can be more or less easy to do depending on the Linux distribution you use, keep in mind that Tomcat should run with the same user that OpenCGA. We will assume that OpenCGA will be installed with user *opencga*. Run the following to set Tomcat to run with user opencga. for example for **Ubuntu 16.04** you can execute the following command:

Shell

```
## You need root permissions
sudo apt-get install tomcat8

## You can check that Tomcat is running by executing
sudo service tomcat8 status

## Run the following to set Tomcat to run with user opencga
cd /var/lib/tomcat8/
sudo service tomcat8 stop
sudo chown -RL opencga:opencga ./*

## Edit the files /etc/init.d/tomcat8 and /etc/default/tomcat8
TOMCAT8_USER=opencga
TOMCAT8_GROUP=opencga

## Then reload the service and restart
sudo systemctl daemon-reload
sudo service tomcat8 start
```

You can install Tomcat ***manually*** by downloading it from http://tomcat.apache.org/download-80.cgi and executing as the OpenCGA installation user, execute the following commands:

Shell

```
## You can uncompress from the Download directory
tar -zxvf apache-tomcat-8.5.3.tar.gz

## You need root permissions to move to /opt and create a symbolic link
sudo mv apache-tomcat-8.5.3 /opt
sudo ln -s /opt/apache-tomcat-8.5.3 tomcat

## You can start and stop Tomcat executing
/opt/tomcat/bin/startup.sh
/opt/tomcat/bin/shutdown.sh
```

We recommend to make same changes in the Tomcat configuration:

- Increase Tomcat **memory**: you can do this by adding *JAVA_OPTS=" $JAVA_OPTS -Xms1024m -Xmx12288m "* to file *bin/catalina.sh*. In this example we have increased the memory to 12GB.
- Enable **compression** to save bandwidth: you can ask Tomcat to compress the HTML and JSON output, go to conf/server.xml and look for the connector 8080 and leave it like this:

```
    <Connector port="8080" protocol="HTTP/1.1"
        connectionTimeout="20000" redirectPort="8443"

        compression="on" compressableMimeType="text/html,text/xml,text
/plain,application/json"

    />
```

MongoDB 3.2

OpenCGA uses **MongoDB 3.2.x+** to store *Catalog* database and also as a possible backened for the *Variant Storage* engine. The best ways of installing MongoDB is from the Linux package managers, you can follow MongoDB tutorials for Ubuntu and RedHat/CentOS at:

- Ubuntu: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/
- RedHat/CentOS: https://docs.mongodb.com/manual/tutorial/install-mongodb-on-red-hat/

You can customise server configuration at */etc/mongodb.conf* or the client at *~/.mongorc.js*. Some recommended configuration for MongoDB:

- Always use WiredTiger engine
- Make sure the *journal* is enabled
- Make sure to have one directory per db with: *directoryperdb=true*
- For cluster installation we recommend to set up a Replica Set, this will improve performance and will make the installation more robust. OpenCGA versions 2.x+ utilise MongoDB's transactions, therefore a Replica Set is required.

Apache Solr

Apache Solr 6.x is used in OpenCGA as a complementary search engine for improving the performance of some queries and aggregations, full *text search* and *faceted* queries to Variant database. Solr is highly reliable, scalable and fault tolerant NoSQL database, it provides distributed indexing, replication, load-balanced querying, automated fail over, recovery, centralised configuration and more. You can learn how to install Solr at **https://cwiki.apache.org/confluence/display/solr/Installing+Solr**.

OpenCGA needs to be able to create new Solr *collections* and provide the Variant search schema. To do this you need to download the Server Configuration if you are using OpenCGA version 1.2.x or 1.3.x, for next version 1.4.0 you must use OpenCGAConfSet-1.4.0.tar.gz, then you just need to uncompress it and copy it into your Solr folder installation (i.e., $SOLR_HOME/server/solr/configsets) .

- For **Solr standalone installation**, you have to copy the OpenCGAConfSet-1.4.0 configuration into the folder *server/solr/configsets* for Solr standalone installation. Now, you can create new *core* or *collections* dynamically from Solr command line or from REST web services, e.g.:

  http://localhost:8983/solr/admin/cores?action=CREATE&name=my-new-core&configSet=OpenCGAConfSet-1.4.0

- For **Solr cloud installation**, you have to upload the OpenCGAConfSet configuation to your cluster by running the Solr zookeeper command line. In the following example command line, the OpenCGAConfSet configuration is located in the folder *server/solr/configsets*:

```
./bin/solr zk upconfig -n OpenCGAConfSet-1.4.0 -d server/solr/configsets
/OpenCGAConfSet-1.4.0 -z localhost:9983
```

  Now we can create collections from our search command line or from REST services, e.g.:

  http://localhost:8983/solr/admin/collections?action=CREATE&name=my-new-collection&numShards=2&collection.configName=OpenCGAConfSet-1.4.0

For more information, https://cwiki.apache.org/confluence/display/solr/Using+ZooKeeper+to+Manage+Configuration+File

Software Dependencies

DeepTools

OpenCGA uses the *bamCoverage* tool to extract a bigwig from every alignment file that is indexed through our system. This tool can be found inside the *DeepTools* suite, and could be easily installed using pip, the package installer for Python.

```
pip install deeptools
```

For more information: https://deeptools.readthedocs.io/en/develop/content/installation.html

Optional Dependencies

Apache Maven

OpenCGA uses **Apache Maven 3.x** as building tool, we use maven to compile, build, install and run tests of OpenCGA, so you do not need Maven unless you want to compile and build the source code for any reason, you can follow **Installation Guide > Building from Source Code** instructions to learn how to do it. Remember that you can always get stable OpenCGA binaries from **Installation#GettingOpenCGA**.

Maven can be easily installed in Linux **via package manager**, you can execute with sudo or as root the following commands in Ubuntu or CentOS:

Shell

```
## Ubuntu 16.04
sudo apt-get install maven

## CentOS 7.x
sudo yum install maven
```

Or you can also install Maven **manually** following this tutorial **http://maven.apache.org/install.html**

You can check that Maven is installed by executing *mvn -v* or *mvn -version*, you should get something like this:

Shell

```
## mvn -v  or  mvn -version
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-
10T16:41:47+00:00)
Maven home: /opt/maven
Java version: 1.8.0_91, vendor: Oracle Corporation
Java home: /opt/jdk1.8.0_91/jre
Default locale: en_GB, platform encoding: UTF-8
OS name: "linux", version: "4.4.0-57-generic", arch: "amd64", family:
"unix"
```

You need to add a Maven Profile to set up some variables that will be injected during the building, you can learn more about this at **Installation Guide > Building from Source Code**.

Redis

OpenCGA uses **Redis 2.8+** to cache queries to data. You can easily install Redis **via package manager**:

Shell

```
## Ubuntu 16.04
sudo apt-get install redis-server redis-tools

## CentOS 7.x
sudo yum install redis
```

This is a new feature in OpenCGA and at the moment we use the default Redis configuration.

Hadoop

This is an optional backend plugin for OpenCGA storage. We have used both Hortonworks and Cloudera as Hadoop distribution. At the moment we recommend **Hortonwork 2.5+**.

Cluster Dependencies

HAProxy

OpenCGA uses **HAProxy 1.5+** in the cluster installation to balance all REST web services call to the different deployed Tomcats. You can easily install HAProxy **via package manager**:

Shell

```
## Ubuntu 16.04
sudo apt-get install haproxy

## CentOS 7.x
sudo yum install haproxy
```

Here you can find an easy example for a very simple configuration, go to HAProxy for more documentation:

haproxy.conf

```
frontend http-in
        bind *:80
                ## Asign 'webservices' to paths starting with 'opencga'
                acl webservices         path_beg                -i
/opencga
                ## Redirect ACL 'webservices' to backend 'tomcat_prod'
                use_backend             tomcat_prod             if
webservices


## backend definition balancing two Tomcat instances
backend tomcat_prod
        cookie SERVERID insert nocache indirect
        balance roundrobin
        option httpclose
        option httpchk HEAD /
        server webprod01 10.21.2.3:8080 cookie webprod01tc check port 8080
inter 5s rise 2 fall 3 maxconn 300
        server webprod02 10.21.2.5:8080 cookie webprod02tc check port 8080
inter 5s rise 2 fall 3 maxconn 300
```