How to setup OpenGrok on Ubuntu

OpenGrok can be installed and used under different use cases. Advanced usage depends on your knowledge of running java applications and command line options. Note, that you need to create the index no matter what is your use case. Without indexes Opengrok will be simply useless.

1. Requirements

a, You need the following:

- JDK 1.8 or higher
- OpenGrok '''binaries''' from https://github.com/OpenGrok/OpenGrok/releases (.tar.gz file with binaries, not the source code tarball !)
- https://github.com/universal-ctags for analysis (avoid Exuberant ctags, they are not maintained anymore)
- A servlet container Tomcat 8.0 or later also running with Java at least 1.8
- A recent browser for clients - IE, Firefox, recent Chrome or Safari

b, Setup

- Install JDK easier by command:

sudo apt-get install default-jdk

(to install latest version)

- Download and extract OpenGrok binaries in somewhere.
- Download Ctags from https://github.com/universal-ctags/ctags and deploy it by command:

git clone https://github.com/universal-ctags/ctags.git

cd ctags

./autogen.sh

./configure

make

sudo make install

Note: Install automake and pkg-config before running ./autogen.sh

- Download and install Tomcat easier by command:

sudo apt-get install tomcat8

or follow link: https://linuxize.com/post/how-to-install-tomcat-8-5-on-ubuntu-18.04/

2. Creating the index

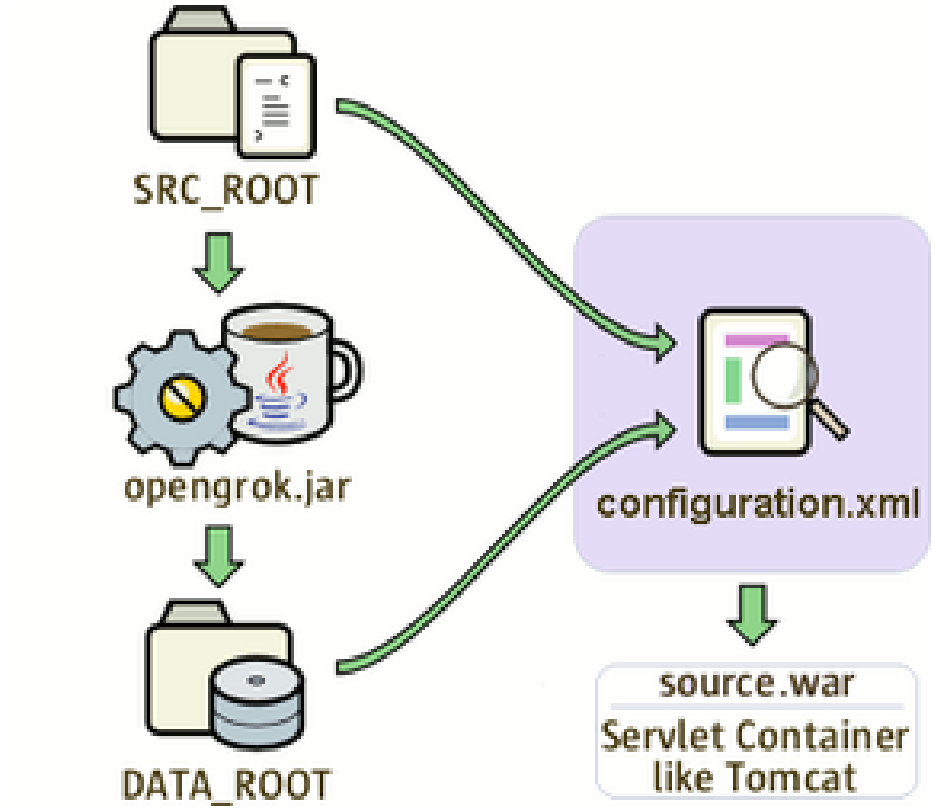
The data to be indexed should be stored in a directory called **source root**. Each subdirectory under this directory is called **project** (projects can be disabled but let's leave this detail aside for now) and usually contains checkout of a **repository** (or it's branch, version, ...) sources. Each project can have multiple repositories.

The indexer will process any input data - be it source code checkouts, plain files, binaries, etc.

The concept of projects was introduced to effectively replace the need for multiple web applications with opengrok .war file (see below) and leave you with one indexer and one web application serving more source code repositories - projects.

That said, OpenGrok can be run in project-less setup where all the input data is always searched at once.

The index data will be created under directory called **data root**.



Step.0 - Setting up the sources / input data

Input data should be available locally for OpenGrok to work efficiently since indexing is pretty I/O intensive. No changes are required to your source tree. If the code is under CVS or SVN, OpenGrok requires the '''checked out source''' tree under source root.

The source root directory needs to be created first, e.g. on Unix: *mkdir -p /var/opengrok/src*

The indexer assumes the input data is stored in the UTF-8 encoding (ASCII works therefore too).

For example, to add 2 sample code checkouts using the default source root on Unix system:

cd /var/opengrok/src

use one of the training modules at GitHub as an example small app.

git clone git@github.com:githubtraining/hellogitworld.git

use Git as an example large app

git clone git@github.com:git/git.git

These 2 directories will be treated as projects if the indexer is run with projects enabled (the -Poption), otherwise the data will be treated as a whole.

Step.1 - Install management tools (optional)

This step is optional, the python package contains wrappers for OpenGrok's indexer and other commands. In the release tarball navigate to tools subdirectory and install the opengrok-tools.tar.gz as a python package. Then you can use defined commands. You can of course run the plain java yourself, without these wrappers. The tools are mainly useful for parallel repository synchronization and indexing and also in case when managing multiple OpenGrok instances with diverse Java installations.

In shell, you can install the package simply by:

$ python3 -m pip install opengrok-tools.tar.gz

Of course, the Python package can be installed into Python virtual environment.

Step.2 - Deploy the web application

Install web application container of your choice (e.g. Tomcat, Glassfish).

The web application is distributed in the form of WAR archive file called source.war by default. To deploy the application, it means to copy the .war file to the location where the application container will detect it and deploy the web application. The container application will usually detect the new file (even if previous version of the web application is already running), unpack the archive and start the web application. Usually, it is not necessary to unpack the archive by hand. It depends on the container server how quickly it will discover the new archive; usually it takes just a couple of seconds. The destination directory varies per application server. For example for Tomcat 8 it might be something like /var/tomcat8/webapps however this could vary based on operating system as well. So, if you copy the archive to say /var/tomcat8/webapps/source.war, the application server will extract the contents of the archive to the /var/tomcat8/webapps/source/ directory.

Once started, the web application will be served on http://ADDRESS:PORT/source/ where ADDRESSand PORT depend on the configuration of your application server. For instance, it could be http://localhost:8080/source. The source part of the URI matches the name of the WAR file, so if you want your application to be available on http://localhost:8080/FooBar/ , copy the file into the destination directory as FooBar.war.

After the initial startup (i.e. before the indexer is run for the first time) the web application will display an error saying that it cannot read the configuration file. This is expected since the configuration file is yet to be generated by the indexer.

After application server unpacks the War file, it will search for the WEB-INF/web.xml file. For example, deployed default War archive in Tomcat 8 on a Unix system might have the file present as /var/tomcat8/webapps/source/WEB-INF/web.xml. Inside this XML file there is an parameter called CONFIGURATION. Inside the XML file it might look like this:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

        http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"

        version="3.1">


    <display-name>OpenGrok</display-name>

    <description>A wicked fast source browser</description>

    <context-param>

        <description>Full path to the configuration file where OpenGrok can read its configuration</description>

        <param-name>CONFIGURATION</param-name>

        <param-value>/var/opengrok/etc/configuration.xml</param-value>

    </context-param>

...
```

This is where the web application will read the configuration from. The default value is /var/opengrok/etc/configuration.xml (notice that in the above example non-default path was used). This configuration file is created by the indexer when using the -W option and the web application reads the file on startup - this is a way how to make the configuration persistent.

If you happen to be using the Python tools distributed with OpenGrok, you can use the opengrok-deploy script to perform the copying of the War file while optionally changing the CONFIGURATIONvalue if the configuration file is stored in non-default location.

See https://github.com/oracle/opengrok/wiki/Webapp-configuration for more configuration options of the web application.

Also see https://github.com/oracle/opengrok/wiki/Security

Step.3 - Indexing

This step consists of these operations:

- create index
- let the indexer generate the configuration file
- notify the web application that new index is available

For the indexing step, the directories that store the output data need to be created first, e.g. for Unix system: mkdir -p /var/opengrok/{data,etc}

The initial indexing can take a lot of time - for large code bases (meaning both amount of source code and history) it can take many hours. Subsequent indexing will be much faster as it is incremental.

To run the indexer you will need the opengrok.jar file that is found in the release tar.gz file plus all the libraries found therein. For example, unpack (assumes GNU tar) the release tarball as follows:

mkdir -p /opengrok/dist &&

tar -C /opengrok/dist --strip-components=1 -xfz opengrok-X.Y.Z.tar.gz

and then the indexer can be run either using opengrok.jar directly:

java -Djava.util.logging.config.file=/var/opengrok/logging.properties ₩

-jar /opengrok/dist/lib/opengrok.jar ₩

-c /path/to/universal/ctags ₩

-s /var/opengrok/src -d /var/opengrok/data -H -P -S -G ₩

-W /var/opengrok/etc/configuration.xml -U http://localhost:8080/source

or using the opengrok-indexer wrapper like so:

opengrok-indexer -J=-Djava.util.logging.config.file=/var/opengrok/logging.properties ₩

-a /opengrok/dist/lib/opengrok.jar -- ₩

-c /path/to/universal/ctags ₩

-s /var/opengrok/src -d /var/opengrok/data -H -P -S -G ₩

-W /var/opengrok/etc/configuration.xml -U http://localhost:8080/source

Notice how the indexer arguments in both commands are the same. The opengrok-indexer script will merely find the Java executable and run it.

At the end of the indexing the indexer automatically attempts to upload newly generated configuration to the web application. Until this is done, the web application will display the old state. The indexer needs to know where to upload the configuration to - this is what the -U option is there for. The URI supplied by this option needs to match the location where the web application was deployed to, e.g. for War file called source.war the URI will be http://localhost:PORT_NUMBER/source.

The above will use /var/opengrok/src as source root, /var/opengrok/data as data root. The configuration will be written to /var/opengrok/etc /configuration.xml and sent to the web application (via the URL passed to the -U option) at the end of the indexing. The location of the configuration file needs to match the configuration location in the web.xml file (see the Deploy section above).

Run the command with -h to get more information about the options, i.e.:

java -jar /opengrok/dist/lib/opengrok.jar -h

or when using the Python scripts:

opengrok-indexer -a /opengrok/dist/lib/opengrok.jar -- -h

Optionally use --detailed together with -h to get extra detailed help, including examples.

It is assumed that any SCM commands are reachable in one of the components of the PATH environment variable (e.g. the git command for Git repositories). Likewise, this should be maintained in the environment of the user which runs the web server instance.

You should now be able to point your browser to *http://YOUR_WEBAPP_SERVER:WEBAPPSRV_PORT/source* to work with your fresh installation.

QUICK GUIDE UPDATE PROJECT FOR OPENGROK ON http://192.168.100.44:8080/

For example, I will update a project name ABC.

1, Go to */var/lib/tomcat8/webapps/*, add more direction *ABC/*

You can add more directory by command:

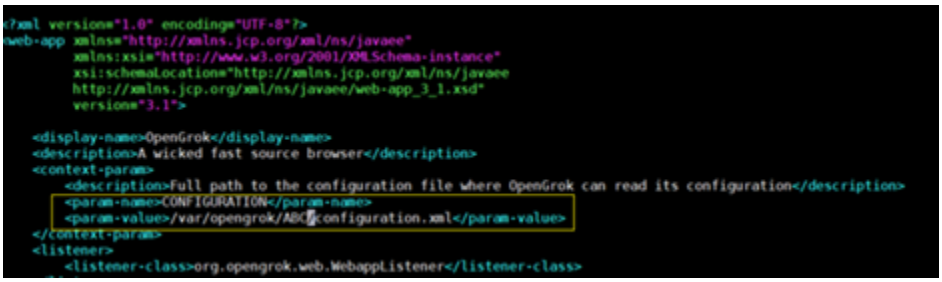
sudo cp –rf /var/lib/tomcat8/webapps/VPEA/ /var/lib/tomcat8/webapps/ABC/

2, Go to */var/lib/tomcat8/webapps/ABC/WEB-INF/* and edit file web.xml by:

cd to /var/lib/tomcat8/webapps/ABC/WEB-INF/

sudoedit web.xml

and change path to file configure like this:



3, Go to */var/opengrok/* add empty direction name *ABC/*. Inside *ABC/* create two folders name *data/* and *src/*.

- */ABC/src*: store source code
- */ABC/data*: where Opengrok indexer for source code

4, Build Jenkins job name ABC on link: http://192.168.100.29:8080/job/OpenGrok/

- Configure for Jenkins pull source code into server 192.168.100.44
- In this case, Jenkins pull source code to */home/vntrong/workspace/OpenGrok/ABC*

5, Make soft link between */home/vntrong/workspace/OpenGrok/ABC and /var/opengrok/*src

sudo ln –s /home/vntrong/workspace/OpenGrok/ABC /var/opengrok/src

(you can copy it, instead)

6, Go to */var/lib/tomcat8/webapps/ROOT/* and add project ABC in *index.html*

Sudoedit index.html

7, Go to configure Jenkins job ABC add script to Build > Execute shell:

sudo /opt/opengrok/opengrok-tools/bin/opengrok-indexer ₩

-J=-Djava.util.logging.config.file=/var/opengrok/logging.properties ₩

-a /home/vntrong/opengrok/opengrok-1.1.2/lib/opengrok.jar -- ₩

-c /usr/local/bin/ctags ₩

-s /var/opengrok/ABC/src -d /var/opengrok/ABC/data -H -P -S -G ₩

-W /var/opengrok/ABC/configuration.xml -U http://localhost:8080/ABC

8, Go to http://192.168.100.44:8080/ . If ABC exist with time and date, your job was successful.