Install a development environment for Cytomine ULiege

⚠ **This guide is designed for an installation on Ubuntu 16.04+ (18.04 tested).**

⚠ **This guide works for Cytomine-bootstrap, Cytomine-Core and Cytomine-IMS development versions released from January 2019.**

Check page history for previous versions of the tutorial.

If you are interested to develop new modules for Cytomine-Core (e.g. new web services, new entity in the data model,...) and/or new modules for Cytomine-IMS (e.g. support of a new image format, new image web service, ...), follow this guide. Note that if you are only interested to plug a new script of image analysis algorithm, we recommend instead to look at the [DOC] Interoperability (data & algorithms) documentation page.

This guide will help you to install Cytomine on your local machine for development. First, we will make a regular installation of Cytomine, then tweak it to allow you to develop and test on your own computer.

Step 1 - Make a regular installation of Cytomine

Follow the guide Install Cytomine on Linux to make a regular installation of Cytomine.

In particular, **install Docker** (step 1), **retrieve Cytomine bootstrap** (step 2) and **configure your instance** (step 3).

In the configuration (`configuration.sh`), set

- the parameter `CORE_DEVELOPMENT` to true if you want to develop on Cytomine-core.
- the parameter `IMS_DEVELOPMENT` to true if you want to develop on Cytomine-IMS.

Development on Core and IMS can be enabled together.

Then, **initialize your instance deployment** (step 4) and **deploy your instance** (step 5).

Step 2 - Install development requirements

Cytomine-Core and Cytomine-IMS mainly rely on Java, Groovy and the Grails framework.

Oracle Java 8

```
sudo apt-get update -y
sudo apt-get install -y git curl software-properties-common wget unzip build-essential locate apt-transport-
https


sudo echo debconf shared/accepted-oracle-license-v1-1 select true | sudo debconf-set-selections
sudo echo debconf shared/accepted-oracle-license-v1-1 seen true | sudo debconf-set-selections
sudo add-apt-repository -y ppa:webupd8team/java
sudo apt-get update -y
sudo apt-get install -y oracle-java8-installer
echo 'export JAVA_HOME="/usr/lib/jvm/java-8-oracle"' >> .bashrc
```

Check Java version with `java -version`. The output should be something like '`java version "1.8.0_191"`'.

Grails & Groovy with sdkman

```
cd /tmp && curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk install grails 2.4.4
sdk install groovy 2.3.7
```

Fix Grails installation for Java 8

Restart a terminal in order the environment variable `$GRAILS_HOME` is effective. Fix Grails 2.4.4 to work with Java 8:

```
wget repo.spring.io/libs-snapshot-local/org/springframework/springloaded/1.2.7.BUILD-SNAPSHOT/springloaded-
1.2.7.BUILD-SNAPSHOT.jar -O $GRAILS_HOME/lib/org.springframework/springloaded/jars/springloaded-1.2.1.RELEASE.
jar
```

It may be also useful to install a modern web browser such as Google Chrome, to navigate into the web interface.

Step 3 - Install an integrated development environment

We recommend to use Intellij IDEA Ultimate Edition. The following instructions considers it is the case. A free license of the application can be obtained with an academical email address.

Install Intellij IDEA

Download Intellij IDEA Ultimate Edition on their website: https://www.jetbrains.com/idea/download/#section=linux

Install it by following the instructions on their website: https://www.jetbrains.com/help/idea/install-and-set-up-intellij-idea.html

Create Cytomine-core project in Intellij IDEA

This tutorial presents the configuration of Cytomine-core project in Intellij IDEA. For Cytomine-IMS, read the supplementary required steps at the end of this document.

Click on **Check out from version control** and choose **Git** in the contextual menu. The repository URL for Cytomine-core is

```
https://github.com/Cytomine-ULiege/Cytomine-core.git
```

You can also manually clone the repsoitory with `git clone` and open the project in Intellij in File > Open.

Configure Java SDK

Go to **File > Project Structure > Project** to choose the project SDK corresponding to the previously installed version of Java.

If the previously installed Java JDK is not available in the dropdown list, click on **New** and reference the SDK with the path `/usr/lib/jvm/java-8-oracle`.

Configure Grails SDK

Go to **Tools > Grails > Configure Grails SDK**. You can retrieve the **Grails SDK Home** by running the command

```
echo $GRAILS_HOME
```

Then, click on **Ok**.

If Grails is not available in the **Tools** menu, first go on the **Project panel** on left, right click on the project name "Cytomine-Core" then choose **Add framework support** and choose Grails.

Install Grails plugins

We will now install required Grails plugins. Go to **Tools > Grails > Run Grails Command** (Shortcut: Ctrl + Alt + G) and enter **`clean-all`** as command.

Run a new Grails Command: **`compile`**. It will download required plugins, install and compile them.

Do not pay attention to potential warnings related to `be.cytomine.customUIController` and `be.cytomine.utils.Task`

Edit running configuration

Go to **Run > Edit Configurations**.

First, **delete the configuration named "Grails: Cytomine"** by clicking on the Minus button on top-left or with Alt+Delete.

Click on the **+ button on top-left corner**, and create a new configuration by selecting "Grails" in the list. Fill the form with

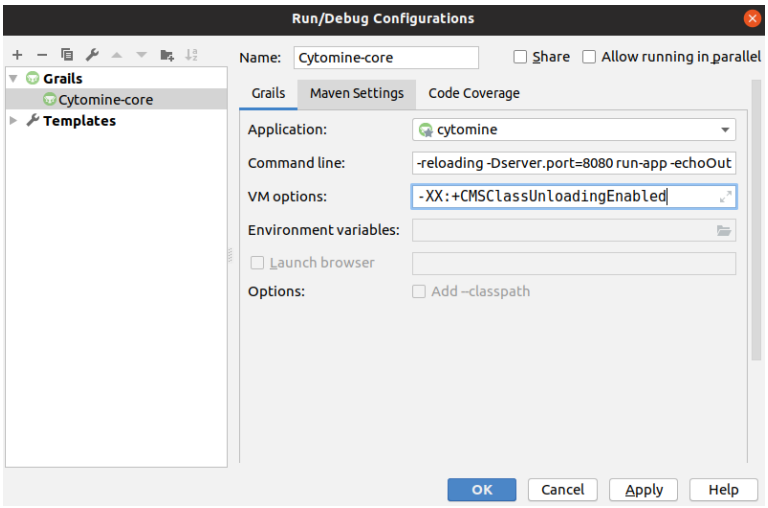
- Cytomine as application,
- for the command line:

```
-reloading -Dserver.port=8080 run-app -echoOut --stacktrace -nocoverage --verbose
```

- for the VM options:

```
-XX:+CMSClassUnloadingEnabled
```

Then click on **Ok**.



Link bootstrap configuration with development environment

The last operation before running Cytomine-core is to **link the configuration established in the bootstrap (see Step 1) with the Cytomine-core development environment**. As the core is no more running in a Docker container, we need to make this link manually.

In your Cytomine-Core project, **open** the `Cytomine-core/grails-app/conf/Config.groovy` file. In the first lines, **update the `grails.config.locations` property in the development environment** with the path to the generated core configuration (`configs/core/cytomineconfig.groovy`) in your bootstrap (see Step 1). For example, at line 9,

```
...
grails.config.locations = [""]
environments {
    production {
        grails.config.locations = ["file:${userHome}/.grails/cytomineconfig.groovy"]
    }
    development {
        // Update the file path so that it matches the generated configuration file in your bootstrap
        grails.config.locations = ["file:${userHome}/Cytomine/Cytomine-bootstrap/configs/core/cytomineconfig.
groovy"]
    }
}
...
```

Step 4 - Run Cytomine locally and develop

Make sure that Cytomine bootstrap is deployed (`start.sh` or `restart.sh` has finished) and run the run configuration we created before in Intellij. You can now access the core development server at http://CORE_URL:8080 (where `CORE_URL` is given in bootstrap `configuration.sh`) to test your core development environment. It will rely on other containers installed by Docker but use your own core development environment.

If you update your bootstrap configuration, run the following command to generate new configuration files and redeploy the containers.

```
sudo bash init.sh
sudo bash restart.sh
```

Then run Cytomine-core in Intellij.

If you set `IMS_DEVELOPMENT` to true in step 1, to run Cytomine-IMS locally, you need to create a new project in Intellij by cloning

https://github.com/Cytomine-ULiege/Cytomine-IMS.git

and perform the supplementary steps detailed below before to run the configuration in Intellij.

Supplementary steps to run IMS locally

Install locally the IMS required libraries

If you want develop on IMS, the IMS will run on your machine and you will need all the dependencies of IMS.

```
sudo apt-get install -y imagemagick libvips libvips-dev libvips-tools libtiff-tools openslide-tools autoconf
sudo chmod -R 777 /data

# Install Openslide-Java bindings
export CFLAGS="-I/usr/lib/jvm/java-8-oracle"
cd /tmp
git clone https://github.com/cytomine/openslide-java
cd /tmp/openslide-java/
autoreconf -i
./configure
make
sudo make install
```

Update your /etc/hosts

If you kept the default configuration in the bootstrap (step 1), append to your `/etc/hosts`:

```
127.0.0.1   localhost-iip-base
127.0.0.1   localhost-iip-jp2000
127.0.0.1   localhost-iip-cyto
```

Fix Background plugin

If you still develop with a version that use Background plugin, in order to avoid the `org.codehaus.groovy.grails.commons.ConfigurationHolder: class not found` error when compiling Cytomine-IMS, edit the file `~/.grails/2.4.4/projects/IMS/plugins/background-thread-1.6/BackgroundThreadGrailsPlugin.groovy`, by replacing

```
import org.codehaus.groovy.grails.commons.ConfigurationHolder as CH
```

by

```
import grails.util.Holders as CH
```

Use an available port

If you run Core and IMS locally at the same time, each component has to run on a dedicated port. In this guide, the port 8080 is used for Cytomine-core. For Cytomine-IMS you can use the port 9080 for example. Go to Run > Edit Configuration and update the port on which IMS will run :

```
-reloading -Dserver.port=9080 run-app -echoOut --stacktrace -nocoverage --verbose
```

Link bootstrap configuration with development environment

In your Cytomine-IMS project, in the `Cytomine-IMS/grails-app/conf/Config.groovy` file of your Cytomine-IMS project,

```
...
//image manipulation executable
cytomine.vips = "vips"
cytomine.tiffinfo = "tiffinfo"
cytomine.identify = "identify"
cytomine.vipsthumbnail = "vipsthumbnail"
...
```

Supplementary steps to run new VueJS front-end locally

We have started to migrate our front-end into a brand new VueJS front-end. For now, it has not been integrated into Cytomine-core (still running in Grails 2.4) and will be done in the future when the whole application will be migrated to Grails 3.

To develop on the new front-end, clone the Github repository locally (https://github.com/Cytomine-ULiege/cytomine-frontend) and install it in your favorite Javascript editor. The package is configured with Webpack.

To run the new front-end, first deploy the application with Cytomine-core listening on `localhost-core:8080`. Then, simply run

```
npm run dev
```

to run a development server. It will launch a server on `localhost:8085` which will act as a proxy to `localhost-core:8080` so that the VueJS front-end will override the old one on `localhost:8085`.

If the hot reload does not seem to work, a possible issue can be related to Intellij. Check Inotify Watches Limit in Intellij IDEA documentation.

To make the changes persistent, run

```
npm run build
```

and copy the produced bundle in `Cytomine-core/web-app/application/views/v2.0/dist` directory of the Cytomine-core project.

Troubleshooting

- If you encounter a "`java.lang.OutOfMememoryError: PermGen space`" error, add the option `-XX:+CMSClassUnloadingEnabled` to the JVM, when running Cytomine-Core. In Intellij, go to Run > Edit Configurations and add it in the VM options field.
- As nginx will try to contact your machine, you may need to allow docker containers to contact port 8080 & 9080. Here is the command with UFW

  ```
  sudo ufw allow from 172.17.0.1/27
  ```