# How to install a development environment for Cytomine v1.1 with Docker

⚠ **For installation procedure of V1.0, see this page.**

⚠ **This guide is designed for an installation on Ubuntu 16.04 LTS.**

In this page we assume 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, ...).

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.

We will explain all the necessary steps to configure a development environment on your computer. The idea is to install a Cytomine instance on your localhost and all the development tools/libraries required so that you can develop and test on your own computer.

## Step 1 - Install the pre-requirements

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

```
sudo apt-get update -y
sudo apt-get install -y git curl linux-image-extra-$(uname -r) linux-image-extra-virtual apt-transport-https
software-properties-common ca-certificates g++

curl -s "https://get.sdkman.io" | bash
source "/home/{your_home}/.sdkman/bin/sdkman-init.sh"
sdk install groovy 1.8.6
sdk install grails 2.4.4

echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu precise main" | sudo tee -a /etc/apt/sources.list
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
sudo apt-get update
echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-
selections
sudo apt-get install -y oracle-java7-installer
```

Edit your .bashrc to export your JAVA_HOME:

```
export JAVA_HOME="XXX"
```

where XXX is obtained with echo $JAVA_HOME. For example, export JAVA_HOME="/usr/lib/jvm/java7-oracle".

Then, check Java and Groovy version:

```
java -version #(1.7)
groovy -v #(1.8.6)
```

If you prefer, you can visualize the installation with screenshots here.

## Step 2 - Install an integrated development environment

We recommend to use Intellij IDEA Ultimate Edition. The following considers it is the case.

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

Install it :

```
cd Desktop/
tar -zxvf ../Downloads/ideaIU-2017.2.5.tar.gz
cd idea-IU-172.4343.14
./idea.sh
```

If you prefer you can visualize the installation with screenshots here.

# Step 3 - Retrieve source code

Download our source code in a local directory (e.g. $HOME/Cytomine/) by cloning Cytomine-core and Cytomine-IMS from our source repository:

```
mkdir Cytomine/ ; cd Cytomine/
git clone https://github.com/cytomine/Cytomine-core.git
git clone https://github.com/cytomine/Cytomine-IMS.git
```

# Step 4 - Configure projects in Intellij

Import the Cytomine-Core and Cytomine-IMS source project into Intellij, by choosing Maven as external model to import the projects.

Choose the right JDK version.

If needed, right click on the project (left menu) and choose "Add Framework support" to select Groovy.

If Grails SDK is not found (see Event Log), click on "Configure SDK Home". The path is given by

```
echo $GRAILS_HOME
```

If the source root of the project is not detected, in the Project panel on left, right click on `grails-app` folder and then Mark Directory as > Source root.

For more detailed importation of projects if you are not familiar with Intellij, see the screenshots on this page.

# Step 5 - Create a Dockerized development environment.

One of the main interest of Docker is to have the same environment between developers and final users.

In order to do this and to avoid installation of an additional local machine, we will setup a Dockerized development environment. Follow the two first points of an Automatic Installation with Docker (Download Cytomine-bootstrap and install Docker).

# Step 6 - Install IMS required libraries (if needed).

If you want develop on IMS, the IMS will run on your machine and you will need all the dependencies of IMS : *tiffinfo, identify, vips*. You can find into the Dockerfile of IMS the way to install these dependancies. The following commands should be enough to get the related libs:

```
sudo apt-get -y update
sudo apt-get install -y software-properties-common
sudo add-apt-repository -y ppa:semiosis/ppa
sudo apt-get -y update
sudo apt-get install -y ant dnsutils gobject-introspection gtk-doc-tools imagemagick libcairo2-dev libglib2.0-
dev libgsf-1-dev libjpeg8-dev libmemcached-dev libopenjpeg-dev libtiff5-dev libtiff-tools libxml2-dev pkg-
config swig
sudo apt-get -y build-dep openslide iipimage-server libvips-dev

# Openslide
cd /tmp
git clone https://github.com/openslide/openslide
cd /tmp/openslide
sudo autoreconf -i
sudo ./configure
sudo make
sudo make install

# Vips
cd /tmp
wget https://github.com/jcupitt/libvips/releases/download/v8.5.1/vips-8.5.1.tar.gz
tar -zxvf ./vips-8.5.1.tar.gz
cd ./vips-8.5.1
ldconfig
sudo LDFLAGS="-L/usr/local/lib -lopenslide" CPPFLAGS="-I/usr/local/include/openslide"  ./configure
sudo make
sudo make install

# Openslide-java
cd /tmp
git clone https://github.com/cytomine/openslide-java
cd /tmp/openslide-java
sudo autoreconf -i
sudo ./configure
sudo make
sudo make install
```

# Step 7 - Update the bootstrap procedure.

Then, in the Bootstrap folder you have downloaded at step 5, make the following changes :

1) In the `nginx/deploy.sh` file, replace the lines

```
sed -i "s/CORE_ALIAS/core/g" /tmp/nginx.conf.sample
sed -i "s/IMS_ALIAS/ims/g" /tmp/nginx.conf.sample # Only if IMS development

# and (only if IMS development)

sed -i "s/IMS_URLS_CONFIG/                      proxy_pass http:\/\/ims:8080; \\`echo -e '\n\r'`
IMS_URLS_CONFIG/g" /tmp/nginx.conf.sample
```

by

```
LOCAL=$(route -n | awk '/UG[ \t]/{print $2}')
sed -i "s/CORE_ALIAS/$LOCAL/g" /tmp/nginx.conf.sample
sed -i "s/IMS_ALIAS/$LOCAL:9080/g" /tmp/nginx.conf.sample # Only if IMS dev

# and  (only if IMS dev)

sed -i "s/IMS_URLS_CONFIG/                      proxy_pass http:\/\/$LOCAL:9080; \\`echo -e '\n\r'`
IMS_URLS_CONFIG/g" /tmp/nginx.conf.sample
```

2) If you develop IMS, in the `nginx/nginx.conf.sample`, replace the line

```
proxy_pass   http://IMS_ALIAS:8080;
```

by

```
proxy_pass   http://IMS_ALIAS;
```

3) In the `restart_keep_data.sh`, replace the lines

```
#sh create_docker_images.sh

# and

sh start_deploy.sh
```

by

```
cd nginx && docker build -t="cytomine/nginxdev" .
cd ..

# and

sh start_dev.sh
```

6) If you develop IMS, in the `clean_docker_keep_data.sh`, add the lines

```
docker stop iipJ2
docker rm -v iipJ2
```

# Step 8 - Update your /etc/hosts

Don't forget to update it, if you deploy on local host. If you kept the default configuration, update your `/etc/hosts` with:

```
127.0.0.1   localhost-core
127.0.0.1   localhost-ims
127.0.0.1   localhost-ims2
127.0.0.1   localhost-upload
127.0.0.1   localhost-retrieval
127.0.0.1   rabbitmq
127.0.0.1   localhost-iip-base # if IMS dev
127.0.0.1   localhost-iip-jp2000 # if IMS dev
127.0.0.1   localhost-iip-cyto # if IMS dev
```

# Step 9 - Last modifications to your IDE

The last modifications need to be done in your development environment (IDE).

In your Cytomine-Core project:

- In the `Cytomine-core/grails-app/conf/Config.groovy` file of your Cytomine-core project,
  - update the value `grails.serverURL` (development part) by *"http://CORE_URL:8080"*.

    - update the public and private keys of admin, superAdmin, and ImageServer users. You can run the *cat* `/proc/sys/kernel/random/uuid` command to generate keys
    - update the value *grails.adminPassword* by the default password of the admin user and add the *retrievalUsername* and *retrievalPassword* *(*default : cytomine and retrieval_default*)*
- In the `DataSource.groovy` file into the Core project, update the value *username* and *password* (development part) by "*docker"* and the *url* by *"jdbc:postgresql://localhost:5432/docker".*
- In IntelliJ, in the "Run" menu toolbar, Go to "Edit configurations" edit the Command Line by "`-reloading -Dserver.port=8080 run-app -echoOut --stacktrace -nocoverage --verbose`"

In your Cytomine-IMS project:

- In the `Cytomine-ims/grails-app/conf/Config.groovy` file,
    - update the public and private keys of admin, superAdmin, and ImageServer users with the keys generated at previous step.
    - as long as the database is not Dockerized, update the the *cytomine.imageServerPrivateKey* and *cytomine.imageServerPublicKey* values by the private and public keys of the username 'ImageServer1' (in the *sec_user* table) in the database.
- In IntelliJ, in the "Run" menu toolbar, Go to "Edit configurations" edit the Command Line by *"*`-reloading -Dserver.port=9080 run-app -echoOut --stacktrace -nocoverage --verbose`*".*

# Step 10 - Start Docker containers and develop

In Cytomine-boostrap directory, you can now run the script *start_dev.sh.* This will deploy the Docker containers:

```
sudo ./start_dev.sh
```

In IntelliJ, you can now run the Cytomine-core project that will simulate a server at the grails.serverURL mentionned above. So in this example you can now access the core development server at http://CORE_URL:8080 to test your core dev environment. It will rely on other containers installed by Docker but use your own core (and IMS) development container(s).

To restart the docker containers while keeping data, use:

```
sudo ./restart_and_keep_data.sh
```

# 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.
- If you have an error "`org.codehaus.groovy.grails.commons.ConfigurationHolder: class not found`" 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
  ```

- 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
  ```