Creating a development environment

This text is intended as a tutorial for CzechIdM developers with the goal to set up IDE, build CzechIdM in it and run the system. If you prefer quick glance at the code or need to start quickly, you can use Quick start with devstack.

1. Install Java and Maven

First install JDK and Maven:

Download and install Java 11 OpenJDK (Java 21 OpenJDK for CzechIdM 13.1.0+).

Install Maven from your system packages, at least version 3.1 is required.

```
yum install maven
mvn -v
```

Note: If you installed Java separately from your system libraries, you should set the correct Java home for Maven in the file ".mavenrc", which you will create in your home directory. You put there the full path to your JDK:

```
export JAVA_HOME=/path/to/installed/jdk/
```

This way Maven will always use this JDK.

2. Download the source code

Download CzechIdM source code from GitHub:

```
git clone https://github.com/bcvsolutions/CzechIdMng.git
```

Check branch:

```
git branch
```

The development is always done on the branch "develop", so don't forget to switch to this branch:

```
git checkout -b develop origin/develop
```

To develop your own feature or fix a bug create new branch from develop

```
git checkout -b personal/YourName/feature_or_bugfix-name_of_your_branch
```

Create pull request to merge your branch to develop branch.

3. Install PostgreSQL

CzechIdM in "dev" profile uses PostgreSQL as the repository.

You have to folow one of these next steps. Choose one which fits you more.

1. PostgreSQL to computer 3.1.
2. Run docker image with postgres part 3.2.

3.1 Install PostgreSQL to computer

Install necessary packages:

```
yum install postgresql postgresql-server postgresql-init
```

Fedora 25 packages:

```
dnf install postgresql-server postgresql-contrib
```

Ubuntu 22.04 LTS packages:

```
apt install postgresql postgresql-contrib
```

First run of the service:

```
systemctl start postgresql.service
```

If the following error occurs: "Job for postgresql.service failed. See 'systemctl status postgresql.service' and 'journalctl -xn' for details." Try initializing the PostgreSQL first and start the database again:

```
postgresql-setup --initdb --unit postgresql
```

Set automatic start of the service when starting OS:

```
systemctl enable postgresql.service
```

Set the configuration file for the database to accept login and password (change METHOD from ident to md5)

```
vim /var/lib/pgsql/data/pg_hba.conf
```

```
in Ubuntu it is:
vim /etc/postgresql/14/main/pg_hba.conf

in Windows it is:
<Postgress Install Dir>/Data/pg_hba.conf

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local        all                     all
peer
# IPv4 local connections:
host         all                   all                127.0.0.1/32
md5
# IPv6 local connections:
host         all                   all                ::1/128
md5
```

Restart the service to load the changes:

```
systemctl restart postgresql.service
```

Create the database with name:

- Version 13 is for IdM 13 and develop (IdM 14) - bcv_idm_13
- Version 10 is for IdM 12 - 10 - bcv_idm_10
- Below version 10 - bcv_idm_storage
- (Make sure you have installed and allowed en_US locale in your OS)

```
su postgres -c psql postgres
CREATE USER idmadmin PASSWORD 'idmadmin';
CREATE DATABASE "bcv_idm_13" WITH OWNER idmadmin ENCODING 'UTF8' LC_COLLATE
= 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' template template0;
CREATE DATABASE "bcv_idm_10" WITH OWNER idmadmin ENCODING 'UTF8' LC_COLLATE
= 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' template template0;
```

3.2 Run postgreSQL in docker container

Installation instructions for **Docker only** or **Docker desktop** are here: docker engine installation.

After installation of docker. Use next command which will create postgres container with user and password which is used for development.

```
docker run --name postgresql -e POSTGRES_USER=idmadmin -e
POSTGRES_PASSWORD=idmadmin -p 5432:5432 postgres:14.5
```

Connect to docker container and create the databases with names:

- Version 13 is for IdM 13 and develop (IdM 14) - bcv_idm_13

- Version 10 is for IdM 12 - 10 - bcv_idm_10
- Below version 10 - bcv_idm_storage
- (Make sure you have installed and allowed en_US locale in your OS)
- \l command will return list of databases make sure you have properly set owner of databases!

```
docker exec -it postgresql /bin/bash
psql -U idmadmin

CREATE DATABASE "bcv_idm_13" WITH OWNER idmadmin ENCODING 'UTF8' LC_COLLATE
= 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' template template0;
CREATE DATABASE "bcv_idm_10" WITH OWNER idmadmin ENCODING 'UTF8' LC_COLLATE
= 'en_US.UTF-8' LC_CTYPE = 'en_US.UTF-8' template template0;

\l
```

After running command "\l" you should see this:

```
                                    List of databases
         Name          |  Owner   | Encoding |  Collate   |   Ctype    |
Acces
s privileges
-----------------------+----------+----------+------------+------------+----
----
---------------
 bcv_idm_10            | idmadmin | UTF8     | en_US.utf8 | en_US.utf8 |
 bcv_idm_13            | idmadmin | UTF8     | en_US.utf8 | en_US.utf8 |
```

Database will be available on address:

```
 postgresql://localhost:5432/postgres
```

4. Backend development environment

The tutorial is different for each IDE. Select one and continue with a certain tutorial below:

- Eclipse
- IDEA

In the IDE start the Tomcat server with CzechIdM deployed.

In browser access http://localhost:8080/idm-backend/api/v1/status to check whether the backend is running. Following text should be displayed:

> CzechIdM API is running > > If you can see this message, API is running

5. Frontend development environment

BEWARE!!! Please use these instructions

Select IDE for frontend environment

You can use any text editor or JavaScript IDE. We recommend to use ATOM

- See ATOM tutorial

6. Run backend and frontend together

Assume running backend from the previous steps.

In the project root folder:

Start frontend

```
cd CzechIdM/Realization/frontend/czechidm-app
gulp
```

A new browser window or tab will open with https://localhost:3000 automatically (this could take some time).

The frontend is fully started after you see in following line in log:

```
[BS] 1 file changed (app.js)
```

After the application is running, log in:

- username: admin
- password: admin

If the app shows error with permission, log out

To use CzechIdM in more windows or tabs for development purposes go to http://localhost:3001/ to **BrowserSync** administration and then go to **SyncOptions** and disable everything except **CodeSync**

Congratulations! Your CzechIdM development environment is ready to use.

Common development errors & tips

Tomcat server fails to start

If Tomcat server fails to start, try following:

- Increase the timeout for Tomcat server Start (see Install Tomcat)
- Check that PostgreSQL server is running.

```
systemctl status postgresql.service
```

- Check that PostgreSQL database exists and the user can connect: Use the credentials filled in `idm-app/src/main/resources/application-dev.properties` and try to connect to the PostgreSQL database written in the datasource URL. E.g. for this setup:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/bcv_idm_storage
spring.datasource.username=idmadmin
spring.datasource.password=idmadmin
```

use the following command and type the password when asked for it.

```
psql -h 127.0.0.1 -U idmadmin bcv_idm_storage
```

- Check that you can read from the tables of the database. As in the previous point, connect to the PostgreSQL database and try to select data from any of the IdM tables. If you have insufficient access rights, check the owner and grants for the schema "public". If the user idmadmin is not an owner, there must be GRANT on schema public to the user idmadmin or to "public".
- Check the error message in the server output console.

Common error messages

`Initialization of bean failed; nested exception is org.flywaydb.core.api.FlywayException: Validate failed: Migration checksum mismatch for migration 7.07.004` means that your database was already used for some other version of CzechIdM (you can check used Flyway versions in the tables idm_schema_version_xxx). This can happen if you switch to some older development branch. The easiest solution for the **developer** (you will lose all your prepared data in IdM!) is to drop the database (`DROP DATABASE bcv\_idm\_storage`) and create it again (see the part `CREATE DATABASE` of Install PostgreSQL). Or you can create a different database and set its name to the `application-dev.properties` temporarily.

`Initialization of bean failed; nested exception is org.flywaydb.core.api.FlywayException: Found more than one migration with version 7.07.004` means literally that there is a duplicity in Flyway scripts versions. This has to be

fixed by the authors of the scripts by renaming one of the script that use the same version. The log will tell you where to find the duplicity:

```
Offenders:
-> ... idm-core-impl-7.7.0-
SNAPSHOT.jar!/eu/bcvsolutions/idm/core/sql/postgresql/V7_07_004__automatic-
role.sql (SQL)
-> ... idm-core-impl-7.7.0-
SNAPSHOT.jar!/eu/bcvsolutions/idm/core/sql/postgresql/V7_07_004__add-index-
template-name.sql (SQL)
```

Just increase the version in the name of one of the scripts, clean & build the project again.

Could not create Vfs.Dir from url

This isn't error just warning. You may don't interest about this warning. Warning is only when you start local development environment and you configuration for ic connectors path isn't correctly setup. Dont worry about this warning.

```
2019-07-18 10:01:20.436  WARN 31049 --- [ost-startStop-1]
org.reflections.Reflections              : could not create Vfs.Dir from
url. ignoring the exception and continuing

org.reflections.ReflectionsException: could not create Vfs.Dir from url, no
matching UrlType was found [file:/usr/lib/jvm/java-8-openjdk-
amd64/jre/lib/ext/libatk-wrapper.so]
either use fromURL(final URL url, final List<UrlType> urlTypes) or use the
static setDefaultURLTypes(final List<UrlType> urlTypes) or
addDefaultURLTypes(UrlType urlType) with your specialized UrlType.
```

Frontend build fails

Out of memory

If your gulp build fails with the following error:

```
[INFO] [18:43:00] Starting 'browserify'...
[INFO] [18:45:30] Finished 'browserify' after 2.5 min
[INFO] [18:45:30] Finished 'build' after 3.12 min
[INFO]
[INFO] <--- Last few GCs --->
[INFO]
[INFO] 192087 ms: Scavenge 1282.6 (1434.8) -> 1282.1 (1434.8) MB, 3.8 / 0 ms
(+ 3.8 ms in 9 steps since last GC) [allocation failure].
[INFO] 192095 ms: Scavenge 1282.6 (1434.8) -> 1282.2 (1434.8) MB, 3.6 / 0 ms
(+ 3.3 ms in 8 steps since last GC) [allocation failure].
```

```
[INFO] 192102 ms: Scavenge 1282.6 (1434.8) -> 1282.1 (1434.8) MB, 3.7 / 0 ms
(+ 2.9 ms in 7 steps since last GC) [allocation failure].
[INFO]
[INFO]
[INFO] <--- JS stacktrace --->
[INFO] Cannot get stack trace in GC.
[ERROR] FATAL ERROR: Scavenger: semi-space copy
[ERROR] Allocation failed - process out of memory
[ERROR] 1: node::Abort() [gulp]
```

Try following:

- Go to `Realization/frontend/czechidm-app/` and delete `node\_modules`. Then start the build again (`npm install` etc.)
- If you build BE+FE of a project together, check the `pom.xml` of the app module - the configuration properties `nodeVersion` and `npmVersion`. They should be the same as the product currently supports (check the app module of the product):

```
                        <execution>
                                <id>install node and npm</id>
                                <goals>
                                    <goal>install-node-and-npm</goal>
                                </goals>
                                <phase>generate-resources</phase>
                                <configuration>
                                    <nodeVersion>v10.15.3</nodeVersion>
                                    <npmVersion>6.4.1</npmVersion>
                                </configuration>
                        </execution>
```

- try adding `\–max\_old\_space\_size=4096` to the `gulp` command. Or if you build BE+FE together by app module, try adding it in the pom.xml of the app:

```
                                                                <execution>
                                                                        <id>gulp
build</id>
                                                                        <goals>
<goal>gulp</goal>
                                                                        </goals>
<phase>generate-resources</phase>
<configuration>
<arguments>build -p ${frontend.profile} -s ${frontend.stage} --
max_old_space_size=4096</arguments>
<workingDirectory>../../frontend/czechidm-app</workingDirectory>
<installDirectory>target</installDirectory>
</configuration>
                                                                </execution>
```

Wrong layout and NPM version

If your frontend looks like this: https://redmine.czechidm.com/issues/1752, check following:

- there is package-lock.json file in the frontend app module and it looks like this: https://github.com/bcvsolutions/CzechIdMng/blob/develop/Realization/frontend/czechidm-app/package-lock.json
- NPM version which you use when building frontend (or FE+BE) is at least 6.

```
npm -v
6.4.1
```

Developing on Appliance

This section contains howtos and hacks for more effective development on IAM Appliance.

Working-around IdM rebuild when developing backend module

When you are developing a backend module and need to debug it directly on the Appliance, each `iam-czechidm` service restart forces the application to relink... which can take significant amount of time. To speed things up you can, technically, edit the content of existing container without the need for relink.

This is really nonstandard hack and exercising special care is absolutely necessary.

Remarks

- If you are debugging a frontend module, this howto **will not work** because the frontend rebuild will be skipped.
- If you forget to properly persist your changes (see steps below) you are creating rather nasty surprise for anybody who re-creates the `czechidm` docker container. And as the original container gets deleted in the process, you will have no way to find out what happenned. Always clean up after yourself.

Steps

1. (you start developing, add all your modules into proper folders in the appliance, restart the `iam-czechidm` service, let the build process finish)
2. At this point, you have a basis of the application you are going to develop further. You can even have some custom frontend parts put in.
3. Suppose you want to put another JAR into you application or replace one that is already here. (here we work with `ssh-connector-1.0.1.jar` as an example)

1. Copy new / modified files directly into the container.

```
[root@appliance70 ~]# docker cp ssh-connector-1.0.1.jar
czechidm:/opt/tomcat/current/webapps/idm/WEB-INF/lib/
Successfully copied 28.2kB to
czechidm:/opt/tomcat/current/webapps/idm/WEB-INF/lib/
```

2. In case you need to delete something, remove it directly from the container. **This is the way you now have to handle conflicting libraries, by the way.**

```
[root@appliance70 ~]# docker exec czechidm ls -l
/opt/tomcat/current/webapps/idm/WEB-INF/lib | grep ssh
-rw-r--r--. 1 root root    26277 Apr 10 13:05 ssh-
connector-1.0.1.jar

[root@appliance70 ~]# docker exec czechidm rm -fv
/opt/tomcat/current/webapps/idm/WEB-INF/lib/ssh-
connector-1.0.1.jar
removed '/opt/tomcat/current/webapps/idm/WEB-INF/lib/ssh-
connector-1.0.1.jar'

[root@appliance70 ~]# docker exec czechidm ls -l
/opt/tomcat/current/webapps/idm/WEB-INF/lib | grep ssh
(empty listing)
```

3. Restart the IdM container.

```
[root@appliance70 ~]# systemctl restart iam-czechidm
```

4. Let the IdM start as usual. Because you did not change anything in the appliance directories (under `/data/volumes/czechidm/…`) the relink mechanism will not kick in.

4. Repeat the previous step as long as you need to develop / debug your backend module.
5. Persist the changes.
    1. Once you are done working, you need to persist all changes you made. Gather all modules you were working on and put them into directories under `/data/volumes/czechidm/…` where they should be located.
    2. Stop, drop and start the IdM container.

```
[root@appliance70 ~]# systemctl stop iam-czechidm
[root@appliance70 ~]# docker rm czechidm
[root@appliance70 ~]# systemctl start iam-czechidm
```

    3. Let the rebuild finish and wait for IdM to start.
    4. Test that changes you made are really there.