Installation of CzechIdM - Linux

installation, jdbc, quickstart, encryption

We presume that the server is prepared as described in Server preparation - Linux - CentOS8.

This tutorial shows how to install full production-ready version of CzechIdM on standard software setup (java, postgreSQL, Tomcat, Apache httpd). If you are looking for a demo installation please see Getting Started.

If you install CzechIdM on Sql server please skip instruction with setup DB and install JDBC driver and follow the tutorial.

1. Create DB user and database in PostgreSQL

Switch the user from root to postgres and use **psql** to add the user and database into PostgreSQL:

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

Try the access to new database with new user:

```
[root@tomcat1 data]# psql -h localhost -U czechidm
Password for czechidm user:
psql (9.6.1)
For more information type "help".

czechidm=>
```

If yout have the database installed on a different server than the application itself (Tomcat etc.), don't forget to configure PostgreSQL to allow remote SSL connection from that server and add this line to /data/pgsql/9.6/pg_hba.conf replacing {IP address} and {Mask} with the IP and mask of the CzechIdM application server:

```
hostssl czechidm    czechidm    {IP address}/{Mask}    md5
```

and restart PostgreSQL.

2. JDBC driver installation

CentOS

Install the package with PostgreSQL JDBC driver:

```
yum install -y postgresql-jdbc
```

allow Tomcat to use the driver:

```
ln -s /usr/share/java/postgresql-jdbc.jar /usr/share/java/tomcat/
```

Debian

Install the package with PostgreSQL JDBC driver:

```
apt-get install libpostgresql-jdbc-java
```

allow Tomcat to use the driver:

```
ln -s /usr/share/java/postgresql.jar /var/lib/tomcat8/lib/postgresql-jdbc4.jar
```

3. Configure environment properties. Select application profile

Edit tomcat unit - edit the line with environment variable choosing the appropriate application profile. We use **production** profile in our example, which enables you to configure production-ready instace of the identity manager.

The **dev** profile is for development and testing environments and as such it has debug logging enabled. For production deployment, use a profile named **production** as is shown in the example. The profile naming convention is mandatory because other CzechIdM configuration depends on it.

To prevent application startup fails due to Flyway bug, property `-Djava.util.Arrays.useLegacyMergeSort=true` has to be added into environment properties. If property is not set, then application can fail on error:

```
Error creating bean with name 'flywayCore' defined in class path resource [eu/bcvsolutions/idm/core/config/flyway/CoreFlywayConfig.class]: Initialization of bean failed; nested exception is java.lang.IllegalArgumentException: Comparison method violates its general contract!
```

Use `systemctl edit tomcat.service` and change the following line (On Debian make changes file `/etc/default/tomcat8`):

```
Environment='JAVA_OPTS=-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom -
Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true -
Djavax.servlet.request.encoding=UTF-8'
```

into:

```
Environment='JAVA_OPTS=-Djava.awt.headless=true -
Djava.security.egd=file:/dev/./urandom -Dspring.profiles.active=production -
Djava.util.Arrays.useLegacyMergeSort=true -
Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=true -
Djavax.servlet.request.encoding=UTF-8'
```

On CentOS reload systemd after the changes:

```
systemctl daemon-reload
```

4. Create CzechIdM configuration folders

In CzechIdM, you can store all deployment-specific configuration (i.e. database credentials) outside the war file. This is a configure-once approach which greatly simplifies future deployments.

- The **etc** directory stores configuration files.
- The **lib** directory stores additional jar libraries such as database drivers.
- The **backup** directory stored Groovy scripts backups.
- The **data** directory stores various user-attached files.
- The **app** directory stores war files.

Create the directory structure:

```
mkdir -p /opt/czechidm/{etc,lib,backup,data,app}
```

5. Create CzechIdM configuration

Now we will create configuration files the CzechIdM will use.

Code snippets in this chapter can be **mostly** copy-pasted or (but please read through whole chapter to be aware of setting you have to adjust). Configuring the CzechIdM is about altering four or five lines altogether.

- The **/opt/czechidm/etc/secret.key** is a file with confidential storage secret key. This key has to have 128 bit (= 16 bytes).

  ```
  cat /dev/urandom | tr -dc 'a-z0-9' | head -c 16 >
  /opt/czechidm/etc/secret.key
  ```

- The **/opt/czechidm/etc/quartz-production.properties** file stores, the Quartz scheduler

configuration. You can use the following snippet as a production-safe configuration file.

quartz-production.properties

```
org.quartz.scheduler.instanceName=idm-scheduler-instance
org.quartz.scheduler.instanceId=AUTO
org.quartz.scheduler.skipUpdateCheck=true
org.quartz.threadPool.class=org.quartz.simpl.SimpleThreadPool
org.quartz.threadPool.threadCount=10
org.quartz.threadPool.threadPriority=4
org.quartz.jobStore.class=org.quartz.impl.jdbcjobstore.JobStoreTX
org.quartz.jobStore.driverDelegateClass=org.quartz.impl.jdbcjobsto
re.PostgreSQLDelegate
org.quartz.jobStore.useProperties=false
org.quartz.jobStore.misfireThreshold=60000
org.quartz.jobStore.tablePrefix=qrtz_
```

- The **/opt/czechidm/etc/logback-spring.xml** specifies logging configuration. This is the default logging configuration that you can use out of the box.

logback-spring.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- https://springframework.guru/using-logback-spring-boot/ -->
<!-- http://logback.qos.ch/manual/appenders.html -->
<configuration>
    <include
resource="org/springframework/boot/logging/logback/base.xml"/>
    <springProperty name="spring.profiles.active"
source="spring.profiles.active"/>
    <springProfile name="default">
        <logger name="eu.bcvsolutions" level="INFO"/>
        <logger name="org.springframework" level="INFO"/>
        <logger name="org.springframework.web" level="INFO"/>
        <logger name="org.hibernate.SQL" level="INFO"/>
        <logger
name="org.hibernate.type.descriptor.sql.BasicBinder"
level="INFO"/>
    </springProfile>

    <springProfile name="test">
        <logger name="eu.bcvsolutions" level="DEBUG"/>
        <logger name="org.springframework" level="INFO"/>
        <logger name="org.springframework.web" level="INFO"/>
        <logger name="org.hibernate.SQL" level="INFO"/>
        <logger
name="org.hibernate.type.descriptor.sql.BasicBinder"
level="INFO"/>
    </springProfile>
```

```
    <springProfile name="dev">
        <springProperty name="spring.datasource.driver-class-name"
source="spring.datasource.driver-class-name"/>
        <springProperty name="spring.datasource.url"
source="spring.datasource.url"/>
        <springProperty name="spring.datasource.username"
source="spring.datasource.username"/>
        <springProperty name="spring.datasource.password"
source="spring.datasource.password"/>

        <appender name="DB"
class="ch.qos.logback.classic.db.DBAppender">
            <connectionSource
class="ch.qos.logback.core.db.DriverManagerConnectionSource">
                <driverClass>${spring.datasource.driver-class-
name}</driverClass>
                <url>${spring.datasource.url}</url>
                <user>${spring.datasource.username}</user>
                <password>${spring.datasource.password}</password>
            </connectionSource>
        </appender>

        <appender name="DB_ASYNC"
class="ch.qos.logback.classic.AsyncAppender">
            <appender-ref ref="DB" />
            <includeCallerData>true</includeCallerData>
        </appender>

        <logger name="eu.bcvsolutions" level="TRACE">
            <appender-ref ref="DB_ASYNC" />
        </logger>
        <logger name="org.springframework" level="INFO"/>
        <logger name="org.springframework.web" level="DEBUG"/>
        <logger name="org.hibernate.SQL" level="DEBUG"/>
        <logger
name="org.hibernate.type.descriptor.sql.BasicBinder"
level="TRACE"/>
    </springProfile>

    <springProfile name="dev-mysql">
        <logger name="eu.bcvsolutions" level="TRACE" />
        <logger name="org.springframework" level="INFO"/>
        <logger name="org.springframework.web" level="DEBUG"/>
        <logger name="org.hibernate.SQL" level="DEBUG"/>
        <logger
name="org.hibernate.type.descriptor.sql.BasicBinder"
level="TRACE"/>
    </springProfile>
</configuration>
```

- The most important file is **/opt/czechidm/etc/application-production.properties** (application-PROFILE.properties, where the PROFILE is the profile you run the IdM under). You can use most of the file as-is, there is a bit of configuration needed though. This is a template file:

  application-production.properties

```
# Doc: https://wiki.czechidm.com/devel/dev/configuration/backend

idm.pub.app.instanceId=idm-primary
idm.pub.app.stage=production

spring.datasource.url=jdbc:postgresql://localhost:5432/czechidm
spring.datasource.username=czechidm
spring.datasource.password=********** TODO *********
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.validationQuery=SELECT 1
spring.datasource.test-on-borrow=true
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=none
flyway.enabled=true

scheduler.enabled=true
scheduler.task.queue.process=1000
scheduler.event.queue.process=1000
scheduler.properties.location=quartz-production.properties
logging.config=/opt/czechidm/etc/logback-spring.xml
idm.sec.core.demo.data.enabled=false

#spring.cache.ehcache.config=classpath:ehcache.xml

spring.activiti.processDefinitionLocationPrefix=classpath*:/eu/bcv
solutions/idm/workflow/
idm.sec.core.notification.template.folder=classpath*:/eu/bcvsoluti
ons/idm/templates/
idm.sec.core.script.folder=classpath*:/eu/bcvsolutions/idm/scripts
/
# configuration property for default backup
idm.sec.core.backups.default.folder.path=/opt/czechidm/backup


idm.pub.security.allowed-origins=http://localhost
# Generate JWT token security string as "cat /dev/urandom | tr -dc
'a-z0-9' | head -c VALUE" where VALUE can be from 1 to 255.
# We recommend the VALUE to be at least 25.
idm.sec.security.jwt.secret.token=********** TODO *********
idm.sec.security.jwt.expirationTimeout=36000000

# recaptcha
# - recaptchaservice endpoint
```

```
#idm.sec.security.recaptcha.url=https://www.google.com/recaptcha/api/siteverify
# - secret key, can be generated here https://www.google.com/recaptcha/admin
idm.sec.security.recaptcha.secretKey=xxx
# Proxy for HTTP requests
#idm.sec.core.http.proxy=12.34.56.78:1234

# Cipher secret key for crypt values in confidential storage
# for crypt values is used secretKey or secretKey defined by file - secretKeyPath
#cipher.crypt.secret.key=XXXXXXXXXXXXXXXX
cipher.crypt.secret.keyPath=/opt/czechidm/etc/secret.key


idm.sec.core.emailer.test.enabled=true
# http://camel.apache.org/mail.html
idm.sec.core.emailer.protocol=smtp
idm.sec.core.emailer.host=something.tld
idm.sec.core.emailer.port=25
# idm.sec.core.emailer.username=czechidm@domain.tld
# idm.sec.core.emailer.password=password
idm.sec.core.emailer.from=czechidm@localhost

## Global property that allow disable or enable sending notification from WF
idm.sec.core.wf.notification.send=false


# supports delete identity
idm.pub.core.identity.delete=true
#
# default password change type for custom users, one of values:
# DISABLED - password change is disable
# ALL_ONLY - users can change passwords only for all accounts
# CUSTOM - users can choose for which accounts change password
idm.pub.core.identity.passwordChange=ALL_ONLY
#
# required old password for change password
idm.pub.core.identity.passwordChange.requireOldPassword=true
#
# create default identity's contract, when identity is created
idm.pub.core.identity.create.defaultContract.enabled=true


# Default user role will be added automatically, after an identity is logged in
# could contains default authorities and authority policies configuration
# for adding autocomplete or all record read permission etc.
idm.sec.core.role.default=userRole
```

```
# Admin user role
idm.sec.core.role.admin=superAdminRole


# ID system against which to authenticate
idm.sec.security.auth.systemId=

# attachments will be stored under this path.
# new directories for attachment will be created in this folder
(permissions has to be added)
# System.getProperty("user.home")/idm_data will be used if no path
is given
idm.sec.core.attachment.storagePath=/opt/czechidm/data
```

Adjust database configuration

If you followed this howto, the only thing you should need to adjust is a **spring.datasource.password** propetry. Set it to the password for czechidm user in PostgreSQL. If necessary, adjust other database connection properties...

```
spring.datasource.url=jdbc:postgresql://localhost:5432/czechidm
spring.datasource.username=czechidm
spring.datasource.password=********** TODO *********
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.validationQuery=SELECT 1
spring.datasource.test-on-borrow=true
```

Generate JWT token

Set value of the **idm.sec.security.jwt.secret.token** property as is described in the template file:

```
# Generate JWT token security string as "cat /dev/urandom | tr -dc 'a-z0-9'
| head -c VALUE" where VALUE can be from 1 to 255.
# We recommend the VALUE to be at least 25.
idm.sec.security.jwt.secret.token=********** TODO *********
```

Local confidential storage

Local confidential storage is encrypted by AES algoritm. Read more. Confidential storage is encrypted by a key found in **secret.key** file you already created.

There are two properties in application-production.properties that influence the confidential storage:

- You can set the 128bit (16byte) key directly in the property file using **cipher.crypt.secret.key** property or

- you can create separate file (in our case **secret.key**) containing a random string. Then you reference this file with **cipher.crypt.secret.keyPath** property.

CzechIdM doesn't contain any default key for crypt confidential storage. Please define it before you start using the IdM.

Confidential storage uses AES/CBC/PKCS5Padding (more info) algorithm which operates with 128bit key.

Attachment store

In CzechIdM, users can sometimes add attachments (say, attach *.jpeg photo to their employee card request). Those files are stored in the attachment store. With the following property, you can configure, where the store is. If you used sample property file, the store is by-default located under /opt/czechidm/data .

```
# attachments will be stored under this path.
# new directories for attachment will be created in this folder (permissions has to be added)
# System.getProperty("user.home")/idm_data will be used if no path is given
idm.sec.core.attachment.storagePath=/opt/czechidm/data
```

Environment

If you install CzechIdM in multiple environments (typically test and production), you can display a label in the navigation bar which will tell the users in which environment they work. The default value "production" doesn't display any label. Set the value to `test` to display the label marking the Test environment.

```
# Application stage (development, test, production (default))
idm.pub.app.stage=production
```

6. Set correct permissions on CzechIdM files

CentOS

```
chown tomcat:tomcat /opt/czechidm
chown -R tomcat:tomcat /opt/czechidm/{etc,data,backup,app,lib}
chmod 750 /opt/czechidm/{etc,data,backup,app,lib}
chmod 640 /opt/czechidm/etc/*
```

Debian

```
chown tomcat8:tomcat8 /opt/czechidm
chown -R tomcat8:tomcat8 /opt/czechidm/{etc,data,backup,app,lib}
```

```
chmod 750 /opt/czechidm/{etc,data,backup,app,lib}
chmod 640 /opt/czechidm/etc/*
```

7. Adjust Tomcat's classpath

Apache Tomcat has to know where the new configuration is. Because CzechIdM uses SpringBoot project, we simply add the **/opt/czechidm/etc** directory (and others) on the classpath.

Add this line with this comand `systemctl edit tomcat.service`.

```
Environment='CLASSPATH=/opt/czechidm/etc:/opt/czechidm/lib/*'
```

On **Debian** create new file `/usr/share/tomcat8/bin/setenv.sh` with this content:

```
CLASSPATH=/opt/czechidm/etc:/opt/czechidm/lib/*
```

And change owner of the file to tomcat:

```
chown root:tomcat /usr/share/tomcat8/bin/setenv.sh
```

8. Create dedicated Java truststore

Java truststore is a file which contains SSL certificates which we consider trusted. Usually this means some certificates of end systems or their respective certificate authorities. When we need CzechIdM to communicate with some new system with SSL-encrypted way, we need to import particular certificate here and restart the Tomcat container.

At this point, we do not have any certificate to put into the truststore so we create a fake one with only one-day validity.

```
cd /opt/czechidm/etc
openssl genrsa -out fakecert.key
openssl req -new -key fakecert.key -out fakecert.csr -subj "/C=CZ/ST=Czech
Republic/L=Prague/O=BCV/CN=CzechIdM placeholder cert"
openssl x509 -req -in fakecert.csr -signkey fakecert.key -days 1 -sha256 -
out fakecert.crt
keytool -importcert -file fakecert.crt -alias placeholder-cert -keystore
truststore.jks
    Enter keystore password:  ENTER SOME PASSWORD HERE AND REMEMBER IT FOR
LATER
    Re-enter new password:
    ...
    Trust this certificate? [no]:  yes
    Certificate was added to keystore

rm fakecert.key fakecert.csr fakecert.crt
chmod 644 truststore.jks
```

```
chown root:root truststore.jks
```

Edit the Tomcat service file (systemctl edit tomcat.service) and add path to the truststore `-Djavax.net.ssl.trustStore=/opt/czechidm/etc/truststore.jks` and truststore password `-Djavax.net.ssl.trustStorePassword=THE PASSWORD YOU ENTERED WHEN CREATING KEYSTORE` to the Environment='JAVA_OPTS' options. Finally, reload the systemd and restart Tomcat.

```
systemctl daemon-reload
systemctl restart tomcat.service
```

9. Deploy the CzechIdM

Download the latest CzechIdM version. Currently it is idm-app-9.4.0.war.

CentOS

Ensure Tomcat is stopped:

```
systemctl stop tomcat.service
```

Copy the identity manager WAR into webapps folder in Tomcat and name it **idm.war**:

```
cp idm-app-9.4.0.war /opt/czechidm/app/idm.war
chown tomcat:tomcat /opt/czechidm/app/idm.war
```

Start the Tomcat container:

```
systemctl start tomcat.service
```

If everything is set up right, the CzechIdM will deploy. Default log is **/var/log/tomcat/catalina.out**.

Debian

Ensure Tomcat is stopped:

```
systemctl stop tomcat8.service
```

Copy the identity manager WAR into webapps folder in Tomcat and name it **idm.war**:

```
cp idm-app-9.4.0.war /opt/czechidm/app/idm.war
chown tomcat8:tomcat8 /opt/czechidm/app/idm.war
```

Start the Tomcat container:

```
systemctl start tomcat8.service
```

If everything is set up right, the CzechIdM will deploy. Default log is **/var/log/tomcat8/catalina.out**.

10. Final Steps

Allow network services

Firewall may restrict the access to all port except ssh (22/tcp). To be able to use CzechIdM, allow port 443/tcp and reload firewalld:

```
firewall-cmd --permanent --add-port=443/tcp
firewall-cmd --reload
```

Change default admin password

In the fresh CzechIdM installation, there is one user identity - **admin** with password **admin**. Right after you install the application, go to https://yourserver.tld/idm and change the default password.

Configure IdM

Follow some final configuration steps: Installation of CzechIdM - Final steps.

On CentOS set permisive mod on Tomcat

SELinux will deny acces to the database for tomcat and won't allow create files by him. The tomcat will write error to the `/var/log/tomcat/catalina.outor` `/var/log/messages` line similar to `org.postgresql.util.PSQLException: Connection to localhost:5432 refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.`.

To fix this we need set the permissive mode for tomcat:

```
semanage permissive -a tomcat_t
```

Evaluate impact of SELinux adjustments **before** you implement them. Proper mitigation heavily depends on habits and security policies of your organization.

There are some possibilities:

- Set permissive mode for logrotate as above.
- Set permissive mode for whole SELinux. (This will drop the SELinux's protective function.)
- Adjust particular SELinux labels. Example (here).