Cloudchip IoT Platform Build Process

We are constantly looking for feedback from Team on how to improve IoT Platform.

Required tools

To build and run IoT Platform instance make sure that you have **Java** and **Maven** installed onto your system.

Please refer to **Building from sources** section where **Java** and **Maven** install processes are described.

Fork and build IoT Platform repository

Once you have completed installation of required tools please fork official **IoT Platform repository**.

Now you can clone source code of the forked project.

NOTE: We will refer later to the folder where you have cloned repository as **${TB_WORK_DIR}**.

If are building on Windows for the first time, you may need to run these commands to ensure the required npm dependencies are available:

```
npm install -g cross-env
npm install -g webpack
```

Before importing the project into the *IDE* please build it using **Maven** tool from the root folder:

```
cd ${TB_WORK_DIR}
mvn clean install -DskipTests
```

A build will generate all the *protobuf* files in the *application* module that are needed for the correct compilation in your *IDE*.

Next, import the project into your favorite *IDE* as **Maven** project. See separate instructions for **IDEA** and **Eclipse**.

NOTE: If you are using Eclipse, after the maven project is imported to the IDE, We recommend you to disable Maven Project builder on **ui** project. This will improve the Eclipse performance *a lot*, because it will avoid Eclipse Maven builder from digging in node_modules directory (which is unnecessary and only causes Eclipse to hang). To do this, right-click on **ui** project, go to **Properties -> Builders**, and then uncheck the **Maven Project Builder** checkbox and then click **Ok**.

Database

By default, IoT Platform uses embedded HSQLDB instance which is very convenient for evaluation or development purposes.

Alternatively, you can configure your platform to use either scalable Cassandra DB cluster or various SQL databases. If you prefer to use an SQL database, we recommend PostgreSQL.

[Optional] SQL Database: PostgreSQL

NOTE: This is an **optional** step. It is required only for production usage. You can use embedded HSQLDB for platform evaluation or development

Please use this link for the PostgreSQL installation instructions.

Once PostgreSQL is installed you may want to create a new user or set the password for the main user.

When it’s done, connect to the database and create IoT Platform DB:

```
psql -U postgres -d postgres -h 127.0.0.1 -W

CREATE DATABASE IoT Platform;
\q
```

[Optional] NoSQL Database: Cassandra

Please refer to appropriate section where you find instructions on how to install cassandra:

- Cassandra installation on **Linux**
- Cassandra installation on **Windows**

[Optional] Configure IoT Platform to use external database

NOTE: This is an **optional** step. It is required only for production usage. You can use embedded HSQLDB for platform evaluation or development

Edit IoT Platform configuration file:
/application/src/main/resources/IoT Platform.yml

Comment '# HSQLDB DAO Configuration' block.

```
# HSQLDB DAO Configuration
#spring:
#  data:
#    jpa:
#      repositories:
#        enabled: "true"
#  jpa:
#    hibernate:
#      ddl-auto: "validate"
#    database-platform: "org.hibernate.dialect.HSQLDialect"
#  datasource:
#    driverClassName: "${SPRING_DRIVER_CLASS_NAME:org.hsqldb.jdbc.JDBCDriver}"
#    url: "${SPRING_DATASOURCE_URL:jdbc:hsqldb:file:${SQL_DATA_FOLDER:/tmp}/thingsboardDb;sql.enforce_size=false}"
#    username: "${SPRING_DATASOURCE_USERNAME:sa}"
#    password: "${SPRING_DATASOURCE_PASSWORD:}"
```

For **PostgreSQL**:

Uncomment ‘# PostgreSQL DAO Configuration’ block. Be sure to update the postgres databases username and password in the bottom two lines of the block (here, as shown, they are both “postgres”).

```
# PostgreSQL DAO Configuration
spring:
  data:
    jpa:
      repositories:
        enabled: "true"
  jpa:
    hibernate:
      ddl-auto: "validate"
    database-platform: "org.hibernate.dialect.PostgreSQLDialect"
  datasource:
    driverClassName: "${SPRING_DRIVER_CLASS_NAME:org.postgresql.Driver}"
    url: "${SPRING_DATASOURCE_URL:jdbc:postgresql://localhost:5432/thingsboard}"
    username: "${SPRING_DATASOURCE_USERNAME:postgres}"
    password: "${SPRING_DATASOURCE_PASSWORD:postgres}"
```

For **Cassandra DB**:

Locate and set database type configuration parameters to ‘cassandra’.

```
database:
  entities:
    type: "${DATABASE_ENTITIES_TYPE:cassandra}" # cassandra OR sql
  ts:
    type: "${DATABASE_TS_TYPE:cassandra}" # cassandra OR sql (for hybrid mode, only this value should be cassandra)
```

NOTE: If your Cassandra server is installed on the remote machine or it is bind to custom interface/port, you need to specify it in thingsboard.yml as well. Please, tefer to the configuration guide for the detailed description of **thingsboard.yml** file and what properties are used for cassandra connection configuration.

After the IoT Platform.yml file was updated, please rebuild the application module so that the updated IoT Platform.yml gets populated to the target directory:

```
cd ${TB_WORK_DIR}/application
mvn clean install -DskipTests
```

Create Database schema and populate demo data

In order to create the database tables, run the following:

On *Linux*:

```
cd ${TB_WORK_DIR}/application/target/bin/install
chmod +x install_dev_db.sh
./install_dev_db.sh
```

On *Windows*:

```
cd %TB_WORK_DIR%\application\target\windows
install_dev_db.bat
```

Running development environment

Running UI container in hot redeploy mode.

By default, IoT Platform UI is served at 8080 port. However, you may want to run UI in the hot redeploy mode.

NOTE: This step is optional. It is required only if you are going to do changes to UI.

To start UI container in hot redeploy mode you will need to install **node.js** first. Once **node.js** is installed you can start container by executing next command:

```
cd ${TB_WORK_DIR}/ui
mvn clean install -P npm-start
```

This will launch a special server that will listen on 3000 port. All REST API and websocket requests will be forwarded to 8080 port.

Running server-side container

To start server-side container you can use couple options.

As a first option, you can run the main method of **org.thingsboard.server.ThingsboardServerApplication**class that is located in *application* module from your *IDE*.

As a second option, you can start the server from command line as a regular **Spring boot** application:

```
cd ${TB_WORK_DIR}
java -jar application/target/thingsboard-${VERSION}-boot.jar
```

Dry run

Navigate to http://localhost:3000/ or http://localhost:8080/ and login into ThingsBoard using demo data credentials:

- *login* **tenant@thingsboard.org**

- *password* **tenant**

Make sure that you are able to login and everything has started correctly.

Code changes

Now you are ready to start to do some changes to the codebase. Update server-side or UI code. Verify that changes that you have done meet your requirements and expectations from the user perspective.

Verify build

Before you commit your changes to the remote repository build it locally with tests run using *Maven*:

```
mvn clean install
```

Make sure that build is fine and all the tests are successful.

Push changes to your fork

When you are done with code changes commit and push them to your forked repository with some meaningful comment:

```
git commit -m 'Some meaningful comment'
git push origin master
```

Build from Sources

This guide will help you to download and build ThingsBoard from sources. Instructions listed below are tested on Ubuntu 16.04 and CentOS 7.1

Required tools

This section contains installation instructions for build tools.

Java

ThingsBoard is build using Java 8. You can use following instructions to install Java 8.

Maven

ThingsBoard build requires Maven 3.1.0+.

```
Ubuntu: sudo apt-get install maven
```

Please note that maven installation may set Java 7 as a default JVM on certain Linux machines. Use java installation instructions to fix this.

Source code

You can clone source code of the project from the official github repo.

```
git clone git@github.com:thingsboard/thingsboard.git
# checkout latest release branch
git checkout release-2.3
```

Build

Run the following command from the thingsboard folder to build the project:

```
mvn clean install
```

Build local docker images

```
mvn clean install -Ddockerfile.skip=false
```

Build artifacts

You can find debian, rpm and windows packages in the target folder:

```
application/target
```