# Designing a platform for network-related education

## Manual Document

Written by Sigurd Traberg Moth

Last updated: 25/11/2020

Version number: 1.00

## Contents

# Manual specifications

Every manual must be written with a specific purpose in mind such as how to set up a database. Additionally, they must include following:

- A detailed numerated guide preferably with screenshots written step by step.
- Mentioning of potential issues that can happen in the current version.
- Links to software that must be downloaded to proceed.
- A version number to document when a manual is changed.

# Manuals

## Setting up Visual Code

VERSION NUMBER: 1.00

To streamline manuals Visual Code has become the editor of choice. Users are by no means locked into using Visual Code, but every manual is made from the assumption that Visual Code was used.

1. Download Visual Code: https://code.visualstudio.com/download
2. Launch Visual Code and download the following extensions:
    a. Java Extension Pack
    b. Spring Boot Tools
    c. Spring Initializer Java Support
    d. Gradle Tasks
    e. Gradle Language Support
    f. GitLab Pipeline Actions
    g. Docker

   Optional:

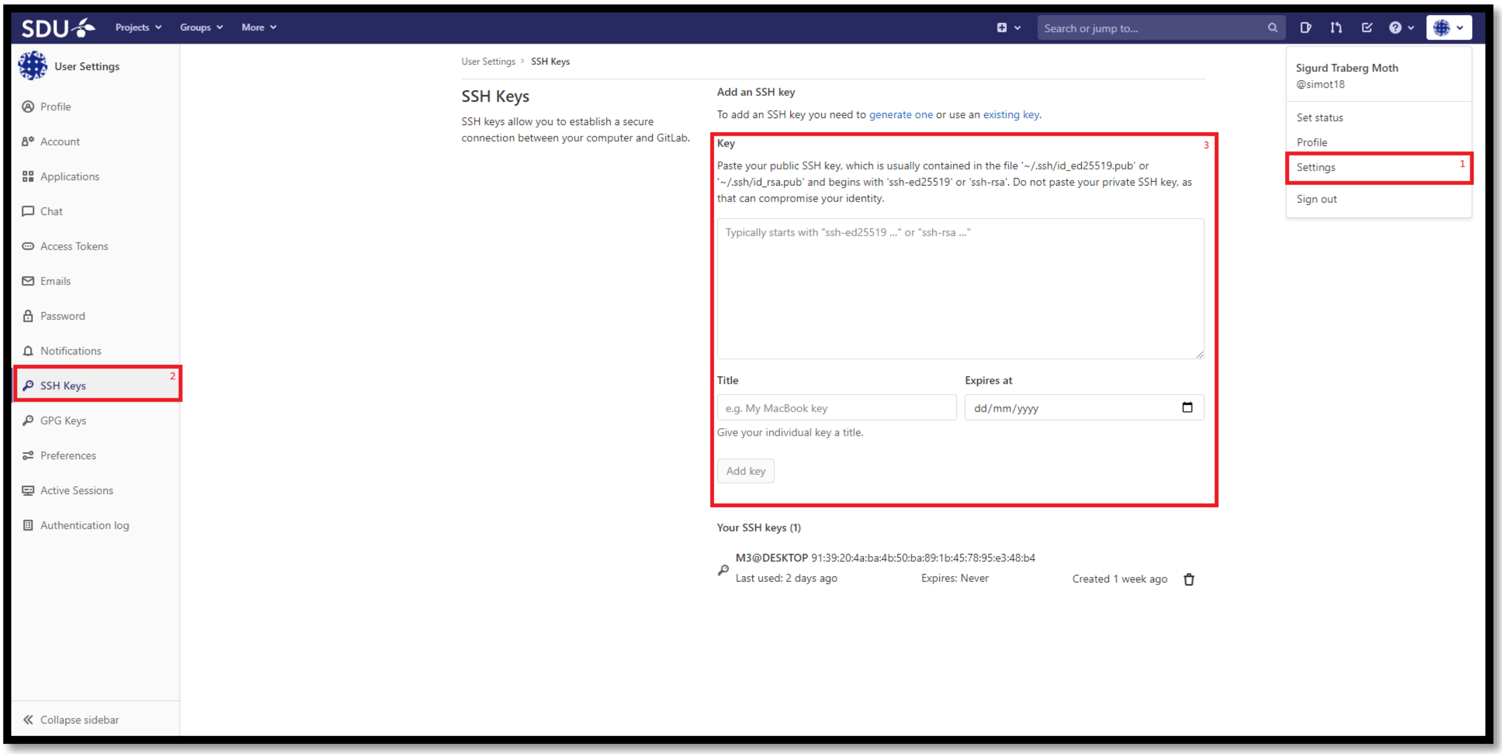
    h. Java Code Generators
    i. Vscode-icons

## Setting up SSH for VIP

VERSION NUMBER: 1.00

1. Generate an SSH keypair or use an existing pair (Not specific for GitLab): https://gitlab.sdu.dk/help/ssh/README#generating-a-new-ssh-key-pair GitHub recommends the key type: ED25519.
   https://gitlab.sdu.dk/help/ssh/README#review-existing-ssh-keys
2. Adding your public key to your profile:
   1. Go to https://gitlab.sdu.dk/
   2. Access your settings on the right side.
   3. On the left, click on ”SSH Keys”
   4. Paste your SSH **public key** in the text field.

5. Git clone with SSH should now work as intended.



SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

To add an SSH key you need to generate one or use an existing key.

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Do not paste your private SSH key, as that can compromise your identity.

Typically starts with "ssh-ed25519 ..." or "ssh-rsa ..."

Title: e.g. My MacBook key

Expires at: dd/mm/yyyy

Give your individual key a title.

Your SSH keys (1)

M3@DESKTOP 91:39:20:4a:ba:4b:50:ba:89:1b:45:78:95:e3:48:b4
Last used: 2 days ago Expires: Never Created 1 week ago

## Downloading the VIP project

Version number: 1.00

This manual explains how to download the VIP project from GitLab for use or development.

### Prerequisites:

Git: https://git-scm.com/downloads
Java 11 JDK (JRE can be used but developers need the JDK anyway):
https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

### Steps

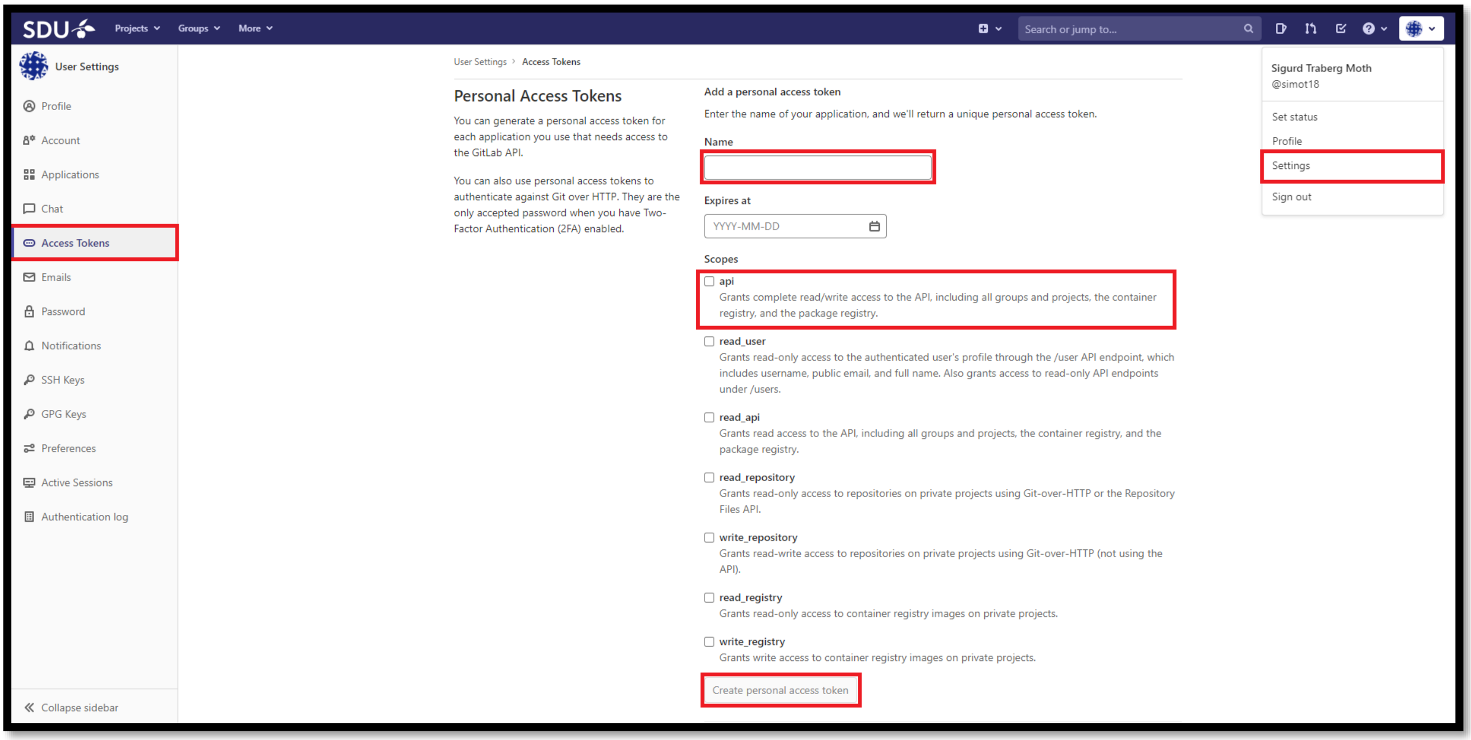
1. The VIP software must first be downloaded. It can be cloned from the GitLab repository and then compiled, or later downloaded as .jar files (WIP).
    a. If SSH is set up (see manual for **Setting up SSH for VIP**).
    Simply paste ”Git clone git@gitlab.sdu.dk:HDCL/virtual-internet-platform.git” into a bash terminal in the folder where you want the repository to reside.
    b. Alternatively write the following in a terminal with bash to clone the project to a folder (requires an account for gitlab.sdu.dk):
    “git clone https://gitlab.sdu.dk/HDCL/virtual-internet-platform.git”
2. The repository should now have been downloaded to a folder and can be opened in an IDE.

# Setting up VIP for development

VERSION NUMBER: 2.00

This manual explains the steps required for a user to correctly configure the system. The manual is created based on the first version of VIP where it is running locally through Visual Code on one PC without Docker.

1. Download the repository (See “**Downloading the VIP project**”).
2. Set up Visual Code (See “**Setting up Visual Code**”).
3. Create a Personal Access Token on GitLab. To create a token the user must be logged in at GitLab. The name of the token should be something related to the VIP, but it is not important as such. Additionally, the “api” tag must be marked or the token will not grant rights to pull and push to the VIP project. *Important*: The key must be saved after creation as it will not be possible to view it after a page refresh. See next step for where to save it.



Personal Access Tokens

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

Add a personal access token

Enter the name of your application, and we'll return a unique personal access token.

Name

Expires at

YYYY-MM-DD

Scopes

□ api
Grants complete read/write access to the API, including all groups and projects, the container registry, and the package registry.

□ read_user
Grants read-only access to the authenticated user's profile through the /user API endpoint, which includes username, public email, and full name. Also grants access to read-only API endpoints under /users.

□ read_api
Grants read access to the API, including all groups and projects, the container registry, and the package registry.

□ read_repository
Grants read-only access to repositories on private projects using Git-over-HTTP or the Repository Files API.

□ write_repository
Grants read-write access to repositories on private projects using Git-over-HTTP (not using the API).

□ read_registry
Grants read-only access to container registry images on private projects.

□ write_registry
Grants write access to container registry images on private projects.

Create personal access token

4. Set up the “.gradle” folder. The .gradle folder (if it does not exist) must be created in the home folder of the user (Next to the .ssh and .m2 folders but is should have been created when installing gradle). The home folder can be found by clicking “This PC” -> followed by the up arrow (moving one directory up from the current) -> Clicking your home folder named your PC’s name.
5. Now in the .gradle folder, create a file named gradle.properties. Make sure that properties is the extension and that it is does not become a “gradle.properties.txt” file (*Important*: file extensions are by default hidden in windows, so make sure to enable it or to closely inspect the file at creation).
6. In the gradle.properties file (can be opened with a text editor), paste the following: “gitLabPrivateToken=” and your personal key so it will look something like this: “gitLabPrivateToken=RGEQjRxks79yJHz3yk1a”. (*Important*: quotation marks should not be written in the file).

7. Now, back in Visual Code gradle commands must be added to every project. This is done by opening a bash terminal in Visual Code and then writing “chmod +x gradlew” in every project folder. “./gradlew” commands will now work.
8. In the terminal type: “./gradlew build” in the project folder. The project should now build with gradle (wait for it to finish building). The project will create a build/libs folder that contains the .jar file and all its downloaded dependencies (from build.gradle).
9. In the terminal type: .”/gradlew bootrun”. The project should now start a springboot container and run the project.

## Creating a springboot microservice

VERSION NUMBER: 1.00

New features are usually created in their own microservices to improve extendibility.  These microservices have to be set up correctly to improve readability and to make every service similar in design. As Java has been used as the core programming language, a manual for creating a springboot service is made.

1. Set up Visual Code for development (See “**Setting Visual Code up for development**”)
2. Create a repository in the GitLab for the microservice.
3. Clone the repository to into a folder (preferably the same folder for other VIP services).
4. In Visual Code, press F1 and use the extension Spring Initializer to set up a new project and select the following:
    a. Springboot version: 2.4.0
    b. Programming language: Java
    c. Group ID: “dk.vip” (standard project structure for VIP)
    d. Artifact ID: Name it what you want the .jar file to be named (this will be the project folder as well).
    e. Packaging: JAR
    f. Version: 11
    g. Select spring web if it is a microservice that is going to be running by itself. Select nothing if it is a library,
    h. Generate into the folder that was cloned from GitLab.
5. Open a terminal in the project and write “chmod +x gradlew” once for gradle commands.
6. Test that it works with ./gradlew
7. Make sure that the .m2 folder is created (see “**Setting up VIP for development**”).
8. A project should now be set up with springboot and it should be working with gradle commands.

## Preparing a microservice architecture

VERSION NUMBER: 1.00

…

# Deprecated Manuals

Manuals that are no relevant but might serve as a track history for the project or if reverting is necessary.

## Setting up VIP for development

VERSION NUMBER: 1.00

This manual explains the steps required for a user to correctly configure the system. The manual is created based on the first version of VIP where it is running locally through Visual Code on one PC without Docker.

1. Download the repository (See "**Downloading the VIP project**").
2. Set up Visual Code (See "**Setting up Visual Code**").
3. Set up the ".m2" folder. This is done by downloading the settings.xml file from the documentation project on GitLab: https://gitlab.sdu.dk/HDCL/virtual-internet-platform-vip/documentation/-/blob/master/settings.xml
   The .m2 folder (if it does not exist) must be created in the home folder of the user (Next to the .ssh folder). The home folder can be found by clicking "This PC" -> followed by the up arrow (moving one directory up from the current) -> Clicking your home folder named your PC's name.
4. Now, back in Visual Code gradle commands must be added to every project. This is done by opening a bash terminal in Visual Code and then writing "chmod +x gradlew" in every project folder. "./gradlew" commands will now work.
5. In the terminal type: "./gradlew build" in the project folder. The project should now build with gradle (wait for it to finish building). The project will create a build/libs folder that contains the .jar file and all its downloaded dependencies (from build.gradle).
6. In the terminal type: ."/gradlew bootrun". The project should now start a springboot container and run the project.