LINKÖPINGS UNIVERSITET

TDP003 Project: Own computer environment

Installation manual

Authors

Dennis Abrikossov, `denab905@student.liu.se`
Philip Bromander, `phibr608@student.liu.se`

INSTITUTIONEN FÖR DATAVETENSKAP

Fall term 2023
Version 1.0

2023-09-21

1 Revision history

Ver.	Revision notes	Date
1.0	Added information for every step and finalized document structure	2023/09/14
0.1	Added illustrations and captions	2023/09/11

2 Troubleshooting

The working operating system for this manual is 64-bit Ubuntu 22.04.3 LTS. For Windows systems the instructions will be very similar but you might have to refer to online documentation for some Windows specific implementations. Most Linux distributions will have same instructions other than which package manager is used.

Installation should be straight-forward however if at any point there is difficulty in completing a step you may refer to this section for general assistance.

2.1 Installing through apt

The package manager 'apt' (Advanced Package Tool) is the primary source for new applications on most Debian systems. 'apt' requires administrator privileges so make sure to include 'sudo' before running it and ensure you are on an account with administrator privileges.

'apt' and 'apt-get' perform the same tasks and are mostly entirely interchangable. 'apt-get' is better suited for scripts whereas 'apt' was designed to be more user friendly. If a step in this manual or a source online asks for you to run either of them then both qualify in nearly all cases. If 'apt' is for any reason unavailable on your system then 'dpkg' might help but that is beyond the scope of this manual.

When installing using apt and a package fails or is not found make sure your apt is updated by running the following:

```
$ sudo apt update
```

If an application is already installed but is not up to date you can run 'upgrade' to update all programs on the system. Do this after running update to have most up to date version.

```
$ sudo apt upgrade
```

Should the terminal fail you, you can almost always resort to the graphical user interface (GUI) instead. The GUI is available for both apt and the alternative package manager on Debian systems, 'snap'.

2.2 Query program version

You can check the version of most programs through the terminal by using the '--version' option, such as:

```
$ python3 --version
> Python 3.10.12
$ git --version
> git version 2.34.1
```

Match the version number you have installed with the most recent version available for the program. Use the version provided in the project specification as minimum version however, unless otherwise specified, you should be running the most up to date version.

3 Python

Python is the main programming language the project uses. The package for python is called 'python3' and is found on apt.

```
      $ sudo apt install python3
```

Python has its own package manager called 'pip' and it is typically installed alongside Python. If it did not install automatically then run:

```
      $ sudo apt install python3-pip
```

3.1 Flask & Jinja2

Flask and Jinja2 are libraries for Python that allow for web development. Flask is a web framework and Jinja is a template-engine that allows for easy integration with HTML. Install both through 'pip'.

```
         $ pip install flask
```

Jinja2 should be automatically installed when installing flask but if it for any reason did not, then install it manually:

```
         $ pip install jinja2
```

4 Git

Git is a version control tool which is extensively used in the world of development. Follow these instructions to install and configure your local Git installation as well as how to clone and configure the repository.

Simply install Git through 'apt':

```
$ sudo apt install git
```

4.1 Credentials

Set your login credentials for Git with these commands. This is required when making any commits as this will be your identity associated with any commits you make. Place your credentials between the quotes.

(Tip: use your Gitlab email!)

```
$ git config --global user.name "your user name"
$ git config --global user.email "youremail@example.com"
```

4.2 Merge method

When making changes to your local branch you might want to receive changes from collaborators. Setting your default merge method to manual merging lets you pull changes from the remote repository without stashing or committing your local changes when you have conflicting changes between the two. This is not a required step for configuring against this project but is recommended when working with other collaborators.

```
$ git config --global pull.rebase false
```

4.3 SSH keys

SSH, or Secure Shell, is a protocol for secure and encrypted communication over a network. SSH keys are cryptographic credentials that enable your computer to securely authenticate itself to a server, such as GitLab in this instance, without the need for a password. Both Windows and Linux have SSH built-in and usage is rather similar but we will be focusing primarily on Linux here. Refer to online documentation for complete Windows instructions on SSH.

4.3.1 Creating a new key

The first step is to generate an SSH key for your system if there is not one already. To find existing keys look for a file called 'id_rsa' (which is the default name) in the folder '.ssh' located in your home catalogue. Remember that it is a hidden folder and may not show up in the file browser.

To generate a new key use the following command:

```
$ ssh-keygen
```

You will be prompted to enter a file location for the new SSH key. Leave this empty unless you already have a default key in use, in which case you will need to name this new key something unique.

4.3.2 Adding SSH key to Gitlab

To use your new (or existing) key with Gitlab you must add the public part of the key to your Gitlab account. This is found at the same place as before where you found out if a key already existed. Make sure you use the '.pub' key as this is the public version. Never publish your private key! The full (default) path should look like this: "/home/.ssh/id_rsa.pub".

Now print or view the public key in order to copy it. 'cat' is a great program for this use on Linux:

```
            $ cat /.ssh/id_rsa.pub
```

Now to add the public key to your Gitlab account, open Gitlab and go to your profile. You can access it through your profile picture. Click 'edit profile' and then the tab named 'SSH Keys'. At the top of the table displaying current SSH keys you click the button in the corner, 'Add new key'.

The large first field is where you paste in the public key. Gitlab should warn you if the key is not recognised as a public key; if it does make sure twice that you have copied the public key! Enter a name for this key, this should be related to where the key will be used such as "Home computer" or "My laptop". You will want to have both authenticating and signing for 'Usage type'. Finally, it is recommended to set an expiration date. If the key is used for a desktop PC at your home then an expiration date is not necessary. If it is a computer at work or a laptop on the go you will want an expiration date.

4.3.3 Cloning project repository

Assuming you have access to the project files, visit the Gitlab page for the project. Find the blue 'Clone' button and copy the 'Clone with SSH' URI. Using this URI you will now tell Git to download the file structure onto your computer. It will configure the local repository for you automatically. To clone the repository use the SSH URI in Git:

```
            $ git clone (URI)
```

After waiting on Git to finish you should have the repository downloaded. Unless an empty folder with the same name already exists, it should have created a new folder. Change into that folder to view the project.

5 IDE

To edit code you will require a form of text editor or IDE. There are typically many choices and here you will find two: Emacs and Visual Studio Code. The former is an old text editor loved by many but requires a lot of learning. The latter is Microsoft's free and open source modern and extendable code editor. Very easy to use and highly customizable with extensions. If you are new to working with IDE's then Visual Studio Code is recommended however if you are on Windows you might find Microsoft's Visual Studio editor to be more suitable. This depends on what you intend to be using it.

5.1 Emacs

Use this to install Emacs:

```
         $ sudo apt install emacs
```

5.2 Visual Studio Code

Typically, software is installed through ’apt,’ just as all previous programs have been. However, some are installed via ’snap,’ which does not require sudo (although it will still prompt you for a password). Visual Studio Code is one such program that installs through snap. The installation process, however, is nearly identical to apt.

```
$ snap install code
```