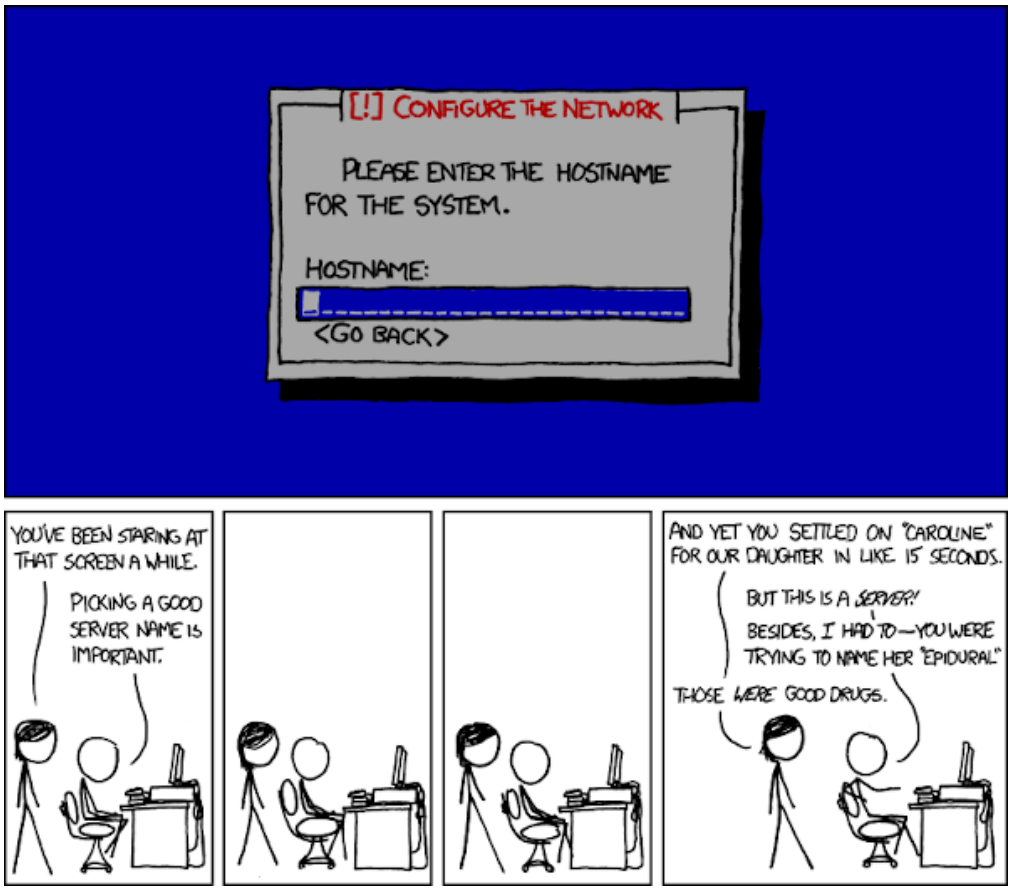
KU LEUVEN

GROUP T LEUVEN CAMPUS

Distributed Applications: Session 0
Setting up the development environment



Master Electronics/ICT – Software Systems

Lab coaches:

- Bert Lagaisse
- Toon Dehaene

Setting your virtual development environment for Distributed Applications

Distributed systems have to operate in heterogenous environments which leads to a plethora of complex configurations on each type of machine and operating system. Hence, for the practical session we will make sure all students start from a common uniform environment regarding operating systems, Java version, build tools, integrated development environment, middleware, web frameworks and versions etc.

The goal of this initial, tutorial-style homework is to set up such a common uniform environment on your personal laptop. This will help you, your team mates and the education team to **get the maximum out of the practical sessions in terms of content and experience regarding distributed applications** and distributed systems in general. We want to avoid that *students loose time searching and exploring for a random configuration issue* with some specific Java version, with some specific web framework on some specific operating system or some specific distro of an operating system. And there are many of those random issues out there …

To accommodate the students that have less experience with Linux, we explicitly adopted a tutorial-style description including quite detailed descriptions, screenshots and commandline instructions.

1. Overview

Our uniform development environment will be based on a virtual machine on your personal laptop. This initial tutorial will help you set this up. Concretely we need to set up the following tools with their own installation and configuration process:

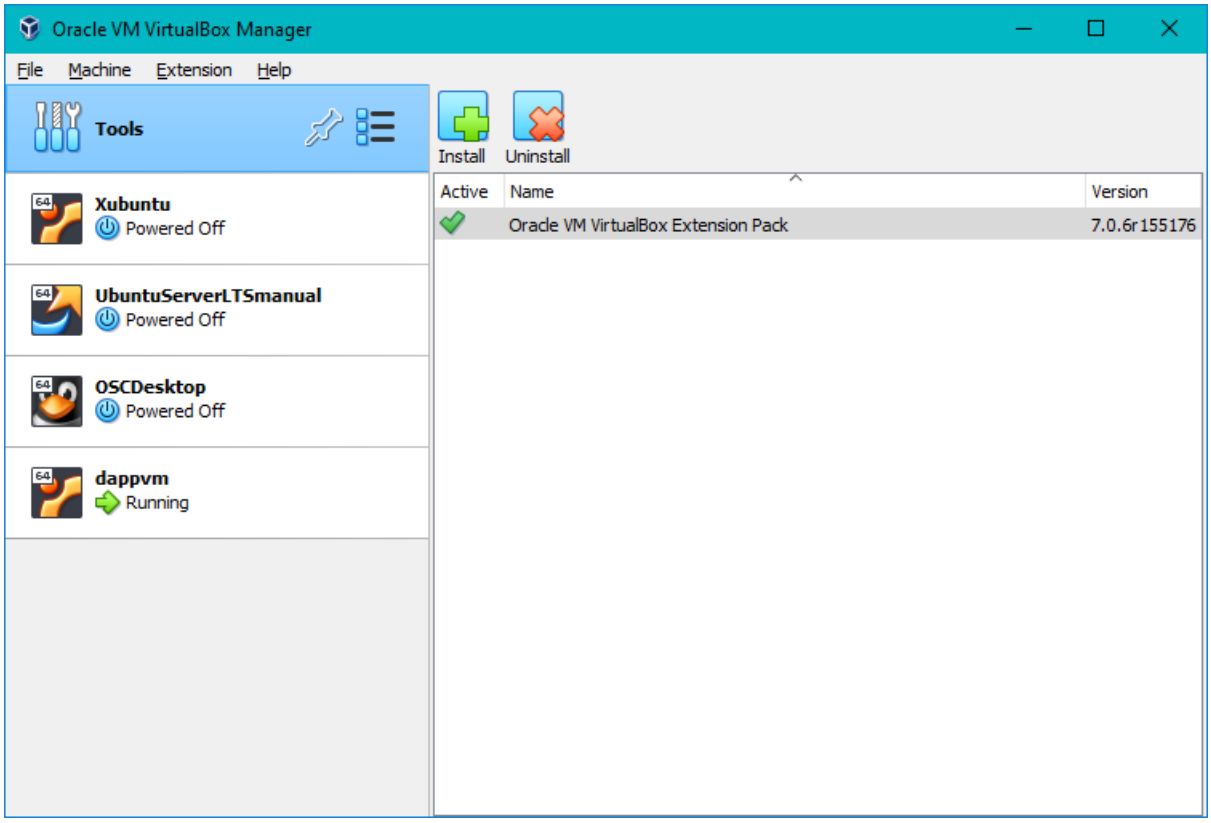
- Virtual Box 7
- Ubuntu 22.04 Desktop LTS
- Virtual box guest additions for Ubuntu such that you can use copy-paste between your native environment and the virtual environment.
- OpenJDK 17
- Apache Ant
- Apache Maven
- Git
- IntelliJ community edition.

2. Installing Virtual Box

Virtual Box 7 is a virtualization environment that works on all laptops and operating systems, including the new Apple MacBook. Hence, this is a good starting point for a uniform environment. You can download Virtual Box for free from here: https://www.virtualbox.org/wiki/Downloads

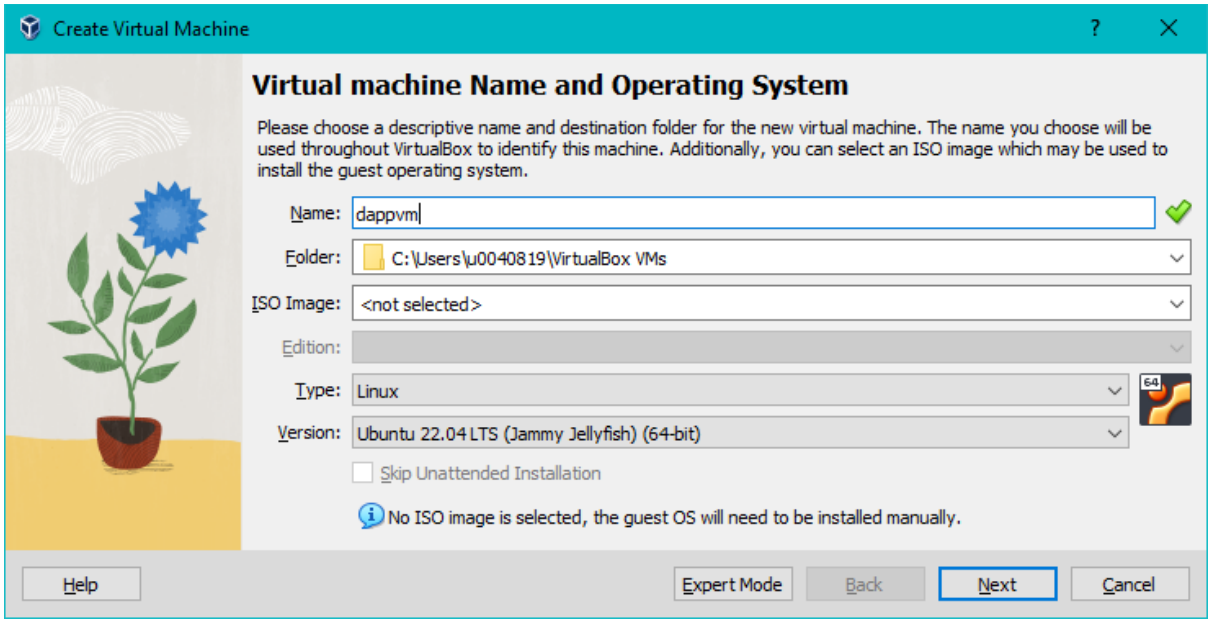
Download both the **platform package** for your operating system as well as the **extension pack**.

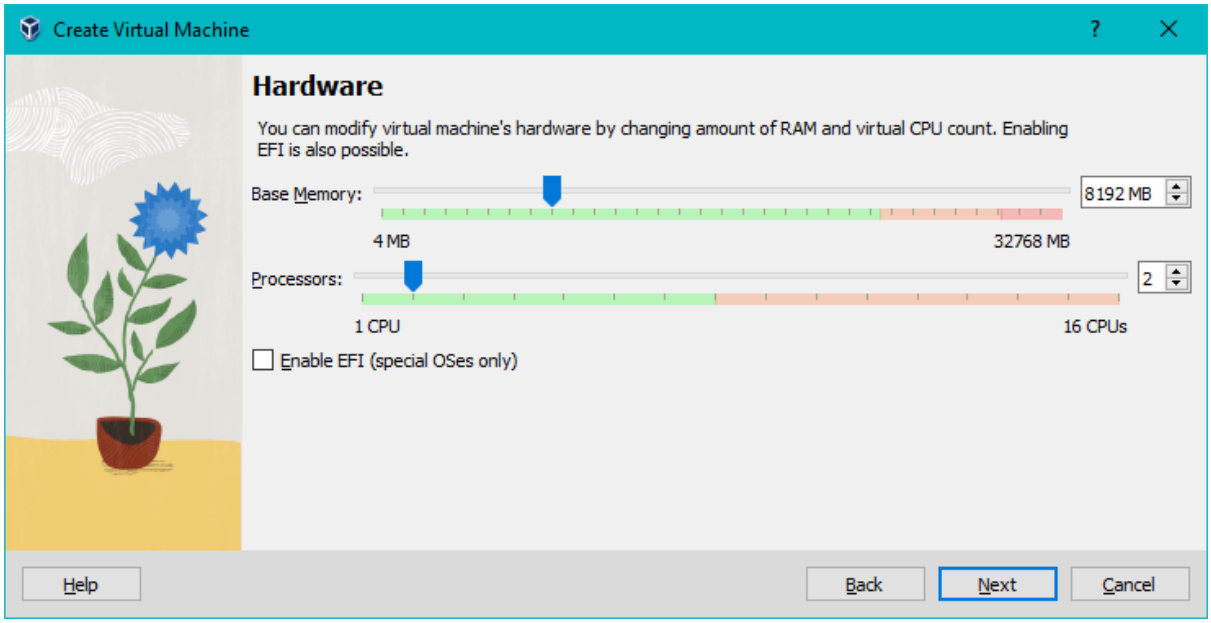
First install the platform, and then under tools, you can install the extension pack.

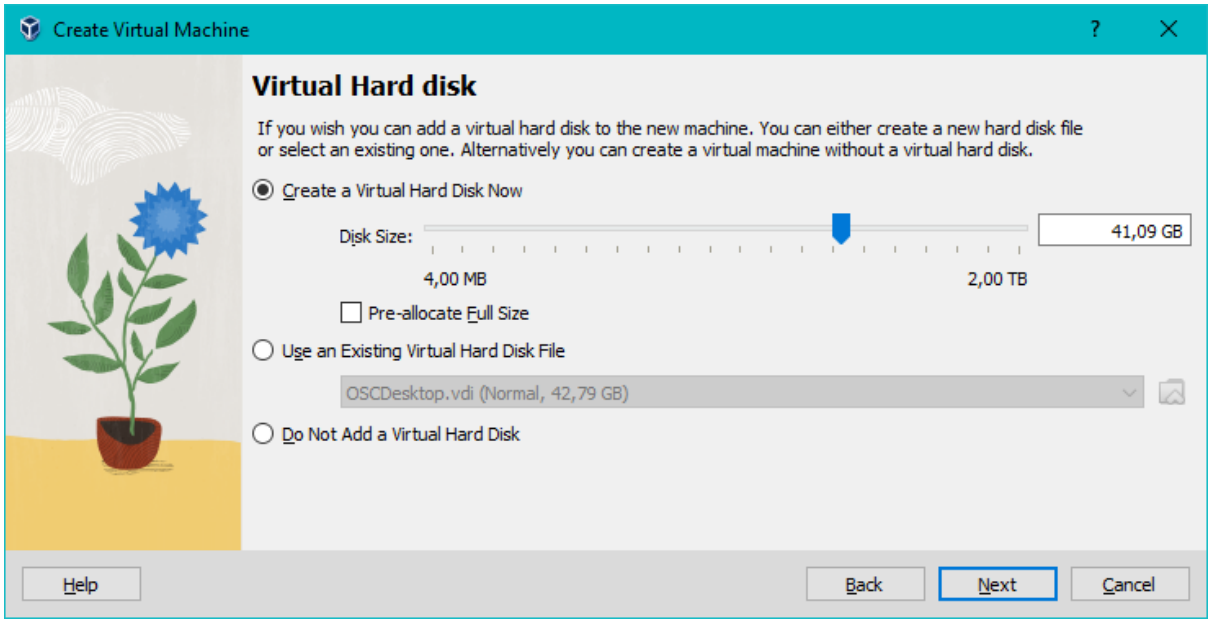


3. Creating a VM

We will create a VM for Ubuntu 22.04 that has a dual core cpu with 8 GB ram and 40 GB of disk space. Do not mount the iso image of Ubuntu yet, as this will boot a live operating system from the ISO rather than install ubuntu onto your VM.





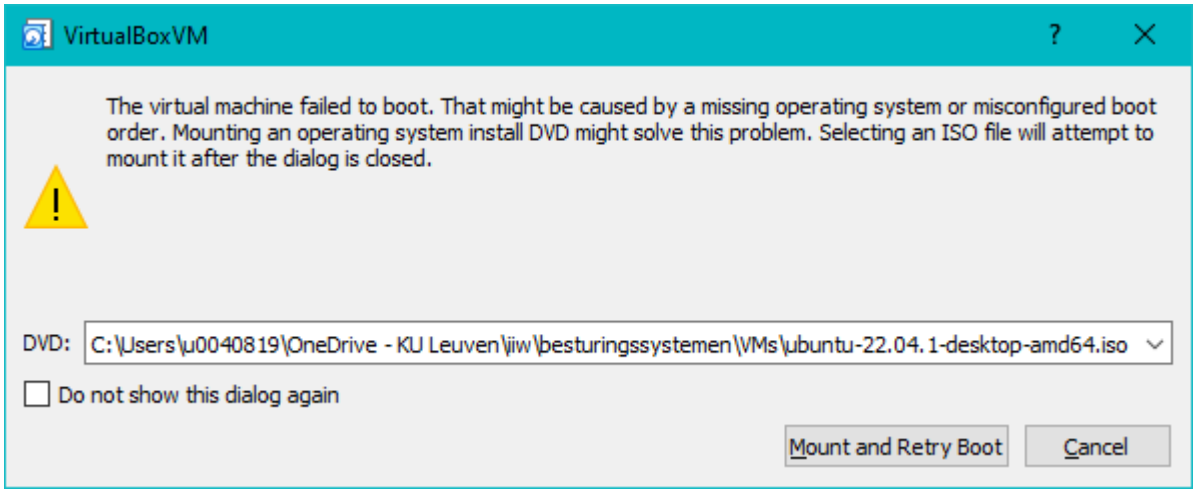


4. Installing Ubuntu 22.04 Desktop.

You can now start up the VM. The VM will find no boot medium and present you the option to mount the install DVD iso image of Ubuntu 22.04 Desktop.

You can download this iso from https://ubuntu.com/download/desktop

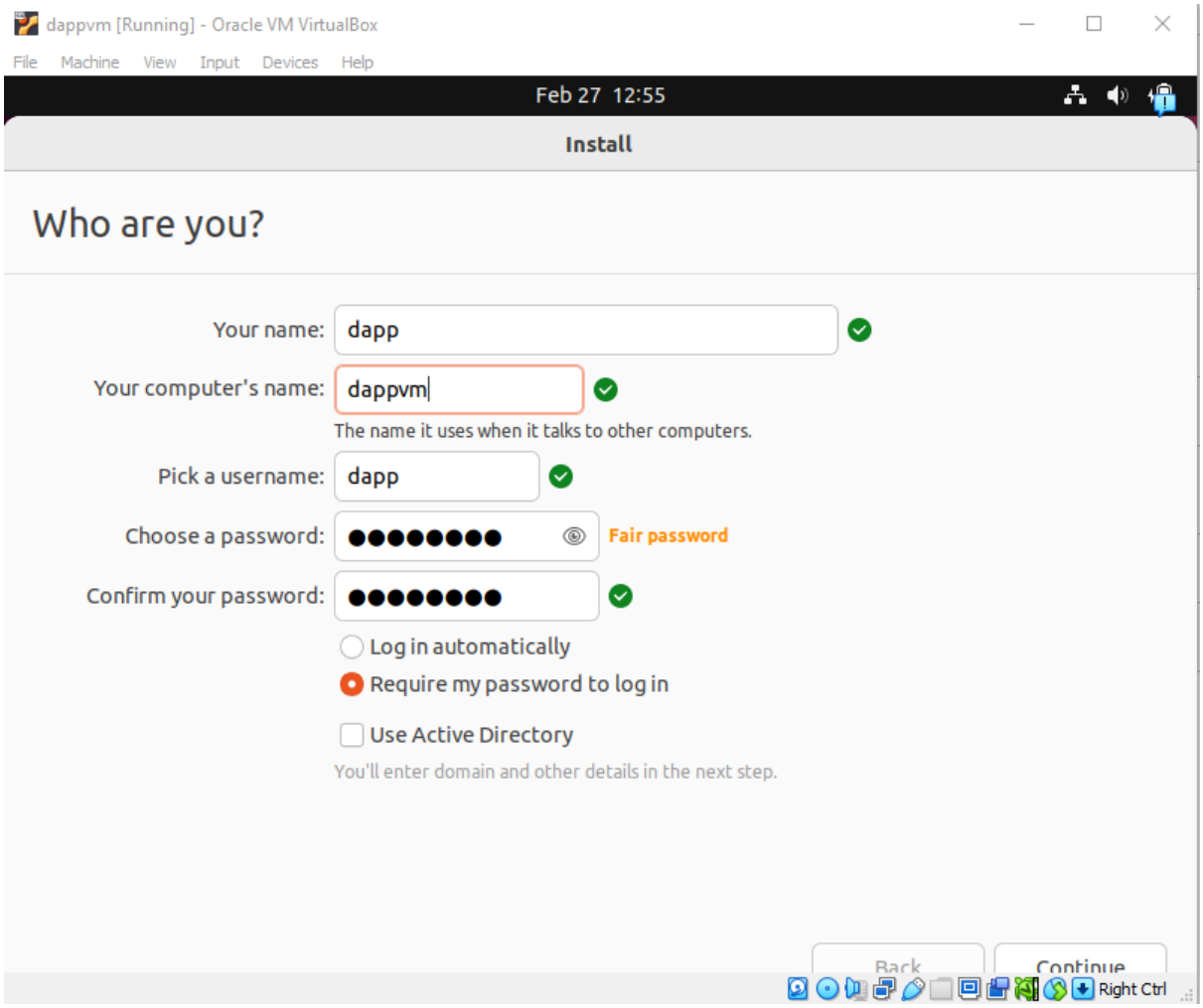
Then tell VirtualBox where to find the iso:



Reboot the VM and choose to *try or install Ubuntu* in the text-based menu interface. After a while you will be presented a GUI to install Ubuntu. Choose to install Ubuntu.

Configure your keyboard layout and language and choose to continue the installation with the default configured options (normal installation, download updates, erase disk and install Ubuntu) and continue with “install now” or “continue”.

You can now create a computer name and user account with a name and password



Choose names and passwords that you can remember and type easily. Your VM is protected from the outside world by your laptop and its operating system.

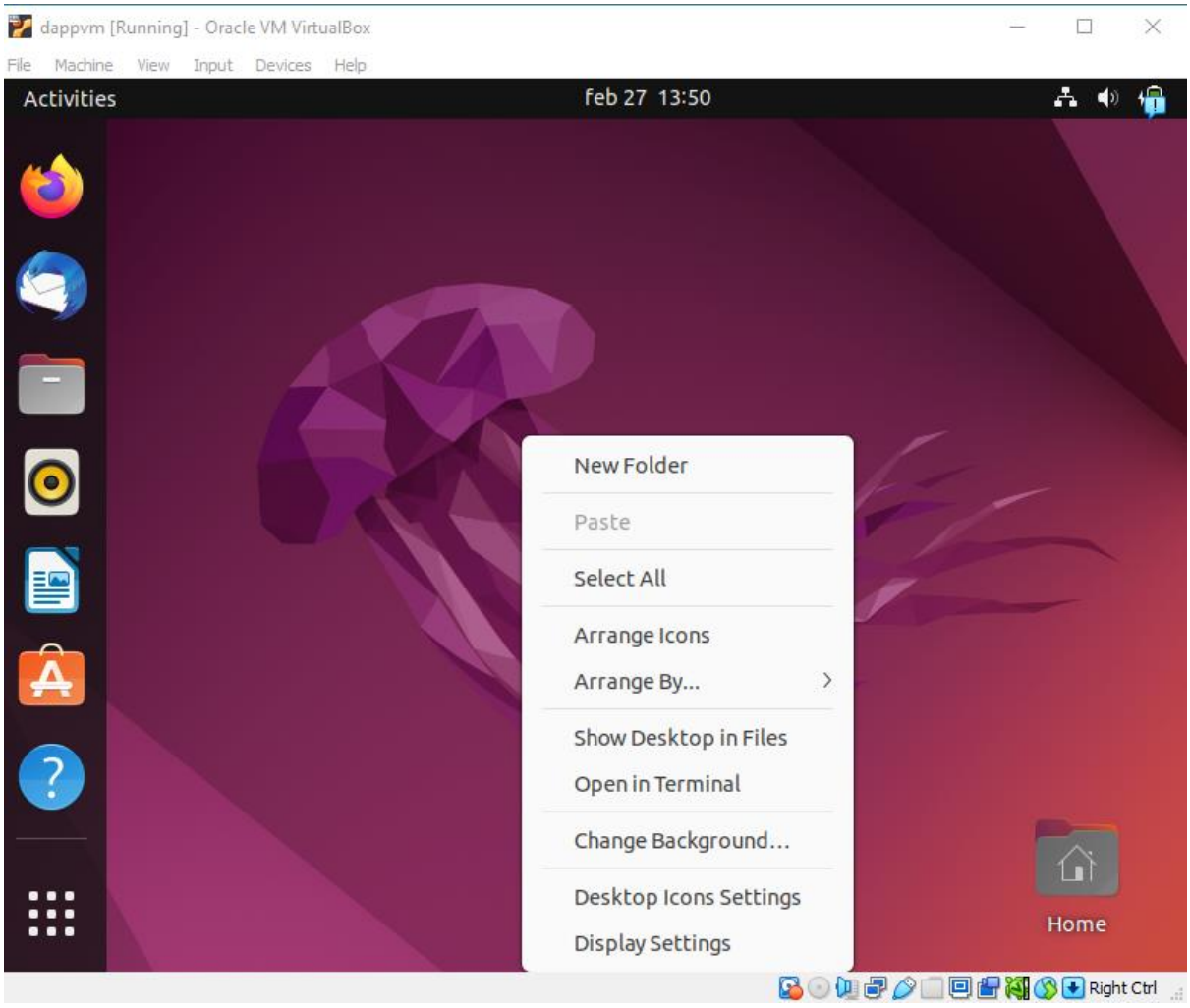
Wait and relax until the installation is finished. At the end, reboot to start Ubuntu Desktop.

Unmount the installation DVD (in the menu Devices) and reboot.

You can skip the start-up screen that wants to configure some bloatware and other software.

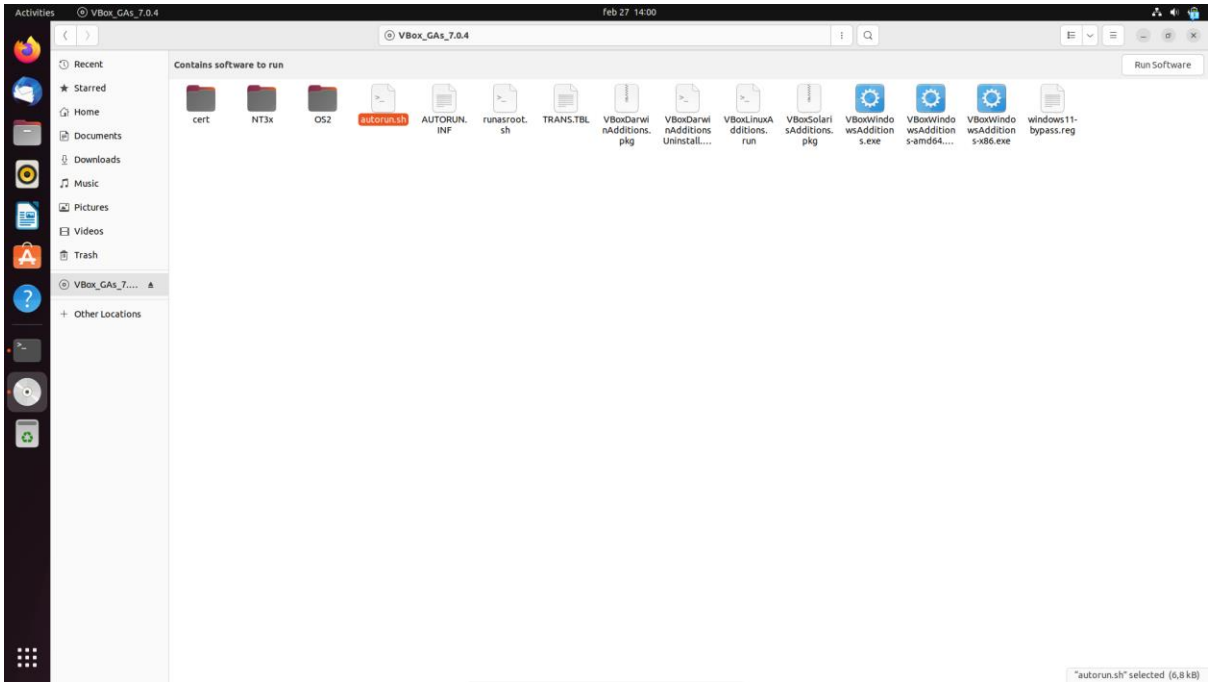
It is advised to install the updates when Ubuntu asks to update. Reboot at the end.

You can change the resolution of the screen on the desktop to a resolution fitting your laptop and put the VM in full screen.

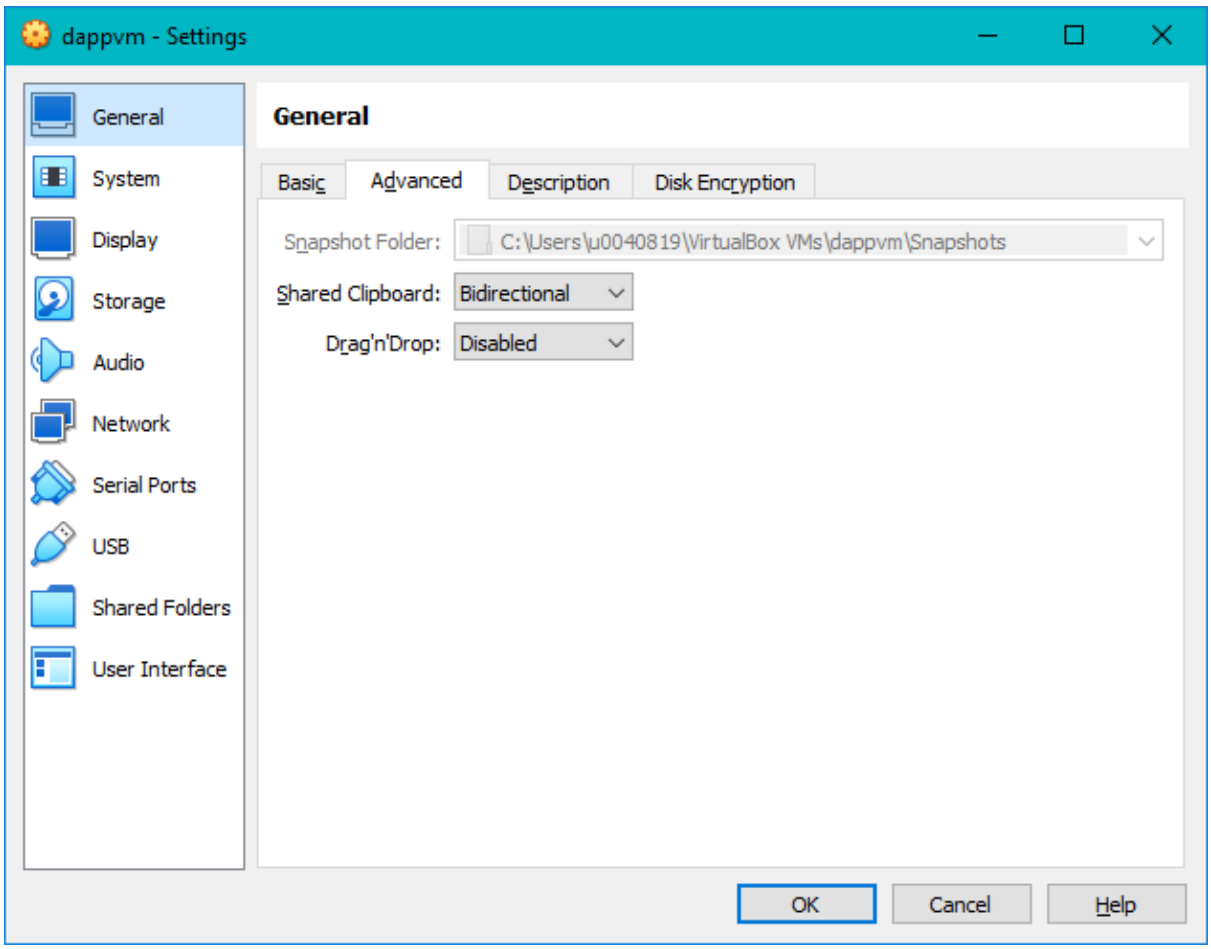


One last thing to have a functional virtual environment is supporting copy paste between your native laptop OS and the VM. To achieve this, we need to install the guest additions DVD iso image and then enable copy paste.

- In “devices”choose to insert the guest cd additions image. This will show the guest additions cd in the files menu.



Right click on autorun.sh and choose “Run as a program”. After entering the root password, the guest additions will install. In the settings of your VM in Virtual Box, you can then enable Shared Clipboard to copy paste text between your host OS and guest OS.



In the machine settings, enable a bidirectional shared clipboard, and reboot. You can now copy paste.

In Ubuntu the installed applications can be found using “show applications” at the left bottom of the screen. Select “terminal” from the applications overview to open a commandline interface (CLI).

You can now control your ubuntu both via the GUI and the CLI.

5. Installing Java 17, development tools and Intellij

We will use the terminal as a CLI to install some packages and tools for Java development. We need to execute this installation as root (administrator) of the operating system using “sudo” aka “do as super user”. We will install OpenJDK 17, the maven build tool and package manager, git and the snap installation tool using *apt install*. We will the install IntelliJ using *snap install*:

```
$ sudo apt install openjdk-17-jdk
$ sudo apt install maven
$ sudo apt install ant
$ sudo apt install git
$ sudo apt install snap
$ sudo snap install intellij-idea-community --classic --edge
```

You can now find IntelliJ Community Edition in your Ubuntu applications. Try starting it up and create a simple Java test project that uses Maven as a build system. Under File > Project Structure you can verify the JDK version. Try putting a break point on some line of code and try to debug.

6. Github Desktop

If you prefer Github Desktop instead of the commandline git, you can install it using the flatpak installation tool:

```
$ sudo apt install flatpak
```

You should reboot the VM before continuing.

```
$ flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo

$ flatpak install -y flathub io.github.shiftey.Desktop
```

First time run:

```
$ flatpak run io.github.shiftey.Desktop
```

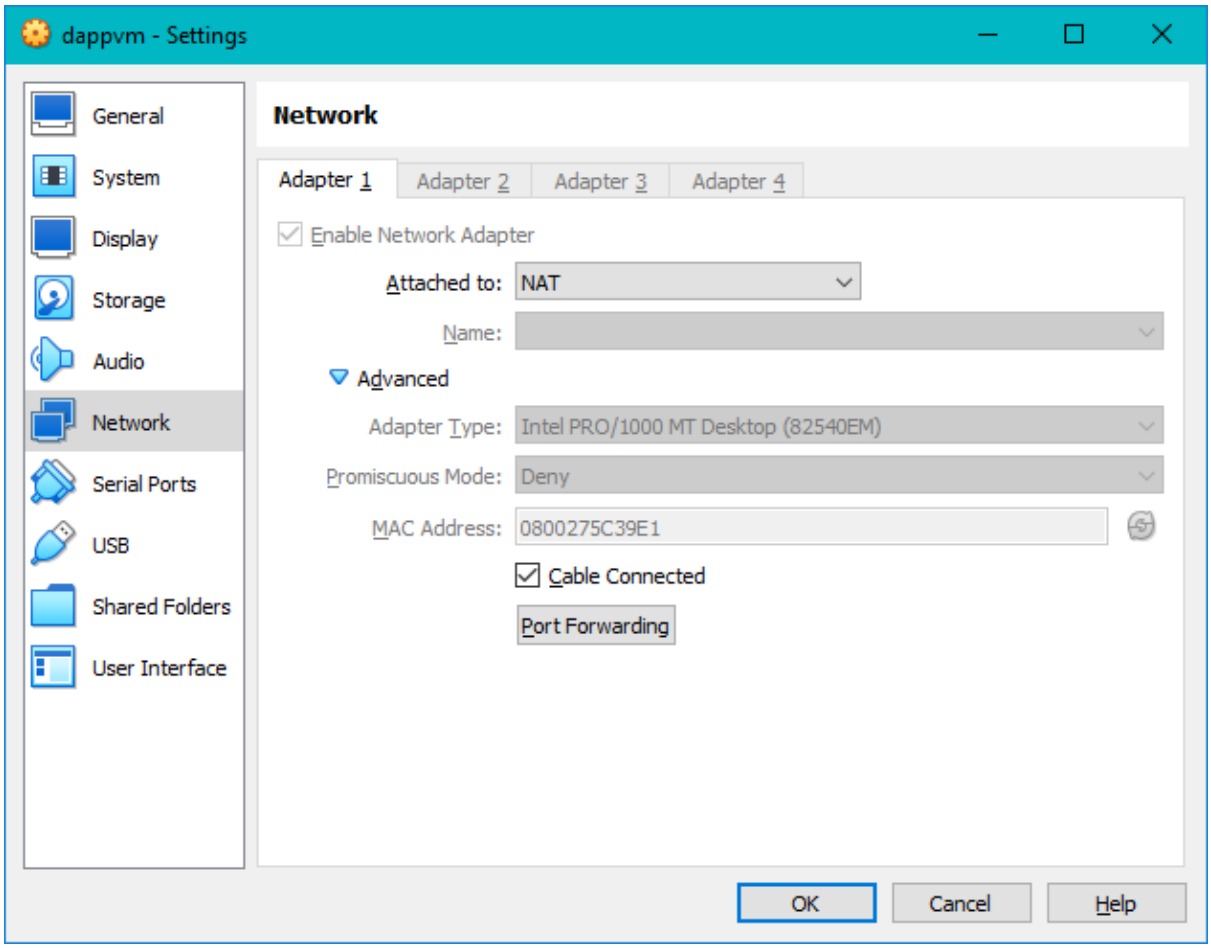
7. Setting up sshd on your ubuntu desktop

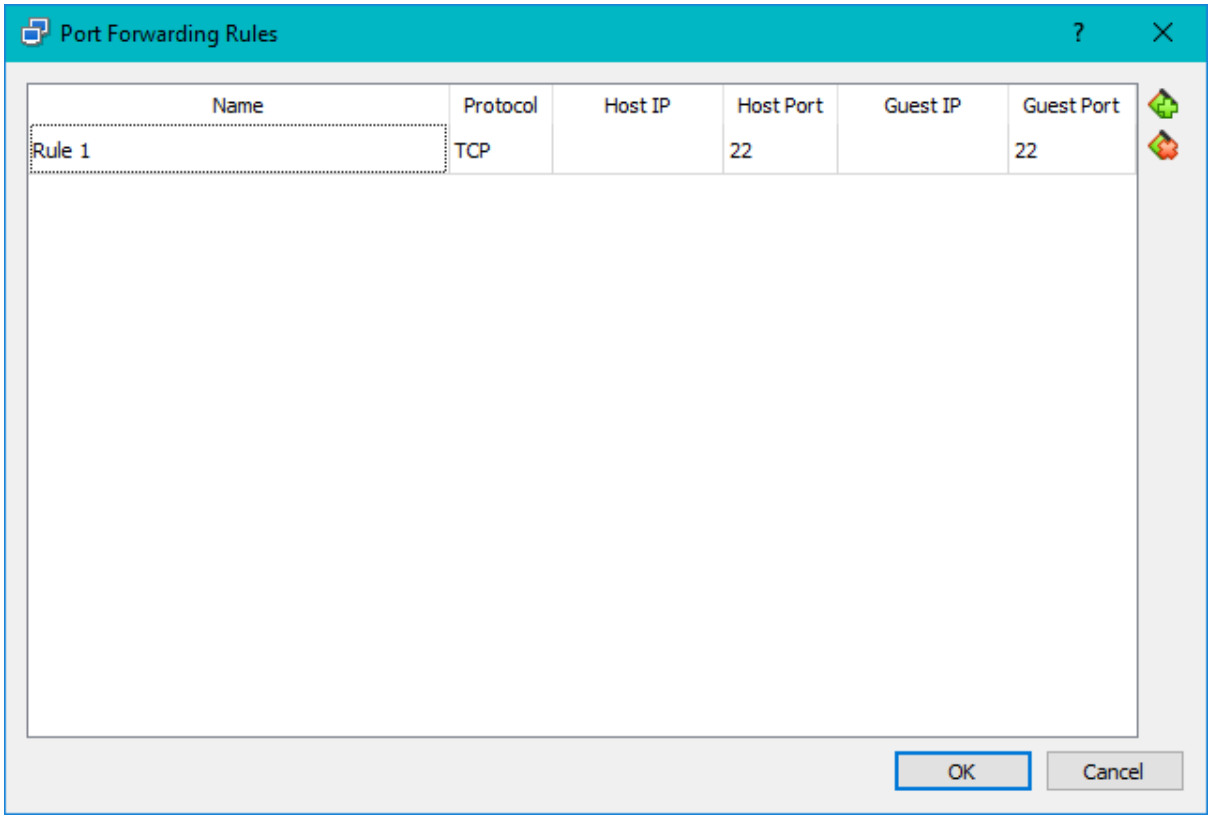
To login to your VM via ssh or to transfer files to and from your VM, you need to install the ssh server on your ubuntu desktop. As such, your ssh client (e.g putty) or sftp client (e.g. filezilla) on your laptop’s native operating system can connect to your VM. The guest port is by default 22. We have also chosen the default port 22 on our native host OS. However, if you already run Linux natively with an sshd on your laptop, you might need to choose another port on the host OS (e.g. 2222)

Installing the ssh server:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install openssh-server
```

You will also need to open up the port in the VirtualBox network configuration of your VM to forward ssh connections to your vm:





8. Testing the setup

Ssh into your vm from your native host OS, e.g. using the ssh commandline client:

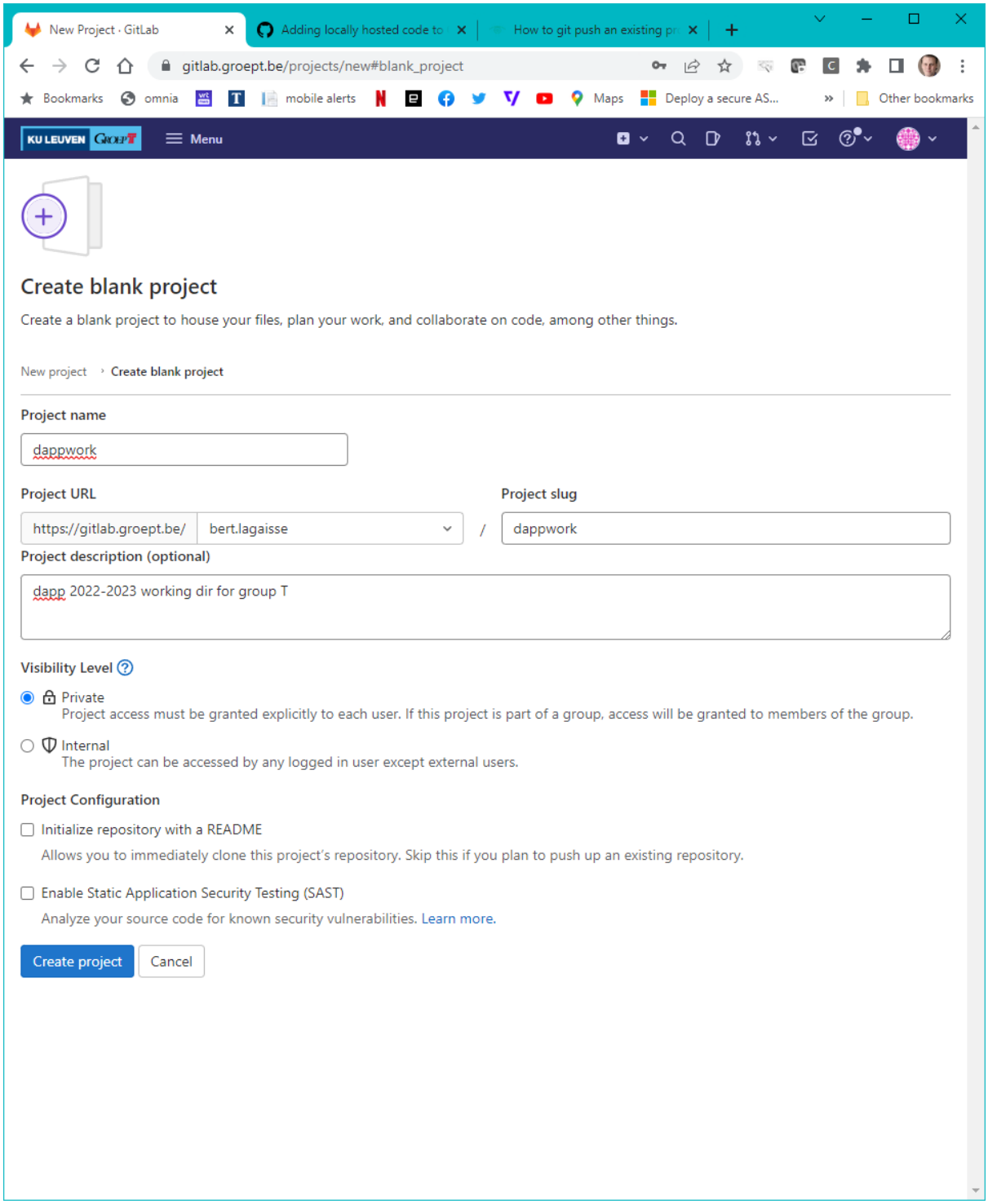
```
C:\users\u0040819> ssh dapp@localhost
```

You can verify the installation and version of the required tools using

```
dapp@dappvm:~$ java -version
openjdk version "17.0.5" 2022-10-18
OpenJDK Runtime Environment (build 17.0.5+8-Ubuntu-2ubuntu122.04)
OpenJDK 64-Bit Server VM (build 17.0.5+8-Ubuntu-2ubuntu122.04, mixed mode, sharing)
dapp@dappvm:~$ javac -version
javac 17.0.5
dapp@dappvm:~$ mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 17.0.5, vendor: Private Build, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.19.0-32-generic", arch: "amd64", family: "unix"
```

9. Backing up your code to the Group T Gitlab server

First create a blank empty project on the Group T Gitlab server for the practical sessions. For example, in the screenshot below, I created the *dappwork* project as a blank project without any initialization of the repository.



Configuring the sshkey on your Ubuntu VM to authenticate against the gitlab server:

1. Search 'passwords and keys' in the ubuntu search menu (On your Ubuntu VM).
2. Find the '+' sign, click it and choose 'secure shell'
3. Fill in anything (eg: dapp-gitlab) in the description, the defaults (RSA2048) should be good. Click 'just create key'
4. If any pop ups come up click allow and fill in no password (leave empty)
5. Go to your created key, double click and go to 'public key'. Copy the whole field
6. Go to https://gitlab.groept.be/-/profile/keys and create a new key pasting the public key in the "key" field.

Initialization on your local development machine (The Ubuntu VM):

Create a local folder, e.g. dappwork, for the practical sessions, and create a subfolder for this homework, e.g. session0. Create a simple text file in this subfolder as a test file.

After navigating in the *dappwork* folder, we are now going to

1) create a local git repository and initialize it as the master branch,
2) add the local files in the folder to the local git repository,
3) add YOUR remote git project url to the local configuration
4) commit locally with a message “initial commit”,
5) and push everything to the remote server.

```
cd dappwork

git init --initial-branch=master

git remote add origin git@gitlab.groept.be:bert.lagaisse/dappwork.git

git add .

git commit -m "Initial commit"

git push -u origin master
```

To clone your repository into a new folder, e.g. after a crash or to work on a copy of the code:

```
dapp@dappvm:~/me$ git clone git@gitlab.groept.be:bert.lagaisse/dappwork.git
Cloning into dappwork...
…
ED25519 key fingerprint is
SHA256:RpLXYht2XJbFMB3TYDmPOdWPpEmK5PC3WJyFR/GcF2o.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
…
```

Using Github Desktop and access tokens instead of commandline and ssh keys

If you don’t like the commandline approach to git using ssh keys, you can also use GitHub Desktop with access tokens and the https:// link to the repository. More info on:

https://github.com/desktop/desktop/blob/development/docs/integrations/gitlab.md

10. Conclusion

This first practical work was a quick tutorial for students new to Virtual Box and Ubuntu desktop. You should now be able to fluently work in the Ubuntu desktop environment and develop Java-based Distributed Applications in IntelliJ. In the next practical sessions we will use this environment to develop Java RMI applications, Spring boot applications for REST and SOAP communication, as well as deploy such applications to cloud-based virtual machines on the Azure IaaS platform.