# Design of an Educational use Virtual Machine

Tuisku Polvinen

December 4, 2019

## Contents

# 1 Course virtualization

This document describes how to design and build pre-installed virtual machines that run on student laptops and contain an OS with all needed software, pre-installed and pre-configured.

## 1.1 Introduction

On most courses, each student is expected to bring their own laptop, on which course exercises are performed. Student laptops contain a variety of operating systems, environments and tools, which have to be configured carefully to ensure compability to avoid problems and errors. The level of expertise in installing and configuring these tools varies with each student, and on some courses there might be a danger of falling behind if a less experienced student does not receive extensive help from their peers or the teacher.

To solve these problems, The Department of Future Technologies of the University of Turku launched a project which aims to bring a solution that:

- reduces the time used for software configuration on courses so that the time can be used for actoual course purposes
- lessens the burden of teachers by reducing the amount of assistance students need for using course tools
- ensures that every student has access to a homogenous environment compatible with other students'.

## 1.2 Virtual machines as a solution

The goal of this project is to solve the abovementioned problems by virtualization. A decided amount of different virtual machines (VM) are built for use in different types of courses and distributed to students, pre-configured. The VMs each run a Linux operating system and come with all needed software pre-installed. Installation of a hypervisor is considerably simpler than installing and configuring large sets of development tools, and to further simplify the process, this project aims to provide students with clear documentation on how to install and use the VMs. This approach is assumed to have several positive effects:

- The teacher is fully aware of the type of environment that students have at their disposal, making it easier to provide technical support.
- The teacher can customize the virtual machine before distributing it to the students, so even fast schedule changes to the machines are possible without bureaucratic processes, if needed
- Unlike a remote environment, the VM operates on the student's own laptop, available whether or not connected to the internet.
- A student can have several different VMs simultaneously at their own, each serving the needs of a different course.
- Installing a virtual machine monitor is simple, and advising students in it's use takes less time from the teacher than all the installation and configuration help that would be needed without a virtual machine.
- The same process and VM model can be used across several different courses, unifying some of the technical support needs of students.

In addition, the usage of virtual machines provides students hands-on experience with virtualization and a low-threshold opportunity to familiarize with unix-based systems.

### 1.2.1 A brief introduction to virtualization

Virtualization is a technology that enables simulating a layer of virtual hardware, a virtual machine, which can be executed in an isolated environment on the host computer. One or more virtual machines can be run on the same host computer, each with their own operating systems and applications. There are two main types of virtualization: bare metal virtualization (type 1), where a hypervisor runs directly on the system's hardware, and hosted virtualization (type 2) where the hypervisor runs within a native operating system as an application (Figure 1) [1, 2].

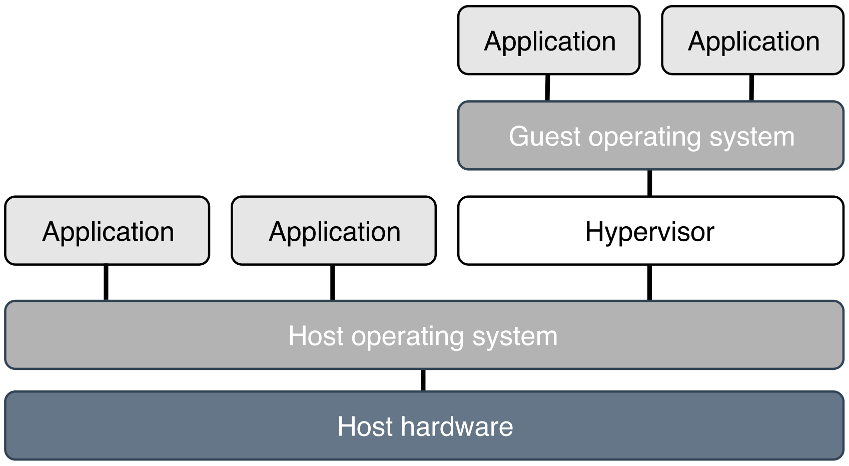

Figure 1: Architecture example of type 2 hypervisor

A hypervisor, or virtual machine monitor, is the software that creates, runs and controls the virtual machines. While a type 1 hypervisor interfaces directly with the host hardware, a type 2 hypervisor interfaces with the host operating system to access hardware resources and distributes them to the guest systems.

The main benefits of virtualization are better control of the hardware resources which can be partitioned and distributed effectively to multiple virtual machines on the same hardware, isolation which has security benefits and encapsulation meaning that the virtual machine state an be saved as a file that can be moved and copied as easily as any file [3].

# 2 Plan

During the beginning of our project, we laid out our goals and a general timeline, and did research on similar projects done in other universities. There were many similar virtualization projects with more or less differences, many of which were intended for computer security or networking courses [3, 4, 5], some for operating system courses [6], some for providing research tools for a specific field [7, 8, 9, 10] and some, sparking our special interest, for computer science studies in general [11, 12, 13]. All of the publications provided valuable insight to what approaches others had chosen.

## 2.1 Prerequisites

We began designing our approach by mapping out which courses would be included, what requirements they had and how they would affect our design choices. We expected there to be a lot of variation in student laptop performance. For the virtual machines to work well and to not take a lot of disc space on student laptops, we wanted them to be as small and lightweight as possible. Since including all of the software needed on any of the courses in one virtual machine would have led to our VM image to be unnecessarily large, we wanted to design a set of VMs with a uniform base and a varying selection of specific tools. The requirements for each course were laid out in a table and studied to find similarities between courses, so that we could deduce which courses would be able to use a similar VM setup. Our build design is presented in figure 2.

| Package | TKO_2038 | TKO_2005 | DTEK1048 | DTEK1049 | DTEK1066 | TKO_8971 | DTEK8101 | DTEK2040 | DTEK2041 | VM1 | VM2 | VM3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Runtime libs** | | | | | | | | | | | | |
| Java JRE | ● | ● | ● | ● | ● | ● | ● | × | × | ● | × | × |
| Java JDK | ● | ● | ● | ● | ● | ● | ● | × | × | ● | × | × |
| Java FX | ○ | ○ | ● | ● | ○ | · | · | × | × | ● | × | × |
| **IDEs** | | | | | | | | | | | | |
| IntelliJ IDEA | ▷ | ▷ | ▷ | ▷ | ▷ | ▷ | · | – | – | ▷ | × | × |
| Eclipse | ● | ● | ● | ● | ● | ● | · | – | – | ● | × | × |
| VS Code | | | – | – | · | · | · | ▷ | ● | × | ● | ● |
| Atom | | | – | – | · | · | · | ▷ | – | × | × | ▷ |
| ... | | | | | | | | | | | | |

| | | |
|---|---|---|
| obligatory<br>● | useful but not required<br>○ | neutral, unrelated<br>· |
| adverse, takes up space<br>– | conflicting<br>× | alternate<br>▷ |

Table 1: An excerpt of the course requirement table.

### 2.1.1 Selecting a hypervisor

A hypervisor is an essential part of creating and running virtual machines. There is a variety of different hypervisors available, and in this project, VMware Workstation [14] and Oracle VirtualBox [15] were both used in the initial design and production process.

To simplify matters and to keep our implementations congruent, we wanted to decide on one hypervisor to use. It would have been possible to provide kernel support for multiple different hypervisors simultaneously, so that the VMs would function optimally regardless of what hypervisor it was run with, but because we wanted to keep the project simple enough to manage in the future, we dropped the idea.

Deciding between VirtualBox and VMware virtualization was not an easy task, as both of them provided all of the functionality we needed, had their own drawbacks and advantages but neither was a clear better option. Some of the pros and cons we considered are presented in table 2.

Creating and managing the virtual machines with VMware software was considerably simpler, since the tools needed for VMware compability are already included in the Debian kernel and no additional installations on the guest OS are needed. Wmware Player can be used in Windows and Linux for free, but for a MacOS user VMware offers Fusion, for which a license is needed.

VirtualBox is free and open source, eliminating any licensing issues. During our research for similar projects we found out that some had had problems with VirtualBox, and many who had initially chosen VirtualBox had later switched to other solutions, most notably VMware [12, 13]. This led us to investigate further. We ran some tests, mostly regarding hypervisor behavior when closing laptop lids, entering sleep mode or turning off the power while the hypervisor is running but found no consistent problematic behavior. We also had a small group of UX test subjects, who didn't show any mentionable preferences except for a couple who had a slight preference for WMware Player's simpler GUI. Other comparisons pointed to the conclusion that the differences between VirtualBox and WMware are not significant performance-wise [16].

| | pros | cons |
|---|---|---|
| VMware | stable<br>backwards compatible<br>simple GUI | different client for MacOS<br>needs license |
| VirtualBox | free and open source<br>multi-platform | more complex setup<br>stability issues |

Table 2: Caption

## 2.2 Designing the initial VM

As the operating system for our virtual machines, Linux was chosen because it is open source software and allows for more customization than other major operating systems. To facilitate the distribution of the VMs and to reduce the amount of disc space they take up on student machines, we wanted the OS to be relatively small, what could be most easily achieved with a suitable Linux distribution.

To choose a distribution for our project, we reviewed the list for VirtualBox's supported Linux distributions and laid out our own criteria to select from among them. We wanted to find a distribution that would be small, lightweight, easily installed and maintained, user-friendly and open-source.

We reviewed some distributions that were designed to be particulary small, like Alpine Linux[1] and Arch Linux[2], but he initial sizes of distributions proved to be somewhat insignificant. The smaller distributions lacked more of the software packages that our chosen

[1]https://alpinelinux.org/
[2]https://www.archlinux.org/

| **Distribution** | **Pros** | **Cons** |
|---|---|---|
| Alpine Linux | can be really lightweight | non-standard libc |
| Arch Linux | can be really lightweight | challenging maintenance |
| CentOS | stable | outdated packages |
| Debian | stable, long-term support | outdated packages |
| Fedora | developer-friendly | large footprint |
| Gentoo | can be really lightweight | challenging maintenance |
| OpenSUSE | solid, stable | rapid update cycle |
| Ubuntu/Xubuntu | well-documented, beginner-friendly | large footprint |

Table 3: The Linux distributions considered for the guest operating system, along with their characteristics, pros and cons. During our project, a new version of Debian was released, resolving the problem with outdated packages.

applications depended on, and they had to be installed anyway, so that the smaller a distribution was, the more it grew with every application installed.

At first we settled for Ubuntu, mostly because it is designed to be user-friendly even for those with no previous experience with Linux, and because it is also well-documented and supported [17]. Debian, on which Ubuntu is based, was initially discarded because it didn't have support for Java 11 nor Python 3, but as a new version was released during the project these problems were fixed and Debian 10 was chosen as the operating system for our VMs.

### 2.2.1 Minimizing the VM

To further reduce the sizes of the virtual machines, irrelevant software packages which were still present in the minimal installation were mapped for removal. Some packages were replaced with a more lightweight alternative. A notable example of this is the default desktop environment, GNOME[3], which was superseded by the smaller Xfce[4]. Size and memory footprint comparison of different desktop environments are shown in table 4 [18, 19].

| Desktop environment | Installed size | Required RAM | Required CPU |
|---|---|---|---|
| GNOME | 2487 MB | 768 MB | 400 MHz |
| Unity | | 1 GB | 1 GHz |
| Cinnamon | 2212 MB | 512 MB | 1 GHz |
| MATE | 1631 MB | 512 MB | 800 MHz |
| KDE | 2198 MB | 615 MB | 1 GHz |
| Xfce | 1529 MB | 192 MB | 300 MHz |

Table 4: ...

### 2.2.2 Desktop modifications

To further accommodate the virtual machines, we decided to make some changes to the default desktop layout and to include specific desktop shortcuts for some applications, even if they could also be found from the application menu. Majority of the student group which our virtual machines are targeted to are Windows users, and to make the environment to feel more familiar to them, we removed the Xfce panel and moved the taskbar from its default position to the bottom of the screen.

[3]https://www.gnome.org/

[4]https://www.xfce.org/

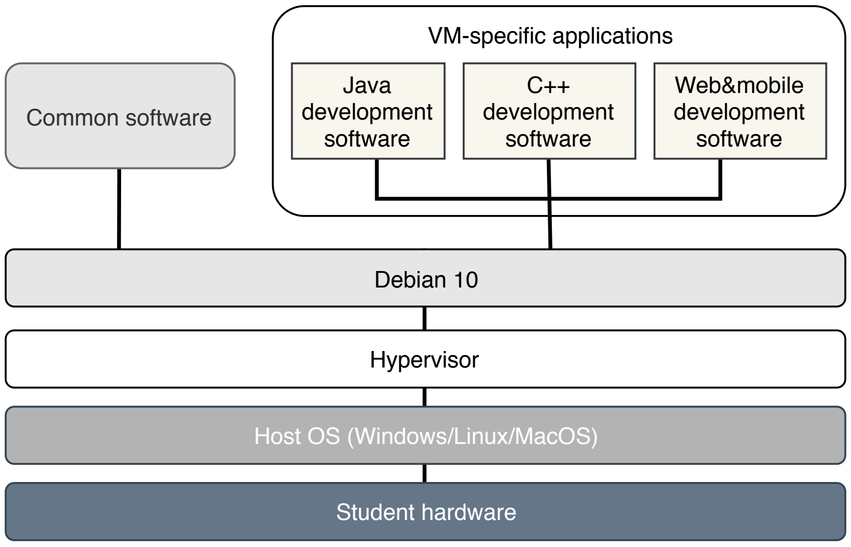

Figure 2: VM Architecture

### 2.2.3 Startup scripts

In addition to the other modifications, we wanted a script that runs when a student logs in to their virtual machine for the first time. We had some tasks planned for the script:

- Asking for localization settings so that they would be set according to the location and preferences of the student right away
- creating an SSH key for the student to use with GitLab
- setting the student's name and email as their user information in the VM's Git configuration file.

We designed two scripts, both of which run when the student uses the VM for the first time. The first script has three tasks. At first it opens a dialog where the student is prompted to enter their name, email and a password, which is used for the second task, where the script creates an SSH key that uses the given password as a passphrase and edits the Git configuration file. The SSH key is created in the student's home .shh folder so they can just paste the public key to GitLab/GitHub. Finally, the script opens the system dialogs for time zone and keyboard layout settings. The second scripts opens a dialog where the student can choose the VM language from our three options, Finnish, English and Swedish.

### 2.2.4 Seafile integration

The University of Turku uses Seafile[5] for file sharing and cloud storage. We wanted to provide students with easy acces to their Seafile libraries on the VMs and decided to include a Seafile applet with the other software. Initially, we pictured having a Seafile Drive client which maps the student's storage space on the university Seafile server as a virtual drive on the VM, automatically linking to the student's account with the login script information, but this proved to be an unstable solution.

[5]https://www.seafile.com/

### 2.2.5 Git guide

Since the students are supposed to track and store their assignment progress using distributed version control, more specifically Git, we produced a compact user guide for Git in PDF to use on the courses included in the virtualization project. To ensure that the file would be easily accessible for anyone using any of the VMs, we decided to distribute it inside the VM, located on the desktop.

### 2.2.6 Preconfigured bookmarks

To make it easier for students to find course material and other related information quickly, we decided to preset bookmarks in the browsers that were pre-installed on the VMs. Our browser of choice was Google Chrome, which stores bookmarks as a json file but exports them as a html file, which cannot be directly inserted to the virtual machine's Chrome folder. The json file could not be edited manually because it is protected by a checksum for errors. We solved this by simply copying the json bookmark file of a browser where the planned bookmarks were set and inserting it to the VM as it was.

## 2.3 Automation of installation

While creating several similar virtual machines, or replicating the creation of one VM while performing subsequental testing on the created machine, it is desirable to replicate the creation as exactly as possible. Manual installation increases the risk of mistakes and takes a lot of time [20]. To automate the install and configuration processes, we used two methods; a preseed file to install the operating system, and Ansible playbooks to automate the installation of software packages in the OS as well as to insert the configuration files and scripts mentioned above.

### 2.3.1 Debian preseed

A preseed file is used to automate the OS installation. All of the options presented during installation can be pre-answered in the preseed file, so that the installation is performed with controlled, predefined settings. Debian offers a preseed file example[6] which was used as the base for this project's preseed.

The preseed file example includes clear documentation in itself. In the file, desired lines are uncommented or added. In this project, we defined the VM's locales (language, country, keyboard, time zone), hostname, mirror settings, account settings (username, password) etc. using the preseed file so that every new VM created was identical without a need to enter all these settings manually. We also decided to modify `sudo` settings through the preseed so that the image would be better prepared for further modification. The effectual lines of our preseed file are presented in appendix B.

The preseed file can be fed to the installer in different ways. In this project, the preseed file was hosted on a web server and fed via the virtual machine's text boot prompt after inserting the Debian installation disc. The text boot prompt can be accessed by pressing ESC key while viewing the boot menu.

At the text boot prompt, the preseed is fed with the command `auto` which takes the preseed url as a parameter[7].

[6]https://www.debian.org/releases/buster/example-preseed.txt
[7]https://www.debian.org/releases/buster/armhf/apbs02.en.html#preseed-auto

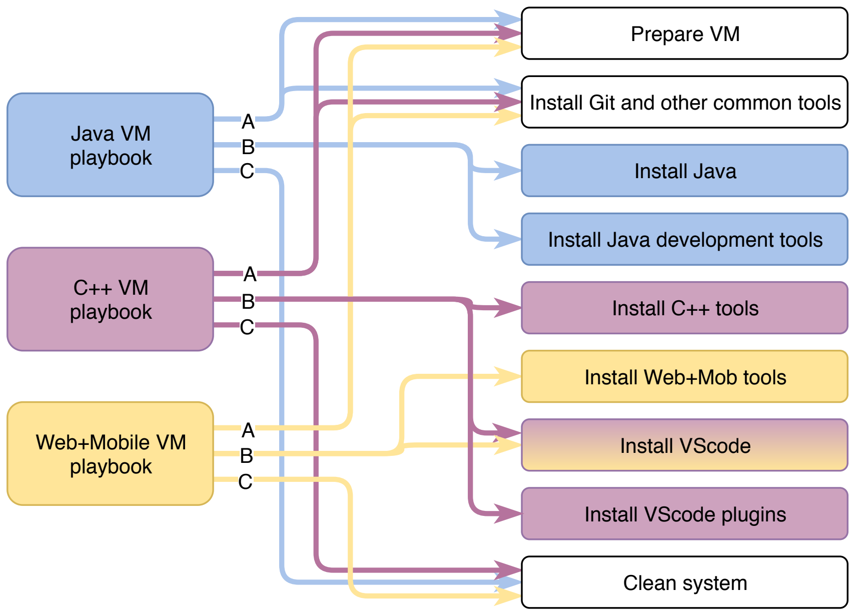

Figure 3: Each management playbook (left) consists of a set of imported playbooks (right). The import sequence for each management playbook is the same. First the preparation and common tool playbooks are imported (A), then the playbooks with specific tools (B) and finally the common cleanup playbook (C).

### 2.3.2 Ansible playbooks

After the OS installation, the virtual machine is prepared for software installations, before which the size of the virtual machine image can be somewhat reduced by removing unnecessary software packages. There are several ways to automate both the clean-up and installation processes, so that every created virtal machine is identical.

In this project, the majority of software installation and uninstallation was made using Ansible [21], which is an open-source software provisioning, configuration management, and application-deployment tool[8].

Ansible playbooks are used to modify the software installed on the virtual machine - to uninstall unnecessarities and to install desired software, like a graphical user interface, a web browser and development tools.

Since in this project we had three VM designs which had some common and some exclusive features, a modular playbook approach was used. The Ansible tasks were divided among smaller playbooks which were imported in management playbooks, which were each designed to build one of the three different VM designs (Figure 3). This way, it can be ensured that all of the common tools were installed the same way on each different machine, while the editing process is also streamlined so that an edit in one playbook takes effect in every management playbook in which it is imported.

In addition to package installations, our Ansible playbooks inserted configuration files mentioned in section 2.2.2 for some of the software. For example, the Prepare VM playbook (Appendix C.1), which installs a login manager, desktop environment and other basic appliances, also creates directories for their configuration files on the VM and copies the files to their respective folders from given locations on the host machine.

[8]https://www.ansible.com/overview/how-ansible-works

Finally, in the Clean system playbook (Appendix C.3), the bash history and other files no longer needed are removed and the startup scripts (Section 2.2.3) are inserted.

## 2.4 Testing

To ensure that the virtual machines would perform their intended purpose, we installed them on different host computers and did a variety of course assignments. No particular problems arose.

Elaborate on testing. Also add student-found problems from oom?

## 2.5 Distribution

how will we distribute the machines? compare different distribution methods in a brief way and then explain the chosen one

At the moment the VMs are distributed through the ftdev.utu.fi site

# 3 Implementation

This section contains a general description of the process assuming the tools are somewhat familiar to the reader. Some examples of the scripts, playbooks etc. used in our project are presented in the appendix as examples of how our solutions were implemented, but it is assumed that thre reader either knows how to create their own files or hass acces to our project repository where the files are stored, accompanied with additional documentation.

## 3.1 Preparations

Before building the actual virtual machine, the tools used in the process should be chosen and aquired, as well as scripts designed and written. Some tasks can be carried out with different methods, like a desktop environment can be installed through preseeding or by Ansible or some other install management tools. While designing the tools, it is advisable to form a clear picture of what will be done by which method before proceeding further.

### 3.1.1 Installing the hypervisor

As stated before, there is a variety of hypervisors available, but this guide provides instructions assuming Oracle VirtualBox is used. The installer files and instructions can be found from the VirtualBox site[9]. The installation itself is a straightforward process, but to enable USB support, among other things, there is an extension pack which needs further installing. It can be found in the download section on the VirtualBox site.

### 3.1.2 Aquiring the OS image

In this project, Debian 10 was chosen as the OS for all machines. The installation image can be downloaded from the Debian official site[10].

[9]https://www.virtualbox.org/
[10]https://www.debian.org

### 3.1.3 Preparing the preseed file

A new preseed file can be obtained from the Debian official site[11]. The file should be saved as `.cfg` and it can then be used as-is, or modified to suit varying needs. In this project, we used the preseed file for several configurements. In addition to settings presented in the initial example file, commands were added to set sudo rights and network settings, among other things. Our additions are presented in appendix B.

The preseed file can be used in three different ways, which are explained in the Debian preseeding guide[12]. In this project, we opted for a netboot method, hosting the file on a http server. Prepare the preseed file for use in your preferred method.

### 3.1.4 Writing the startup scripts

Our scripts are presented in appendix A.1 and A.2. They can be used as is or modified to suit other needs. They should be saved as `.yaml` files in a folder where they can be accessed by Ansible later. Our system cleanup playbook copies the scripts from host directory specified as `../../scripts/`, relative to where the playbooks are run, to VM directory `/etc/profile.d/` as seen in appendix C.3

### 3.1.5 Creating configuration, bookmark and other insertable files

Configuration files can be created in a VM and copied from there or written manually. They should be saved on the host computer so they can be accessed by Ansible later. Also note where the locations where the configuration files should be inserted in the VM. An example of our insertation of our Xfce configuration files can be seen in appendix C.1.

Bookmark procedure: writing a bookmark file as html and importing to chrome or bookmarking in chrome, exporting the json.

### 3.1.6 Creating Ansible playbooks

This guide doesn't go in-depth about how to create and build Ansible playbooks. Documentation and tutorials can be found from the Ansible site[13]. Three playbooks that install the basic software packages used by VMs in our project and insert their configurations are presented in appendix C.

## 3.2 Building the initial VM image

### 3.2.1 creating a virtual machine

Using the selected hypervisor, a new virtual machine is created. The setting used in this project can be seen in table 5.

If VirtualBox is used, additional network adapter settings should be configured for the VM before installing the operation system. These settings can be found the virtual machine's preference menu. In the Network section, Adapter 2 settins should be set as follows:

```
☑ Enable Network adapter
Attached to: Host-only Adapter
Name: VirtualBox Host-Only Ethernet Adapter #2
```

[11] https://www.debian.org/releases/buster/example-preseed.txt
[12] https://www.debian.org/releases/buster/armhf/apb.en.html
[13] https://www.ansible.com/overview/how-ansible-works

| | |
|---|---|
| Type | Linux |
| Version | Debian (64-bit) |
| Processors | 2 |
| Cores per processor | 1 |
| Memory size | 2048 MB |
| Network type | NAT |
| Virtual disc size | 10 GB |
| Dynamically allocated | yes |

Table 5: Virtual machine settings

### 3.2.2 Installing the operating system (preseed)

After creating a suitable virtual machine, the operating system (OS) is installed on it using the OS image aquired in section 3.1.2 and a preseed file. The image is inserted into the virtual machine which is then booted. To use a Debian preseed file for automated installation, the text boot prompt should be opened by pressing the ESC key while viewing the Debian boot menu. The preseed and its creation are explained in section 3.1.3. Using a http-hosted preseeding method, the preseed file is fed as follows[14]:

```
auto url=http://192.168.1.2/path/to/mypreseed.file
```

The url should be replaced with the actual url of the preseed file used.

### 3.2.3 VM modifications needed for VirtualBox

While using VirtualBox, certain drivers are needed to be installed separately inside the virtual machine from a guest additions disc. Through the virtual machine settings in VirtualBox, an optical drive is added and `VBoxGuestadditions.iso` inserted. Further instructions for installing the additions are provided on the VirtualBox site[15].

To enable Ansible to connect to the virtual machine, the network settings need to be modified. Inside the virtual machine, `/etc/netplan/*.yaml` should be edited so that it is as follows:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: yes
    enp0s8:
      dhcp4: yes
```

While logged in to the VM, the IP address needed in the next section can be checked from the VM's terminal with `networctl status`.

### 3.2.4 Running Ansible playbooks

Before running the playbooks, Ansible needs contact information for the virtual machine. On the host machine, a file is created in directory `/etc/ansible/` and named `hosts`, and

[14]https://www.debian.org/releases/jessie/mips/apbs02.html.en#preseed-auto
[15]https://www.virtualbox.org/manual/ch04.html#additions-linux

the address of the VM is written in the file. In this project, we used Avahi[16] and Bonjour[17], allowing us to use .local hostnames with Ansible, but an IP address works just as well. Our `hosts` file contained the VMs ID and hostname:

```
[javavm]
utuvm.local
```

Ansible connects to the virtual machine via SSH. To permit the host to connect to the VM, the host's SSH key is copied to the VM:

```
ssh-copy-id utu@utuvm.local
```

Alternatively, using an IP addess:

```
ssh-copy-id utu@xxx.xxx.xxx.xxx
```

When all preparations are done and playbooks ready for use, they can be run:

```
ansible-playbook -u utu playbook-install-xxx.yaml
```

## 3.3 Exporting and distributing the VM

Finally, the virtual machine is exported as an `OVA` file and distributed to the students by a suitable method.

## References

[1] Gerald J Popek and Robert P Goldberg. "Formal requirements for virtualizable third generation architectures". In: *Communications of the ACM* 17.7 (1974), pp. 412–421.

[2] Mendel Rosenblum and Tal Garfinkel. "Virtual machine monitors: Current technology and future trends". In: *Computer* 38.5 (2005), pp. 39–47.

[3] Dale L Lunsford. "Virtualization Technologies in Information Systems Education". English. In: *Journal of Information Systems Education* 20.3 (Oct. 2009), p. 339. URL: `https://search.proquest.com/docview/200134934`.

[4] Peng Li. "Integrating Virtualization Technology into Remote Lab: A Three- Year Experience". In: *American Society for Engineering Education.* American Society for Engineering Education. 2009.

[5] Networking Courses. *A Virtual Lab Model to Integrate Computer.* 2017.

[6] Jason Nieh and Chris Vaill. "Experiences teaching operating systems using virtual platforms and Linux". In: *ACM SIGOPS Operating Systems Review* 40.2 (2006), pp. 100–104.

[7] Tobias Kind et al. "Software platform virtualization in chemistry research and university teaching". In: *Journal of cheminformatics* 1.1 (2009), p. 18.

[8] Samuel V Angiuoli et al. "CloVR: a virtual machine for automated and portable sequence analysis from the desktop using cloud computing". In: *BMC bioinformatics* 12.1 (2011), p. 356.

[9] Florian Metze, Eric Fosler-Lussier, and Rebecca Bates. "The Speech Recognition Virtual Kitchen". In: (Aug. 2013). DOI: `10.1184/R1/6473750.v1`. URL: `https://kilthub.cmu.edu/articles/The_Speech_Recognition_Virtual_Kitchen/6473750`.

[10] Martin Dahlö et al. "Biolmg. org: A Catalog of Virtual Machine Images for the Life Sciences". In: *Bioinformatics and Biology insights* 9 (2015), BBI–S28636.

[16]https://www.avahi.org/

[17]https://support.apple.com/bonjour

[11] Mohammed Ketel. “A virtualized environment for teaching IT/CS laboratories”. In: *Proceedings of the 48th Annual Southeast Regional Conference*. ACM. 2010, p. 92.

[12] David J Malan. “From cluster to cloud to appliance”. In: *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. ACM. 2013, pp. 88–92.

[13] Andy Sayler et al. “Supporting CS education via virtualization and packages”. English. In: SIGCSE ’14. ACM, Mar. 2014, pp. 313–318. DOI: `10.1145/2538862.2538928`. URL: `http://dl.acm.org/citation.cfm?id=2538928`.

[14] *VMware – Official Site*. `https://www.vmware.com/`.

[15] *Oracle VM VirtualBox*. `https://www.virtualbox.org/`.

[16] J Horalek and T Svoboda. “Analysis of Virtualization Tools for Education Purposes”. In: *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 10.1-8 (2018), pp. 89–94.

[17] Budoor Al Housani, Bakhita Mutrib, and Hend Jaradi. “The Linux review-Ubuntu desktop edition-version 8.10”. In: *2009 International Conference on the Current Trends in Information Technology (CTIT)*. IEEE. 2009, pp. 1–6.

[18] Charles Craig. *Desktop Environments for Linux*. `https://renewablepcs.wordpress.com/about-linux/kde-gnome-or-xfce/`. Accessed: 2019-08-01.

[19] *D.2. Disk Space Needed for Tasks*. `https://www.debian.org/releases/jessie/amd64/apds02.html.en`. Accessed: 2019-08-01.

[20] Changhua Sun et al. “Simplifying Service Deployment with Virtual Appliances”. English. In: vol. 2. IEEE, July 2008, pp. 265–272. DOI: `10.1109/SCC.2008.53`. URL: `https://ieeexplore.ieee.org/document/4578533`.

[21] *Ansible is Simple IT Automation*. `https://www.ansible.com/`.

## Todo list

# A Startup Scripts

## A.1 startupScript.sh

```
#!/bin/bash
FILE=/home/utu/.firstrun.flag
SSHKEYFILE=/home/utu/.ssh/id_rsa
if [ ! -e "$FILE" ]; then
    echo Running settings script...
    ENTRY=‘zenity --forms --title="Information gathering" \
    --text="Information needed for Your new Ssh-key and git-configuration" \
    --add-entry="Your name" --add-entry="Your email address" --add-password="Password" ‘
    NAME=‘echo $ENTRY | cut -d’|’ -f1‘
    EMAIL=‘echo $ENTRY | cut -d’|’ -f2‘
    PWD=‘echo $ENTRY | cut -d’|’ -f3‘
    if ! [ -z ’$ENTRY’ ]; then
        case $? in
        0)

#Jos SSH-avainta ei ole, se tehdään
[ ! -e "$SSHKEYFILE" ] &&  ssh-keygen -P $PWD -C $EMAIL -f ~/.ssh/id_rsa -t rsa -b 4096

#modify .gitconfig
                sed -i "s/^name =.*$/name = $NAME/g" /home/utu/.gitconfig
                sed -i "s/^email =.*$/email = $EMAIL/g" /home/utu/.gitconfig
        ;;
         1)
                echo "Stop login.";;
        -1)
                echo "An unexpected error has occurred.";;
        esac
    fi
    touch $FILE
fi

if [ ! -f /home/utu/.keyboard.flag ]
then
xfce4-keyboard-settings
touch /home/utu/.keyboard.flag
fi

if [ ! -f /home/utu/.tz.flag ]
then
xfce4-terminal -e ’sudo dpkg-reconfigure tzdata’
touch /home/utu/.tz.flag
fi
```

## A.2 debianlanguage.sh

```
#!/bin/bash
if [ ! -f /home/utu/.language.flag ]
then
yad --button="English":1 --button="Suomi":2 \
--button="Svenska":3 --center --undecorated --text="Select Language"
case $? in
1)
touch /home/utu/.language.flag
if [[ ! "$LANG" = "en_US.UTF-8" ]] ; then
sudo localectl set-locale LANG=en_US.UTF-8
sudo reboot
fi ;;#&& exit 0
2)
touch /home/utu/.language.flag
      if [[ ! "$LANG" = "fi_FI.UTF-8" ]] ; then
sudo localectl set-locale LANG=fi_FI.UTF-8
sudo reboot
fi ;; #&& exit 0
3)
touch /home/utu/.language.flag
      if [[ ! "$LANG" = "sv_FI.UTF-8" ]] ; then
sudo localectl set-locale LANG=sv_FI.UTF-8
sudo reboot
fi ;; #&& exit 0
esac
fi
```

## B Debian Preseed

A condensed version of the Debian preseed file, with all but relevant lines removed.

```
d-i debian-installer/locale string en_US
d-i debian-installer/language string en
d-i debian-installer/country string FI
d-i debian-installer/locale string en_US.UTF-8
d-i localechooser/supported-locales multiselect fi_FI.UTF-8, sv_FI.UTF-8
d-i keyboard-configuration/xkb-keymap select fi
d-i netcfg/choose_interface select auto
d-i netcfg/get_hostname string utuVM
d-i netcfg/get_domain string unassigned-domain
d-i netcfg/hostname string utuVM
d-i netcfg/wireless_wep string
d-i netcfg/dhcp_hostname string utuVM
d-i mirror/country string finland
d-i mirror/http/hostname string www.nic.funet.fi
d-i mirror/http/directory string /debian
d-i mirror/http/proxy string
d-i mirror/suite string stable
d-i mirror/udeb/suite string stable
d-i passwd/root-login boolean false
d-i passwd/user-fullname string Utu Student
d-i passwd/username string utu
d-i passwd/user-password password ttlaitos
d-i passwd/user-password-again password ttlaitos
d-i clock-setup/utc boolean true
d-i time/zone string Europe/Helsinki
d-i clock-setup/ntp boolean true
d-i partman-auto/method string regular
d-i partman-lvm/device_remove_lvm boolean true
d-i partman-md/device_remove_md boolean true
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true
d-i partman-auto/choose_recipe select atomic
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
d-i partman-md/confirm boolean true
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
d-i base-installer/kernel/image string linux-image-amd64
d-i apt-setup/non-free boolean true
d-i apt-setup/contrib boolean true
tasksel tasksel/first multiselect ssh-server
d-i pkgsel/include string ncdu, avahi-daemon, open-vm-tools-desktop, curl
d-i pkgsel/upgrade select none
d-i grub-installer/only_debian boolean true
d-i grub-installer/bootdev  string /dev/sda
d-i grub-installer/bootdev  string default
d-i finish-install/reboot_in_progress note
d-i preseed/late_command string echo "%sudo ALL=(ALL:ALL) NOPASSWD:ALL"> /target/etc/sudoers.d/sudogroup; \
chmod 0440 /target/etc/sudoers.d/sudogroup; \
in-target bash -c "/usr/bin/systemctl enable avahi-daemon systemd-networkd"; \
echo "utuVM" > /target/etc/hostname; \
sed -i 's/debian/utuVM/g' /target/etc/hosts; \
echo "W01hdGNoXQpOYW1lPSoKW05ldHdvcmtdCkRIQ1A9eWVz" > /target/home/utu/.netconf; \
in-target bash -c 'base64 -d /home/utu/.netconf > /etc/systemd/network/20-wired.network'; \
in-target bash -c 'base64 -d /home/utu/.netconf > /etc/systemd/network/25-wireless.network'
```

## C Ansible playbooks

### C.1 prepare-vm.yaml

```
---
- hosts: all
  tasks:
  - name: install gpg
    become: yes
    apt:
      name: gnupg
      state: present
  - name: add key
    apt_key:
      keyserver: hkp://keyserver.ubuntu.com:80
      id: 8756C4F765C9AC3CB6B85D62379CE192D401AB61
    become: true
  - name: add repo
    become: yes
    become_method: sudo
    apt_repository:
      repo: deb http://deb.seadrive.org bionic main
      state: present
  - name: Install xfce4 and slim window manager
    become: yes
    become_method: sudo
    apt:
      update_cache: yes
      state: present
      name: "{{ item }}"
    loop:
    - xfce4
    - slim
    - xfce4-terminal
    - seadrive-gui
    - seadrive-daemon
    - zenity
    - yad
    - curl
    - jq
  - name: remove unnessassary software
    apt:
      name: "{{ item }}"
      state: absent
      purge: yes
    loop:
      - xscreensaver
      - gnome-terminal
      - ubuntu-advantage-tools
      - wireless-regdb
      - tango-icon-theme
      - friendly-recovery
      - geoip-database
      - iptables
      - installation-report
      - install-info
      - laptop-detect
      - linux-firmware
      - light-locker
    become: true
  - name: Remove useless packages from the cache
    apt:
      autoclean: yes
    become: true
  - name: create directory for xfce config files
    file:
      path: /home/utu/.config/xfce4/xfconf/xfce-perchannel-xml
      state: directory
  - name: create desktop directory
    file:
      path: /home/utu/Desktop
      state: directory
  - name: create documents directory
    file:
      path: /home/utu/Documents
      state: directory
  - name: copy background image
    become: yes
    copy:
      src: "../../xfce_config/taustakuva.jpg"
      dest: "/usr/share/backgrounds/xfce/xfce-teal.jpg"
  - name: copy config files to vm
    copy:
      src: "../../xfce_config/{{ item }}.xml"
      dest: "/home/utu/.config/xfce4/xfconf/xfce-perchannel-xml/{{ item }}.xml"
    loop:
      - keyboards
      - thunar
      - xfce4-desktop
      - xfce4-keyboard-shortcuts
      - xfce4-panel
      - xfce4-session
      - xfwm4
  - name: Remove dependencies that are no longer required
    apt:
      autoremove: yes
    become: true
  - name: create link to Documents directory
    file:
      src: /home/utu/Documents
      dest: /home/utu/Desktop/Documents
      state: link
  - name: copy git-manuals
    copy:
      src: "../../git-opas/pieni_git_opas_fi.pdf"
      dest: "/home/utu/Documents"
  - name: reconfigure login manager
    command: dpkg-reconfigure slim
    become: true
...
```

## C.2 playbook-install-git.yaml

```
---
- hosts: all
  tasks:
  - name: Install git and other system tools
    become: true
    become_method: sudo
    apt:
      name: "{{ item }}"
      state: present
      update_cache: true
    loop:
    - git
    - gitk
    - meld
    - evince
    - doxygen
    - geany
    - htop
    - iftop
    - iotop
    - gdmap

  - name: Install chromium debian
    become: true
    become_method: sudo
    apt:
      name: chromium
      state: present
    when: ansible_distribution == 'Debian'

  - name: Install chromium ubuntu
    become: true
    become_method: sudo
    apt:
      name: chromium-browser
      state: present
    when: ansible_distribution == 'Ubuntu'

  - name: create folder for chromium bookmarks
    file:
      path: /home/utu/.config/chromium/Default
      state: directory
  - name: copy bookmarks to guest system
    copy:
      src: ../../scripts/bookmarks
      dest: /home/utu/.config/chromium/Default/Bookmarks
  - name: copy gitconfig base to system
    copy:
      src: ../../scripts/git.config
      dest: /home/utu/.gitconfig
  - name: Remove useless packages from the cache
    apt:
      autoclean: yes
    become: true
  - name: Remove dependencies that are no longer required
    apt:
      autoremove: yes
    become: true
...
```

## C.3 playbook-clean-system.yaml

```
---
- hosts: all
  tasks:
  - name: clear files no longer needed
    become: yes
    become_method: sudo
    file:
      path: "{{ item }}"
      state: absent
    loop:
    - /root/.bash_history
    - /home/utu/.bash_history
    - /home/utu/.ansible/
    - /var/cache/apt/
    - /var/lib/apt/lists/
  - name: set autologin for user utu
    blockinfile:
      path: /etc/slim.conf
      block: |
        default_user  utu
        auto_login  yes
    become: true
  - name: copy firstrun script to profile.d
    copy:
      src: ../../scripts/startupScript.sh
      dest: /etc/profile.d/startupScript.sh
      owner: root
      group: root
      mode: u=rw,g=r,o=r
    become: true

  - name: Install debian language script
    become: true
    become_method: sudo
    copy:
      src: ../../scripts/debianlanguage.sh
      dest: /etc/profile.d/xlanguage.sh
      owner: root
      group: root
      mode: u=rw,g=r,o=r
    when: ansible_distribution == 'Debian'

  - name: Install ubuntu language script
    become: true
    become_method: sudo
    copy:
      src: ../../scripts/ubuntulanguage.sh
      dest: /etc/profile.d/xlanguage.sh
      owner: root
      group: root
      mode: u=rw,g=r,o=r
    when: ansible_distribution == 'Ubuntu'

  - name: reboot for changes to take effect
    become: true
    reboot:
...
```