CLOUD COMPUTING DEPLOYMENT AND MANAGEMENT

A Collection of Exercises and Tasks with Solutions

Aleš ZAMUDA

University of Maribor Press

University of Maribor

Faculty of Electrical Engineering
and Computer Science

Cloud Computing Deployment and Management

A Collection of Exercises and Tasks with Solutions

Author
Aleš Zamuda

September 2020

Title	**Cloud Computing Deployment and Management**
Subtitle	**A Collection of Exercises and Tasks with Solutions**
Author	Aleš Zamuda (University of Maribor, Faculty of Electrical Engineering and Computer Science)
Translator	Aleš Zamuda (University of Maribor, Faculty of Electrical Engineering and Computer Science)
Original title	Postavitev in upravljanje računalniških oblakov : zbirka vaj in nalog z rešitvami
Review	Matjaž Potočnik (University of Maribor, Faculty of Electrical Engineering and Computer Science)
Language editing	Shelagh Hedges
Technical editors	Aleš Zamuda (University of Maribor, Faculty of Electrical Engineering and Computer Science) Jan Perša (University of Maribor, University Press)
Computer pagination	Aleš Zamuda (University of Maribor, Faculty of Electrical Engineering and Computer Science)
Cover designer	Jan Perša (University of Maribor, University Press)
Cover graphics	Author
Graphic material	Author

Published by
University of Maribor
University Press
Slomškov trg 15, 2000 Maribor, Slovenia
https://press.um.si, zalozba@um.si

Co-published by
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, 2000 Maribor, Slovenia
https://www.feri.um.si, feri@um.si

Edition	1st
Published at	Maribor, September 2020
Publication type	E-book
Version	R1
Attachments	N04-MPI-Cluster-z05.zip, N01-Linux-VM.zip, N02-Linux-Cluster-z01.zip, 02-Linux-Cluster.zip, N02-Linux-Cluster-z02.zip, N02-Linux-Cluster-z03.zip, N02-Linux-Cluster-z04.zip, N02-Linux-Cluster-z05.zip, N03-MPI-PingPong.zip, N03-MPI-PingPong-z01.zip,N03-MPI-PingPong-z02.zip, N03-MPI-PingPong-z03.zip, N03-MPI-PingPong-z04.zip, N03-MPI-PingPong-z05.zip, N03-MPI-PingPong-z06.zip, N04-MPI-Cluster.zip), N04-MPI-Cluster-z01.zip, N04-MPI-Cluster-z02.zip, N04-MPI-Cluster-z03.zip, N04-MPI-Cluster-z04.zip, N04-MPI-Cluster-z05.zip, N04-MPI-Cluster-z06.zip, N07-Hadoop-Single.zip, N08-Hadoop-Cluster.zip, N08-Hadoop-Cluster-z01.zip, N08-Hadoop-Cluster-z02.zip.
System requirements	Computer compatible with IBM PC models, operating system Linux, Mac, or Windows, 8 GB RAM; internet access
Software requirements	Internet browser, program Okular or Adobe Reader, Bash, VirtualBox
Dostopno na	https://press.um.si/index.php/ump/catalog/book/499

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004(0.034.2)

ZAMUDA, Aleš
    Cloud computing deployment and management [Elektronski vir] : a collection of exercises and tasks with solutions / author Aleš Zamuda. - 1st ed. - E-knjiga. - Maribor : Univerzitetna založba Univerze, 2020

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/499
ISBN 978-961-286-371-5
doi: doi.org/10.18690/978-961-286-371-5
1. Drugi var. nasl.
COBISS.SI-ID 27208195

ISBN 978-961-286-371-5 (pdf)

DOI https://doi.org/10.18690/978-961-286-371-5

Price Free copy

For publisher prof. dr. Zdravko Kačič,
Rector of University of Maribor

Cloud Computing Deployment and Management

A Collection of Exercises and Tasks with Solutions

Aleš Zamuda

University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia. E-mail: ales.zamuda@um.si

Abstract The subject Cloud Computing Deployment and Management is included in the second-cycle Bologna Study Programme Computer Science and Information Technologies as a full-time study unit. This document presents study material for computer exercises in this subject. After an introduction, then individual tasks with solutions to the computer exercises in the subsequent chapters are provided. This is followed by a list of archived files containing the computer format of the provided solutions.

Keywords: cloud computing; virtualization; parallel programming; distributed programming; virtual computing environments; high-performance computing

University of Maribor Press

DOI https://doi.org/10.18690/978-961-286-371-5
ISBN 978-961-286-371-5

Active Table of Contents

Active Term (Index) Table

Apache Apache Web Server: A program that serves web requests to clients. 46, 54

Apache Hadoop Open source software package for reliable, scalable, and distributed computing. 8, 46–48, 50, 52, 54, 55, 59, 61

APT Advanced Package Tool. 17

CD-ROM Compact Disc Read-Only Memory. 7

DHCP Dynamic Host Configuration Protocol. 19, 20, 22, 55

DVD Digital Versatile Disc. 7

HDD Hard Disk Drive. 6

HDFS Hadoop Distributed File System. 48, 50

IP Internet Protocol. 18–21, 31, 54, 55

ISO image International Organisation for Standardisation Disk Image. 5, 7

MPI Message Passing Interface. 34

NAT Network Address Translation. 27, 35, 46

OpenMPI A computer tool for deploymnet and management of communication interface among processes on different computers, using messaging, MPI denotes Message Passing Interface. 26, 28–31, 34, 35, 43

OpenSSH A computer tool for deployment and management of secure remote shell logins, denotes OpenBSD Secure Shell, where BSD stands for Berkeley Software Distribution.. 4, 7, 8, 14, 15, 20, 21, 23, 24, 28, 35, 46, 48, 52, 55

OVA Open Virtualisation Format. 5, 8, 16, 25, 27, 32, 34, 43, 46, 52, 54, 62, 63

PNG Portable Network Graphics. 7

PSSH Parallel SSH. 7, 17–19, 24

RSA Rivest-Shamir-Adleman. 21, 24

SATA Serial Advanced Technology Attachment. 6

SSH Secure Shell. 24

X server X Window System. 7, 16, 17

VDI Virtual Disk Image. 6

VirtualBox A computer tool for deployment and management of virtual environments of computers. 4, 7, 18, 19

YARN Yet Another Resource Negotiator. 8, 51, 52, 54, 57, 59, 60

Active Table of Figures

Foreword

The subject Cloud Computing Deployment and Management is included in the second-cycle Bologna Study Programme Computer Science and Information Technologies as a full-time study unit. This document presents study material for computer exercises in this subject. The next page is followed by an Introduction, then by individual tasks with solutions to the computer exercises in the subsequent chapters. This is followed by a list of archived files containing the computer format of the provided solutions.

Thanks

Thanks to the expert reviewer, language reviewer, the e-publishing commission, and the publisher of this work. In preparing the study materials, I would also like to thank the Heads of the Laboratories of the Institute of Computer Science, as well as the students of the Study Programme Computer Science and Information Technologies, who helped guide the preparation of these materials.

Special thanks for supporting participation in the action IC1406: *High-Performance Modelling and Simulation for Big Data Applications (cHiPSet)*, which was implemented within European Cooperation in Science & Technology (COST). Within this action, we exchanged views, experiences and good practices in teaching cloud computing from several universities in Europe, collected by Dave Freenan under the leadership of Joanna Kolodziej, and available with other materials on the website `http://chipset-cost.eu`.

Introduction — General Information about the Subject, Exercises, and Tasks

The study unit of the course Cloud Computing Deployment and Management (CCDM) consists of lectures, tutorials, computer exercises, and independent student work. The list below includes some basic information about the course, followed by content overview for computer exercises.

- **Lecturer:** Asst. Prof. Dr. Aleš Zamuda.
- **E-mail:** ales.zamuda@um.si.
- **Office hours:** `https://www.um.si/univerza/imenik-zaposlenih/Strani/imenik.aspx`. Office hours are **weekly** and it is also possible to ask for advice regarding matters of this course. To request a time slot reservation, contact the lecturer via e-mail.
- **Course description in the UM catalog**: published at `https://aips.um.si/PredmetiBP5/UcnaEnotaInfo.asp?UEID=25860&Leto=2020&Jezik=A`.
- **Lectures:** 30 hours (2 hours weekly), might be reduced for English.
- **Computer exercises:** 28 hours (2 hours per week, by groups).
- **Seminars:** 2 hours (initial week, combined).
- **Schedule:** published at `https://feri.um.si/studij/urniki/`.
- **Dates for tests:** Scheduled dates for tests are published on the website of the Institute of Computer Science: `https://cs.feri.um.si/za-studente/vmesni-izpiti/`.
- **Exam dates:** Published at AIPS: `https://aips.um.si/en/IzpitniRokiEN.aspx`.
- **Evaluation rules:** The website of the Institute of Computer Science publishes these as: *Ocenjevanje pri predmetih in projektih*, `https://cs.feri.um.si/site/assets/files/1037/ocenjevanje_pri_predmetih_in_projektih-2018-2019.pdf`.

Content Overview

For computer exercises the content is provided initially at the lectures, for which slides are also available, i.e., the subject content is first lectured. Through lab work at seminars with a computer, the content from the lectures is then mastered in more detail using the practical examples of exercises. Web links to a computer classroom (e-study), which is accessible to online users, are also listed.

Slides from lectures (in English)
https://estudij.um.si/mod/resource/view.php?id=197935

- Example exercise: VirtualBox installation and SSH user accounts in Linux.

  Exercise N01-Linux-VM
  https://estudij.um.si/mod/assign/view.php?id=163010

- Example exercise: Cloning virtual environments, cluster with VirtualBox.

  Exercise N02-Linux-Cluster
  https://estudij.um.si/mod/assign/view.php?id=162992

- Example exercise: Cluster network settings and parallel configuration management.

  Exercise N03-MPI-PingPong
  https://estudij.um.si/mod/assign/view.php?id=163046

- Example exercise: MPI installation, MPI_Send and MPI_Receive code.

  Exercise N04-MPI-Cluster
  https://estudij.um.si/mod/assign/view.php?id=163036

- Example exercise: MapReduce with Hadoop and YARN on 1 computer.

  Exercise N07-Hadoop-Single
  https://estudij.um.si/mod/assign/view.php?id=139826

- Example exercise: Multiple nodes for MapReduce with Hadoop and YARN, and use of distributed HDFS storage with Hadoop in the cloud.

  Exercise N08-Hadoop-Cluster
  https://estudij.um.si/mod/assign/view.php?id=139831

Chapter 1

Virtual Environments — Exercise N01-Linux-VM

Instructions

Create one instance of a virtual environment VirtualBox (`https://www.virtualbox.org/`) on one physical host computer.

Then install Linux with an OpenSSH server (`https://www.openssh.com/`) and create OpenSSH access keys for one remote user.

Then show that you can connect to this hosted virtual environment using an OpenSSH client from the host computer.

Submit a report about this exercise that demonstrates execution of the solution, its implementation, and scripts/code.

FOR HALF POINTS: Without client demonstration for OpenSSH

Material for hints during the explanation of the exercise:

`https://help.ubuntu.com/community/SSH/OpenSSH/Keys`
`https://www.virtualbox.org/`
`https://www.ubuntu.com/download/server`
`http://ftp.arnes.si/pub/mirrors/ubuntu-cdimage/18.04.2/ubuntu-18.04.2-live-server-amd64.iso` (without X server)
`https://www.alpinelinux.org/about/`
`http://www.damnsmalllinux.org/`

Solution

When setting up computer systems, it is good to use procedures that can be upgraded and automated later. One such is the use of a shell language for commands to an operating system such as Linux, i.e., the Bash Command Interpreter. Hence, the Bash Command Interpreter is applied in the continuation of this solution.

Attachment file of the solution: Computer virtual image in Open Virtualisation Format (OVA)

N01-Linux-VM.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`

First, install the VirtualBox software on the computer, which can be found on the website `https://www.virtualbox.org/`. In the continuation of this solution (here, it is synced with the Slovenian materials edition), we assume that we use the Linux Ubuntu 18.04.2 LTS distribution as a host system for VirtualBox equipment, which can be installed on both personal and server computers. Both VirtualBox and Ubuntu distribution can be used free of charge.[1] The implementation steps of this process are provided in the following.

1. To download the Ubuntu installation image file in the shell, run the command `wget`; the result of the command is a downloaded International Organisation for Standardisation Disk Image (ISO image):

```
$ wget http://old-releases.ubuntu.com/releases
    ↪ /18.04.2/ubuntu-18.04.2-live-server-amd64.
    ↪ iso
```

```
--2020-05-12 09:57:23-- http://old-releases.ubuntu.com/releases/18.04.2/ubuntu-18.04.2-live-
      ↪ server-amd64.iso
Resolving old-releases.ubuntu.com (old-releases.ubuntu.com)... 91.189.88.153, 2001:67c
      ↪ :1360:8001::25
Connecting to old-releases.ubuntu.com (old-releases.ubuntu.com)|91.189.88.153|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 874512384 (834M) [application/x-iso9660-image]
Saving to: 'ubuntu-18.04.2-live-server-amd64.iso'

ubuntu-18.04.2-live-server-amd64.iso 100%[=================================================>]
      ↪ 834.00M 32.8MB/s in 29s

2020-05-12 09:57:53 (28.6 MB/s) - 'ubuntu-18.04.2-live-server-amd64.iso' saved
      ↪ [874512384/874512384]
```

[1] The task can also be solved by using another or more recent Linux distribution or using Unix systems, for example macOS (`http://apple.com/macos/`) or use one of the Windows (`https://www.microsoft.com/sl-si/windows`) — however, in these cases, the License to use may also require payment.

2. To be able to use the image, install VirtualBox:

```
$ sudo apt−get install virtualbox virtualbox−
    ↪ guest−additions−iso
```

3. Create a new VirtualBox environment and name it `N01-Linux-VM`:

```
$ VBoxManage createvm −−name N01−Linux−VM −−
    ↪ ostype Ubuntu_64 −−register
```

```
Virtual machine 'N01−Linux−VM' is created and registered.
UUID: ab4b145d−8f14−4e33−a39f−973d01c8cf14
Settings file: '/home/ales/.VirtualBox/Machines/N01−Linux−VM/N01−Linux−VM
    ↪ .vbox'
```

4. For this new environment N01-Linux-VM set the memory size to 4GB:

```
$ VBoxManage modifyvm N01−Linux−VM −−memory 4096
```

5. For this new environment N01-Linux-VM, create a Virtual Disk Image (VDI) type disk image with a size of 8GB and save it:

```
$ VBoxManage createhd −−filename ~/.VirtualBox/
    ↪ Machines/N01−Linux−VM/N01−Linux−VM.vdi −−
    ↪ size 8000 −−format VDI
```

```
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Medium created. UUID: 66102b9d−228e−43c1−b5ab−144d92e8a0d1
```

6. Create a Serial Advanced Technology Attachment (SATA) controller for virtual environments:

```
$ VBoxManage storagectl N01−Linux−VM −−name "SATA
    ↪  Controller" −−add sata −−controller
    ↪ IntelAhci
```

7. Attach the SATA controller to the first port of the first device as a disk type Hard Disk Drive (HDD) and connect the saved VDI disk image to it:

```
$ VBoxManage storageattach N01−Linux−VM −−
    ↪ storagectl "SATA Controller" −−port 0 −−
    ↪ device 0 −−type hdd −−medium ~/.VirtualBox/
    ↪ Machines/N01−Linux−VM/N01−Linux−VM.vdi
```

8. Add a controller for the CD player — Compact Disc Read-Only Memory (CD-ROM):

```
$ VBoxManage storagectl N01-Linux-VM --name "IDE
   ↪ Controller" --add ide --controller PIIX4
```

9. Connect the CD controller to the other port of the first device as a type Digital Versatile Disc (DVD), and insert the downloaded Ubuntu file into the device, stored as an ISO image:

```
$ VBoxManage storageattach N01-Linux-VM --
   ↪ storagectl "IDE Controller" --port 1 --device
   ↪  0 --type dvddrive --medium ~/N01-Linux-VM
   ↪ /ubuntu-18.04.2-live-server-amd64.iso
```

10. Then run the configured image of the virtual environment in VirtualBox and hence put the Ubuntu installation image in a running state.

```
$ VBoxManage startvm N01-Linux-VM
Waiting for VM "N01-Linux-VM" to power on...
VM "N01-Linux-VM" has been successfully started.
```

An alternative solution for steps 1–10: Here, the exercise can also be solved interactively with the help of a Window System X server, where the special use of a mouse input device is required, which can be a bit slower if we want to learn an automated approach to setting up and deploying virtual environments. In this case, we use the steps as indicated in the screenshots seen in Figures 1.2–1.4. We use the default settings, and change only the memory size and set the image name.

11. Install the Ubuntu operating system in a newly run virtual environment state with default settings (Figures 1.6–1.11). Before that, we prepare the program to capture raster images for the X Window System (X server), which we will use to produce a report document. We save the images in a file that is in Portable Network Graphics (PNG) format, using the command `spectacle`:

```
$ spectacle -u -b -o "N01-Linux-VM-$(date).png"
```

To make it easier to use the spectacle command immediately while working in a running virtual environment, connect a keyboard shortcut `Meta+P` to the trigger of this command in the program `ksystemsettings`, as shown in Figure 1.1.

12. Inside the Ubuntu hosted system, use the key generation process OpenSSH from https://help.ubuntu.com/community/SSH/OpenSSH/Keys (Figure 1.12):

```
$ mkdir ~/.ssh
$ chmod 700 ~/.ssh
$ ssh−keygen −t rsa
```

13. Save the virtual environment and put it in a saved suspended state:

```
$ vboxmanage controlvm N01−Linux−VM savestate
```

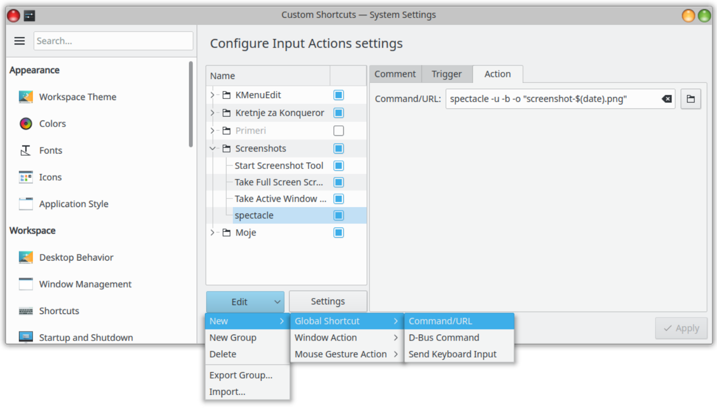
14. Export the OVA image for the created environment.

```
$ vboxmanage export N01−Linux−VM −o N01−Linux−VM
   ↪ .ova
```

Executing this command prints the following result (produced image OVA is about the size of 0.8 GB):

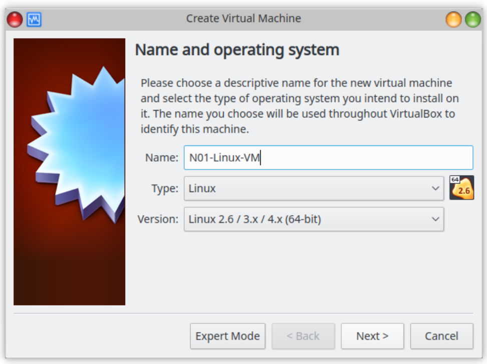
```
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully exported 1 machine(s).
```

15. Write down an exercise report (for example, in the same manner that the solution was described here in this material).



Command/URL: spectacle -u -b -o "screenshot-$(date).png"

Figure 1.1: Setting the automatic screen capture button for continuous capturing of screenshots to a file on demand, in the program `ksystemsettings`.


Create Virtual Machine

Name and operating system

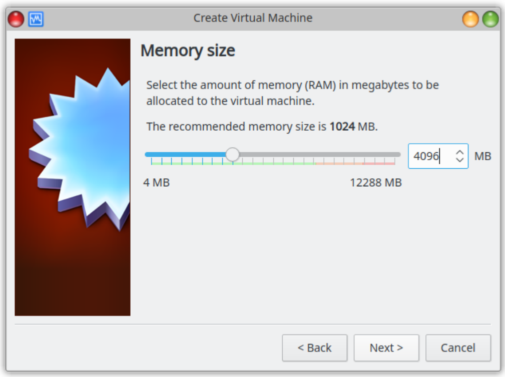
Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name: N01-Linux-VM

Type: Linux

Version: Linux 2.6 / 3.x / 4.x (64-bit)

(a) Step 01-1.


Create Virtual Machine

Memory size

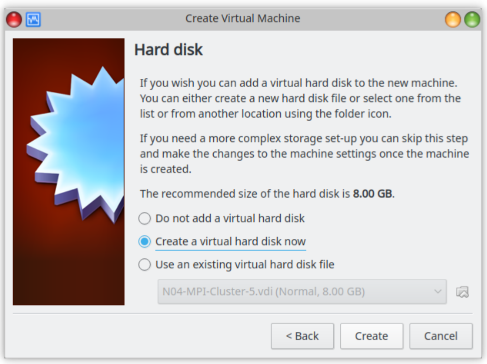
Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024** MB.

4096 MB

4 MB 12288 MB

(b) Step 01-2.


Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

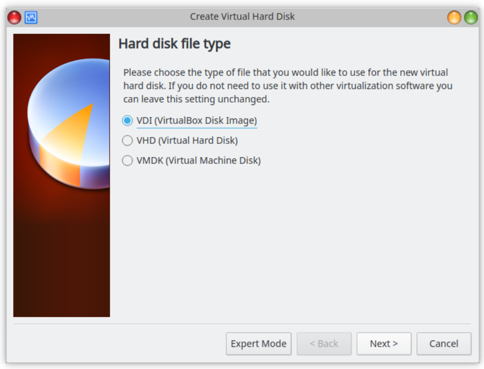
The recommended size of the hard disk is **8.00 GB**.

Do not add a virtual hard disk [✓ Create a virtual hard disk now] Use an existing virtual hard disk file

N04-MPI-Cluster-5.vdi (Normal, 8.00 GB)

(c) Step 01-3.

Figure 1.2: Installing the VirtualBox virtual environment deployment and management tool (interactive steps 01-1 to 01-3).

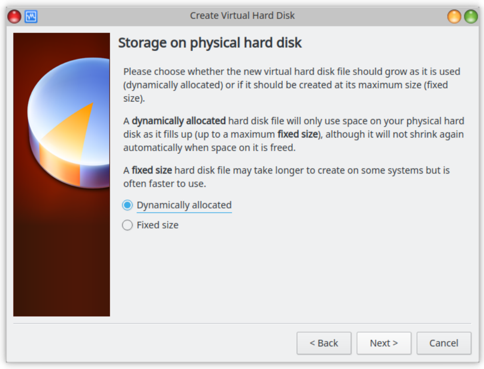

Create Virtual Hard Disk

Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

[✓ VDI (VirtualBox Disk Image)] VHD (Virtual Hard Disk) VMDK (Virtual Machine Disk)

(a) Step 01-4.


Create Virtual Hard Disk

Storage on physical hard disk

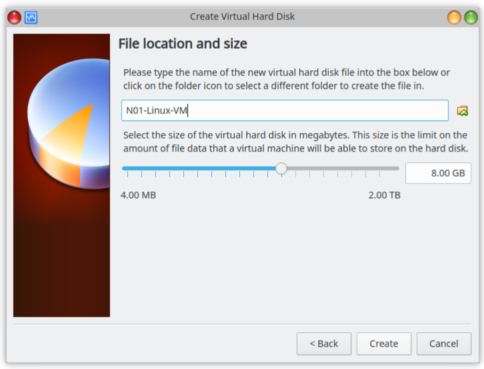
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

[✓ Dynamically allocated] Fixed size

(b) Step 01-5.


Create Virtual Hard Disk

File location and size

Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

N01-Linux-VM

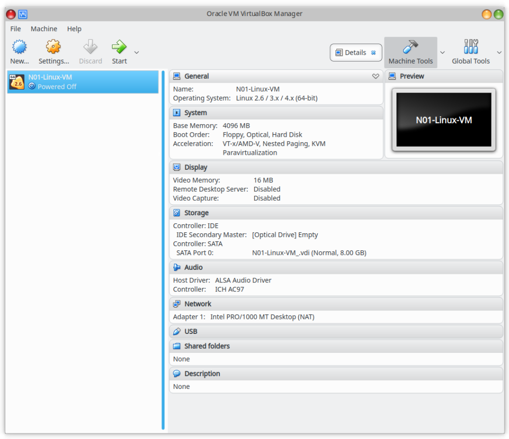
Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

8.00 GB

4.00 MB 2.00 TB

(c) Step 01-6.

Figure 1.3: Installing the VirtualBox virtual environment deployment and management tool (interactive steps 01-4 to 01-6).



General

```
Name:              N01-Linux-VM
Operating System:  Linux 2.6 / 3.x / 4.x (64-bit)
```

System

```
Base Memory:   4096 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  VT-x/AMD-V, Nested Paging, KVM
               Paravirtualization
```

Display

```
Video Memory:           16 MB
Remote Desktop Server:  Disabled
Video Capture:          Disabled
```

Storage

```
Controller: IDE
 IDE Secondary Master:  [Optical Drive] Empty
Controller: SATA
 SATA Port 0:           N01-Linux-VM_.vdi (Normal, 8.00 GB)
```

Audio

```
Host Driver:  ALSA Audio Driver
Controller:   ICH AC97
```

Network

```
Adapter 1:  Intel PRO/1000 MT Desktop (NAT)
```

USB

Shared folders

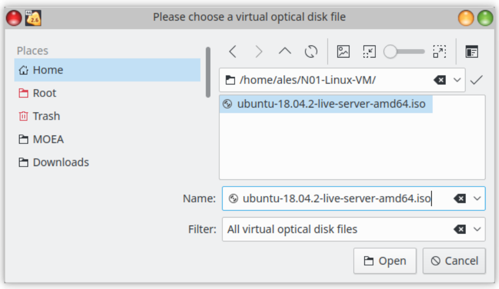
None

Description

None

(a) Step 01-7.

Figure 1.4: Installing the VirtualBox virtual environment deployment and management tool (interactive step 01-7). The final window is displayed with the settings of the new virtual environment N01-Linux-VM.



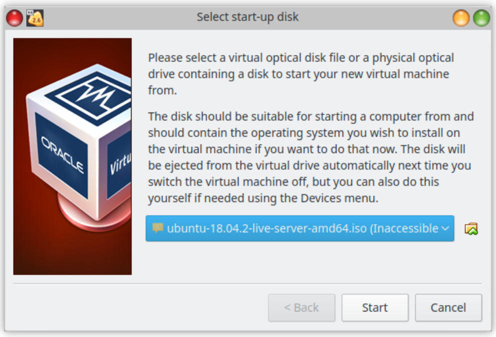
/home/ales/N01-Linux-VM/

ubuntu-18.04.2-live-server-amd64.iso

Name: ubuntu-18.04.2-live-server-amd64.iso

Filter: All virtual optical disk files

(a) Step 03-1.

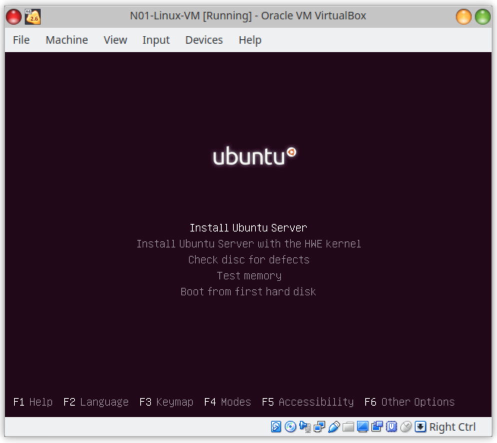


Please select a virtual optical disk file or a physical optical drive containing a disk to start your new virtual machine from.

The disk should be suitable for starting a computer from and should contain the operating system you wish to install on the virtual machine if you want to do that now. The disk will be ejected from the virtual drive automatically next time you switch the virtual machine off, but you can also do this yourself if needed using the Devices menu.

ubuntu-18.04.2-live-server-amd64.iso (Inaccessible

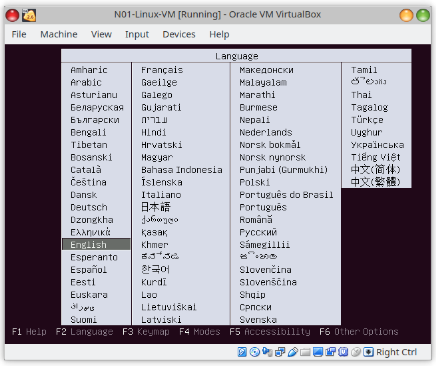
(b) Step 03-2.



```
Install Ubuntu Server
Install Ubuntu Server with the HWE kernel
Check disc for defects
Test memory
Boot from first hard disk

F1 Help  F2 Language  F3 Keymap  F4 Modes  F5 Accessibility  F6 Other Options
```

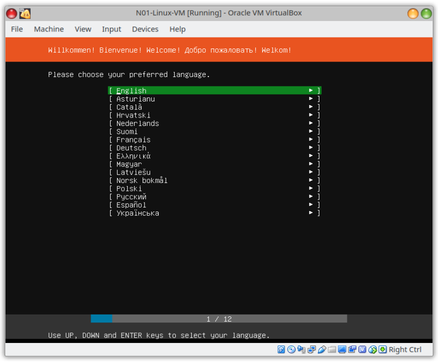
(c) Step 03-3.

Figure 1.5: Creating a new virtual environment in VirtualBox to prepare for Ubuntu installation (interactive steps 03-1 to 03-3).


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

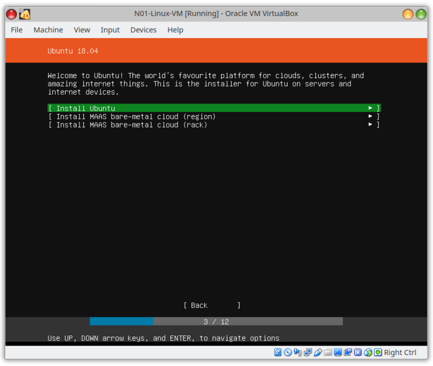
```
                                  Language
 Amharic       Français           Македонски             Tamil
 Arabic        Gaeilge            Malayalam              తెలుగు
 Asturianu     Galego             Marathi                Thai
 Беларуская    Gujarati           Burmese                Tagalog
 Български     עברית              Nepali                 Türkçe
 Bengali       Hindi              Nederlands             Uyghur
 Tibetan       Hrvatski           Norsk bokmål           Українська
 Bosanski      Magyar             Norsk nynorsk          Tiếng Việt
 Català        Bahasa Indonesia   Punjabi (Gurmukhi)     中文(简体)
 Čeština       Íslenska           Polski                 中文(繁體)
 Dansk         Italiano           Português do Brasil
 Deutsch       日本語              Português
 Dzongkha      ქართული            Română
 Ελληνικά      Қазақ              Русский
 English       Khmer              Sámegillii
 Esperanto     ಕನ್ನಡ              සිංහල
 Español       한국어              Slovenčina
 Eesti         Kurdî              Slovenščina
 Euskara       Lao                Shqip
 فارسی         Lietuviškai        Српски
 Suomi         Latviski           Svenska
F1 Help  F2 Language  F3 Keymap  F4 Modes  F5 Accessibility  F6 Other Options
```

(a) Step 04-1.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
    Willkommen! Bienvenue! Welcome! Добро пожаловать! Welkom!

    Please choose your preferred language.

                  [ English                              ▸ ]
                  [ Asturianu                            ▸ ]
                  [ Català                               ▸ ]
                  [ Hrvatski                             ▸ ]
                  [ Nederlands                           ▸ ]
                  [ Suomi                                ▸ ]
                  [ Français                             ▸ ]
                  [ Deutsch                              ▸ ]
                  [ Ελληνικά                             ▸ ]
                  [ Magyar                               ▸ ]
                  [ Latviešu                             ▸ ]
                  [ Norsk bokmål                         ▸ ]
                  [ Polski                               ▸ ]
                  [ Русский                              ▸ ]
                  [ Español                              ▸ ]
                  [ Українська                           ▸ ]

                                1 / 12

    Use UP, DOWN and ENTER keys to select your language.
```

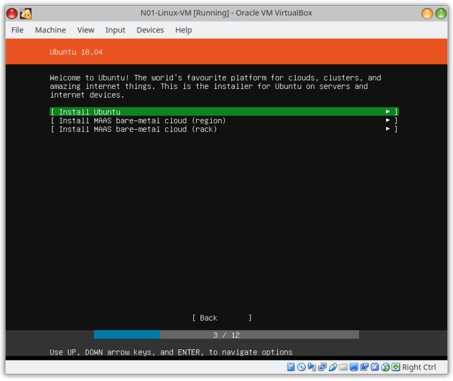
(b) Step 04-2.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
    Ubuntu 18.04

    Welcome to Ubuntu! The world's favourite platform for clouds, clusters, and
    amazing internet things. This is the installer for Ubuntu on servers and
    internet devices.

    [ Install Ubuntu                                                        ▸ ]
    [ Install MAAS bare-metal cloud (region)                                ▸ ]
    [ Install MAAS bare-metal cloud (rack)                                  ▸ ]

                              [ Back         ]

                                3 / 12

    Use UP, DOWN arrow keys, and ENTER, to navigate options
```

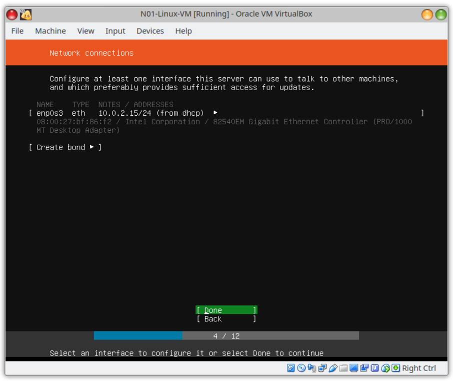
(c) Step 04-3.

Figure 1.6: Installing Ubuntu in a virtual environment (interactive steps 04-1 to 04-3).


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

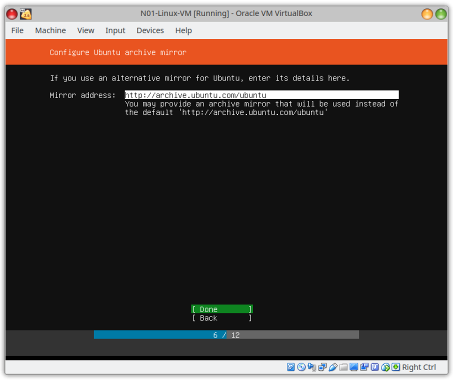
```
    Ubuntu 18.04

    Welcome to Ubuntu! The world's favourite platform for clouds, clusters, and
    amazing internet things. This is the installer for Ubuntu on servers and
    internet devices.

    [ Install Ubuntu                                                        ▸ ]
    [ Install MAAS bare-metal cloud (region)                                ▸ ]
    [ Install MAAS bare-metal cloud (rack)                                  ▸ ]

                              [ Back         ]

                                3 / 12

    Use UP, DOWN arrow keys, and ENTER, to navigate options
```

(a) Step 04-4.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
    Network connections

    Configure at least one interface this server can use to talk to other machines,
    and which preferably provides sufficient access for updates.

   NAME    TYPE  NOTES / ADDRESSES
 [ enp0s3  eth   10.0.2.15/24 (from dhcp)  ▸                                       ]
   08:00:27:bf:86:f2 / Intel Corporation / 82540EM Gigabit Ethernet Controller (PRO/1000
   MT Desktop Adapter)

 [ Create bond ▸ ]

                              [ Done         ]
                              [ Back         ]

                                4 / 12

    Select an interface to configure it or select Done to continue
```

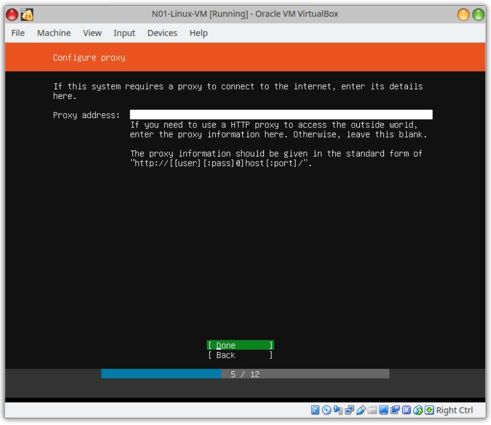
(b) Step 04-5.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
    Configure Ubuntu archive mirror

    If you use an alternative mirror for Ubuntu, enter its details here.

    Mirror address:  http://archive.ubuntu.com/ubuntu
                     You may provide an archive mirror that will be used instead of
                     the default 'http://archive.ubuntu.com/ubuntu'

                              [ Done         ]
                              [ Back         ]

                                6 / 12
```

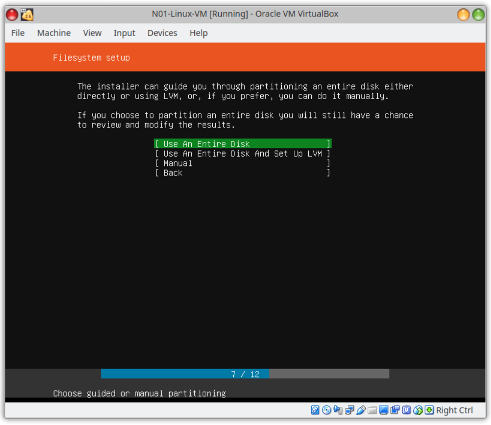
(c) Step 04-6.

Figure 1.7: Installing Ubuntu in a virtual environment (interactive steps 04-4 to 04-6).


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

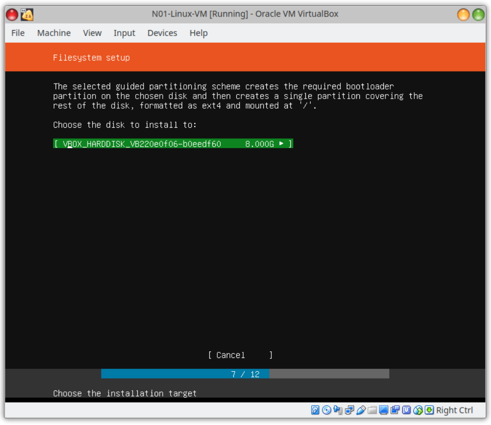
```
Configure proxy

If this system requires a proxy to connect to the internet, enter its details
here.

Proxy address: [                                                  ]
               If you need to use a HTTP proxy to access the outside world,
               enter the proxy information here. Otherwise, leave this blank.

               The proxy information should be given in the standard form of
               "http://[[user][:pass]@]host[:port]/".


                                [ Done      ]
                                [ Back      ]

                                 5 / 12
```

(a) Step 04-7.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Filesystem setup

    The installer can guide you through partitioning an entire disk either
    directly or using LVM, or, if you prefer, you can do it manually.

    If you choose to partition an entire disk you will still have a chance
    to review and modify the results.

                [ Use An Entire Disk                 ]
                [ Use An Entire Disk And Set Up LVM ]
                [ Manual                             ]
                [ Back                               ]

                                 7 / 12

Choose guided or manual partitioning
```

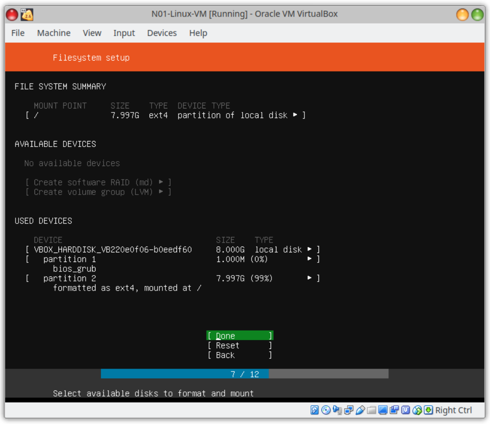
(b) Step 04-8.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Filesystem setup

The selected guided partitioning scheme creates the required bootloader
partition on the chosen disk and then creates a single partition covering the
rest of the disk, formatted as ext4 and mounted at '/'.

Choose the disk to install to:

[ VBOX_HARDDISK_VB220e0f06-b0eedf60      8.000G ► ]


                                [ Cancel    ]

                                 7 / 12

Choose the installation target
```

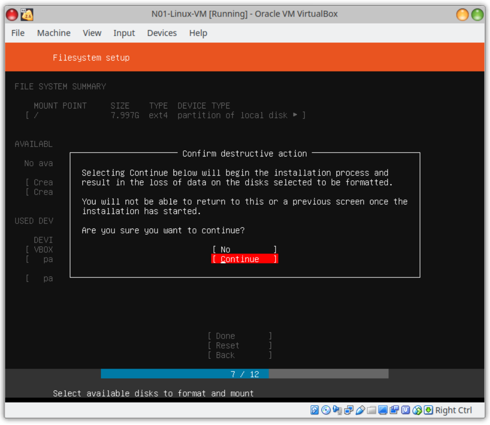
(c) Step 04-9.

Figure 1.8: Installing Ubuntu in a virtual environment (interactive steps 04-7 to 04-9).


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

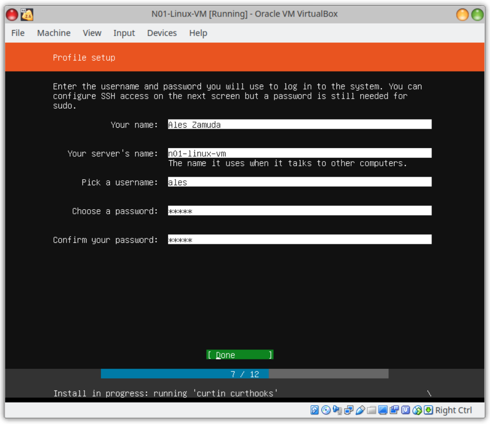
```
Filesystem setup

FILE SYSTEM SUMMARY

    MOUNT POINT     SIZE    TYPE  DEVICE TYPE
  [ /               7.997G  ext4  partition of local disk ► ]


AVAILABLE DEVICES

  No available devices

  [ Create software RAID (md) ► ]
  [ Create volume group (LVM) ► ]


USED DEVICES

    DEVICE                                 SIZE    TYPE
  [ VBOX_HARDDISK_VB220e0f06-b0eedf60     8.000G  local disk ► ]
  [   partition 1                         1.000M (0%)        ► ]
        bios_grub
  [   partition 2                         7.997G (99%)       ► ]
        formatted as ext4, mounted at /


                                [ Done      ]
                                [ Reset     ]
                                [ Back      ]

                                 7 / 12

Select available disks to format and mount
```

(a) Step 04-10.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Filesystem setup

FILE SYSTEM SUMMARY

    MOUNT POINT     SIZE    TYPE  DEVICE TYPE
  [ /               7.997G  ext4  partition of local disk ► ]


AVAILABL   ┌──────────────── Confirm destructive action ────────────────┐
           │                                                            │
  No ava   │ Selecting Continue below will begin the installation process and
           │ result in the loss of data on the disks selected to be formatted.
  [ Crea   │                                                            │
  [ Crea   │ You will not be able to return to this or a previous screen once the
           │ installation has started.                                  │
USED DEV   │                                                            │
           │ Are you sure you want to continue?                         │
    DEVI   │                                                            │
  [ VBOX   │                      [ No        ]                         │
  [   pa   │                      [ Continue  ]                         │
           │                                                            │
  [   pa   └────────────────────────────────────────────────────────────┘


                                [ Done      ]
                                [ Reset     ]
                                [ Back      ]

                                 7 / 12

Select available disks to format and mount
```

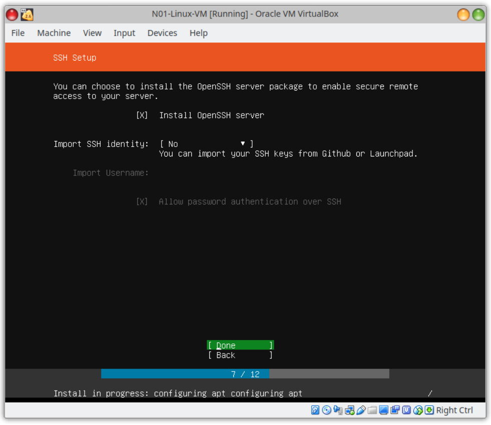
(b) Step 04-11.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Profile setup

Enter the username and password you will use to log in to the system. You can
configure SSH access on the next screen but a password is still needed for
sudo.

             Your name: Ales Zamuda

      Your server's name: n01-linux-vm
                          The name it uses when it talks to other computers.

        Pick a username: ales

       Choose a password: *****

   Confirm your password: *****


                                [ Done      ]

                                 7 / 12

Install in progress: running 'curtin curthooks'                              \
```

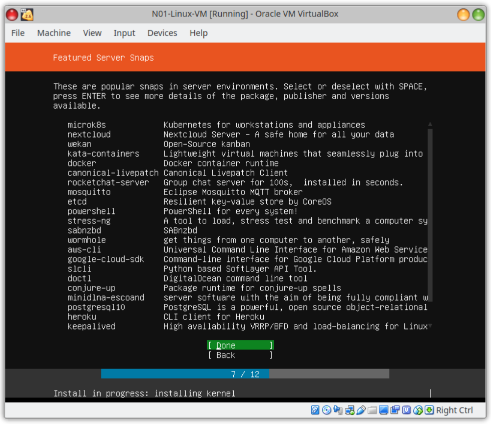
(c) Step 04-12.

Figure 1.9: Installing Ubuntu in a virtual environment (interactive steps 04-10 to 04-12).


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

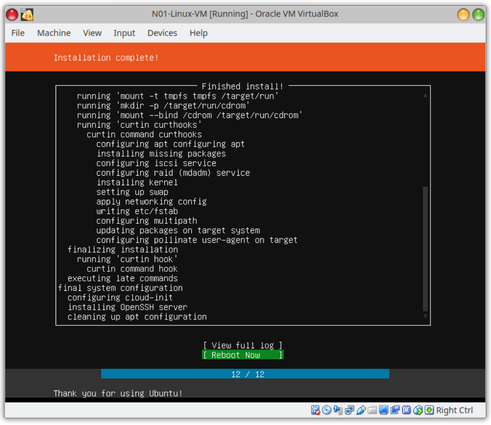
```
SSH Setup

You can choose to install the OpenSSH server package to enable secure remote
access to your server.

                  [X]  Install OpenSSH server


Import SSH identity:  [ No               ▾ ]
                      You can import your SSH keys from Github or Launchpad.

    Import Username:


                  [X]  Allow password authentication over SSH



                            [ Done       ]
                            [ Back       ]

                               7 / 12

Install in progress: configuring apt configuring apt                          /
```

(a) Step 04-13.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Featured Server Snaps

These are popular snaps in server environments. Select or deselect with SPACE,
press ENTER to see more details of the package, publisher and versions
available.

   microk8s            Kubernetes for workstations and appliances
   nextcloud           Nextcloud Server - A safe home for all your data
   wekan               Open-Source kanban
   kata-containers     Lightweight virtual machines that seamlessly plug into
   docker              Docker container runtime
   canonical-livepatch Canonical Livepatch Client
   rocketchat-server   Group chat server for 100s,  installed in seconds.
   mosquitto           Eclipse Mosquitto MQTT broker
   etcd                Resilient key-value store by CoreOS
   powershell          PowerShell for every system!
   stress-ng           A tool to load, stress test and benchmark a computer sy
   sabnzbd             SABnzbd
   wormhole            get things from one computer to another, safely
   aws-cli             Universal Command Line Interface for Amazon Web Service
   google-cloud-sdk    Command-line interface for Google Cloud Platform produc
   slcli               Python based SoftLayer API Tool.
   doctl               DigitalOcean command line tool
   conjure-up          Package runtime for conjure-up spells
   minidlna-escoand    server software with the aim of being fully compliant w
   postgresql10        PostgreSQL is a powerful, open source object-relational
   heroku              CLI client for Heroku
   keepalived          High availability VRRP/BFD and load-balancing for Linux

                            [ Done       ]
                            [ Back       ]

                               7 / 12

Install in progress: installing kernel                                        |
```

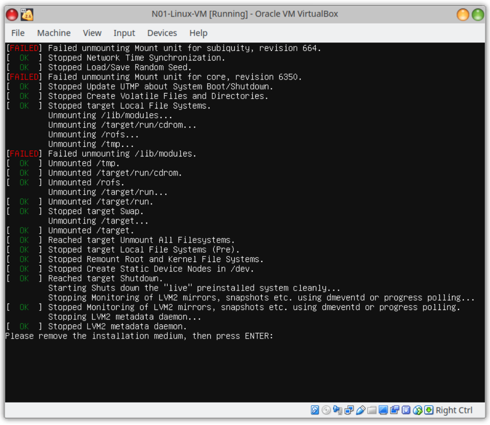
(b) Step 04-14.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Installation complete!

 ┌──────────────────────── Finished install! ─────────────────────────┐
 │    running 'mount -t tmpfs tmpfs /target/run'                      │
 │    running 'mkdir -p /target/run/cdrom'                            │
 │    running 'mount --bind /cdrom /target/run/cdrom'                 │
 │    running 'curtin curthooks'                                      │
 │      curtin command curthooks                                      │
 │        configuring apt configuring apt                             │
 │        installing missing packages                                 │
 │        configuring iscsi service                                   │
 │        configuring raid (mdadm) service                            │
 │        installing kernel                                           │
 │        setting up swap                                             │
 │        apply networking config                                     │
 │        writing etc/fstab                                           │
 │        configuring multipath                                       │
 │        updating packages on target system                          │
 │        configuring pollinate user-agent on target                  │
 │  finalizing installation                                           │
 │    running 'curtin hook'                                           │
 │      curtin command hook                                           │
 │  executing late commands                                           │
 │final system configuration                                          │
 │  configuring cloud-init                                            │
 │  installing OpenSSH server                                         │
 │  cleaning up apt configuration                                     │
 └────────────────────────────────────────────────────────────────────┘

                           [ View full log ]
                           [ Reboot Now    ]

                               12 / 12

Thank you for using Ubuntu!
```

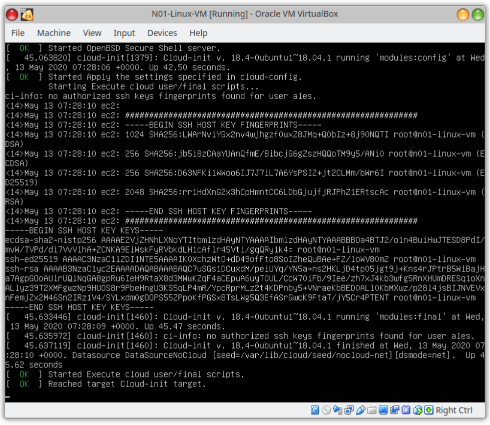
(c) Step 04-15.

Figure 1.10: Installing Ubuntu in a virtual environment (interactive steps 04-13 to 04-15).


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

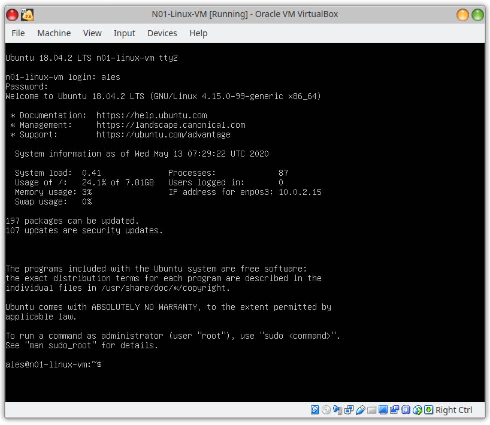
```
[FAILED] Failed unmounting Mount unit for subiquity, revision 664.
[  OK  ] Stopped Network Time Synchronization.
[  OK  ] Stopped Load/Save Random Seed.
[FAILED] Failed unmounting Mount unit for core, revision 6350.
[  OK  ] Stopped Update UTMP about System Boot/Shutdown.
[  OK  ] Stopped Create Volatile Files and Directories.
[  OK  ] Stopped target Local File Systems.
         Unmounting /lib/modules...
         Unmounting /target/run/cdrom...
         Unmounting /rofs...
         Unmounting /tmp...
[FAILED] Failed unmounting /lib/modules.
[  OK  ] Unmounted /tmp.
[  OK  ] Unmounted /target/run/cdrom.
[  OK  ] Unmounted /rofs.
         Unmounting /target/run...
[  OK  ] Unmounted /target/run.
[  OK  ] Stopped target Swap.
         Unmounting /target...
[  OK  ] Unmounted /target.
[  OK  ] Reached target Unmount All Filesystems.
[  OK  ] Stopped target Local File Systems (Pre).
[  OK  ] Stopped Remount Root and Kernel File Systems.
[  OK  ] Stopped Create Static Device Nodes in /dev.
[  OK  ] Reached target Shutdown.
         Starting Shuts down the "live" preinstalled system cleanly...
         Stopping Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progress polling...
[  OK  ] Stopped Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progress polling.
         Stopping LVM2 metadata daemon...
[  OK  ] Stopped LVM2 metadata daemon.
Please remove the installation medium, then press ENTER:
```

(a) Step 04-16.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
[  OK  ] Started OpenBSD Secure Shell server.
[   45.063820] cloud-init[1379]: Cloud-init v. 18.4-0ubuntu1~18.04.1 running 'modules:config' at Wed
, 13 May 2020 07:28:06 +0000. Up 42.50 seconds.
[  OK  ] Started Apply the settings specified in cloud-config.
         Starting Execute cloud user/final scripts...
ci-info: no authorized ssh keys fingerprints found for user ales.
<14>May 13 07:28:10 ec2:
<14>May 13 07:28:10 ec2: #############################################################
<14>May 13 07:28:10 ec2: -----BEGIN SSH HOST KEY FINGERPRINTS-----
<14>May 13 07:28:10 ec2: 1024 SHA256:LWArNviYGx2nv4wjhgzf0wxZ8JMq+Q0bIz+8j9ONQTI root@n01-linux-vm (
DSA)
<14>May 13 07:28:10 ec2: 256 SHA256:jb5i8zCAaYUAnQfmE/BibcjG6gZszHQQoTM9y5/ANi0 root@n01-linux-vm (E
CDSA)
<14>May 13 07:28:10 ec2: 256 SHA256:D63NFKi1WWoo6IJ7J7iL7A6YsPSIZ+jt2CLMm/bWr6I root@n01-linux-vm (E
D25519)
<14>May 13 07:28:10 ec2: 2048 SHA256:rr1HdXnG2x3hCpHmntCC6LDbGjujfjRJPhZ1ERtscAc root@n01-linux-vm (
RSA)
<14>May 13 07:28:10 ec2: -----END SSH HOST KEY FINGERPRINTS-----
<14>May 13 07:28:10 ec2: #############################################################
-----BEGIN SSH HOST KEY KEYS-----
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBOa4BTJ2/o1n4BuiHwJTESD8PdI/
mvW/TVPd/di7VvV1hA+ZCNKA9EiWskFyRVbkdLH1cAflr45Vti/gqQRylk4= root@n01-linux-vm
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIK0XchzWtO+dD49ofFto8SoIZheQuBAe+FZ/1oWV80mZ root@n01-linux-vm
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC7uSGs1DCuxdM/peiUYq/YN5a+ns2HKLjD4tp05jgt9j+Kns4rJPtrB5WiBajH
a7AgpGOoAUlrUQiNqGA8gpRu6IeH9RtaX8d3MWwKZqF4aCEpuA6uyTOUL/CcW7OiFb/9Iee/zh7xJ4kb3wfg5RnXHUmDRESq1oXn
ALlyz39T2XMFgwzNp9HUOS8r9PbeHngU3KS5qLP4mR/YpcRprMLz2t4KDPnby5+VNraeKbBED0AL10KbMXwz/p2814jsBIJNVEVx
nFemjZx2M46Sn2IRz1V4/SYLxdm0gOOPS55ZPpoKfPGSxBTsLWg5Q3EfASrGwcK9FtaT/jY5Cr4PTENT root@n01-linux-vm
-----END SSH HOST KEY KEYS-----
[   45.633446] cloud-init[1460]: Cloud-init v. 18.4-0ubuntu1~18.04.1 running 'modules:final' at Wed,
 13 May 2020 07:28:09 +0000. Up 45.47 seconds.
[   45.635972] cloud-init[1460]: ci-info: no authorized ssh keys fingerprints found for user ales.
[   45.637119] cloud-init[1460]: Cloud-init v. 18.4-0ubuntu1~18.04.1 finished at Wed, 13 May 2020 07
:28:10 +0000. Datasource DataSourceNoCloud [seed=/var/lib/cloud/seed/nocloud-net][dsmode=net].  Up 4
5.62 seconds
[  OK  ] Started Execute cloud user/final scripts.
[  OK  ] Reached target Cloud-init target.
_
```

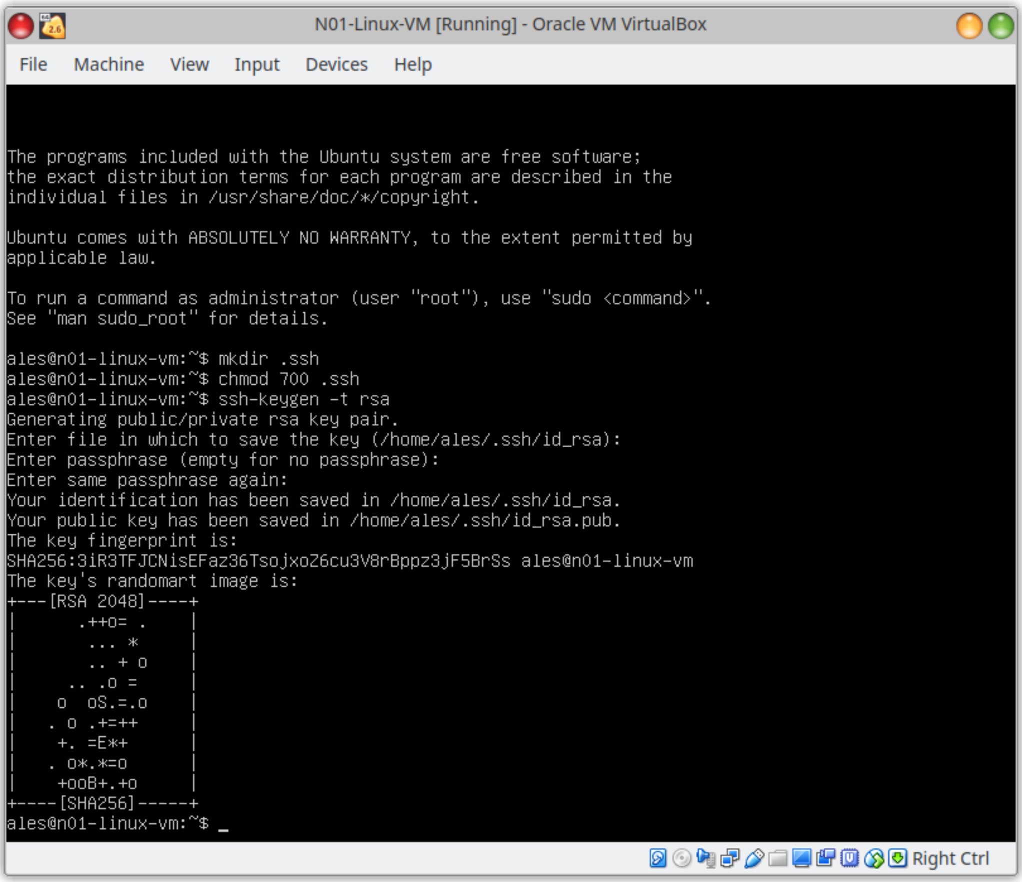
(b) Step 04-17.


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Ubuntu 18.04.2 LTS n01-linux-vm tty2

n01-linux-vm login: ales
Password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-99-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed May 13 07:29:22 UTC 2020

  System load:  0.41              Processes:            87
  Usage of /:   24.1% of 7.81GB   Users logged in:      0
  Memory usage: 3%                IP address for enp0s3: 10.0.2.15
  Swap usage:   0%

197 packages can be updated.
107 updates are security updates.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ales@n01-linux-vm:~$
```

(c) Step 04-18.

Figure 1.11: Installing Ubuntu in a virtual environment (interactive steps 04-16 to 04-18).


N01-Linux-VM [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ales@n01-linux-vm:~$ mkdir .ssh
ales@n01-linux-vm:~$ chmod 700 .ssh
ales@n01-linux-vm:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ales/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ales/.ssh/id_rsa.
Your public key has been saved in /home/ales/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:3iR3TFJCNisEFaz36TsojxoZ6cu3V8rBppz3jF5BrSs ales@n01-linux-vm
The key's randomart image is:
+---[RSA 2048]----+
|      .++o= .    |
|       ... *     |
|       .. + o    |
|     .. .o =     |
|    o  oS.=.o    |
|   . o .+=++     |
|    +. =E*+      |
|   . o*.*=o      |
|    +ooB+.+o     |
+----[SHA256]-----+
ales@n01-linux-vm:~$ _
```

Figure 1.12: Creation of keys for OpenSSH.

Chapter 2

Deploying Instances and Managing Virtual Networks in a Cluster — Exercise N02-Linux-Cluster

Instructions

Clone the virtual environment from the exercise **N01** to a total of 10 instances of this environment.

Then edit the network settings for each cloned instance so that it has its own local Internet Protocol (IP) address number and the communication to the OpenSSH server is enabled between these instances and to them.

Then show that you can connect to each hosted environment (`computer 1`) from the host computer (`computer 0`), and from this hosted environment to another hosted environment (`computer 2`), and then run the `` `hostname -I` `` command at the very latest (`computer 2`).

Show this for all combinations, and since there are 90 combinations of links, make this printout using a Bash script and command `` `parallel-ssh` `` (example for a few combinations: `ssh comp1 parallel-ssh -i -H comp2 -H comp3 -H comp4 -H comp5 -H comp6 -H comp7 -H comp8 -H comp9 -H comp10 hostname -I`).

Material for hints during the explanation of the task:

```
guest# apt-get update; apt-get install pssh
host$ for comp in comp{1..10}; do echo ssh $comp parallel-ssh
      $(echo comp{1..10} | sed -e 's/'$comp'//g'); done
```

```
host$ vboxmanage clonevm Ubuntu
host$ vboxmanage controlvm Linux savestate
host$ vboxmanage startvm Linux -type headless
```

Submit a report about this exercise that demonstrates execution of the solution, its implementation, and scripts/code.

FOR HALF POINTS: Without creating a script for all 90 combinations (only 1 complete combination)

Solution

The solution of this exercise builds on the solution of the previous exercise. As already mentioned, it is good to use procedures that can be upgraded and later automated when setting up systems, so Linux and the Bash Command Interpreter are used here again.

Attachment file of the solution: Computer virtual image in OVA

N02-Linux-Cluster.ova (password for `sudo`: `12345`)
https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng

Use the host computer which was prepared in the previous exercise N01-Linux-VM, where the VirtualBox software is installed. We also have a Linux distribution Ubuntu 18.04.2 LTS installed, located in the VirtualBox virtual image environment `N01-Linux-VM`.

1. First, we clone the existing virtual environment `N01-Linux-VM` once, give it a new name `N02-Linux-Cluster-1`, and register it:

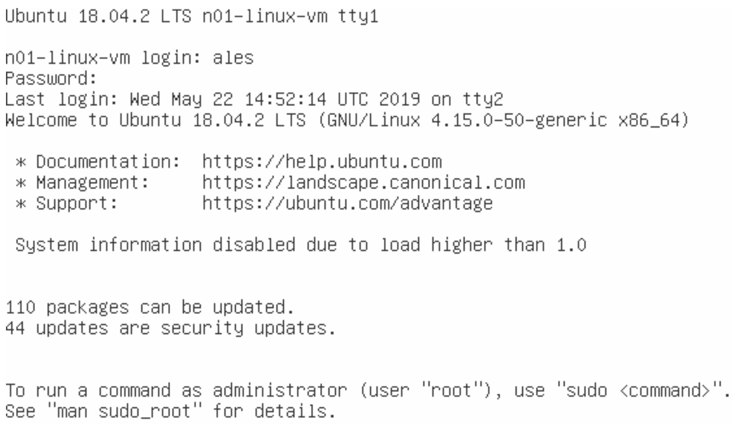
```
$ vboxmanage clonevm N01-Linux-VM \
   --name N02-Linux-Cluster-1 --register
```

2. The cloned system is then run in interactive mode. Because the network of the guest system happens to be unresponsive due to the use of cloned network interfaces, the virtual environment is stopped and restarted.[1]

[1] In case you install some other distribution and the network interface is not connecting, first check that the DHCP server is running on the host and run the `dhclient -r` in the guest.

```
$ vboxmanage startvm N02−Linux−Cluster−1
$ vboxmanage controlvm N02−Linux−Cluster−1
   ↪ poweroff
$ vboxmanage startvm N02−Linux−Cluster−1
```

3. In the interactive environment window shown for X server, log in to the newly started cloned system (password `12345`), as seen in Figure 2.1.



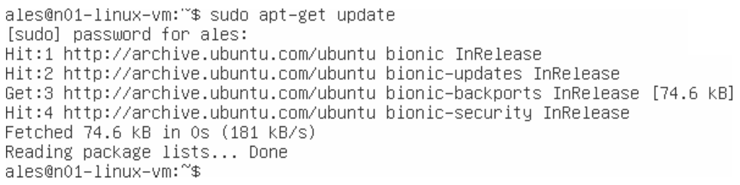
```
Ubuntu 18.04.2 LTS n01-linux-vm tty1

n01-linux-vm login: ales
Password:
Last login: Wed May 22 14:52:14 UTC 2019 on tty2
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-50-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 System information disabled due to load higher than 1.0


110 packages can be updated.
44 updates are security updates.


To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Figure 2.1: Interactive login to a cloned system `N02-Linux-Cluster-1`.

4. We then run the resource update command Advanced Package Tool (APT) for the software packages it manages:

```
ales@n01−linux−vm:~$ sudo apt−get update
```

After updating the resources on the X server screen, we see the result as shown in Figure 2.2.



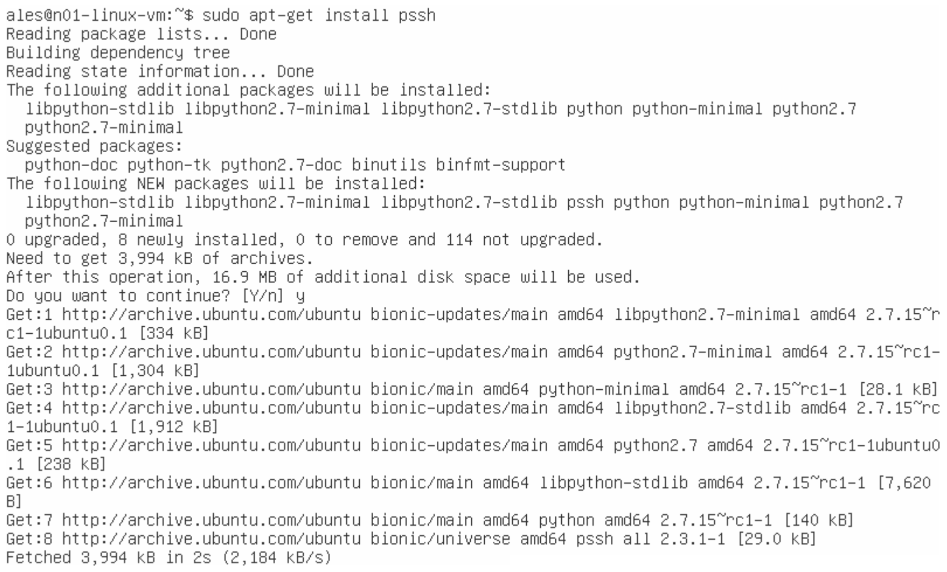
```
ales@n01-linux-vm:~$ sudo apt-get update
[sudo] password for ales:
Hit:1 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Get:3 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Hit:4 http://archive.ubuntu.com/ubuntu bionic-security InRelease
Fetched 74.6 kB in 0s (181 kB/s)
Reading package lists... Done
ales@n01-linux-vm:~$
```

Figure 2.2: Updating of the resources in the list of software packages in exercise `N02-Linux-Cluster-1`.

5. After updating the resources, install the software package for Parallel SSH (PSSH).

```
ales@n01−linux−vm:~$ sudo apt−get install pssh
```

We can see the receipt of the necessary packages in Figure 2.3 and continuing of the execution of the command by installing these packages in Figure 2.4. If you leave the computer running during the command to update the package resources, it may happen that the automatic installation of security patches starts in the meantime, so we wait for the completion of this update before installing the package.



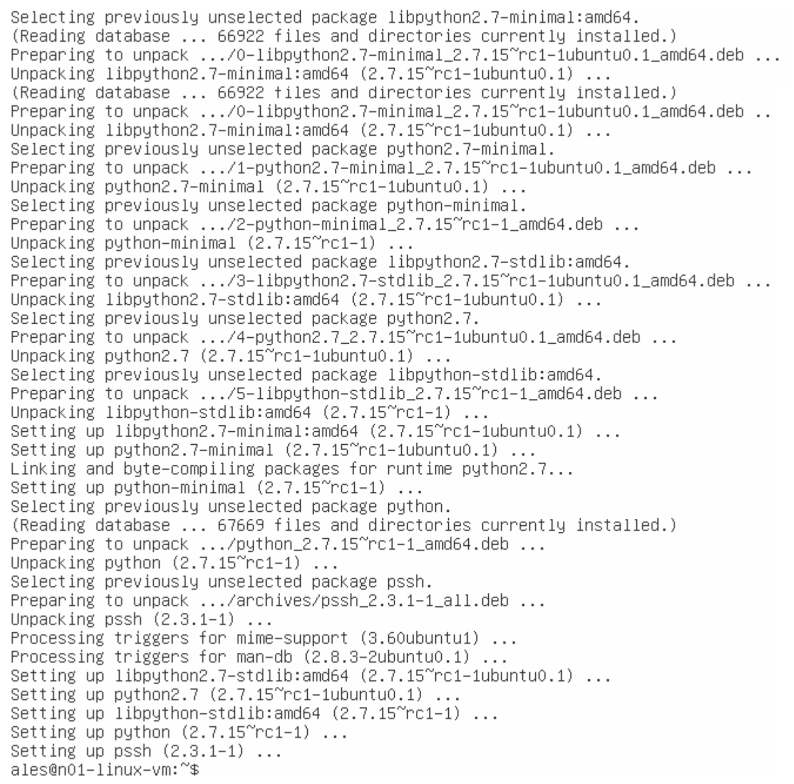
```
ales@n01-linux-vm:~$ sudo apt-get install pssh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib python python-minimal python2.7
  python2.7-minimal
Suggested packages:
  python-doc python-tk python2.7-doc binutils binfmt-support
The following NEW packages will be installed:
  libpython-stdlib libpython2.7-minimal libpython2.7-stdlib pssh python python-minimal python2.7
  python2.7-minimal
0 upgraded, 8 newly installed, 0 to remove and 114 not upgraded.
Need to get 3,994 kB of archives.
After this operation, 16.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libpython2.7-minimal amd64 2.7.15~r
c1-1ubuntu0.1 [334 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 python2.7-minimal amd64 2.7.15~rc1-
1ubuntu0.1 [1,304 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 python-minimal amd64 2.7.15~rc1-1 [28.1 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libpython2.7-stdlib amd64 2.7.15~rc
1-1ubuntu0.1 [1,912 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 python2.7 amd64 2.7.15~rc1-1ubuntu0
.1 [238 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpython-stdlib amd64 2.7.15~rc1-1 [7,620
B]
Get:7 http://archive.ubuntu.com/ubuntu bionic/main amd64 python amd64 2.7.15~rc1-1 [140 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/universe amd64 pssh all 2.3.1-1 [29.0 kB]
Fetched 3,994 kB in 2s (2,184 kB/s)
```

Figure 2.3: Receiving PSSH installation packages in `N02-Linux-Cluster-1`.

6. Because we will change the network settings, we stop the virtual environment after the successful installation of the new packages:

```
ales@n01−linux−vm:~$ sudo shutdown −h
```

7. Now let's prepare the environment VirtualBox for the parallel use of multiple instances of virtual environments. Because the new virtual environments will also access each other and allow access from the host, we will use the network access setting via the host interface (`hostonly`). We will connect these virtual environments to the network that uses Internet Protocol (IP) automatically. Hence, first, in the environment VirtualBox, we create a new network interface that will be named `vboxnet1` automatically:



```
Selecting previously unselected package libpython2.7-minimal:amd64.
(Reading database ... 66922 files and directories currently installed.)
Preparing to unpack .../0-libpython2.7-minimal_2.7.15~rc1-1ubuntu0.1_amd64.deb ...
Unpacking libpython2.7-minimal:amd64 (2.7.15~rc1-1ubuntu0.1) ...
(Reading database ... 66922 files and directories currently installed.)
Preparing to unpack .../0-libpython2.7-minimal_2.7.15~rc1-1ubuntu0.1_amd64.deb ..
Unpacking libpython2.7-minimal:amd64 (2.7.15~rc1-1ubuntu0.1) ...
Selecting previously unselected package python2.7-minimal.
Preparing to unpack .../1-python2.7-minimal_2.7.15~rc1-1ubuntu0.1_amd64.deb ...
Unpacking python2.7-minimal (2.7.15~rc1-1ubuntu0.1) ...
Selecting previously unselected package python-minimal.
Preparing to unpack .../2-python-minimal_2.7.15~rc1-1_amd64.deb ...
Unpacking python-minimal (2.7.15~rc1-1) ...
Selecting previously unselected package libpython2.7-stdlib:amd64.
Preparing to unpack .../3-libpython2.7-stdlib_2.7.15~rc1-1ubuntu0.1_amd64.deb ...
Unpacking libpython2.7-stdlib:amd64 (2.7.15~rc1-1ubuntu0.1) ...
Selecting previously unselected package python2.7.
Preparing to unpack .../4-python2.7_2.7.15~rc1-1ubuntu0.1_amd64.deb ...
Unpacking python2.7 (2.7.15~rc1-1ubuntu0.1) ...
Selecting previously unselected package libpython-stdlib:amd64.
Preparing to unpack .../5-libpython-stdlib_2.7.15~rc1-1_amd64.deb ...
Unpacking libpython-stdlib:amd64 (2.7.15~rc1-1) ...
Setting up libpython2.7-minimal:amd64 (2.7.15~rc1-1ubuntu0.1) ...
Setting up python2.7-minimal (2.7.15~rc1-1ubuntu0.1) ...
Linking and byte-compiling packages for runtime python2.7...
Setting up python-minimal (2.7.15~rc1-1) ...
Selecting previously unselected package python.
(Reading database ... 67669 files and directories currently installed.)
Preparing to unpack .../python_2.7.15~rc1-1_amd64.deb ...
Unpacking python (2.7.15~rc1-1) ...
Selecting previously unselected package pssh.
Preparing to unpack .../archives/pssh_2.3.1-1_all.deb ...
Unpacking pssh (2.3.1-1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up libpython2.7-stdlib:amd64 (2.7.15~rc1-1ubuntu0.1) ...
Setting up python2.7 (2.7.15~rc1-1ubuntu0.1) ...
Setting up libpython-stdlib:amd64 (2.7.15~rc1-1) ...
Setting up python (2.7.15~rc1-1) ...
Setting up pssh (2.3.1-1) ...
ales@n01-linux-vm:~$
```

Figure 2.4: Installation of PSSH packages in `N02-Linux-Cluster-1`.

```
$ vboxmanage hostonlyif create
```

8. The new network on the interface `vboxnet1` shall assign the IP numbers automatically and use Dynamic Host Configuration Protocol (DHCP) to do this. The server for DHCP is added in VirtualBox with the following command:

```
$ vboxmanage dhcpserver add --ifname vboxnet1 \
    --ip 192.168.57.2 --netmask 255.255.255.0 \
    --lowerip 192.168.57.3 \
     --upperip 192.168.57.254 \
    --enable
```

That way, it will be that the first IP number, assigned by the DHCP server, will match `192.168.57.3`, and the remaining extra numbers will follow each other as `192.168.57.4`, `192.168.57.5`, `192.168.57.6` etc., possibly all the way to the number 254 in the last part of the

IP address. Network mask `255.255.255.0` specifies that, for the network accessed with addresses outside the IP range `192.168.57.0` – `192.168.57.255`, network traffic will be directed outwards[2] via the host interface `vboxnet1` for the host and any virtual environments that use this same interface `vboxnet1`.

9. We now define new network settings for the stopped virtual environment. We configure network access via the host interface (`hostonly`):

```
$ vboxmanage modifyvm N02-Linux-Cluster-1 \
   --nic1 hostonly
$ vboxmanage modifyvm N02-Linux-Cluster-1 \
   --hostonlyadapter1 vboxnet1
```

10. For later simultaneous execution of instances of this virtual environment, the memory capacity of the virtual machine is reduced to 1GB:

```
$ vboxmanage modifyvm N02-Linux-Cluster-1 \
   --memory 1024
```

11. The virtual environment is then started in screenless (`headless`) mode, and we test if the prepared access works through its OpenSSH server.

```
$ vboxmanage startvm N02-Linux-Cluster-1 \
     --type headless;
```

12. We wait for the system boot to be completed successfully in the virtual environment and OpenSSH is loaded, then test connect to the server and accept the public key from OpenSSH, located at the first IP address in the network space from the DHCP server, which is `192.168.57.3`:

```
$ ssh 192.168.57.3
```

[2] If we have more than one host computer with VirtualBox, we can also create a real physically separate cluster in this way, by forwarding network traffic to the IPs of these computers, which can be divided into different physical locations.

```
The authenticity of host '192.168.57.3 (192.168.57.3)' can't be established.
ECDSA key fingerprint is SHA256:7CMTDNfnvTST7GWQNfA90bsUZIcHTzVaE0ki5l8YXog.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.57.3' (ECDSA) to the list of known hosts.
ales@192.168.57.3's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-50-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

  System information as of Thu May 23 07:25:27 UTC 2019

  System load: 0.17 Processes: 89
  Usage of /: 25.2% of 7.81GB Users logged in: 0
  Memory usage: 13% IP address for enp0s3: 192.168.57.3
  Swap usage: 0%


74 packages can be updated.
0 updates are security updates.


Last login: Thu May 23 07:08:00 2019
ales@n01-linux-vm:~$
```

13. Login to the virtual environment via the server OpenSSH at IP address `192.168.57.3` in the network `vboxnet1` has succeeded. Therefore, to facilitate future logins to this system, we copy the public key Rivest-Shamir-Adleman (RSA) from the host computer to the virtual environment `N02-Linux-Cluster-1` at the network IP address `192.168.57.3` to the user's folder `~/.ssh/`:

```
$ cat ~/.ssh/id_rsa.pub | ssh 192.168.57.3 \
  "cat >> ~/.ssh/authorized_keys"
```

14. We stop the environment again to be ready for re-cloning:

```
ales@n01-linux-vm:~$ sudo shutdown -h
```

```
Shutdown scheduled for Thu 2019-05-23 07:29:48 UTC, use 'shutdown -c' to
    ↪ cancel.
```

15. Then, we clone the virtual environment `N02-Linux-Cluster-1` nine times, and assign a new name to each clone:

```
$ for i in {2..10}; do
    vboxmanage clonevm N02-Linux-Cluster-1 \
      --name N02-Linux-Cluster-$i --register;
  done
```

The procedure takes about 10 minutes, depending on the speed of the hardware. In the shell we see a printout of the progress of this process:

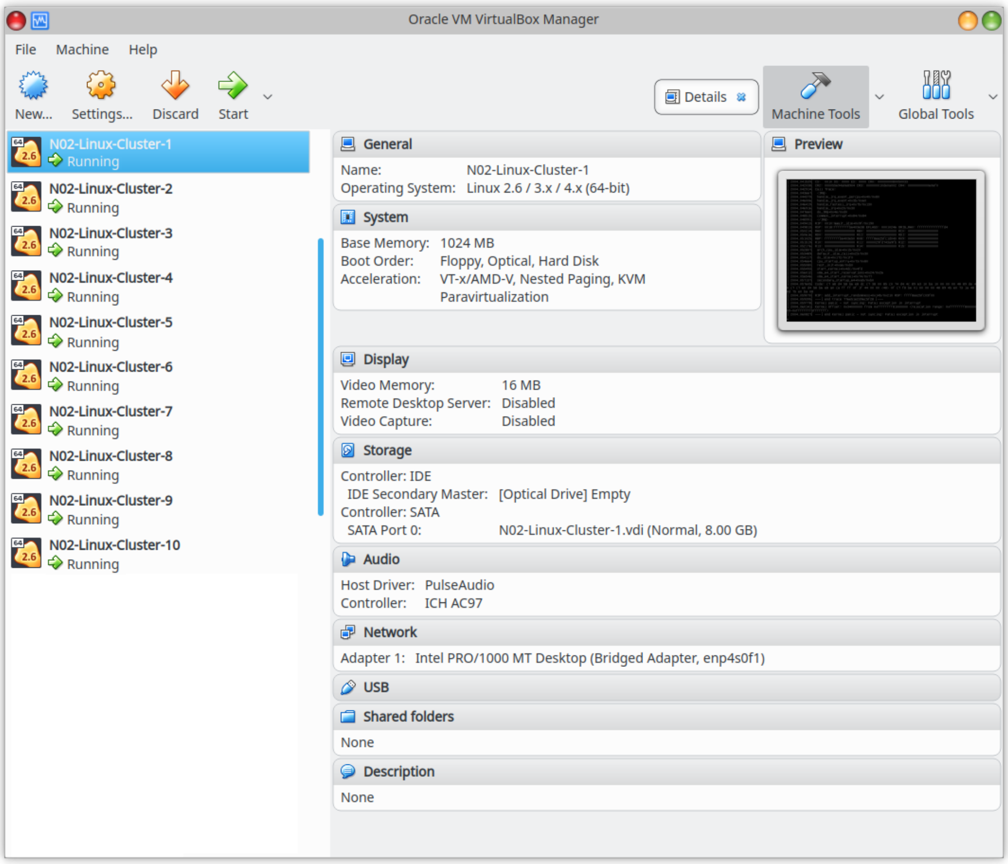
```
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-1"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-2"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-3"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-4"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-5"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-6"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-7"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-8"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-9"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Machine has been successfully cloned as "N02-Linux-Cluster-10"
```

16. We now explore each of these virtual environments in screenless mode (`headless` type).

```
$ for i in {1..10}; do
    vboxmanage startvm N02-Linux-Cluster-$i \
      --type headless;
  done
```

```
Waiting for VM "N02-Linux-Cluster-1" to power on...
VM "N02-Linux-Cluster-1" has been successfully started.
Waiting for VM "N02-Linux-Cluster-2" to power on...
VM "N02-Linux-Cluster-2" has been successfully started.
Waiting for VM "N02-Linux-Cluster-3" to power on...
VM "N02-Linux-Cluster-3" has been successfully started.
Waiting for VM "N02-Linux-Cluster-4" to power on...
VM "N02-Linux-Cluster-4" has been successfully started.
Waiting for VM "N02-Linux-Cluster-5" to power on...
VM "N02-Linux-Cluster-5" has been successfully started.
Waiting for VM "N02-Linux-Cluster-6" to power on...
VM "N02-Linux-Cluster-6" has been successfully started.
Waiting for VM "N02-Linux-Cluster-7" to power on...
VM "N02-Linux-Cluster-7" has been successfully started.
Waiting for VM "N02-Linux-Cluster-8" to power on...
VM "N02-Linux-Cluster-8" has been successfully started.
Waiting for VM "N02-Linux-Cluster-9" to power on...
VM "N02-Linux-Cluster-9" has been successfully started.
Waiting for VM "N02-Linux-Cluster-10" to power on...
VM "N02-Linux-Cluster-10" has been successfully started.
```

17. During the execution of virtual environments starting up in the background (Figure 2.5), we can observe their progress in the main window of the VirtualBox application, or in the shell:



N02-Linux-Cluster-1 Running
N02-Linux-Cluster-2 Running
N02-Linux-Cluster-3 Running
N02-Linux-Cluster-4 Running
N02-Linux-Cluster-5 Running
N02-Linux-Cluster-6 Running
N02-Linux-Cluster-7 Running
N02-Linux-Cluster-8 Running
N02-Linux-Cluster-9 Running
N02-Linux-Cluster-10 Running

General
Name: N02-Linux-Cluster-1
Operating System: Linux 2.6 / 3.x / 4.x (64-bit)

System
Base Memory: 1024 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Preview

Display
Video Memory: 16 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: N02-Linux-Cluster-1.vdi (Normal, 8.00 GB)

Audio
Host Driver: PulseAudio
Controller: ICH AC97

Network
Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, enp4s0f1)

USB

Shared folders
None

Description
None

Figure 2.5: Launch of 10 instances of Ubuntu virtual environments.

```
$ vboxmanage controlvm N02-Linux-Cluster-10 \
   screenshotpng screenshot-10.png
$ display screenshot-10.png
```

18. We wait for the successful completion of booting and connection of Ubuntu hosted systems in virtual environments with the network via the previously added DHCP server, e.g. cca. 5 minutes. If we may have forgotten to perform any of the above steps for any reason, we need to repeat the process again: Remove current network changes and cloned virtual environments (note — this step is not performed if we performed the above steps correctly, it is only listed as possible troubleshooting):

```
$ for rac in rac{1..10}; do
    vboxmanage unregistervm N02-Linux-Cluster-1
       ↪ --delete;
  done
$ vboxmanage dhcpserver remove --ifname vboxnet1
```

19. On the host computer, after running virtual environments with OpenSSH servers successfully, we then receive fingerprints of keys from the OpenSSH servers on all virtual environments:

```
$ for i in {3..12}; do
    ssh-keyscan -H 192.168.57.$i >> ~/.ssh/
       ↪ known_hosts;
  done
```

20. The same receipt of fingerprints is also done in the virtual environments:

```
$ for i in {3..12}; do
    ssh 192.168.57.$i \
     'for j in {3..12}; do \
       ssh-keyscan -H 192.168.57.$j \
       >> ~/.ssh/known_hosts; \
      done';
  done
```

21. We share the RSA generated public key from the virtual environments from each to each other, to be able to log in to their OpenSSH servers:

```
$ for i in {3..12}; do
    for j in {3..12}; do
      ssh 192.168.57.$i cat .ssh/id_rsa.pub \
        | ssh 192.168.57.$j \
          "cat >> ~/.ssh/authorized_keys";
    done
    echo -n ".";
  done
```

22. Because the package Parallel SSH (PSSH) enables that the Secure Shell (SSH) is run in parallel on multiple computers with the same commands, we can now run multiple commands in parallel. On the host computer, we execute a script for the parallel execution of commands:

```
$ for i in {3..12}; do
    ssh 192.168.57.$i \
      parallel−ssh −i \
        $(echo "-H "192.168.57.{3..12} \
          | sed −e 's/−H 192.168.57.'$i'//g') \
        hostname −I;
  done
```

The command prints the next result, starting as:

```
[1] 09:30:48 [SUCCESS] 192.168.57.5
192.168.57.5
[2] 09:30:48 [SUCCESS] 192.168.57.9
192.168.57.9
[3] 09:30:48 [SUCCESS] 192.168.57.10
192.168.57.10
[4] 09:30:48 [SUCCESS] 192.168.57.4
192.168.57.4
[5] 09:30:48 [SUCCESS] 192.168.57.7
192.168.57.7
[6] 09:30:48 [SUCCESS] 192.168.57.6
192.168.57.6
[7] 09:30:48 [SUCCESS] 192.168.57.12
192.168.57.12
[8] 09:30:48 [SUCCESS] 192.168.57.8
192.168.57.8
[9] 09:30:48 [SUCCESS] 192.168.57.11
192.168.57.11
[1] 09:30:51 [SUCCESS] 192.168.57.5
192.168.57.5
```

and (after 154 lines of printout omitted here), completes as:

```
[7] 09:31:17 [SUCCESS] 192.168.57.8
192.168.57.8
[8] 09:31:17 [SUCCESS] 192.168.57.6
192.168.57.6
[9] 09:31:17 [SUCCESS] 192.168.57.10
192.168.57.10
```

23. Properly functioning virtual environments are then stored and put to sleep:

```
$ for i in {1..10}; do
    vboxmanage controlvm N02−Linux−Cluster−$i \
      savestate;
  done
```

24. We export the OVA image for the created environments. The process takes some time because of the writing of a large file, e.g. approximately

fifteen minutes. The exported file size is approximately 13 GB. During this time we see the progress of the process:

```
$ vboxmanage export N02-Linux-Cluster-{1..10} \
   -o N02-Linux-Cluster.ova
```

```
0%...10%...20%...30%...40%...50%...60%...70%...
80%...90%...100%
Successfully exported 10 machine(s).
```

25. We write down an exercise report (for example, in the same manner that the solution was described here in this material).

Chapter 3

Deploying a Cloud for a Distributed Service and Messaging Communication — Exercise N03-MPI-PingPong

Exercise: N03-MPI-PingPong

Install OpenMPI in virtual environments of **N02**.

Then write a **PingPong** program that sends one floating point number (random number between 0.00 ... 180.00) to each client (rank 1 ... rank 9) in OpenMPI from the master node (rank 0).

Then, each client sends the number back to the master node (rank 0) that adds the received number to a sum by modulo with 360.

Execute the ping-pong until the sum of numbers on the arc is an angle in the interval between 270.505 and 270.515 (the sum of remainders after modulo with 360).

In the end, the master rank prints a number of ping-pong message pairs on the terminal and in file `RESULT.TXT`. Submit a report about this exercise that demonstrates execution of the solution, its implementation, scripts/code, and the file RESULT.TXT.

Solution

The solution of this exercise builds on the solution of the previous exercise, N02-Linux-Cluster. As already mentioned, it is good to use procedures that can be upgraded and later automated when setting up systems, so Linux and the Bash Command Interpreter are used here again.

Attachment file of the solution: Computer virtual image in OVA

N03-MPI-PingPong.ova (password for `sudo`: `12345`)
https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng

We use the host computer again, which we have already prepared in the assignment N01-Linux-VM, where we have VirtualBox software installed. We also reuse the prepared virtual environments from the exercise N02-Linux-Cluster.

1. First, we copy all the virtual environments from the exercise N02-Linux-Cluster:

```
$ for i in {1..10}; do
    vboxmanage clonevm N02-Linux-Cluster-$i \
      --name N03-MPI-PingPong-$i --register;
  done
```

2. We then run the newly cloned virtual environments:

```
$ for i in {1..10}; do
    vboxmanage startvm N03-MPI-PingPong-$i \
      --type headless;
  done
```

3. For this task, we will need to install additional software packages that we will download from the Internet. Therefore, we will change the network connection settings in the virtual environments. The settings could be changed graphically in interactive mode, but the process of changing the settings for 10 virtual environments may be somewhat impractical time-wise or, at an even larger number of environments (e.g. a few thousand), this might also become quite less controllable and less error prone. So, we reuse the shell and programme the iteration process for all virtual environments. Hence, we first change the network setting from the host interface (`hostonly`) to the setting for Network Address Translation (NAT):

```
$ for i in {1..10}; do
    vboxmanage controlvm N03-MPI-PingPong-$i \
      nic1 nat
  done
```

4. Then we add the port forwarding for the OpenSSH server from the host ports 2000+i (hence, values 2001, 2002, ..., 2010) to a guest's port 22, where the server at i-th virtual environment is listening:

```
$ for i in {1..10}; do
    vboxmanage controlvm N03-MPI-PingPong-$i \
      natpf1 "OpenSSH,tcp,,200$i,,22"
  done
```

5. In this way, we will be able to access an individual OpenSSH server, but, due to the newly changed network access for the server, we need to retake the OpenSSH key fingerprint in the file `known_hosts` on the guest computer:

```
$ for i in {1..10}; do
    ssh-keyscan -p 200$i -H 127.0.0.1 >> ~/.ssh/
        ↪ known_hosts;
  done
```

6. We can now install object libraries on all of these virtual environments that implement an OpenMPI communication interface. These libraries and their software development support are included in the packages `openmpi-bin` and `mpi-default-dev`. Next to the libraries, we install the package `g++`, which contains a compiler so we can translate the code for these versions of libraries. Let's use the program `apt-get` and install the mentioned packages in the virtual environments:

```
$ stty -echo; echo "Password: "; read pass; stty echo
    ↪ ;
$ for i in {1..10}; do \
    echo $pass | ssh -p 200$i 127.0.0.1 \
      sudo -S apt-get install -y \
        libopenmpi-dev openmpi-bin g++;
  done
```

Here, the parameter `-S` with command `sudo` reads the entered password so that we do not have to re-enter it for each server separately. Installation on all virtual environments can take approximately one hour, depending on the speed of the hardware.[3] Therefore, during this time, we implement our own code for `N03-MPI-PingPong.c`, as written in the next step.

7. In a text editor (e.g. `vim`) we write a computer program for OpenMPI in the language C[4] and save it as a file `N03-MPI-PingPong.c`:

```
#include <mpi.h>
#include <stdlib.h>
#include <stdio.h>

void rank0() {
  MPI_Status status;

  int nPingPongs = 0;
  float sum = 0;

  while (1) {
    for (int i = 1; i < 10; i++) {
      float pongReceive, pingSend = rand()%18000 / 100.0;

      MPI_Send(&pingSend, 1, MPI_FLOAT, i, 42,
        ↪ MPI_COMM_WORLD);
      MPI_Recv(&pongReceive, 1, MPI_FLOAT, i, 42,
        ↪ MPI_COMM_WORLD, &status);
      nPingPongs++;

      sum += pongReceive;
      if (sum > 360) sum -= 360;

      if (sum >= 270.505 && sum <= 270.515) {
        printf("Number of Ping-Pongs: %d\n", nPingPongs);

        pingSend = -42; // exit ping pong
```

[3]If necessary, the operation could be speeded up by first installing the software in a single virtual environment and then cloning that environment. However, in this case, one should repeat the key exchange after cloning OpenSSH from the previous exercise.

[4]We can also use other languages. For example, if we use the Python language, we need to install a software package `python-mpi4py` on the virtual environments.

```
        for (int i = 1; i < 10; i++)
          MPI_Send(&pingSend, 1, MPI_FLOAT, i, 42,
              ↪ MPI_COMM_WORLD);
        return;
      }
    }
  }
}

void rankN(int N) {
  MPI_Status status;

  while (1) {
    float pingpong42;

    MPI_Recv(&pingpong42, 1, MPI_FLOAT, 0, 42,
        ↪ MPI_COMM_WORLD, &status);
    if (pingpong42 == -42) return;
    MPI_Send(&pingpong42, 1, MPI_FLOAT, 0, 42,
        ↪ MPI_COMM_WORLD);
  }
}

int main(int argc, char* argv[]) {
  int rank;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  if (!rank) rank0(); else rankN(rank);

  MPI_Finalize();
}
```

8. Once we have prepared the code for OpenMPI and also installed packages for OpenMPI in virtual environments successfully, we restore the network settings so that the nodes will communicate with each other, as in the previous exercise, via the network `vboxnet1`:

```
$ for i in {1..10}; do
   vboxmanage controlvm N03−MPI−PingPong−$i \
    nic1 hostonly vboxnet1;
  done
```

9. We copy the file `N03-MPI-PingPong.c` to all virtual environments with a command:

```
$ for i in {3..12}; do
   scp N03−MPI−PingPong.c 192.168.57.$i:;
  done
```

10. Then we translate the code with OpenMPI in the file `main.c` using the command:

```
$ parallel−ssh −i \
   $(echo "-H "192.168.57.{3..12}) \
   mpicc N03−MPI−PingPong.c \
    −o N03−MPI−PingPong
```

11. Then we execute the programs with OpenMPI on all nodes by entering their names in the run command:

```
$ time ssh 192.168.57.3 \
  mpiexec −n 10 \
   $(echo "-H "192.168.57.{3..12}) \
   −−mca btl\_base\_warn\_component\_unused 0 \
   N03−MPI−PingPong
```

After execution, the following result is displayed:

```
Number of Ping−Pongs: 7592

real 1m16,476s
user 0m0,039s
sys 0m0,018s
```

12. During the execution of the parallel program, we check whether the program actually runs over different virtual machines and is run exactly once on each:

```
$ parallel-ssh -i \
    $(echo "-H "192.168.57.{3..12}) \
    '(echo -n MACHINE=; hostname -I;) \
       | xargs; ps aux' \
  | grep -e MACHINE=192.168.57. \
         -e N03-MPI-PingPong
```

Which prints `mpiexec` program startup for the first virtual environment and, in addition, the program runs on all virtual environments `N03-MPI-PingPong`, as seen separately for each IP address:

```
MACHINE=192.168.57.3
ales 23027 0.1 1.8 467736 18972 ? Ssl 16:18 0:00 mpiexec -n 10 -H
    ↪ 192.168.57.3 -H 192.168.57.4 -H 192.168.57.5 -H 192.168.57.6 -H
    ↪ 192.168.57.7 -H 192.168.57.8 -H 192.168.57.9 -H 192.168.57.10 -H
    ↪ 192.168.57.11 -H 192.168.57.12 --mca btl_base_warn_component_unused
    ↪  0 N03-MPI-PingPong
ales 23043 92.4 1.1 325420 12024 ? Rl 16:18 0:33 N03-MPI-PingPong
MACHINE=192.168.57.12
ales 18068 89.8 1.1 325164 11548 ? Rl 16:18 0:33 N03-MPI-PingPong
MACHINE=192.168.57.10
ales 15684 93.2 1.1 325164 11600 ? Rl 16:18 0:34 N03-MPI-PingPong
MACHINE=192.168.57.6
ales 15472 93.4 1.1 325164 11564 ? Rl 16:18 0:34 N03-MPI-PingPong
MACHINE=192.168.57.9
ales 15586 90.5 1.1 325160 11496 ? Rl 16:18 0:34 N03-MPI-PingPong
MACHINE=192.168.57.5
ales 15689 88.2 1.1 325160 11516 ? Rl 16:18 0:34 N03-MPI-PingPong
MACHINE=192.168.57.8
ales 15705 88.0 1.1 325160 11416 ? Rl 16:18 0:34 N03-MPI-PingPong
MACHINE=192.168.57.11
ales 16635 90.4 1.1 325164 11472 ? Rl 16:18 0:34 N03-MPI-PingPong
MACHINE=192.168.57.4
ales 16357 87.9 1.1 325160 11528 ? Rl 16:18 0:34 N03-MPI-PingPong
MACHINE=192.168.57.7
ales 15520 88.7 1.1 325160 11884 ? Rl 16:18 0:34 N03-MPI-PingPong
```

13. Save the properly functioning virtual environments and put them in a saved suspended state:

```
$ for i in {1..10}; do
    vboxmanage controlvm N03-MPI-PingPong-$i \
      savestate;
  done
```

14. Export the OVA image for the created environments. The process takes some time because of the writing of a large file, e.g. approximately fifteen minutes. The exported file size is approximately 13 GB. During this time we see the progress of the process:

```
$ vboxmanage export N03-MPI-PingPong-{1..10} \
   -o N03-MPI-PingPong.ova
```

```
0%...10%...20%...30%...40%...50%...60%...70%
...80%...90%...100%
Successfully exported 10 machine(s).
```

15. Write down an exercise report (for example, in the same manner that the solution was described here in this material).

Chapter 4

Deploying Cloud Computing by Sharing Resources between Multiple Distributed Services and Communication by Sending Multiple Types of Messages — Exercise N04-MPI-Cluster

Exercise: N04-MPI-Cluster

In the environment from task **N02**, implement a **multiplayer Battleship game** (https://en.wikipedia.org/wiki/Battleship_(game)).

Implement the Graphical User Interface of the game online (for example HTML) by accessing one of your hosted computers (for example, **computer 1**) on the web browser from the host computer (**computer 0**), which represents the web server for the external viewers of the game. Display the game field as boxes that are filled or not filled, for individual ships, and also indicate the success of the players and the boats hit.

The players play the game through steps as **virtual players** within computer virtual environments (e.g., **computer 2** ... **computer 10**), and are controlled by the main computer (**computer 1**) as a referee. In other words of parallel programming, in the game, one computer is leading (master, **computer 1**), others are following (clients, slaves).

Players play by the moves so that the player **computer 2** starts, then continues to the player **computer 3**, etc., all the way to the player **computer 10**, then it is again the turn of **computer 2**. One move takes 1 second. In each single move, a player first receives the state of the game from the referee, then replies to the referee about his game play move (field coordinate), and then receives feedback from the referee regarding the continuation of the game. The host computer (**computer 0**) and master computer (referee – **computer 1**) do not play the game and, hence, do not have their own ships.

The state of the Battleship game is a **2D playing field** (for example, an array with size of 20x20 units) on which ships of different lengths are installed: 5 units, 4 units, 3 units, 2 units, and 1 unit (e.g. two ships for each length type). In the game state, also keep the moves (coordinates) played, and their success (which ships are already hit and which player has hit which ship).

Use Message Passing Interface (MPI) to connect between virtual players. The MPI master node first sends the status to the player (MPI_Send), and this player receives the status (MPI_Recv). With subsequent calls MPI_Send / MPI_Recv then the referee and the player exchange the sent-in move and success. Then the next player is on the move. The game ends when all the ships are sunk. The player with most hits wins.

Submit a report about this exercise that demonstrates execution of the solution, its implementation, and scripts/code.

FOR HALF POINTS: Only 2 players

HINT — use the refresh method:
`https://www.w3schools.com/tags/att_meta_http_equiv.asp`

Solution

The solution of this exercise builds on the solution of the previous exercise, N03-MPI-PingPong[5]. As already mentioned, it is good to use procedures that can be upgraded and later automated when setting up systems, so Linux and the Bash Command Interpreter are used here again. We will also use a web server already located in the Ubuntu distribution.

Attachment file of the solution: Computer virtual image in OVA

N04-MPI-Cluster.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`

We use the host computer again, which we have already prepared in the assignment N01-Linux-VM, where we have installed VirtualBox software. We also reuse the prepared virtual environments from the exercise N02-Linux-Cluster. As in the exercise N03-MPI-PingPong we already installed Open-MPI, we start by upgrading its virtual environments.

1. First, we copy all the virtual environments from the exercise N03-MPI-PingPong:

```
$ for i in {1..10}; do
   vboxmanage clonevm N03-MPI-PingPong-$i \
     --name N04-MPI-Cluster-$i --register;
  done
```

2. We then start up the first cloned virtual environment:

```
$ vboxmanage startvm N04-MPI-Cluster-1 \
     --type headless;
```

3. Since we will be accessing the web folder on the first virtual environment through a server that will run in that environment, we will first install the web server in that environment. Let's switch to internet connection mode NAT, connect to the first virtual environment through an OpenSSH connection through redirection on port 2001, and install software package `apache2`:

[5]As an option, this exercise can also be done by merely upgrading the task N02-Linux-Cluster, such that in the step of installing additional packages, we add packages for Open-MPI.

```
$ vboxmanage controlvm N04−MPI−Cluster−$i \
     nic1 nat
$ ssh −p 2001 127.0.0.1 \
     'sudo −S apt update; sudo apt install −y apache2'
```

4. We reset the network settings for the host interface as (`hostonly`):

```
$ vboxmanage controlvm N04−MPI−Cluster−1 \
     nic1 hostonly vboxnet1
```

5. We allow the user to write to a file that will be served via a web server:

```
$ ssh 192.168.57.3 sudo −S chown ales.ales /var/
   ↪ www/html/index.html
```

6. In a programming development environment (e.g. QtCreator or `vim`) we write a game program and save it to a file `N04-MPI-Cluster.c`. The program is divided into two parts: First the code for the main node (function `void rank0()`) and then the code for the remaining nodes (function `void rankN(int N)`). Both functions are called from the main program, depending on the type of current node executing it:

```c
#include <mpi.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

void rankN(int N);
void rank0();

int main(int argc, char* argv[]) {
    int rank;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    if (!rank) rank0(); else rankN(rank);

    MPI_Finalize();
}
```

Data that will be exchanged between the main node (referee) and other nodes (players) are defined as arrays. The vessels' playing field to be exchanged will contain a value of -1 for each element if no player has attempted to play a move on that field. This value will be 0 if any of the players have already tried to play on this field and the result was a miss (no vessel). Otherwise, the value will be set to the serial number of the player who played on this field and hit part of the vessel. In addition to the playing field, we also define the size of the array, horizontally 20 (`areaSizeX`) and vertically 20 (`areaSizeY`), which then determine a one-dimensional array:

```
// data as shared for all players
#define areaSizeX 20
#define areaSizeY 20
#define NUM_PLAYERS 9
int sharedMapOfHits[areaSizeY*areaSizeX]; // as 1-D array,
   ↪ otherwise serialize the data
int gameMove[2];
```

The player code contains one call to `MPI_Recv` and one call to `MPI_Send`, which are repeated in a loop until the end of the game. The logic of the player is simple – sets its seed to a random number generator with a function call `srand()`, and plays a random move that was not played by other players:

```
void rankN(int N) {
   MPI_Status status;


   srand(N);

   while (1) {
      MPI_Recv(sharedMapOfHits, areaSizeY*areaSizeX, MPI_INT,
         ↪ 0, 42, MPI_COMM_WORLD, &status);
      if (sharedMapOfHits[0] == -42) // game over?
         return;

      do { // select an empty (non-played) spot as a new move
         gameMove[0] = rand() % areaSizeY;
         gameMove[1] = rand() % areaSizeX;
      } while (sharedMapOfHits[gameMove[0]*areaSizeX +
         ↪ gameMove[1]] != -1);

      MPI_Send(gameMove, 2, MPI_INT, 0, 42, MPI_COMM_WORLD);
   }
}
```

We store more detailed information about the game on the main node, on the basis of which we will be informing the players about the state of the game. In addition to hits, we also keep ship locations here:

```
// data & functions only for the judge (game master)
int shipsMap[areaSizeY][areaSizeX];
int privateMapOfHitsAndAttacks[areaSizeY][areaSizeX];
```

The code of the main node is divided further into 3 more additional functions for the sake of clarity, and these are called from the master node code (function `void rank0()`) that follows below. All three of these functions also use the data stored by the main node (judge). In the basic function of the main node, we merely loop the game moves for scenarios generated in `void generateNewBattleshipGameScenario()` function, and then, in this loop, every second (call to `sleep(1)`, for which we included the library `unistd.h` at the beginning of the program) we ask the players for moves (call `MPI_Recv` and `MPI_Send`). During the game, we collect and print the statistics of the game state via the call to the function `int applyGameMoveAndCheckGameOver(int x, int y, int player)` and check whether the game is over:

```
void generateNewBattleshipGameScenario();
int applyGameMoveAndCheckGameOver(int x, int y, int player);
void publishGameAsHTML(int isGameOver);

void rank0() {
    MPI_Status status;

    generateNewBattleshipGameScenario();

    while (1)
        for (int player = 0; player < NUM_PLAYERS; player++) {
            // send shared game state to player
            MPI_Send(sharedMapOfHits, areaSizeY*areaSizeX,
                ↪ MPI_INT, 1 + player, 42, MPI_COMM_WORLD);

            // receive move from player: (y, x) to play
            MPI_Recv(gameMove, 2, MPI_INT, 1 + player, 42,
                ↪ MPI_COMM_WORLD, &status);

            if (applyGameMoveAndCheckGameOver(gameMove[1],
                ↪ gameMove[0], player)) {
                for (int player = 0; player < NUM_PLAYERS; player
                    ↪ ++) {
```

```
                    sharedMapOfHits[0] = -42; // end of game
                        ↪ message for players
                    MPI_Send(sharedMapOfHits, areaSizeY*areaSizeX
                        ↪ , MPI_INT, 1 + player, 42,
                        ↪ MPI_COMM_WORLD);
                }
                return; // game over
            }

            sleep(1);
        }
}
```

The function for generating the game scenario arranges vessels in random locations in such a way that that they do not intersect and touch. Vessel data are stored in the variable `shipsMap`, for which an element of the array is set at value -1 if there is no vessel at a certain location, otherwise it is set to the value of the vessel type:

```
#define NUM_SHIP_TYPES 5 // Carrier, Battleship, Cruiser,
    ↪ Submarine, Destroyer
const int shipsCountPerType[NUM_SHIP_TYPES] = {1, 2, 3, 4, 5};
const int shipsSizesPerType[NUM_SHIP_TYPES] = {5, 4, 3, 3, 2};

// places ships on the battleground in such manner that
// the stips do not come close one to another or cross
void generateNewBattleshipGameScenario() {
    for (int y = 0; y < areaSizeY; y++) // initialize to empty
        ↪ map
        for (int x = 0; x < areaSizeX; x++) {
            sharedMapOfHits[y*areaSizeX + x] = -1;
            shipsMap[y][x] = -1;
            privateMapOfHitsAndAttacks[y][x] = -1;
        }

    for (int t = 0; t < NUM_SHIP_TYPES; t++)
        for (int s = 0; s < shipsCountPerType[t]; s++)
            while (1) {
                int randX = rand() % areaSizeX;
                int randY = rand() % areaSizeY;
                int direction = rand() % 2; // 0=X; 1=Y

                int checkAreaIsEmptyOK = 1;
                for (int border = -1; border <= 1; border++) {
                    for (int i = -1; i < shipsSizesPerType[t] +
                        ↪ 1; i++) {
                        int iIdxX = !direction ? randX + i : randX
                            ↪  + border;
                        int iIdxY = direction ? randY + i : randY
                            ↪ + border;

                        if (iIdxY < 0 || iIdxY >= areaSizeY ||
                                iIdxX < 0 || iIdxX >= areaSizeX ||
                                shipsMap[iIdxY][iIdxX] > 0) {
                            checkAreaIsEmptyOK = 0;
                            border=2; break;
                        }
                    }
                }

                if (checkAreaIsEmptyOK) { // put ships on the map
                    ↪  of ships
                    for (int i = 0; i < shipsSizesPerType[t]; i
                        ↪ ++) {
                        int iIdxX = !direction ? randX + i : randX
                            ↪ ;
                        int iIdxY = direction ? randY + i : randY;

                        shipsMap[iIdxY][iIdxX] = 1 + t;
                    }
                    break;
                }
            }
}
```

The game status control, and monitoring if it is finished, works as follows:

```
// applies hit attempt and updates game state,
// including checking if game is over
int applyGameMoveAndCheckGameOver(int x, int y, int player) {
    if (privateMapOfHitsAndAttacks[y][x] == -1) // assert
        ↪ replayed hits
        privateMapOfHitsAndAttacks[y][x] = 1 + player;

    // reveal data to player about hit/misss from all players
    if (sharedMapOfHits[y * areaSizeX + x] == -1) // allow
        ↪ players to play same spot only once
        sharedMapOfHits[y * areaSizeX + x] = shipsMap[y][x] > 0
            ↪ ? 1 + player : 0;

    int isGameOver = 1;
    for (int y = 0; y < areaSizeY; y++)
        for (int x = 0; x < areaSizeX; x++)
            if (shipsMap[y][x] > 0 && privateMapOfHitsAndAttacks
                ↪ [y][x] == -1)
                isGameOver = 0;

    publishGameAsHTML(isGameOver);

    return isGameOver;
}
```

At the end of the program is a printout of statistics. We display the statistics of each player's points and the state of the game map, where we mark the played moves:

```
void publishGameAsHTML(int isGameOver) {
    // save leaderboard and check if game is over
    int scoresForPlayers[NUM_PLAYERS];
    for (int p = 0; p < NUM_PLAYERS; p++) scoresForPlayers[p] =
        ↪ 0;

    for (int y = 0; y < areaSizeY; y++)
        for (int x = 0; x < areaSizeX; x++)
            if (shipsMap[y][x] > 0 && privateMapOfHitsAndAttacks
                ↪ [y][x] > 0)
                scoresForPlayers[privateMapOfHitsAndAttacks[y][x]
                    ↪  - 1]++;

    FILE *fout;
    fout = fopen("/var/www/public_html/index.html", "w");

    static int numMoves = 0;
    fprintf(fout, "<!DOCTYPE html><html><head><style>"
                  ".table{display:table;font-size:20px;font-color
                      ↪ :blue;}"
                  ".row{display:table-row;}"
                  ".cell{display:table-cell; text-align:center;
                      ↪ border:1px solid black; width:30px;
                      ↪ height: 30px; }</style>"
                  "<meta http-equiv=\"refresh\" content=\"1\"></
                  ↪ head><body>"
                "<h1>N04-MPI-Cluster</h1>\n"
                "<div>Move #%d. Player scores: \n", numMoves++)
                  ↪ ;

    int scoreMax = 1;
    for (int p = 0; p < NUM_PLAYERS; p++) {
        fprintf(fout, "P%d: <b>%d</b> ", p, scoresForPlayers[p])
            ↪ ;
        if (scoresForPlayers[p] > scoreMax) scoreMax =
            ↪ scoresForPlayers[p];
    }

    fprintf(fout, "</div><div class=\"table\">\n");
    for (int y = 0; y < areaSizeY; y++) {
        fprintf(fout, "<div class=\"row\">\n");
        for (int x = 0; x < areaSizeX; x++) {
            fprintf(fout, "<div class=\"cell\"");

            if (shipsMap[y][x] == -1) {
                if (privateMapOfHitsAndAttacks[y][x] == -1)
                    ↪ fprintf(fout, ">");
                else fprintf(fout, "style=\"background-color: rgb
                    ↪ (0,%d,0);\">%d", 100+156/NUM_PLAYERS*
                    ↪ privateMapOfHitsAndAttacks[y][x],
                    ↪ privateMapOfHitsAndAttacks[y][x]);
            } else {
                if (privateMapOfHitsAndAttacks[y][x] == -1)
                    ↪ fprintf(fout, " style=\"background-color:
                    ↪ lightblue;\">");
                else fprintf(fout, " style=\"background-color:
                    ↪ red;\">%d", privateMapOfHitsAndAttacks[y][
                    ↪ x]);
            }

            fprintf(fout, "</div>\n");
        }
        fprintf(fout, "</div>\n");
    }
    fprintf(fout, "</div>");

    if (isGameOver) {
        fprintf(fout, "<h2>WINNER:");
        for (int p = 0; p < NUM_PLAYERS; p++)
            if (scoresForPlayers[p] == scoreMax) fprintf(fout, "
                ↪  P%d ", p);
        fprintf(fout, "with %d points.", scoreMax);
        fprintf(fout, "</h2>");
    }

    fprintf(fout, "</body></html>\n");
    fclose(fout);
}
```

7. We then run all the remaining cloned virtual environments:

```
$ for i in {2..10}; do
   vboxmanage startvm N04−MPI−Cluster−$i \
     −−type headless;
  done
```

8. We copy the file `N04-MPI-Cluster.c` to all virtual environments with the command:

```
$ for i in {3..12}; do
   scp N04−MPI−Cluster.c 192.168.57.$i:;
  done
```

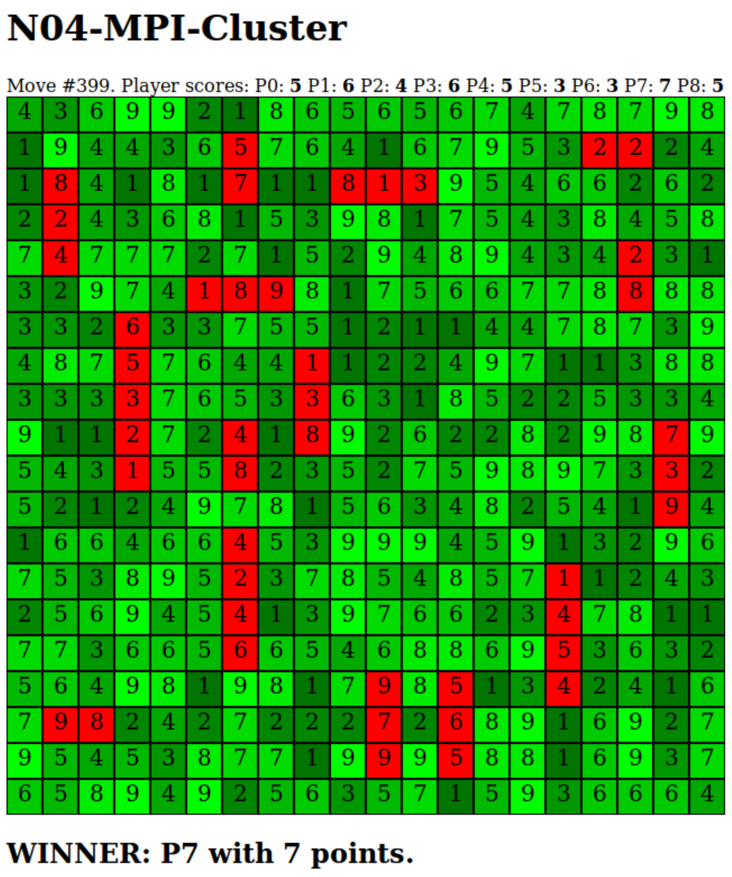
9. Then the code with OpenMPI in the file `main.c` is translated with the command:

```
$ parallel−ssh −i \
   $(echo "-H "192.168.57.{3..12}) \
   mpicc N04−MPI−Cluster.c \
     −o N04−MPI−Cluster
```

10. Finally, we run the programs with OpenMPI on all nodes by entering their names at startup:

```
$ time ssh 192.168.57.3 \
   mpiexec −n 10 \
     $(echo "-H "192.168.57.{3..12}) \
     −−mca btl\_base\_warn\_component\_unused 0 \
     N04−MPI−Cluster
```

During operation, the main node (`rank=0`) refreshes a file `gamestate.html` written to the web folder every second. If, at the end of the game, we open the web address for serving this file in a web browser via the web server, we get a rendering as shown in Figure 4.1.



N04-MPI-Cluster

Move #399. Player scores: P0: **5** P1: **6** P2: **4** P3: **6** P4: **5** P5: **3** P6: **3** P7: **7** P8: **5**

4	3	6	9	9	2	1	8	6	5	6	5	6	7	4	7	8	7	9	8
1	9	4	4	3	6	5	7	6	4	1	6	7	9	5	3	2	2	2	4
1	8	4	1	8	1	7	1	1	8	1	3	9	5	4	6	6	2	6	2
2	2	4	3	6	8	1	5	3	9	8	1	7	5	4	3	8	4	5	8
7	4	7	7	7	2	7	1	5	2	9	4	8	9	4	3	4	2	3	1
3	2	9	7	4	1	8	9	8	1	7	5	6	6	7	7	8	8	8	8
3	3	2	6	3	3	7	5	5	1	2	1	1	4	4	7	8	7	3	9
4	8	7	5	7	6	4	4	1	1	2	2	4	9	7	1	1	3	8	8
3	3	3	3	7	6	5	3	3	6	3	1	8	5	2	2	5	3	3	4
9	1	1	2	7	2	4	1	8	9	2	6	2	2	8	2	9	8	7	9
5	4	3	1	5	5	8	2	3	5	2	7	5	9	8	9	7	3	3	2
5	2	1	2	4	9	7	8	1	5	6	3	4	8	2	5	4	1	9	4
1	6	6	4	6	6	4	5	3	9	9	9	4	5	9	1	3	2	9	6
7	5	3	8	9	5	2	3	7	8	5	4	8	5	7	1	1	2	4	3
2	5	6	9	4	5	4	1	3	9	7	6	6	2	3	4	7	8	1	1
7	7	3	6	6	5	6	6	5	4	6	8	8	6	9	5	3	6	3	2
5	6	4	9	8	1	9	8	1	7	9	8	5	1	3	4	2	4	1	6
7	9	8	2	4	2	7	2	2	2	7	2	6	8	9	1	6	9	2	7
9	5	4	5	3	8	7	7	1	9	9	9	5	8	8	1	6	9	3	7
6	5	8	9	4	9	2	5	6	3	5	7	1	5	9	3	6	6	6	4

WINNER: P7 with 7 points.

Figure 4.1: Final result shown after executing exercise `N03-MPI-PingPong`.

11. Save the properly functioning virtual environments and put them in a saved suspended state:

```
$ for i in {1..10}; do
    vboxmanage controlvm N04-MPI-Cluster-$i \
      savestate;
  done
```

12. Export the OVA image for the created environments. The process takes some time because of the writing of a large file, e.g. approximately fifteen minutes. The exported file size is approximately 13 GB. During this time we see the progress of the process:

```
$ vboxmanage export N04-MPI-Cluster-{1..10} \
   -o N04-MPI-Cluster.ova
```

```
0%...10%...20%...30%...40%...50%...60%...70%
...80%...90%...100%
Successfully exported 10 machine(s).
```

13. Write down an exercise report (for example, in the same manner that the solution was described here in this material).

Chapter 5

Apache Hadoop, MapReduce, and YARN in a Single-node Virtual Environment: Deploying and Managing a Distributed System — Exercise N07-Hadoop-Single

Exercise: N07-Hadoop-Single

For a computer cloud, create the following system image (only 1 computer / VM virtual system) using VirtualBox.

In the created system image, a Hadoop cluster is deployed with a single node that demonstrates the working of the MapReduce method (run "hadoop-mapreduce-examples"). Along with Hadoop, install YARN. To implement this exercise, use Apache **single node instructions** (`https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html`, PDF: `tutorial-Hadoop-single.pdf`).

Submit a report about this exercise that demonstrates execution of the solution, its implementation, and scripts/code.

FOR HALF POINTS: MapReduce (without YARN)

tutorial-Hadoop-single.pdf (required login to e-study)
`https://estudij.um.si/pluginfile.php/368910/mod_assign/introattachment/0/tutorial-Hadoop-single.pdf?forcedownload=1`

Solution

We reuse the host computer again, which we have already prepared in the N01-Linux-VM exercise, where we have installed the VirtualBox software. We can also reuse the prepared virtual environment from the exercise N02-Linux-Cluster or, since we have already installed the multi-node web server Apache in the exercise N04-MPI-Cluster, we can start by upgrading the virtual environment `N04-MPI-Cluster-1`. As already mentioned, it is good to use procedures that can be upgraded and automated later when setting up systems, so Linux and the Bash Command Interpreter are used here again. We will also use a web server already located in the Ubuntu distribution. We will also use the server Apache Hadoop, which we will download from the web using the official Apache Hadoop website.

Attachment file of the solution: Computer virtual image in OVA

N07-Hadoop-Single.ova (password for `sudo`: `12345`)
https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng

1. Since we will only need one virtual environment in this task, we first clone the first virtual environment from the exercise N04-MPI-Cluster:

```
$ vboxmanage clonevm N04-MPI-Cluster-1 \
  --name N07-Hadoop-Single-1 --register
```

2. We then start the cloned virtual environment:

```
$ vboxmanage startvm N07-Hadoop-Single-1 \
    --type headless
```

3. Since we will need to access the web from the virtual environment to install the software, we will first switch to the Internet connection mode NAT:

```
$ vboxmanage controlvm N07-Hadoop-Single-1 \
  nic1 nat
```

4. Since we will be using Apache Hadoop, we will download the software package Apache Hadoop from the Apache Hadoop official website into a virtual environment for it. We login to the first virtual environment via the OpenSSH connection redirected to port 2001 and download the software package with the `wget` command; in this solution the version Apache Hadoop 2.9.2 is used:

```
$ ssh -p 2001 127.0.0.1 \
   'wget https://www.apache.si/hadoop/common/
      ↪ hadoop-2.9.2/hadoop-2.9.2.tar.gz'
```

5. Version Apache Hadoop 2.9.2 supports Java 7 and 8, therefore, we install the runtime environment for Java 8 in the software package `openjdk-8-jre-headless`:

```
$ ssh -p 2001 127.0.0.1 sudo -S \
   apt install openjdk-8-jre-headless
```

6. We reset the network setting to the host interface (`hostonly`):

```
$ vboxmanage controlvm N07-Hadoop-Single-1 \
    nic1 hostonly vboxnet1
```

7. We unpack the downloaded software package for Apache Hadoop to the home folder:

```
$ ssh 192.168.57.3 tar xvf hadoop-2.9.2.tar.gz
```

8. Let's set the environment variable `JAVA_HOME`, which for locally running Apache Hadoop, determines where to find the Java environment. This way, we will not have to set the variable over and over again after shell login. We also store the setting in the file `hadoop-env.sh`:

```
$ ssh 192.168.57.3 "echo -e export JAVA_HOME=/usr/lib/
   ↪ jvm/java-8-openjdk-amd64 >> ~/.bashrc"
$ ssh 192.168.57.3 "sed -i 's|export JAVA_HOME=\${
   ↪ JAVA_HOME}|export JAVA_HOME=/usr/lib/jvm/java-8-
   ↪ openjdk-amd64|g' hadoop-2.9.2/etc/hadoop/hadoop-env.
   ↪ sh"
```

9. We now set the layout for pseudo-distributed Apache Hadoop execution on one node. We write the files `core-site.xml` and `hdfs-site.xml`:

```
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/
   ↪ core-site.xml <<EOF
<configuration>
   <property>
      <name>fs.defaultFS</name>
      <value>hdfs://localhost:9000</value>
   </property>
</configuration>
EOF'
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/
   ↪ hdfs-site.xml <<EOF
<configuration>
   <property>
      <name>dfs.replication</name>
      <value>1</value>
   </property>
</configuration>
EOF'
```

10. We format the file system for Apache Hadoop, Hadoop Distributed File System (HDFS).

```
$ ssh 192.168.57.3 'bash -ic "hadoop-2.9.2/bin/hdfs
   ↪ namenode -format"'
```

11. For running Apache Hadoop locally we add a local key OpenSSH among trustworthy:

```
$ ssh 192.168.57.3 ssh-keyscan -H 127.0.0.1
```

12. We can now test the locally running Apache Hadoop, and in a Bash command use the switch `-i`, which reads the above setting for `JAVA_HOME`. Let's run the server for the NameNode and the DataNode:

```
$ ssh 192.168.57.3 hadoop-2.9.2/sbin/start-dfs.sh
```

The command starts the servers and prints the following response:

```
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/ales/
   ↪ hadoop-2.9.2/logs/hadoop-ales-namenode-n01-
   ↪ linux-vm.out
localhost: starting datanode, logging to /home/ales/
   ↪ hadoop-2.9.2/logs/hadoop-ales-datanode-n01-
   ↪ linux-vm.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/
   ↪ ales/hadoop-2.9.2/logs/hadoop-ales-
   ↪ secondarynamenode-n01-linux-vm.out
```

13. Let's test the operation of the web interface. This can be done by tunnelling the network traffic between the virtual environment and the host computer, and then visiting the web address `http://localhost:50070/`, and we get a rendering of the web page as seen in Figure 5.1 on the next page.

```
$ ssh 192.168.57.3 -L50070:localhost:50070
```

14. We create new directories to start MapReduce loads:

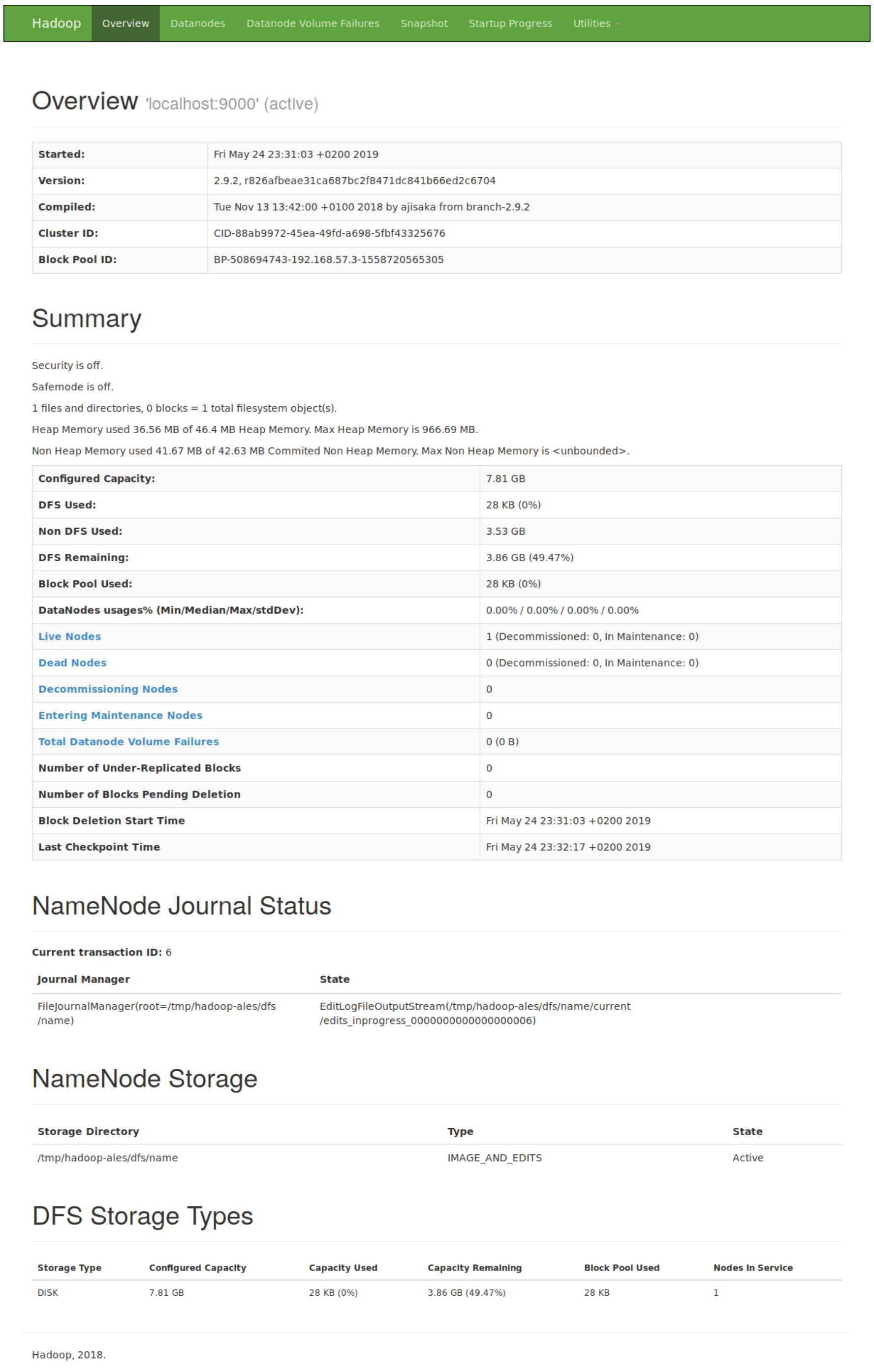
```
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hdfs dfs -
   ↪ mkdir /user
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hdfs dfs -
   ↪ mkdir /user/ales
```

15. We copy the input files into a distributed file system:

```
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hdfs dfs -put
   ↪  hadoop-2.9.2/etc/hadoop input
```

16. We can now run an example with MapReduce, as required in this exercise:

```
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hadoop jar
   ↪ hadoop-2.9.2/share/hadoop/mapreduce/hadoop-
   ↪ mapreduce-examples-2.9.2.jar grep input
   ↪ output 'dfs[a-z.]+'
```



Overview 'localhost:9000' (active)

Started:	Fri May 24 23:31:03 +0200 2019
Version:	2.9.2, r826afbeae31ca687bc2f8471dc841b66ed2c6704
Compiled:	Tue Nov 13 13:42:00 +0100 2018 by ajisaka from branch-2.9.2
Cluster ID:	CID-88ab9972-45ea-49fd-a698-5fbf43325676
Block Pool ID:	BP-508694743-192.168.57.3-1558720565305

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 36.56 MB of 46.4 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 41.67 MB of 42.63 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	7.81 GB
DFS Used:	28 KB (0%)
Non DFS Used:	3.53 GB
DFS Remaining:	3.86 GB (49.47%)
Block Pool Used:	28 KB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	0
Number of Blocks Pending Deletion	0
Block Deletion Start Time	Fri May 24 23:31:03 +0200 2019
Last Checkpoint Time	Fri May 24 23:32:17 +0200 2019

NameNode Journal Status

Current transaction ID: 6

Journal Manager	State
FileJournalManager(root=/tmp/hadoop-ales/dfs/name)	EditLogFileOutputStream(/tmp/hadoop-ales/dfs/name/current/edits_inprogress_0000000000000000006)

NameNode Storage

Storage Directory	Type	State
/tmp/hadoop-ales/dfs/name	IMAGE_AND_EDITS	Active

DFS Storage Types

Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes In Service
DISK	7.81 GB	28 KB (0%)	3.86 GB (49.47%)	28 KB	1

Hadoop, 2018.

Figure 5.1: Screenshot of the web page for the NameNode in exercise `N07-Hadoop-Single`.

The command is executed and a longer printout is displayed, ending with the following lines:

```
      File System Counters
            FILE: Number of bytes read=1339410
            FILE: Number of bytes written=3084996
            FILE: Number of read operations=0
            FILE: Number of large read operations=0
            FILE: Number of write operations=0
            HDFS: Number of bytes read=177604
            HDFS: Number of bytes written=1196
            HDFS: Number of read operations=151
            HDFS: Number of large read operations=0
            HDFS: Number of write operations=16
      Map-Reduce Framework
            Map input records=13
            Map output records=13
            Map output bytes=298
            Map output materialized bytes=330
            Input split bytes=129
            Combine input records=0
            Combine output records=0
            Reduce input groups=5
            Reduce shuffle bytes=330
            Reduce input records=13
            Reduce output records=13
            Spilled Records=26
            Shuffled Maps =1
            Failed Shuffles=0
            Merged Map outputs=1
            GC time elapsed (ms)=22
            Total committed heap usage (bytes)=271884288
      Shuffle Errors
            BAD_ID=0
            CONNECTION=0
            IO_ERROR=0
            WRONG_LENGTH=0
            WRONG_MAP=0
            WRONG_REDUCE=0
      File Input Format Counters
            Bytes Read=488
      File Output Format Counters
            Bytes Written=220
```

17. The printout can be copied from HDFS at Apache Hadoop to the local file system on a virtual environment, into the folder `output`[6]:

```
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hdfs dfs -get
   ↪ output output
$ ssh 192.168.57.3 cat output/*
```

[6]We see the same printout if we run the command `hadoop-2.9.2/bin/hdfs dfs -cat output/*` on the distributed file system itself.

Which prints the following result to the shell:

```
6 dfs.audit.logger
4 dfs.class
3 dfs.logger
3 dfs.server.namenode.
2 dfs.audit.log.maxbackupindex
2 dfs.period
2 dfs.audit.log.maxfilesize
1 dfs.log
1 dfs.file
1 dfs.servers
1 dfsadmin
1 dfsmetrics.log
1 dfs.replication
```

18. Since we want to add YARN, we first configure the file settings for it in **mapred-site.xml** and **yarn-site.xml**:

```
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/
   ↪ mapred-site.xml <<EOF
<configuration>
   <property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
   </property>
</configuration>
EOF'
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/yarn
   ↪ -site.xml <<EOF
<configuration>
   <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
   </property>
</configuration>
EOF'
```

19. We run Yet Another Resource Negotiator (YARN), with the following command:

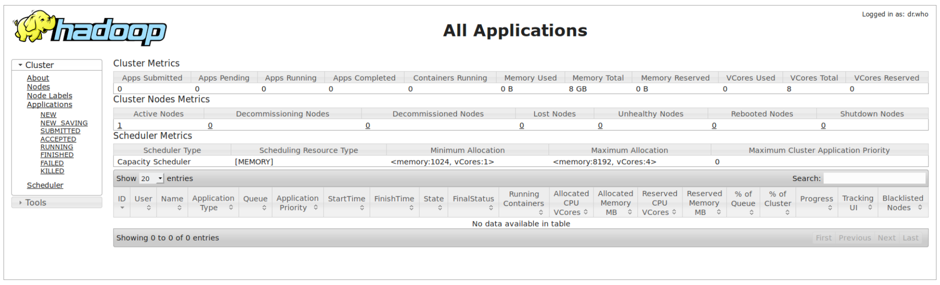
```
$ ssh 192.168.57.3 hadoop-2.9.2/sbin/start-yarn.
   ↪ sh
```

Running the command in the shell displays the following response:

```
starting yarn daemons
starting resourcemanager, logging to /home/ales/hadoop
    ↪ -2.9.2/logs/yarn-ales-resourcemanager-n01-linux
    ↪ -vm.out
localhost: starting nodemanager, logging to /home/ales
    ↪ /hadoop-2.9.2/logs/yarn-ales-nodemanager-n01-
    ↪ linux-vm.out
```

20. In order to access the website through the web interface of the YARN tool (as seen in Figure 5.2), we use the OpenSSH tunnel again:

```
$ ssh 192.168.57.3 -L8088:localhost:8088
```



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	8 GB	0 B	0	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
No data available in table																			

Showing 0 to 0 of 0 entries

Figure 5.2: The YARN website displayed running inside of the Apache Hadoop installation in exercise `N07-Hadoop-Single`.

21. Save the properly functioning virtual environment `N07-Hadoop-Single-1` and put it in a saved suspended state:

```
$ vboxmanage controlvm N07-Hadoop-Single-1 \
    savestate
```

22. Export the OVA image for the created environment. The exported file is approximately 2 GB. During this time we see the progress of the process:

```
$ vboxmanage export N07-Hadoop-Single-1 \
    -o N07-Hadoop-Single.ova
```

```
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully exported 1 machine(s).
```

23. Write down an exercise report (for example, in the same manner that the solution was described here in this material).

Chapter 6

Multiple Nodes with Hadoop — Exercise N08-Hadoop-Cluster

Exercise: N08-Hadoop-Cluster

For a computer cloud, create the following system image (only 1 computer / VM virtual system) using VirtualBox.

In the system image, a Hadoop cluster is deployed to support multiple nodes, that demonstrates the execution of the MapReduce method (run "hadoop-mapreduce-examples"). Along with Hadoop, install YARN. To implement this task, use Apache **instructions for running multiple nodes**: `https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html`. Use the tutorial, until the heading "*Configuring the Data Nodes*", at "tutorial-Hadoop-multi.pdf" (PDF printed from `https://dwbi.org/etl/bigdata/183-setup-hadoop-cluster`).

Submit a report about this exercise that demonstrates execution of the solution, its implementation, and scripts/code.

FOR HALF POINTS: MapReduce (without YARN)

tutorial-Hadoop-multi.pdf (required login to e-study)
`https://estudij.um.si/pluginfile.php/368920/mod_assign/introattachment/0/tutorial.pdf?forcedownload=1`

Solution

We reuse the host computer again, which we have already prepared in the N01-Linux-VM exercise, where we have the VirtualBox software installed. We can also reuse the prepared virtual environment after the exercise N02-Linux-Cluster, already installed as a multi-node web server Apache in the exercise N04-MPI-Cluster, where we installed Apache web server and multiple nodes. Since we have already set up Apache Hadoop and YARN with a single node in the N07-Hadoop-Single exercise, we will design the solution by upgrading the `N07-Hadoop-Single-1` virtual environment. As already mentioned, it is good to use procedures that can be upgraded and later automated when setting up systems, so Linux and the Bash Command Interpreter are used here again.

Attachment file of the solution: Computer virtual image in OVA

File: N07-Hadoop-Single.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`

1. Since, in this exercise, we will be upgrading the first virtual environment from exercise N07-Hadoop-Single, we will first clone the first virtual environment from the exercise N07-Hadoop-Single:

```
$ vboxmanage clonevm N07-Hadoop-Single-1 \
   --name N08-Hadoop-Cluster-1 --register
```

2. Then we start the cloned virtual environment:

```
$ vboxmanage startvm N08-Hadoop-Cluster-1 \
     --type headless
```

3. The Apache Hadoop with a single node configuration and YARN server are currently running in a virtual environment. As we are going to change the settings, we first stop both of these:

```
$ ssh 192.168.57.3 hadoop-2.9.2/sbin/stop-dfs.sh
$ ssh 192.168.57.3 hadoop-2.9.2/sbin/stop-yarn.sh
```

4. We will now prepare this node for deployment for true distributed implementation of Apache Hadoop on multiple nodes. Each node will be contained in its own virtual environment. As already mentioned

in the N02-Linux-Cluster task, virtual nodes could also be hosted at different physical host computers, which could also be hosted on different physical locations (connected via an IP network). However, in this solution, we assume that we will host virtual environments on only one host, who will virtualise the network and provide for the environment the host network `vboxnet1` (addresses IP 192.168.57.0/24). Any VirtualBox virtual environment as a node for Apache Hadoop service will, hence, in the network `vboxnet1`, have, as in previous exercises, its own leashed IP, assigned via the DHCP server for network `vboxnet1` in VirtualBox. These addresses are assigned sequentially, as already shown in previous tasks.

NameNodes for Apache Hadoop, which we link to the assigned IP addresses on the network `vboxnet1`, we merely enter in the file `/etc/hosts`:

```
$ ssh 192.168.57.3 'sudo -S bash -c "(
echo 192.168.57.3 NameNode;
echo 192.168.57.4 DataNode1
echo 192.168.57.5 DataNode2) >> /etc/hosts"'
```

5. For new NameNodes, we add the option for login among nodes via the OpenSSH keys:

```
$ for H in NameNode DataNode1 DataNode2; do
    for i in {3..5}; do
      ssh 192.168.57.$i "ssh-keyscan $H >> ~/.ssh/
          ↪ known_hosts";
    done; done
```

6. Now we configure the layout for the real distributed execution of Apache Hadoop on several nodes. First, we write the file `core-site.xml`:

```
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/core-site.xml <<EOF
<?xml version="1.0"?>
<!-- core-site.xml -->
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://NameNode:8020/</value>
</property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
</property>
</configuration>
EOF'
```

We also write the file **`hdfs-site.xml`**:

```
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/
   ↪ hdfs-site.xml <<EOF
<?xml version="1.0"?>
<!-- hdfs-site.xml -->
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/home/ales/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/home/ales/hdfs/datanode</value>
</property>
<property>
<name>dfs.namenode.checkpoint.dir</name>
<value>file:/home/ales/hdfs/namesecondary</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.block.size</name>
<value>134217728</value>
</property>
</configuration>
EOF'
```

7. Enter the MapReduce settings in the file `mapred-site.xml`:

```
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/
   ↪ mapred-site.xml <<EOF
<?xml version="1.0"?>
<!-- mapred-site.xml -->
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.jobhistory.address</name>
<value>NameNode:10020</value>
</property>
<property>
<name>mapreduce.jobhistory.webapp.address</name>
<value>NameNode:19888</value>
</property>
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/user/app</value>
</property>
<property>
<name>mapred.child.java.opts</name>
<value>-Djava.security.egd=file:/dev/../dev/
   ↪ urandom</value>
</property>
</configuration>
EOF'
```

8. Enter the settings for YARN to a file `yarn-site.xml`:

```
$ ssh 192.168.57.3 'cat > hadoop-2.9.2/etc/hadoop/yarn-site.xml <<EOF
<?xml version="1.0"?>
<!-- yarn-site.xml -->
<configuration>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>NameNode</value>
</property>
<property>
<name>yarn.resourcemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<property>
<name>yarn.nodemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.log-aggregation-enable</name>
<value>true</value>
</property>
<property>
<name>yarn.nodemanager.local-dirs</name>
<value>file:/usr/local/hadoop_work/yarn/local</value>
</property>
<property>
<name>yarn.nodemanager.log-dirs</name>
<value>file:/usr/local/hadoop_work/yarn/log</value>
</property>
<property>
<name>yarn.nodemanager.remote-app-log-dir</name>
<value>hdfs://NameNode:8020/var/log/hadoop-yarn/apps</value>
</property>
</configuration>
EOF'
```

9. Configure the division of the main (NameNode) and other (DataNode) nodes:

```
$ ssh 192.168.57.3 'echo NameNode > hadoop-2.9.2/
    ↪ etc/hadoop/masters'
$ ssh 192.168.57.3 'echo -e DataNode1\\nDataNode2
    ↪ > hadoop-2.9.2/etc/hadoop/slaves'
```

10. The prepared virtual environment is cloned twice so that we get the two slave DataNodes (`DataNode1` and `DataNode2`). For this purpose, we use the first created virtual environment, which we put to sleep mode and then wake up:

```
$ vboxmanage controlvm N08-Hadoop-Cluster-1\
   savestate
$ vboxmanage clonevm N08-Hadoop-Cluster-1 \
   --name N08-Hadoop-Cluster-2 --register
$ vboxmanage clonevm N08-Hadoop-Cluster-1 \
   --name N08-Hadoop-Cluster-3 --register
$ for i in {1..3}; do vboxmanage \
   startvm N08-Hadoop-Cluster-$i \
     --type headless; done
```

11. Let's start Apache Hadoop and YARN, configured for multiple nodes.

```
$ ssh 192.168.57.3 hadoop-2.9.2/sbin/start-dfs.sh
$ ssh 192.168.57.3 hadoop-2.9.sbin/start-yarn.sh
```

Command printouts:

```
$ Starting namenodes on [NameNode]
NameNode: starting namenode, logging to /home/ales/hadoop-2.9.2/logs/hadoop-ales-namenode-n01-
     ↪ linux-vm.out
DataNode1: starting datanode, logging to /home/ales/hadoop-2.9.2/logs/hadoop-ales-datanode-n01-
     ↪ linux-vm.out
DataNode2: starting datanode, logging to /home/ales/hadoop-2.9.2/logs/hadoop-ales-datanode-n01-
     ↪ linux-vm.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/ales/hadoop-2.9.2/logs/hadoop-ales-
     ↪ secondarynamenode-n01-linux-vm.out
```

12. We format the file system for Apache Hadoop, Hadoop Distributed File System (HDFS).

```
$ ssh 192.168.57.3 'sudo -S bash -ic "hadoop-2.9.2/
   ↪ bin/hdfs namenode -format"'
```

The command prints a longer text, within which we can trace the following lines:

```
19/05/31 08:31:07 INFO common.Storage: Storage directory /home/ales/hdfs/namenode has been
     ↪ successfully formatted.
19/05/31 08:31:07 INFO namenode.FSImageFormatProtobuf: Saving image file /home/ales/hdfs/namenode/
     ↪ current/fsimage.ckpt_0000000000000000000 using no compression
19/05/31 08:31:07 INFO namenode.FSImageFormatProtobuf: Image file /home/ales/hdfs/namenode/current/
     ↪ fsimage.ckpt_0000000000000000000 of size 323 bytes saved in 0 seconds .
```

13. To start MapReduce loads, we create new directories:

```
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hdfs dfs -mkdir /user
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hdfs dfs -mkdir /user/ales
```

14. We copy the input files to the distributed file system:

```
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hdfs dfs -put
   ↪  hadoop-2.9.2/etc/hadoop input
```

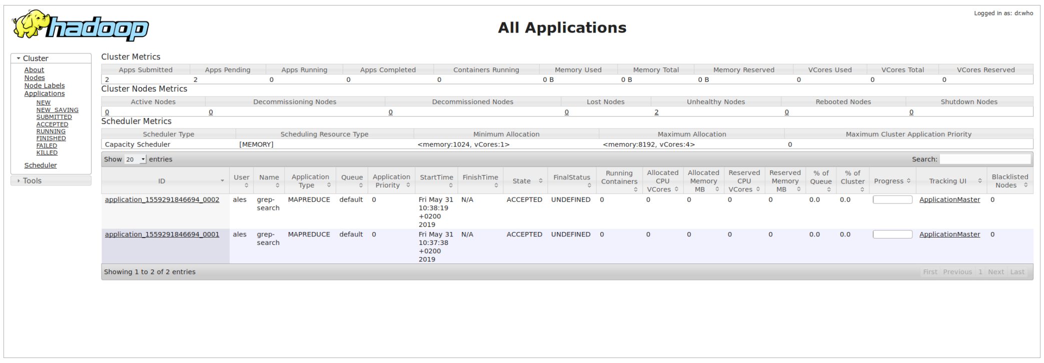
15. We can now run the MapReduce example as required in this exercise:

```
$ ssh 192.168.57.3 hadoop-2.9.2/bin/hadoop jar
   ↪ hadoop-2.9.2/share/hadoop/mapreduce/hadoop-
   ↪ mapreduce-examples-2.9.2.jar grep input
   ↪ output 'dfs[a-z.]+'
```

The command is executed, and a longer printout is displayed, ending with the following lines:

```
19/05/31 08:38:16 INFO client.RMProxy: Connecting to ResourceManager at NameNode/192.168.57.3:8032
19/05/31 08:38:17 INFO input.FileInputFormat: Total input files to process : 31
19/05/31 08:38:18 INFO mapreduce.JobSubmitter: number of splits:31
19/05/31 08:38:18 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.
      ↪ enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
19/05/31 08:38:19 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1559291846694_0002
19/05/31 08:38:19 INFO impl.YarnClientImpl: Submitted application application_1559291846694_0002
19/05/31 08:38:19 INFO mapreduce.Job: The url to track the job: http://NameNode:8088/proxy/
      ↪ application_1559291846694_0002/
19/05/31 08:38:19 INFO mapreduce.Job: Running job: job_1559291846694_0002
```

16. We open the web address http://192.168.57.3:8088/cluster in the local web browser (Figure 6.1), where we can see the running applications. At the web address http://192.168.57.3:50070 (Figure 6.2) we see additional statistics for the deployed cloud.



All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
2	2	0	0	0	0 B	0 B	0 B	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
0	0	0	0	2	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

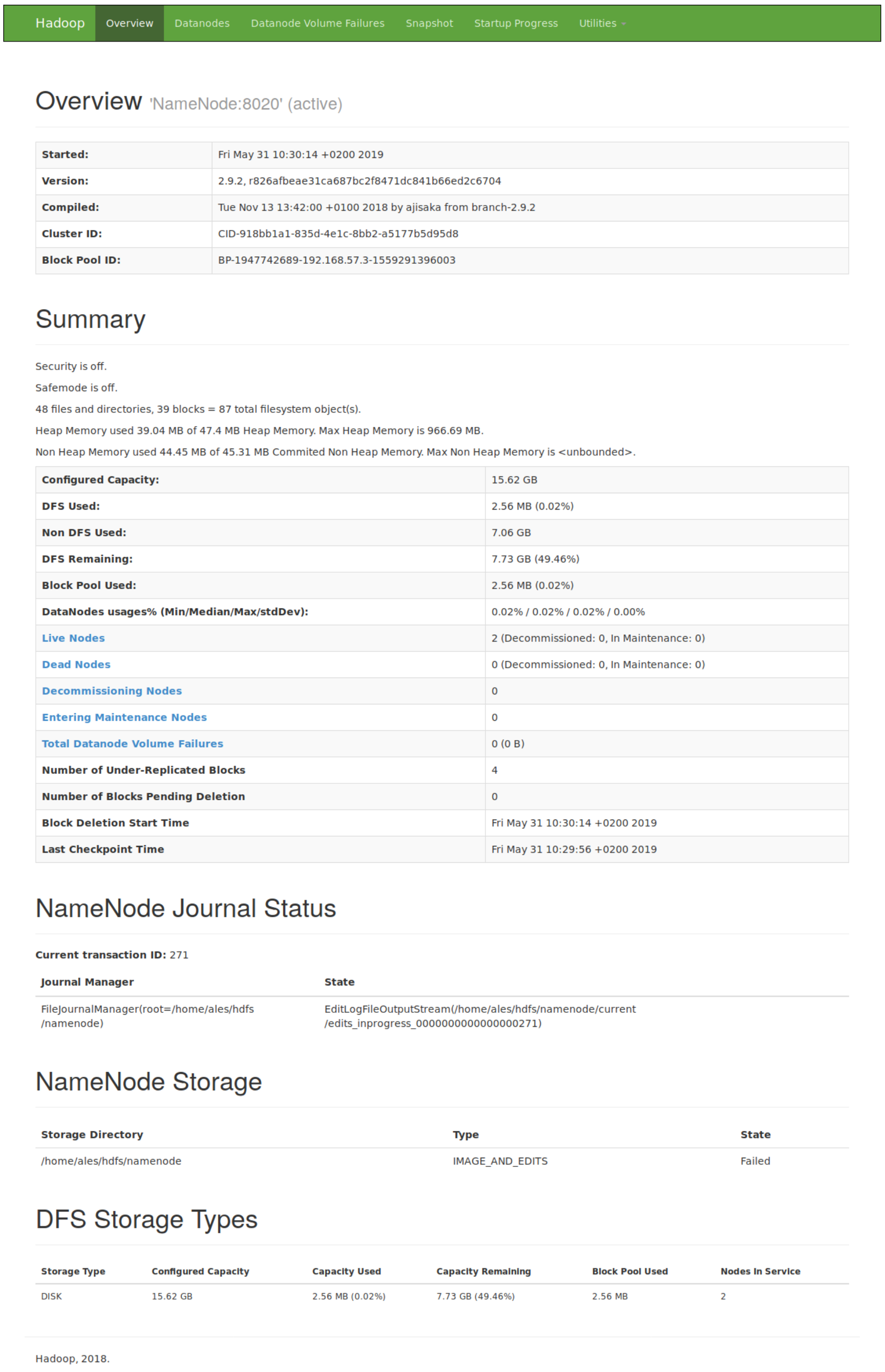
Show 20 entries Search:

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
application_1559291846694_0002	ales	grep-search	MAPREDUCE	default	0	Fri May 31 10:38:19 +0200 2019	N/A	ACCEPTED	UNDEFINED	0	0	0	0	0	0.0	0.0		ApplicationMaster	0
application_1559291846694_0001	ales	grep-search	MAPREDUCE	default	0	Fri May 31 10:37:38 +0200 2019	N/A	ACCEPTED	UNDEFINED	0	0	0	0	0	0.0	0.0		ApplicationMaster	0

Showing 1 to 2 of 2 entries

First Previous 1 Next Last

Figure 6.1: Screenshot from the app information web page YARN with multiple nodes.



Overview 'NameNode:8020' (active)

Started:	Fri May 31 10:30:14 +0200 2019
Version:	2.9.2, r826afbeae31ca687bc2f8471dc841b66ed2c6704
Compiled:	Tue Nov 13 13:42:00 +0100 2018 by ajisaka from branch-2.9.2
Cluster ID:	CID-918bb1a1-835d-4e1c-8bb2-a5177b5d95d8
Block Pool ID:	BP-1947742689-192.168.57.3-1559291396003

Summary

Security is off.

Safemode is off.

48 files and directories, 39 blocks = 87 total filesystem object(s).

Heap Memory used 39.04 MB of 47.4 MB Heap Memory. Max Heap Memory is 966.69 MB.

Non Heap Memory used 44.45 MB of 45.31 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	15.62 GB
DFS Used:	2.56 MB (0.02%)
Non DFS Used:	7.06 GB
DFS Remaining:	7.73 GB (49.46%)
Block Pool Used:	2.56 MB (0.02%)
DataNodes usages% (Min/Median/Max/stdDev):	0.02% / 0.02% / 0.02% / 0.00%
Live Nodes	2 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0
Entering Maintenance Nodes	0
Total Datanode Volume Failures	0 (0 B)
Number of Under-Replicated Blocks	4
Number of Blocks Pending Deletion	0
Block Deletion Start Time	Fri May 31 10:30:14 +0200 2019
Last Checkpoint Time	Fri May 31 10:29:56 +0200 2019

NameNode Journal Status

Current transaction ID: 271

Journal Manager	State
FileJournalManager(root=/home/ales/hdfs /namenode)	EditLogFileOutputStream(/home/ales/hdfs/namenode/current /edits_inprogress_0000000000000000271)

NameNode Storage

Storage Directory	Type	State
/home/ales/hdfs/namenode	IMAGE_AND_EDITS	Failed

DFS Storage Types

Storage Type	Configured Capacity	Capacity Used	Capacity Remaining	Block Pool Used	Nodes In Service
DISK	15.62 GB	2.56 MB (0.02%)	7.73 GB (49.46%)	2.56 MB	2

Hadoop, 2018.

Figure 6.2: Screenshot from the information web page on Apache Hadoop with multiple nodes.

17. Save the properly functioning virtual environments and put them in a saved suspended state:

```
$ for i in {1..3}; do vboxmanage \
   controlvm N08—Hadoop—Cluster—$i \
     savestate; done
```

18. Export the OVA image for the created environment. The exported file size is approximately 6 GB. During this time we see the progress of the process:

```
$ vboxmanage export N08—Hadoop—Cluster—{1..3} \
   —o N08—Hadoop—Cluster.ova
```

```
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully exported 3 machine(s).
```

19. Write down an exercise report (for example, in the same manner that the solution was described here in this material).

Chapter 7

Conclusion

An overview of collected study materials was provided through the chapters of this document, intended for computer exercises in the subject Cloud Computing Deployment and Management, included in the second-cycle Bologna Study Programme Computer Science and Information Technologies as a full-time study unit. In the continuation of the work within the preparation of study materials, an extended collection of additional exercises and study materials is especially planned for this subject in the form of a textbook.

The next page provides the attachments in which the solutions to the exercises are given in computerised form. Due to limitations, some files are divided into several smaller files. After unpacking, from these OVA images are obtained.

List of e-material linked attachments at DKUM

These attachments were archived in the Slovenian version, links are below.

File: N01-Linux-VM.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`
ZIP: N01-Linux-VM.zip

File: N02-Linux-Cluster.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`
ZIP: N02-Linux-Cluster.zip, N02-Linux-Cluster.z01, N02-Linux-Cluster.z02, N02-Linux-Cluster.z03, N02-Linux-Cluster.z04, N02-Linux-Cluster.z05

File: N03-MPI-PingPong.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`
ZIP: N03-MPI-PingPong.zip, N03-MPI-PingPong.z01, N03-MPI-PingPong.z02, N03-MPI-PingPong.z03, N03-MPI-PingPong.z04, N03-MPI-PingPong.z05, N03-MPI-PingPong.z06

File: N04-MPI-Cluster.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`
ZIP: N04-MPI-Cluster.zip, N04-MPI-Cluster.z01, N04-MPI-Cluster.z02, N04-MPI-Cluster.z03, N04-MPI-Cluster.z04, N04-MPI-Cluster.z05, N04-MPI-Cluster.z06

File: N07-Hadoop-Single.ova (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`
ZIP: N07-Hadoop-Single.zip

File: N08-Hadoop-Cluster (password for `sudo`: `12345`)
`https://dk.um.si/IzpisGradiva.php?id=77676&lang=eng`
ZIP: N08-Hadoop-Cluster.zip, N08-Hadoop-Cluster.z01, N08-Hadoop-Cluster.z02

University of Maribor

Faculty of Electrical Engineering
and Computer Science