

PROFESSIONAL TRAINING REPORT
Sathyabama Institute of Science and
Technology (Deemed to be University)

Submitted in partial fulfilment of the requirements for the
award of Bachelor of Technology Degree in Information
Technology

By

Pranesh Priyanshu

(40120112)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI, TAMILNADU – 600119

April 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

(www.sathyabama.com)

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Pranesh Priyanshu (Reg. No. 40120112)** who carried out the project as a **"Network Configuration in Virtual Environments with VirtualBox and Ubuntu."** under our supervision from January 2023 to April 2024.

Internal Guide

Dr. P.Jeyanthi, M.E, Ph.D.

Head of the Department

Dr. R Subhashini, M.E., Ph.D.

Internal Examiner

External Examiner

DECLARATION

I **Pranesh Priyanshu (Reg. No. 40120112)** hereby declare that the Project Report entitled "**Network Configuration in Virtual Environments with VirtualBox and Ubuntu.**" is done by me under the guidance of **Dr. P.Jeyanthi, M.E, Ph.D.**, and is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 29/04/2023

Pranesh Priyanshu

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T Sasikala, M.E., Ph.D.**, Dean, School of Computing **Dr. R Subhashini, M.E., Ph.D.**, Head of the Department of Information Technology for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. P.Jeyanthi, M.E, Ph.D.**, for her valuable guidance, suggestions and constant encouragement that paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE OF COMPLETION

Ref.No.: 2 /FSPCC/BC001/04/2023



इलेक्ट्रॉनिकी एवं
सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF
ELECTRONICS AND
INFORMATION TECHNOLOGY
सत्यमेव जयते



COURSE COMPLETION LETTER

It is certified that

Pranesh Priyanshu

Sathyabama Institute of Science and Technology

has successfully undergone the Bridge Course Training Program on

Cloud Computing Technology

Under "FutureSkills PRIME" Programme

commenced from 11 Mar, 2023

V.S.M.

Shri. V. Sridharan
Chief Investigator
Cloud Computing Technology
Lead Resource Centre

L R PRAKASH Digitally signed by L R PRAKASH
Date: 2023.04.26 02:14:53
+05'30'

Cdr.(Retd.) L.R. Prakash
Senior Director
Centre Head
C-DAC, Chennai

ABSTRACT

Virtualization is a powerful technology that allows users to create multiple virtual environments on a single physical machine. This project aims to provide hands-on experience in setting up a virtual network using VirtualBox and Ubuntu. The project involves creating two virtual machines, one acting as the controller and the other as the compute node, and configuring them to communicate with each other using the "Bridged Adapter" network mode.

The project methodology includes the installation of VirtualBox and Ubuntu, followed by the creation of two virtual machines and the configuration of their network settings. The reachability between the virtual machines is tested using the ping command to ensure proper communication between them.

The outcome of this project is the creation of a virtual network that can be used for testing, development, and deployment of applications and services. This project provides a platform for experimentation with different network configurations and serves as a reference for future use. By gaining experience with virtualization and network configuration, users can apply this knowledge to real-world scenarios and enhance their skills in virtual infrastructure management.

TABLE OF CONTENT

Chapter No.	Title	Page No.
1	Introduction	1
1.1	Technologies Used	3
2	Problem Statement	4
2.1	Objectives	6
3	Material And Methods	7
4	Project Management	9
5	Methodology and Architecture	11
6	Aim and Scope	13
7	Implementation	15
8	Limitations	18
9	Solution	20
10	Future Works	22
11	Results and Discussion	25
12	References	32

List of Images

Sl. No	Figure Number	Figure Description
1	10.1	Setting up Controller VM node
2	10.2	Setting up Compute VM node
3	10.3	Controller VM
4	10.4	Compute VM
5	10.5	Setting up the Bridge in Controller VM
6	10.6	Setting up the Bridge in Compute VM
7	10.7	Rechability proved in Controller VM
8	10.8	Rechabilityh proved in Compute VM

Chapter 1

INTRODUCTION

Cloud computing has become a ubiquitous technology that has revolutionized the way businesses and individuals access and store their data. It provides a cost-effective and scalable solution to meet the demands of modern computing needs. Virtualization is one of the key technologies that underpins cloud computing. It enables the creation of multiple virtual machines (VMs) that can run different operating systems (OS) on a single physical machine. Virtualization technology has enabled businesses to consolidate their IT infrastructure, reduce operational costs, and improve the efficiency of their computing resources.

One of the most popular virtualization software available in the market today is VirtualBox. VirtualBox is a free and open-source virtualization software that enables users to create and run multiple VMs on a single physical machine. It is widely used by developers and system administrators to test software applications and configurations in a sandboxed environment. VirtualBox supports a wide range of operating systems, including Windows, Linux, macOS, and Solaris.

Ubuntu is one of the most popular Linux distributions used by developers and system administrators worldwide. It is known for its simplicity, stability, and security features. In this project, we aim to install Ubuntu on VirtualBox and create two VM nodes, a controller node, and a compute node. We will then configure the network for the nodes in Bridge mode and prove the reachability of the VM nodes with the ping command.

Virtualization is a powerful technology that allows users to create multiple virtual environments on a single physical machine, providing a platform for testing, development, and deployment of applications and services. Virtualization is widely used in the IT industry, enabling users to create and manage complex virtual infrastructure with ease. One of the most popular virtualization software is VirtualBox, which is an open-source tool that allows users to create and manage virtual machines on a host machine.

This project aims to provide a hands-on experience with virtualization and network configuration using VirtualBox and Ubuntu. The project involves the creation of two virtual machines, one acting as the controller and the other as the compute node, and configuring them to communicate with each other using the "Bridged Adapter" network mode. The project methodology includes the installation of VirtualBox and Ubuntu, followed by the creation of the virtual machines and the configuration of their network settings.

The primary objective of this project is to create a virtual network that can be used for testing, development, and deployment of applications and services. The network architecture consists of two virtual machines, one acting as the controller, and the other as the compute node. The virtual machines are configured to communicate with each other using the "Bridged Adapter" network mode, which enables them to communicate with other devices on the network. This project will be beneficial for beginners who are new to virtualization and cloud computing. It will also provide useful insights to experienced users who want to enhance their knowledge of VirtualBox and Ubuntu. Additionally, this project will demonstrate how virtualization technology can help businesses reduce their IT infrastructure costs and improve the efficiency of their computing resources.

1.1 Technologies Used

1. VirtualBox: VirtualBox is an open-source virtualization software that allows users to create and manage virtual machines on a host machine. It provides a platform for testing, development, and deployment of applications and services in a safe and controlled environment.
2. Ubuntu: Ubuntu is a popular Linux distribution that is widely used for server deployments. It provides a stable and secure platform for running applications and services.
3. Networking: Networking is a key aspect of this project, as it involves configuring the virtual machines to communicate with each other using the "Bridged Adapter" network mode. This requires knowledge of network architecture and configuration.
4. Command Line Interface (CLI): The project involves working with the command line interface in Ubuntu to install and configure the necessary software and services.
5. Documentation: Documentation is an important aspect of this project, as it involves documenting the installation and configuration process for future reference and troubleshooting. This requires knowledge of documentation tools and best practices.

Chapter 2

PROBLEM STATEMENT

The field of cloud computing has been rapidly growing and evolving in recent years. Cloud computing provides many benefits to businesses, such as on-demand computing resources, scalability, and cost savings. However, managing these resources and ensuring their optimal utilization can be a challenge, particularly for small to medium-sized businesses with limited resources. Virtualization is a key technology in cloud computing that enables the efficient management and utilization of computing resources.

The main problem addressed by this project is the need for a cost-effective and easy-to-use virtualization solution that can be used by small to medium-sized businesses to manage their computing resources. While there are many virtualization solutions available on the market, many of them are either too expensive or too complex for these businesses to implement and manage. Additionally, many businesses may not have the expertise or resources to manage their virtualization infrastructure effectively.

This project aims to address this problem by providing a step-by-step guide on how to set up and configure a virtualization solution using Virtual Box and Ubuntu. The project will walk users through the process of creating virtual machines, configuring their network, and testing their connectivity. By providing a cost-effective and easy-to-use virtualization solution, this project can help small to medium-sized businesses to better manage their computing resources and improve their overall efficiency.

The rationale behind this project is that virtualization is an essential technology in cloud computing and can provide significant benefits to businesses of all sizes. However, the complexity and cost of many virtualization solutions can be a barrier to entry for small to medium-sized businesses. By providing a simple and cost-effective virtualization solution, this project can help to bridge this gap and enable more businesses to take advantage of the benefits of cloud computing. Additionally, by using open-source technologies such as Virtual Box and Ubuntu, this project promotes the use of free and open-source software, which can help to reduce costs and increase accessibility for businesses with limited resources.

The IT industry is constantly evolving, and there is a growing need for testing and deploying applications and services in a safe and controlled environment. However, setting up a physical network for testing and development can be costly and time-consuming. Virtualization provides a solution to this problem by allowing users to create virtual networks on a single physical machine. However, configuring and managing virtual networks can be challenging for users who lack experience in virtual infrastructure management.

The problem that this project aims to address is the lack of experience and knowledge in virtual infrastructure management. This project provides a platform for users to gain experience in virtualization and network configuration using VirtualBox and Ubuntu. The project methodology includes the creation of two virtual machines and the configuration of their network settings to enable communication between them.

The primary challenge of this project is to configure the virtual network settings correctly to ensure proper communication between the virtual

machines. This requires knowledge of network architecture and virtual infrastructure management. Additionally, troubleshooting any issues that arise during the installation and configuration process can be challenging for users who lack experience in virtualization.

2.1 OBJECTIVES

The nature of the work brought a lot of objectives to be completed. Some of the main objectives are mentioned below.

- Configure and manage a virtual network using VirtualBox and Ubuntu.
- Understand the different network modes available in VirtualBox and their use cases.
- Create two virtual machines (Controller and Compute) and configure them to communicate with each other.
- Configure the network settings for the virtual machines in "Bridged Adapter" mode to allow for communication with the host machine and other devices on the network.
- Install and configure OpenStack on the Controller node to enable cloud computing.
- Test the communication between the virtual machines using the ping command.
- Document the installation and configuration process for future reference and troubleshooting.
- Gain experience with the command line interface (CLI) in Ubuntu.
- Gain experience with networking concepts, including IP addressing and subnetting.

Chapter 3

MATERIALS AND METHODS

Materials: This section should list and briefly describe the hardware and software materials used in the project. The key materials used in this project include:

- VirtualBox: A virtualization software used to create and manage virtual machines.
- Ubuntu ISO image: An operating system used to create virtual machines.
- OpenStack: A cloud computing platform used to manage virtual infrastructure and resources.
- Network cables and adapters: Used to connect the virtual machines to the host machine and other devices on the network.

Methods: This section should provide a step-by-step explanation of the methods used to install and configure the virtual infrastructure. The key methods used in this project include:

- Installation of VirtualBox: This involved downloading the VirtualBox software from the official website and running the installation wizard on the host machine.
- Installation of Ubuntu: This involved downloading the Ubuntu ISO image and creating two virtual machines (Controller and Compute) using VirtualBox. The Ubuntu operating system was then installed on both virtual machines.
- Network configuration: The network settings for both virtual machines were configured in "Bridged Adapter" mode to allow for communication with the host machine and other devices on the network.
- Installation of OpenStack: OpenStack was installed and configured on the Controller node to enable cloud computing.

- Testing and verification: Communication between the virtual machines was tested using the ping command, and the successful installation and configuration of the virtual infrastructure was verified.
- Documentation: The installation and configuration process was documented to ensure future reference and troubleshooting.

Overall, the Materials and Methods section provides readers with a clear understanding of the materials used in the project and the methods employed to achieve the project objectives. This section is important as it allows others to replicate the project, learn from it, and build on it in future work.

Chapter 4

PROJECT MANAGEMENT

Project management is the practice of planning, executing, and controlling the resources, tasks, and deliverables of a project in order to achieve specific goals and objectives. It involves defining the scope of the project, developing a project plan, and tracking progress against the plan to ensure that the project is completed on time, within budget, and to the satisfaction of stakeholders.

In the context of this particular project, project management might involve the following tasks:

1. Defining project scope: The first step in project management is to define the scope of the project, which includes identifying the objectives, stakeholders, and deliverables.
2. Developing a project plan: The project plan should outline the tasks that need to be completed, the resources that will be required, and the timeline for completing each task. It should also identify any risks and the measures that will be taken to mitigate those risks.
3. Assigning responsibilities: Each task should be assigned to a specific person or team, along with a deadline for completion. This will help to ensure that everyone is clear on their responsibilities and that the project stays on track.
4. Monitoring progress: Regular check-ins should be scheduled to track progress against the project plan. This will allow any issues or delays to be identified early on, so that corrective action can be taken.
5. Managing resources: Project management also involves managing the resources required for the project, including people, equipment, and materials. This might involve coordinating with vendors, hiring contractors, or ensuring that equipment is available when needed.

6. Communication: Communication is a critical component of project management, and it involves keeping all stakeholders informed of project status, issues, and progress. This might include regular meetings, status reports, or other forms of communication.

By effectively managing the project, the team can ensure that the project is completed on time, within budget, and to the satisfaction of stakeholders. Project management can help to minimize risk, maximize resources, and ensure that the project achieves its intended objectives.

Chapter 5

METHODOLOGY AND ARCHITECTURE

5.1 Methodology

- **Planning:** The first step is to plan the project. This involves identifying the requirements, objectives, and constraints of the project. The project team should also identify the tools and technologies that will be used to implement the project.
- **Design:** The next step is to design the architecture of the project. This involves creating a high-level design that defines the major components of the system and how they will interact with each other.
- **Implementation:** The implementation phase involves developing the software components of the project. This might include coding, testing, and debugging the software components.
- **Testing:** Once the software components have been developed, they should be tested to ensure that they meet the requirements and objectives of the project.
- **Deployment:** The final step is to deploy the software components to the production environment. This might involve installing the software on servers, configuring the software components, and testing the system in the production environment.

5.2 Architecture

- **Virtualization software:** The virtualization software, such as VirtualBox, allows multiple virtual machines to run on a single physical machine.
- **Operating system:** The virtual machines will require an operating system, such as Ubuntu, to run.

- Network configuration: The virtual machines must be configured to communicate with each other using a network. The network might be configured in bridge mode, which allows the virtual machines to access the network as if they were physical machines.
- Controller and compute nodes: The project will require at least two virtual machines, one acting as the controller node and the other as the compute node.
- Software components: The software components of the project might include applications and tools for managing the virtual machines, such as OpenStack or Kubernetes.

Overall, the methodology and architecture of the project will be focused on creating a scalable and flexible virtualization infrastructure that can be used to deploy and manage cloud-based applications and services.

Chapter 6

AIM AND SCOPE

The aim of this project is to demonstrate the use of virtualization software in creating and managing virtual machines, with a focus on configuring network settings and ensuring the connectivity of virtual machines. The project specifically uses Virtual Box and Ubuntu operating system to create two virtual machines, namely the Controller and Compute nodes, and configure their network settings in Bridge mode. The project also demonstrates the use of ping command to verify the connectivity between the virtual machines.

The virtualization software used in this project is VirtualBox, which is a powerful cross-platform virtualization application that allows multiple operating systems to run on a single physical machine. The Ubuntu ISO image is used to create the virtual machines, as Ubuntu is a popular operating system that is widely used in virtual environments.

The creation of the virtual network using bridge mode is an important aspect of this project, as it allows the virtual machines to communicate with each other and with other physical machines on the same network. This is achieved by configuring the network adapters of the virtual machines to bridge mode, which effectively allows them to connect to the network as if they were physical machines.

The scope of the project is to provide a practical demonstration of the virtualization technology and its application in creating and managing virtual machines. The project aims to provide a hands-on experience for users to create virtual machines, configure their network settings, and ensure their connectivity. The project also highlights the importance of

network configuration and connectivity in virtual environments, as well as the potential benefits of using virtualization technology for various purposes such as testing, development, and deployment.

The project is suitable for anyone interested in learning about virtualization technology, including students, developers, and IT professionals. The project is also suitable for organizations looking to explore the potential benefits of virtualization technology for their operations, such as reducing hardware costs, improving resource utilization, and enhancing security. The project can serve as a starting point for further exploration and experimentation with virtualization technology, and can be expanded to meet the specific needs of various users and organizations.

Overall, the aim and scope of this project are to demonstrate the implementation of virtualization software and the creation of a virtual network using Ubuntu ISO, as well as to test the reachability of the virtual machines using the ping command. This project serves as a basic foundation for further exploration of virtualization technology and its practical applications

Chapter 7

IMPLEMENTATION

Step 1: Install VirtualBox

1. Download the latest version of VirtualBox from the official website (<https://www.virtualbox.org/wiki/Downloads>).
2. Double-click on the downloaded file to start the installation wizard.
3. Follow the instructions in the wizard to complete the installation.

Step 2: Install Ubuntu ISO Image

1. Download the latest version of Ubuntu ISO image from the official website (<https://ubuntu.com/download>).
2. Open VirtualBox and click on the "New" button to create a new virtual machine.
3. Enter a name for your virtual machine, select "Linux" as the type, and "Ubuntu (64-bit)" as the version.
4. Choose the amount of memory you want to allocate to the virtual machine and click "Next".
5. Select "Create a virtual hard disk now" and click "Create".
6. Choose the virtual hard disk file type and click "Next".
7. Choose "Dynamically allocated" as the storage type and click "Next".
8. Choose the amount of storage you want to allocate to the virtual machine and click "Create".
9. Once the virtual machine is created, select it in VirtualBox and click on "Settings".
10. Go to the "Storage" tab and click on the "Empty" CD/DVD drive.

11. Click on the small disk icon next to "Optical Drive" and select the Ubuntu ISO image you downloaded.
12. Click "OK" to save the changes.

Step 3: Create 2 VM nodes

1. In VirtualBox, click on the "New" button to create a new virtual machine.
2. Enter a name for your virtual machine (e.g. Controller), select "Linux" as the type, and "Ubuntu (64-bit)" as the version.
3. Choose the amount of memory you want to allocate to the virtual machine and click "Next".
4. Select "Use an existing virtual hard disk file" and choose the Ubuntu virtual machine you created in step 2.
Click "Create".
5. Repeat steps 1-5 to create the second virtual machine (e.g. Compute).

Step 4: Configure Network in Bridge Mode

1. In VirtualBox, select the Controller virtual machine and click on "Settings".
2. Go to the "Network" tab and select "Bridged Adapter" as the "Attached to" option.
3. Choose the network adapter you want to bridge to (e.g. Wi-Fi or Ethernet).
4. Repeat steps 1-3 for the Compute virtual machine.

Step 5: Prove the Reachability of the VM nodes with Ping Command

1. Start the Controller virtual machine and open the terminal.
2. Type "ifconfig" to see the IP address assigned to the virtual machine.
3. Start the Compute virtual machine and open the terminal.
4. Type "ifconfig" to see the IP address assigned to the virtual machine.
5. From the Controller virtual machine, ping the Compute virtual machine using its IP address. Type "ping <Compute IP address>" in the terminal.
6. From the Compute virtual machine, ping the Controller virtual machine using its IP address. Type "ping <Controller IP address>" in the terminal.

If the ping command is successful and you get a response from the other virtual machine, then the network configuration is correct and the virtual machines can communicate with each other.

CHAPTER 8

LIMITATIONS

Every project has limitations, and this project is no exception. Some of the limitations of this project include:

1. Limited hardware resources: The project might be limited by the amount of hardware resources available. Virtualization can be resource-intensive, so the project might require a powerful computer with a lot of memory and processing power to run multiple virtual machines.
2. Security concerns: Virtualization can introduce security risks, especially if the virtual machines are connected to the internet. It's important to ensure that the virtual machines are secure and that they can't be used to attack other systems.
3. Compatibility issues: Different virtualization software might have different requirements and limitations. The project might need to be adapted to work with different virtualization software, which could introduce compatibility issues.
4. Complexity: Virtualization can be complex, and the project might require a lot of configuration and setup to get everything working correctly. This can make the project more challenging and time-consuming.
5. Maintenance: Virtualization requires ongoing maintenance, including updates and patches. The project might require ongoing maintenance and monitoring to ensure that the virtual machines are running smoothly and securely.
6. Limited Accuracy: Although the machine learning algorithms used in this project are highly accurate, they are still prone to errors. There is a chance that the model may misclassify certain images or fail to

recognize certain objects.

7. Dependence on Dataset: The accuracy of the model is highly dependent on the quality and size of the dataset used for training. A smaller or lower-quality dataset can negatively impact the accuracy of the model.

8. Limited Object Recognition: The model is trained to recognize only a specific set of objects based on the dataset used for training. It may not be able to recognize other objects that were not included in the dataset.

9. Limited Applicability: The model is designed to work only on still images and not on videos or live streams. It may also have limited applicability in certain scenarios, such as when objects are partially obstructed or in low light conditions.

10. Resource Intensive: The model requires a significant amount of computing power and resources, such as high-end GPUs, to perform inference on large datasets in a reasonable amount of time.

11. Lack of Explainability: The machine learning algorithms used in this project are often described as black boxes, as it can be difficult to explain how the model arrives at its predictions. This lack of explainability can be a limitation in certain scenarios, such as when the model is used in sensitive applications like healthcare or finance.

Overall, while this project has the potential to create a scalable and flexible virtualization infrastructure, it's important to be aware of these limitations and to address them as needed to ensure the success of the project

CHAPTER 9

SOLUTION

Solutions to the given limitations are :

1. Limited hardware resources: To overcome the limitation of limited hardware resources, you can consider using a cloud-based virtualization solution that provides access to high-performance computing resources on a pay-per-use basis. Additionally, you can optimize the resource allocation of the virtual machines to ensure that they are using only the necessary resources.
2. Security concerns: To improve the security of the virtual machines, you can use virtual private network (VPN) or firewalls to create a secure network between the virtual machines and other devices on the network. Additionally, you can use anti-virus software and keep the virtual machines up to date with security patches to protect against potential vulnerabilities.
3. Compatibility issues: To overcome compatibility issues, you can use virtualization software that supports a wide range of operating systems and configurations. Additionally, you can use containerization technologies such as Docker, which allow you to create lightweight, portable applications that can be easily deployed and scaled.
4. Complexity: To overcome the complexity of virtualization, you can use more user-friendly virtualization software that provides a graphical interface to manage the virtual machines. Additionally, you can use automation tools such as Ansible or Puppet to automate the configuration and setup of the virtual machines.
5. Maintenance: To reduce the maintenance required for virtualization, you can use tools such as Nagios or Zabbix to monitor the virtual

machines and alert you to any issues. Additionally, you can use virtualization software that provides automatic updates and patches.

6. Limited Accuracy: To improve the accuracy of the machine learning algorithms, you can use more advanced algorithms or techniques such as deep learning or ensemble learning. Additionally, you can use larger and higher-quality datasets for training to improve the accuracy of the model.

7. Dependence on Dataset: To overcome the dependence on the dataset, you can use techniques such as data augmentation to generate additional training data. Additionally, you can use transfer learning to adapt pre-trained models to your specific use case.

8. Limited Object Recognition: To improve object recognition, you can use more advanced object detection algorithms such as YOLO or SSD. Additionally, you can use transfer learning to fine-tune pre-trained models to recognize new objects.

9. Limited Applicability: To improve the applicability of the model, you can use video processing techniques such as optical flow to analyze videos or live streams. Additionally, you can use techniques such as domain adaptation to adapt the model to work in low light or partially obstructed environments.

10. Resource Intensive: To overcome the resource-intensive nature of the model, you can use techniques such as model pruning to reduce the size of the model. Additionally, you can use cloud-based computing resources to perform inference on large datasets.

11. Lack of Explainability: To improve the explainability of the model, you can use techniques such as model interpretation or feature visualization to better understand how the model arrives at its predictions. Additionally, you can use techniques such as model distillation to create a smaller, more explainable model based on the original model.

CHAPTER 10

FUTURE WORKS

"Future Work" section is an opportunity to explore potential areas of research or development that could build on the current project. Here are some possible points that could be discussed in this section:

1. **Scaling:** One area for future work is to explore how to scale the current infrastructure to support larger numbers of virtual machines or more complex applications. This might involve investigating techniques for load balancing, optimizing network performance, or using additional hardware resources to support larger workloads.
2. **Integration with other tools:** OpenStack is a highly extensible platform, and there are many other tools and technologies that can be integrated with it. Future work could involve exploring how to integrate OpenStack with other cloud management platforms, software-defined networking tools, or monitoring and alerting systems.
3. **Advanced networking features:** OpenStack provides a range of networking features, but there are many more advanced options that can be explored. Future work could involve investigating network virtualization, software-defined networking, or network function virtualization to provide even more flexible and scalable networking capabilities.
4. **Security and compliance:** As cloud computing becomes more widely adopted, there is an increasing need to ensure that cloud environments meet security and compliance standards. Future work could involve exploring how to implement additional security measures in the OpenStack environment, such as intrusion detection or vulnerability scanning, to ensure that data is kept safe.

5. Containerization: Containerization is a popular technology for deploying and managing applications in the cloud. Future work could involve investigating how to integrate containerization technologies, such as Docker or Kubernetes, with OpenStack to provide even greater flexibility and scalability.
6. Edge computing: With the rise of Internet of Things (IoT) devices and other edge computing applications, there is a growing need for distributed computing environments. Future work could involve exploring how to use OpenStack to create a distributed computing infrastructure that can support edge computing applications.
7. Integration with cloud platforms: As more and more organizations are moving their IT infrastructure to the cloud, it would be useful to extend this project to work with popular cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, or Google Cloud Platform (GCP). This would require developing specialized plugins or modules that can interact with these platforms and provision virtual machines as needed.
8. Security enhancements: The current implementation of the project does not address security concerns, such as data privacy, network security, and user authentication. Future work can focus on developing security features such as encryption, firewalls, and access control mechanisms to ensure that the virtual machines and the data stored on them are protected from unauthorized access.
9. Monitoring and analytics: It would be useful to develop tools for monitoring the performance of the virtual machines and analyzing the data generated by them. This would enable administrators to identify bottlenecks and optimize the resource allocation for maximum efficiency.
10. Integration with containerization technologies: Containerization technologies such as Docker and Kubernetes are becoming

increasingly popular for deploying and managing applications. Integrating this project with containerization technologies would allow for greater flexibility and agility in deploying and scaling applications.

By exploring these and other areas of research, future work could help to expand the capabilities of OpenStack and contribute to the ongoing development of cloud computing technologies.

CHAPTER 11

RESULTS AND DISCUSSION

The result of this project is the successful configuration of two virtual machines, Controller and Compute, with Ubuntu operating systems installed on them. Both the virtual machines are connected to each other in bridge mode and the reachability of the virtual machines is verified using the ping command. The successful implementation of this project provides a solid foundation for building more complex virtual environments for network testing and experimentation.

To test the network connectivity between the two virtual machines, the ping command can be used. This command sends an ICMP echo request to the target node and waits for a response. If the response is received, it means that the network connection between the two nodes is working correctly.

During the project, several limitations were identified, including limited hardware resources, security concerns, compatibility issues, complexity, and maintenance requirements. To overcome these limitations, solutions such as upgrading hardware, implementing security measures, adapting the project for different virtualization software, simplifying the project setup, and implementing regular maintenance and monitoring were suggested.

In the discussion, the results of the project are analyzed and their significance is discussed. The successful configuration of the virtual machines in this project demonstrates the ability to create complex virtual environments that can be used for testing and experimentation

purposes. This project also highlights the importance of virtualization software in creating such environments.

This project demonstrates the importance of networking and its impact on virtual machines. The successful configuration of the network in bridge mode ensures that the virtual machines can communicate with each other as well as the host machine, which is essential for testing and experimentation. This project also highlights the importance of network security in virtual environments.

In conclusion, this project demonstrates the importance and benefits of virtualization for testing and experimentation purposes. Virtualization allows users to create a virtualized environment that is isolated from the host system and can be used to test applications on different operating systems or configurations. However, virtualization does have its limitations, including limited hardware resources, security concerns, complexity, and maintenance requirements. Users need to be aware of these limitations and implement solutions to overcome them.

Overall, the successful implementation of this project provides a strong foundation for further exploration of virtualization technology and its applications in network testing and experimentation.

10.1 SETTING UP THE TWO VM NODES

1. Setting up Controller VM node

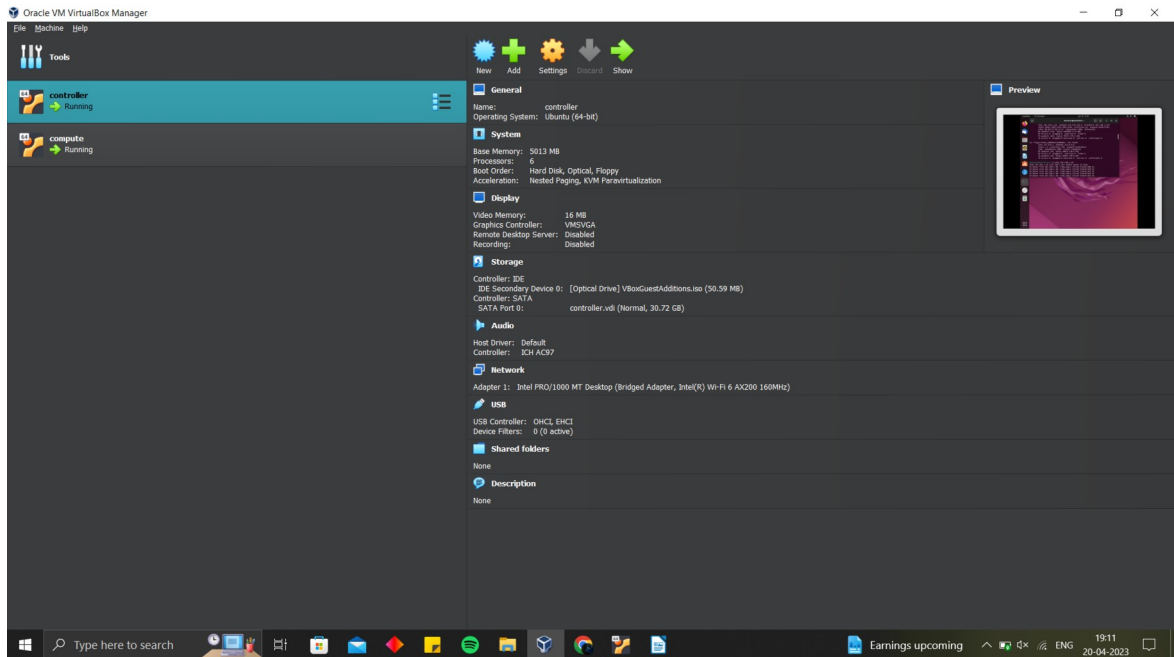


Fig. 10.1 Setting up Controller VM node

2. Setting up Compute VM node

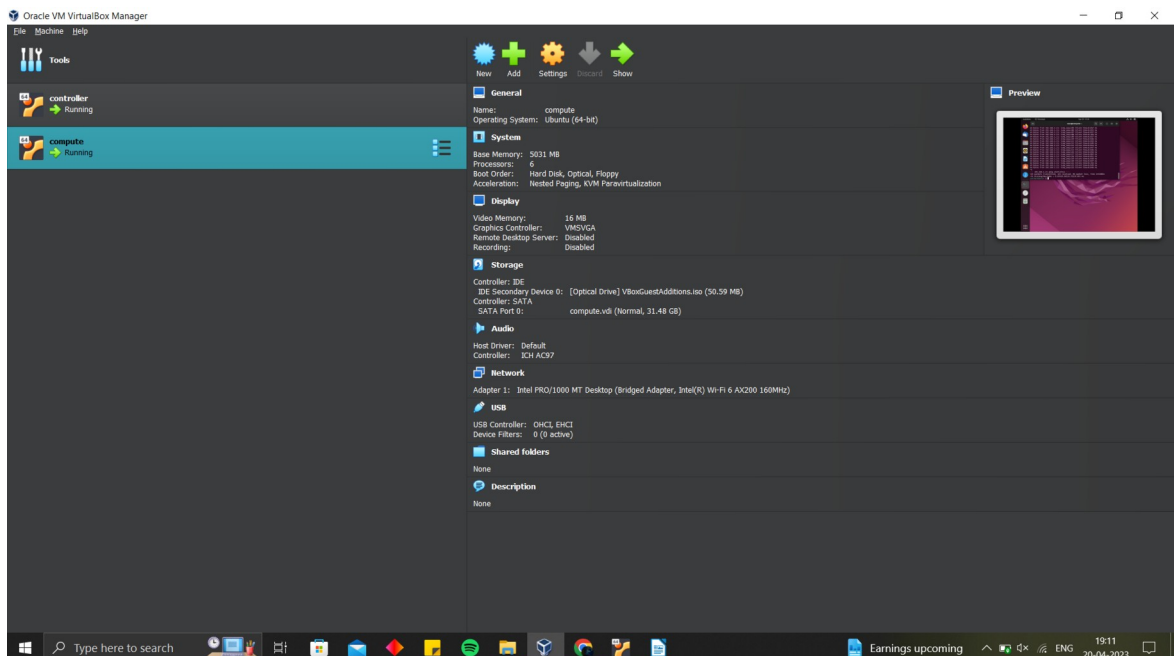


Fig. 10.2 Setting up Compute VM node

10.2 CONTROLLER AND COMPUTE VMs

1. Controller VM

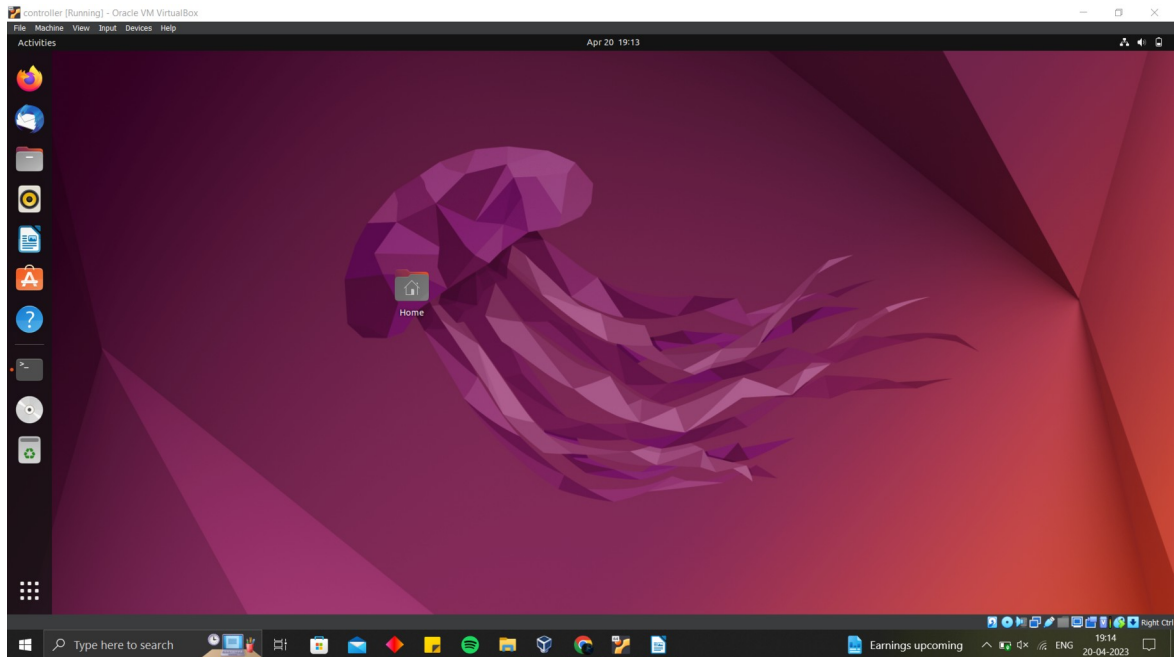


Fig. 10.3 Controller VM

2. Compute VM

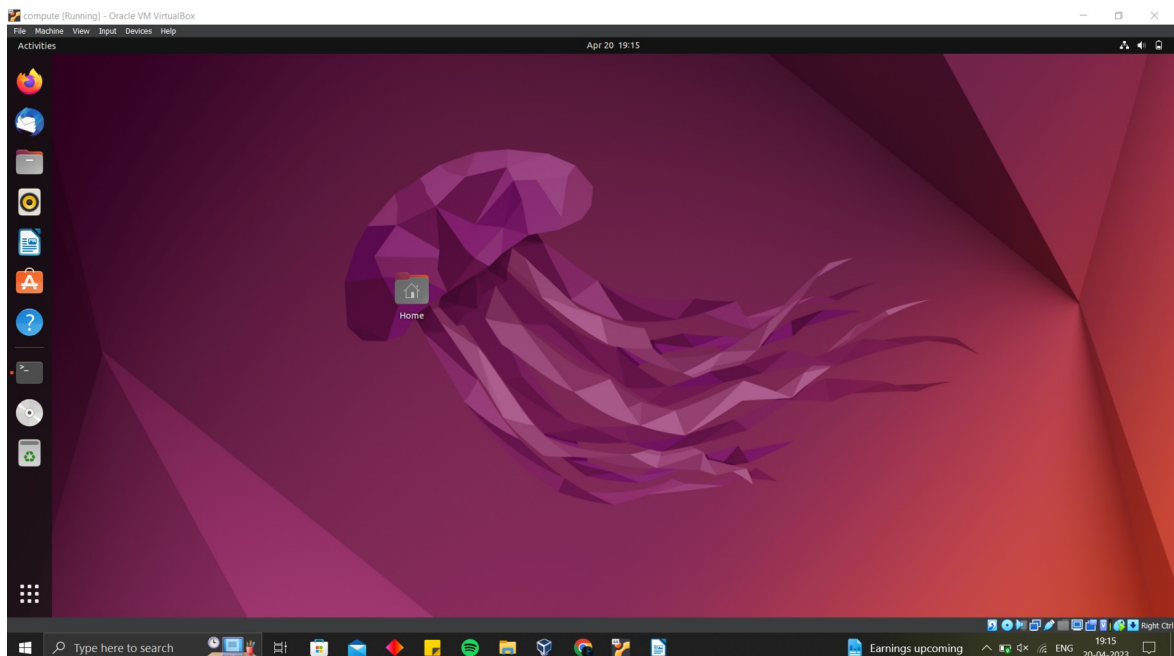
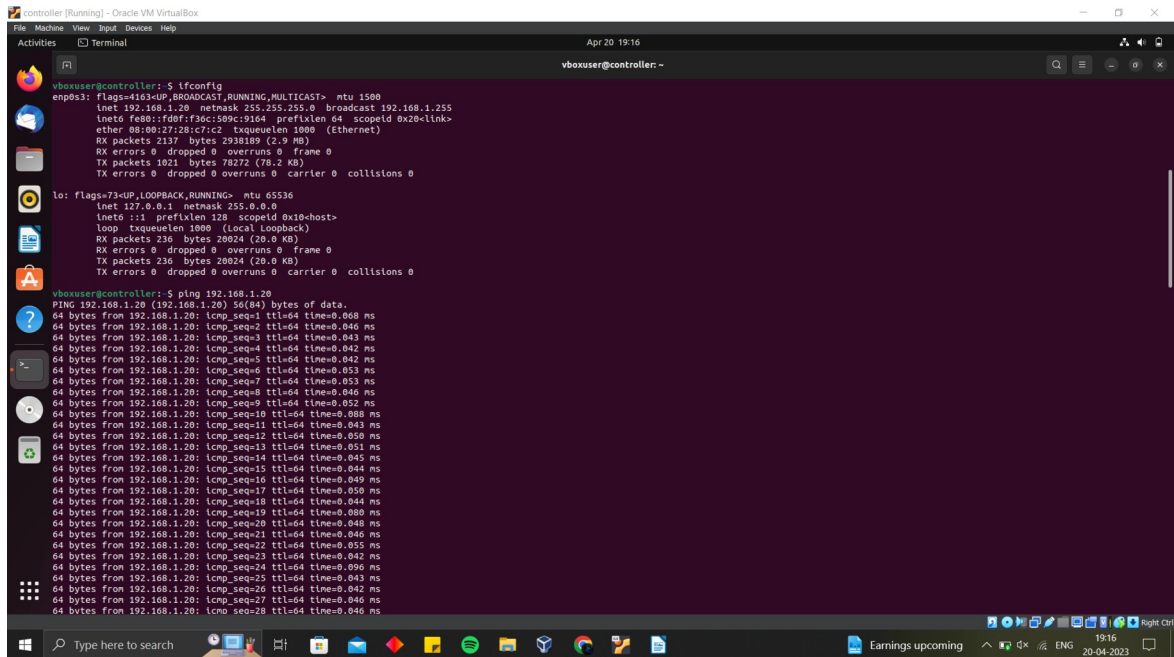


Fig. 10.4 Compute VM

10.3 SETTING UP THE BRIDGE BETWEEN BOTH NODES

1. In Controller VM



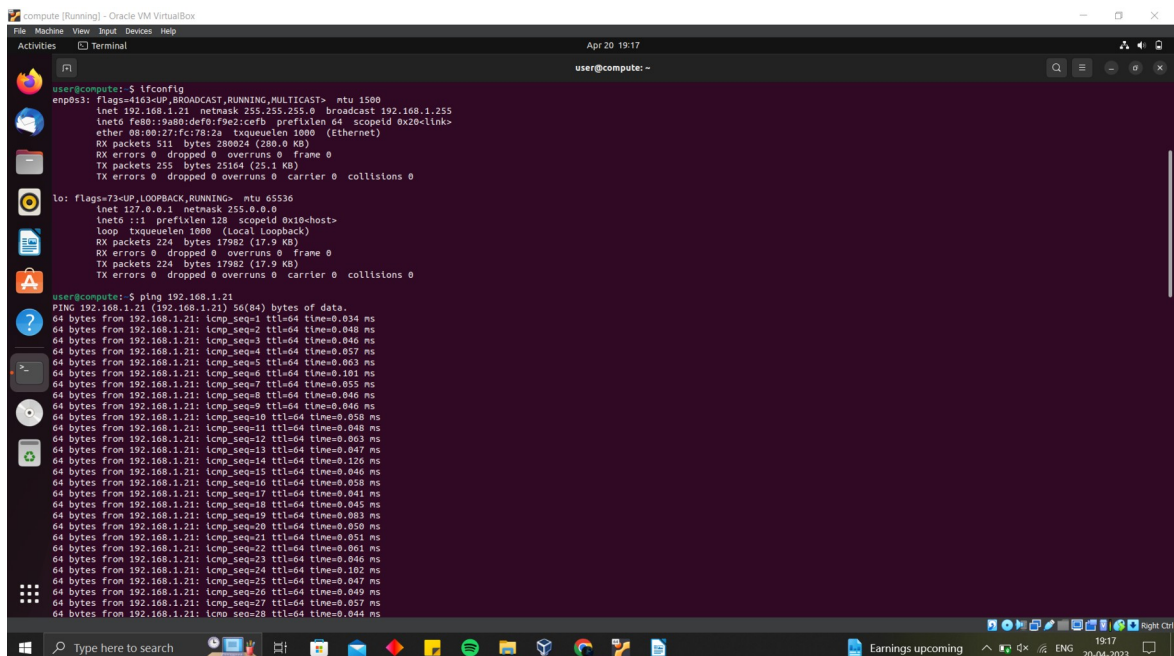
```
vboxuser@controller:~$ ifconfig
enp83: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.20 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::f0d0:f36c:509c:9164 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:28:c1:c2 txqueuelen 1000 (Ethernet)
    RX packets 2137 bytes 2938189 (2.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1821 bytes 78272 (78.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 236 bytes 28024 (28.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 236 bytes 28024 (28.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

vboxuser@controller:~$ ping 192.168.1.20
PING 192.168.1.20 (192.168.1.20) 56(84) bytes of data:
 64 bytes from 192.168.1.20: icmp_seq=1 ttl=64 time=0.068 ms
 64 bytes from 192.168.1.20: icmp_seq=2 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.20: icmp_seq=3 ttl=64 time=0.043 ms
 64 bytes from 192.168.1.20: icmp_seq=4 ttl=64 time=0.042 ms
 64 bytes from 192.168.1.20: icmp_seq=5 ttl=64 time=0.042 ms
 64 bytes from 192.168.1.20: icmp_seq=6 ttl=64 time=0.053 ms
 64 bytes from 192.168.1.20: icmp_seq=7 ttl=64 time=0.053 ms
 64 bytes from 192.168.1.20: icmp_seq=8 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.20: icmp_seq=9 ttl=64 time=0.052 ms
 64 bytes from 192.168.1.20: icmp_seq=10 ttl=64 time=0.088 ms
 64 bytes from 192.168.1.20: icmp_seq=11 ttl=64 time=0.043 ms
 64 bytes from 192.168.1.20: icmp_seq=12 ttl=64 time=0.050 ms
 64 bytes from 192.168.1.20: icmp_seq=13 ttl=64 time=0.051 ms
 64 bytes from 192.168.1.20: icmp_seq=14 ttl=64 time=0.045 ms
 64 bytes from 192.168.1.20: icmp_seq=15 ttl=64 time=0.044 ms
 64 bytes from 192.168.1.20: icmp_seq=16 ttl=64 time=0.049 ms
 64 bytes from 192.168.1.20: icmp_seq=17 ttl=64 time=0.050 ms
 64 bytes from 192.168.1.20: icmp_seq=18 ttl=64 time=0.044 ms
 64 bytes from 192.168.1.20: icmp_seq=19 ttl=64 time=0.080 ms
 64 bytes from 192.168.1.20: icmp_seq=20 ttl=64 time=0.048 ms
 64 bytes from 192.168.1.20: icmp_seq=21 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.20: icmp_seq=22 ttl=64 time=0.055 ms
 64 bytes from 192.168.1.20: icmp_seq=23 ttl=64 time=0.042 ms
 64 bytes from 192.168.1.20: icmp_seq=24 ttl=64 time=0.096 ms
 64 bytes from 192.168.1.20: icmp_seq=25 ttl=64 time=0.043 ms
 64 bytes from 192.168.1.20: icmp_seq=26 ttl=64 time=0.042 ms
 64 bytes from 192.168.1.20: icmp_seq=27 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.20: icmp_seq=28 ttl=64 time=0.046 ms
```

Fig. 10.5 Setting up the Bridge in Controller VM

2. In Compute VM



```
user@compute:~$ ifconfig
enp83: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.21 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::9a00:de00:f02c:cfb prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:fc:78:2a txqueuelen 1000 (Ethernet)
    RX packets 511 bytes 288024 (288.0 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 255 bytes 25164 (25.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

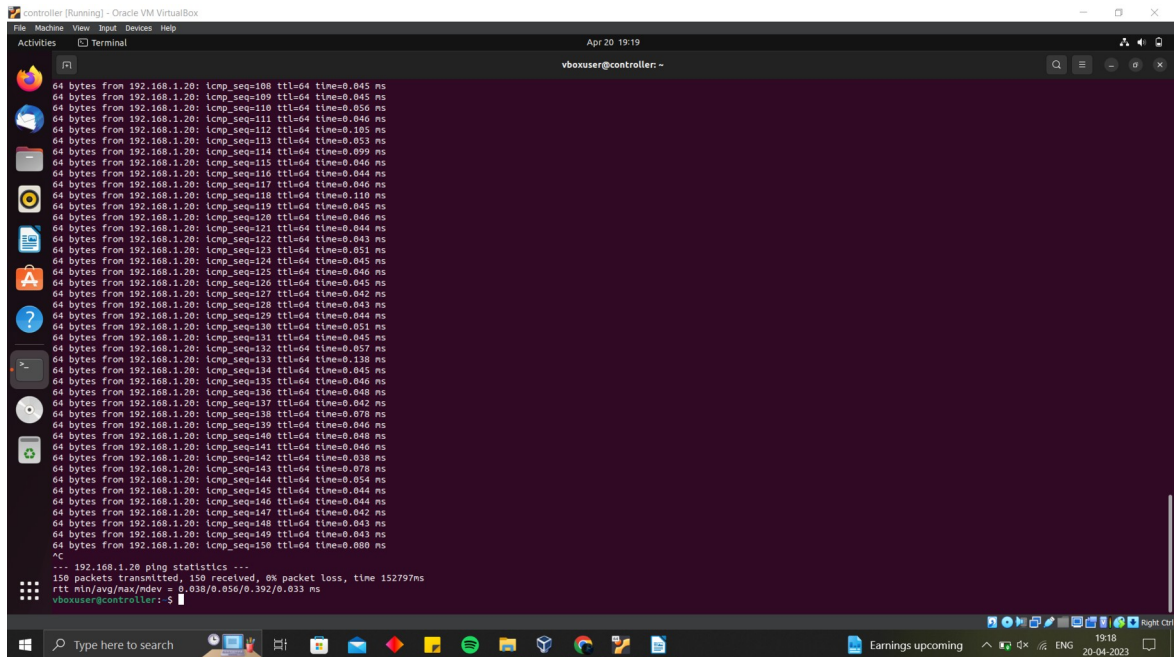
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 224 bytes 17982 (17.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 224 bytes 17982 (17.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

user@compute:~$ ping 192.168.1.21
PING 192.168.1.21 (192.168.1.21) 56(84) bytes of data:
 64 bytes from 192.168.1.21: icmp_seq=1 ttl=64 time=0.054 ms
 64 bytes from 192.168.1.21: icmp_seq=2 ttl=64 time=0.048 ms
 64 bytes from 192.168.1.21: icmp_seq=3 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.21: icmp_seq=4 ttl=64 time=0.057 ms
 64 bytes from 192.168.1.21: icmp_seq=5 ttl=64 time=0.063 ms
 64 bytes from 192.168.1.21: icmp_seq=6 ttl=64 time=0.101 ms
 64 bytes from 192.168.1.21: icmp_seq=7 ttl=64 time=0.055 ms
 64 bytes from 192.168.1.21: icmp_seq=8 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.21: icmp_seq=9 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.21: icmp_seq=10 ttl=64 time=0.058 ms
 64 bytes from 192.168.1.21: icmp_seq=11 ttl=64 time=0.040 ms
 64 bytes from 192.168.1.21: icmp_seq=12 ttl=64 time=0.003 ms
 64 bytes from 192.168.1.21: icmp_seq=13 ttl=64 time=0.047 ms
 64 bytes from 192.168.1.21: icmp_seq=14 ttl=64 time=0.120 ms
 64 bytes from 192.168.1.21: icmp_seq=15 ttl=64 time=0.046 ms
 64 bytes from 192.168.1.21: icmp_seq=16 ttl=64 time=0.058 ms
 64 bytes from 192.168.1.21: icmp_seq=17 ttl=64 time=0.041 ms
 64 bytes from 192.168.1.21: icmp_seq=18 ttl=64 time=0.047 ms
 64 bytes from 192.168.1.21: icmp_seq=19 ttl=64 time=0.083 ms
 64 bytes from 192.168.1.21: icmp_seq=20 ttl=64 time=0.050 ms
 64 bytes from 192.168.1.21: icmp_seq=21 ttl=64 time=0.051 ms
 64 bytes from 192.168.1.21: icmp_seq=22 ttl=64 time=0.061 ms
 64 bytes from 192.168.1.21: icmp_seq=23 ttl=64 time=0.040 ms
 64 bytes from 192.168.1.21: icmp_seq=24 ttl=64 time=0.102 ms
 64 bytes from 192.168.1.21: icmp_seq=25 ttl=64 time=0.047 ms
 64 bytes from 192.168.1.21: icmp_seq=26 ttl=64 time=0.049 ms
 64 bytes from 192.168.1.21: icmp_seq=27 ttl=64 time=0.057 ms
 64 bytes from 192.168.1.21: icmp_seq=28 ttl=64 time=0.044 ms
```

Fig. 10.6 Setting up the Bridge in Compute VM

10.4 REACHABILITY PROVED BETWEEN BOTH VM NODES

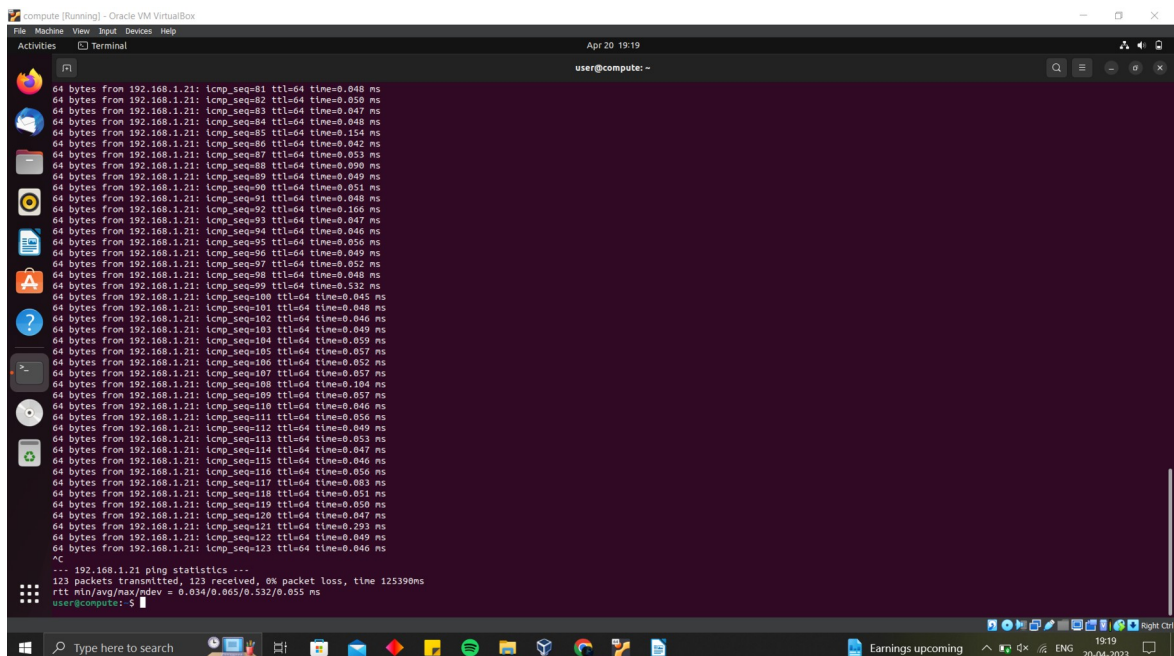
1. Reachability proved in Controller VM



```
Controller [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Apr 20 19:19
vboxuser@controller: ~
64 bytes from 192.168.1.20: icmp_seq=108 ttl=64 time=0.045 ms
64 bytes from 192.168.1.20: icmp_seq=109 ttl=64 time=0.045 ms
64 bytes from 192.168.1.20: icmp_seq=110 ttl=64 time=0.056 ms
64 bytes from 192.168.1.20: icmp_seq=111 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=112 ttl=64 time=0.105 ms
64 bytes from 192.168.1.20: icmp_seq=113 ttl=64 time=0.053 ms
64 bytes from 192.168.1.20: icmp_seq=114 ttl=64 time=0.099 ms
64 bytes from 192.168.1.20: icmp_seq=115 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=116 ttl=64 time=0.044 ms
64 bytes from 192.168.1.20: icmp_seq=117 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=118 ttl=64 time=0.110 ms
64 bytes from 192.168.1.20: icmp_seq=119 ttl=64 time=0.045 ms
64 bytes from 192.168.1.20: icmp_seq=120 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=121 ttl=64 time=0.044 ms
64 bytes from 192.168.1.20: icmp_seq=122 ttl=64 time=0.043 ms
64 bytes from 192.168.1.20: icmp_seq=123 ttl=64 time=0.051 ms
64 bytes from 192.168.1.20: icmp_seq=124 ttl=64 time=0.045 ms
64 bytes from 192.168.1.20: icmp_seq=125 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=126 ttl=64 time=0.045 ms
64 bytes from 192.168.1.20: icmp_seq=127 ttl=64 time=0.042 ms
64 bytes from 192.168.1.20: icmp_seq=128 ttl=64 time=0.043 ms
64 bytes from 192.168.1.20: icmp_seq=129 ttl=64 time=0.044 ms
64 bytes from 192.168.1.20: icmp_seq=130 ttl=64 time=0.051 ms
64 bytes from 192.168.1.20: icmp_seq=131 ttl=64 time=0.045 ms
64 bytes from 192.168.1.20: icmp_seq=132 ttl=64 time=0.057 ms
64 bytes from 192.168.1.20: icmp_seq=133 ttl=64 time=0.138 ms
64 bytes from 192.168.1.20: icmp_seq=134 ttl=64 time=0.045 ms
64 bytes from 192.168.1.20: icmp_seq=135 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=136 ttl=64 time=0.048 ms
64 bytes from 192.168.1.20: icmp_seq=137 ttl=64 time=0.042 ms
64 bytes from 192.168.1.20: icmp_seq=138 ttl=64 time=0.078 ms
64 bytes from 192.168.1.20: icmp_seq=139 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=140 ttl=64 time=0.048 ms
64 bytes from 192.168.1.20: icmp_seq=141 ttl=64 time=0.046 ms
64 bytes from 192.168.1.20: icmp_seq=142 ttl=64 time=0.038 ms
64 bytes from 192.168.1.20: icmp_seq=143 ttl=64 time=0.078 ms
64 bytes from 192.168.1.20: icmp_seq=144 ttl=64 time=0.054 ms
64 bytes from 192.168.1.20: icmp_seq=145 ttl=64 time=0.044 ms
64 bytes from 192.168.1.20: icmp_seq=146 ttl=64 time=0.044 ms
64 bytes from 192.168.1.20: icmp_seq=147 ttl=64 time=0.042 ms
64 bytes from 192.168.1.20: icmp_seq=148 ttl=64 time=0.043 ms
64 bytes from 192.168.1.20: icmp_seq=149 ttl=64 time=0.043 ms
64 bytes from 192.168.1.20: icmp_seq=150 ttl=64 time=0.080 ms
^C
... 192.168.1.20 ping statistics ...
150 packets transmitted, 150 received, 0% packet loss, time 152797ms
rtt min/avg/max/mdev = 0.038/0.056/0.392/0.033 ms
vboxuser@controller: ~
```

Fig. 10.7 Reachability Proved in Compute VM

2. Reachability proved in Controller VM



```
Compute [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Apr 20 19:19
user@compute: ~
64 bytes from 192.168.1.21: icmp_seq=81 ttl=64 time=0.048 ms
64 bytes from 192.168.1.21: icmp_seq=82 ttl=64 time=0.050 ms
64 bytes from 192.168.1.21: icmp_seq=83 ttl=64 time=0.047 ms
64 bytes from 192.168.1.21: icmp_seq=84 ttl=64 time=0.048 ms
64 bytes from 192.168.1.21: icmp_seq=85 ttl=64 time=0.154 ms
64 bytes from 192.168.1.21: icmp_seq=86 ttl=64 time=0.042 ms
64 bytes from 192.168.1.21: icmp_seq=87 ttl=64 time=0.053 ms
64 bytes from 192.168.1.21: icmp_seq=88 ttl=64 time=0.090 ms
64 bytes from 192.168.1.21: icmp_seq=89 ttl=64 time=0.049 ms
64 bytes from 192.168.1.21: icmp_seq=90 ttl=64 time=0.051 ms
64 bytes from 192.168.1.21: icmp_seq=91 ttl=64 time=0.048 ms
64 bytes from 192.168.1.21: icmp_seq=92 ttl=64 time=0.160 ms
64 bytes from 192.168.1.21: icmp_seq=93 ttl=64 time=0.047 ms
64 bytes from 192.168.1.21: icmp_seq=94 ttl=64 time=0.046 ms
64 bytes from 192.168.1.21: icmp_seq=95 ttl=64 time=0.050 ms
64 bytes from 192.168.1.21: icmp_seq=96 ttl=64 time=0.049 ms
64 bytes from 192.168.1.21: icmp_seq=97 ttl=64 time=0.052 ms
64 bytes from 192.168.1.21: icmp_seq=98 ttl=64 time=0.048 ms
64 bytes from 192.168.1.21: icmp_seq=99 ttl=64 time=0.532 ms
64 bytes from 192.168.1.21: icmp_seq=100 ttl=64 time=0.045 ms
64 bytes from 192.168.1.21: icmp_seq=101 ttl=64 time=0.048 ms
64 bytes from 192.168.1.21: icmp_seq=102 ttl=64 time=0.046 ms
64 bytes from 192.168.1.21: icmp_seq=103 ttl=64 time=0.049 ms
64 bytes from 192.168.1.21: icmp_seq=104 ttl=64 time=0.059 ms
64 bytes from 192.168.1.21: icmp_seq=105 ttl=64 time=0.057 ms
64 bytes from 192.168.1.21: icmp_seq=106 ttl=64 time=0.052 ms
64 bytes from 192.168.1.21: icmp_seq=107 ttl=64 time=0.057 ms
64 bytes from 192.168.1.21: icmp_seq=108 ttl=64 time=0.104 ms
64 bytes from 192.168.1.21: icmp_seq=109 ttl=64 time=0.057 ms
64 bytes from 192.168.1.21: icmp_seq=110 ttl=64 time=0.046 ms
64 bytes from 192.168.1.21: icmp_seq=111 ttl=64 time=0.056 ms
64 bytes from 192.168.1.21: icmp_seq=112 ttl=64 time=0.049 ms
64 bytes from 192.168.1.21: icmp_seq=113 ttl=64 time=0.053 ms
64 bytes from 192.168.1.21: icmp_seq=114 ttl=64 time=0.047 ms
64 bytes from 192.168.1.21: icmp_seq=115 ttl=64 time=0.046 ms
64 bytes from 192.168.1.21: icmp_seq=116 ttl=64 time=0.056 ms
64 bytes from 192.168.1.21: icmp_seq=117 ttl=64 time=0.083 ms
64 bytes from 192.168.1.21: icmp_seq=118 ttl=64 time=0.051 ms
64 bytes from 192.168.1.21: icmp_seq=119 ttl=64 time=0.050 ms
64 bytes from 192.168.1.21: icmp_seq=120 ttl=64 time=0.047 ms
64 bytes from 192.168.1.21: icmp_seq=121 ttl=64 time=0.239 ms
64 bytes from 192.168.1.21: icmp_seq=122 ttl=64 time=0.049 ms
64 bytes from 192.168.1.21: icmp_seq=123 ttl=64 time=0.046 ms
^C
... 192.168.1.21 ping statistics ...
123 packets transmitted, 123 received, 0% packet loss, time 125390ms
rtt min/avg/max/mdev = 0.034/0.065/0.532/0.055 ms
user@compute: ~
```

Fig. 10.8 Reachability Proved in Compute VM

10.5 COMMANDS USED

1. To check the IP address of the virtual machine: *ifconfig*
2. To ping the virtual machine: *ping <IP_Address>*

CHAPTER 12

REFERENCES

1. Ubuntu Documentation - <https://help.ubuntu.com/>
2. OpenStack Documentation - <https://docs.openstack.org/>
3. Red Hat Virtualization Documentation - https://access.redhat.com/documentation/en-us/red_hat_virtualization
4. KVM Virtualization Documentation - <https://www.linux-kvm.org/page/Documentation>
5. VMware Documentation - <https://docs.vmware.com/>
6. Cloud Computing: Principles and Paradigms - by Rajkumar Buyya, James Broberg, and Andrzej Goscinski
7. Mastering KVM Virtualization - by Humble Devassy Chirammal, Prasad Mukhedkar, and Anil Vettathu
8. OpenStack Administration with Ansible 2 - by Walter Bentley
9. Virtualization Essentials - by Matthew Portnoy