RTB | RWANDA TVET BOARD

RQF LEVEL 5

SWDDT501

SOFTWARE DEVELOPMENT

DevOps Techniques Application

TRAINEE'S MANUAL

October, 2024

DEVOPS TECHNIQUES APPLICATION

KOICA Korea International Cooperation Agency

TQUM TVET Quality Management Project

RTB RWANDA TVET BOARD

2024

AUTHOR'S NOTE PAGE (COPYRIGHT)

© Rwanda TVET Board

Copies available from:

- *HQs: Rwanda TVET Board-RTB*
- *Web: www.rtb.gov.rw*
- **KIGALI-RWANDA**

Original published version: October 2024

ACKNOWLEDGEMENTS

The publisher would like to thank the following for their assistance in the elaboration of this training manual:

Rwanda TVET Board (RTB) extends its appreciation to all parties who contributed to the development of the trainer's and trainee's manuals for the TVET Certificate V in Software Development, specifically for the module **"SWDDT501: DevOps Techniques Application"**

We extend our gratitude to KOICA Rwanda for its contribution to the development of these training manuals and for its ongoing support of the TVET system in Rwanda.

We extend our gratitude to the TQUM Project for its financial and technical support in the development of these training manuals.

We would also like to acknowledge the valuable contributions of all TVET trainers and industry practitioners in the development of this training manual.

The management of Rwanda TVET Board extends its appreciation to both its staff and the staff of the TQUM Project for their efforts in coordinating these activities.

This training manual was developed:

Under Rwanda TVET Board (RTB) guiding policies and directives

RTB | RWANDA TVET BOARD

Under Financial and Technical support of

KOICA Korea International Cooperation Agency

TQUM TVET Quality Management Project

COORDINATION TEAM

RWAMASIRABO Aimable

MARIA Bernadette M. Ramos

MUTIJIMA Asher Emmanuel

Production Team

Authoring and Review

NIYOMWUNGERI Aphrodis

NYANDWI Rongin

Validation

NIRINGIYIMANA Augustin

SEKABANZA Jean de la Paix

Conception, Adaptation and Editorial works

HATEGEKIMANA Olivier

GANZA Jean Francois Regis

HARELIMANA Wilson

NZABIRINDA Aimable

DUKUZIMANA Therese

NIYONKURU Sylvestre

UMEREWENEZA Naasson

Formatting, Graphics, Illustrations, and infographics

YEONWOO Choe
SUA Lim
SAEM Lee
SOYEON Kim
WONYEONG Jeong
MANIRAKORA Alexis

Financial and Technical support

KOICA through TQUM Project

TABLE OF CONTENT

ACRONYMS

AWS: Amazon Web Services

CD: Continuous Delivery

CI: Continuous Integration

DevOps: Development and Operations

DevSecOps: Development, Security, and Operations

DNS: Domain Name System

IaaS: Infrastructure as a Service

IaC: Infrastructure as Code

JSON: JavaScript Object Notation

K8s: Kubernetes

KOICA: Korea International Cooperation Agency

OS: Operating system

RTB: Rwanda TVET Board

RTB: Rwanda TVET Board

SSH: Secure Shell

TQUM: TVET Quality Management

TVET: Technical and Vocational Education and Training

VCS: Version Control System

YAML: yet another markup language

INTRODUCTION

This trainee's manual includes all the knowledge and skills required in Software Development, specifically for the module of **“DevOps Techniques Application".** Trainees enrolled in this module will engage in practical activities designed to develop and enhance their competencies. The development of this training manual followed the Competency-Based Training and Assessment (CBT/A) approach, offering ample practical opportunities that mirror real-life situations.

The trainee's manual is organized into Learning Outcomes, which is broken down into indicative content that includes both theoretical and practical activities. It provides detailed information on the key competencies required for each learning outcome, along with the objectives to be achieved.

As a trainee, you will start by addressing questions related to the activities, which are designed to foster critical thinking and guide you towards practical applications in the labor market. The manual also provides essential information, including learning hours, required materials, and key tasks to complete throughout the learning process.

All activities included in this training manual are designed to facilitate both individual and group work. After completing the activities, you will conduct a formative assessment, referred to as the end learning outcome assessment. Ensure that you thoroughly review the key readings and the 'Points to Remember' section.

MODULE CODE AND TITLE: SWDDT501 DEVOPS TECHNIQUES APPLICATION

Learning Outcome 1: Perform server configuration

Learning Outcome 2: Deploy the system

Learning Outcome 3: Implement monitoring

Learning Outcome 1: Perform Server Configuration

Indicative contents

1.1 Preparation of environment

1.2 Applying Linux basics commands

1.3 Management of server services

Key Competencies for Learning Outcome 1: Perform server configuration

Knowledge	Skills	Attitudes
• Description of DevOps key terms • Identification of Linux distributions • Description of Linux basic command • Description of server services	• Installing Linux operating system • Applying basics Linux Commands • Managing server services	• Being collaborative while installing Linux operating system • Being competent when managing Linux server • Being innovative • Being Practical oriented

Duration: 20 hrs

Learning outcome 1 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Describe correctly the key terms related to the development operations (DevOps)
2. Identify accurately the Linux distributions based on selected platform.
3. Describe properly Linux basic commands based on selected Linux distribution
4. Describe properly the server services based on selected platform.
5. Install properly Linux operating system based od on selected Linux distribution
6. Apply properly basics Linux commands based on Linux distribution.
7. Manage properly the server services based on Linux distribution

Resources

Equipment	Tools	Materials
• Computer • Server Machine • storage device	• Linux OS • Oracle Virtual Box	• Internet • Electricity

Indicative content 1.1: Preparation of Environment

Duration: 8 hrs

Theoretical Activity 1.1.1: Description of key terms related to DevOps

Tasks:

1: Answer the following questions:

i. What do you understand by the terms below?

a) Server
b) Linux
c) Development Operations (DevOps).
d) DevSecOps
e) IaC
f) IaaS
g) CI/CD
h) Container
i) Node

ii. What is the difference between Continuous Integration (CI) and Continuo us Deployment (CD)?

2: Write your answers on flipchart/paper, blackboard or whiteboard.

3: Present your findings to the whole class and trainer

4: Ask questions for clarification if necessary

5: Read the key readings 1.1.1

Key readings 1.1.1.: Description of key terms

- **Definition of key terms**
  - ✓ **Server**

A server is a computer that provides services to other computers over a network. In DevOps, servers are crucial for hosting applications, databases, and other components of the software development and delivery process.

Key functions of a server include:

Sharing data: Storing and distributing files, documents, and other information.

Providing services: Offering applications or functionalities, such as email, web hosting, or database management.

Managing resources: Allocating and controlling network access, hardware, and software.

- **Types of servers:**

Web server: Handles requests from web browsers and delivers web pages.

File server: Stores and shares files across a network.

Database server: Manages databases and provides access to data.

Mail server: Handles email messages.

Application server: Runs applications on behalf of clients.

- **Server architecture:**

Client-server model: Clients request services from a server, which processes the requests and returns results.

Peer-to-peer (P2P) model: Computers act as both clients and servers, directly sharing resources with each other.

- **Server hardware:**

Processor: Executes instructions and processes data.

Memory: Stores data and instructions.

Storage: Stores data persistently (e.g., hard drives, SSDs).

Network interface: Connects the server to a network.

✓ **Linux**

Linux is a family of open-source operating systems that are widely used in servers and other devices. It is known for its stability, security, and flexibility, making it a popular choice for DevOps environments.

✓ **DevOps**

DevOps is a set of practices that aim to shorten the development life cycle and provide continuous delivery of high-quality software. It emphasizes collaboration between development and operations teams, automation, and continuous improvement.

In DevOps contexts development team is typically responsible for writing and maintaining the code that makes up the software product. It primarily focuses on software development activities, such as coding, testing, and debugging.

Collaboration team may include members from various roles within the organization.

Application of DevOps

DevOps has applications in a wide range of fields, including:

Web development: DevOps is commonly used in web development, where it helps to speed up release cycles and improve collaboration between teams.

Mobile development: DevOps is also used in mobile development, where it helps to ensure that new features are rolled out smoothly and efficiently.

Cloud computing: The use of DevOps in cloud computing helps to improve the automation and scalability of cloud-based applications.

IoT development: DevOps can help to improve the development and deployment of Internet of Things (IoT) applications, which often require a high degree of automation and continuous integration.

Benefits of DevOps:

- **Faster time to market:** DevOps accelerates the delivery of new features and improvements to customers.
- **Improved quality:** By automating testing and deployment, DevOps helps to ensure that software is of high quality and free from defects.
- **Increased reliability:** DevOps practices improve system reliability and resilience by automating routine tasks and monitoring for potential issues.
- **Enhanced collaboration:** DevOps fosters collaboration between development and operations teams, leading to better communication and alignment.

Greater agility: DevOps enables organizations to respond more quickly to changing market conditions and customer needs.

✓ **DevSecOps**

DevSecOps is an extension of DevOps that incorporates security practices throughout the entire software development lifecycle. It ensures that security is considered from the beginning, rather than as an afterthought.

- ✓ **Infrastructure as Code (IaC)**

IaC is a practice that treats infrastructure (such as servers, networks, and storage) as code. It allows for automation, version control, and repeatable provisioning of infrastructure resources.

- ✓ **Infrastructure as a Service (IaaS)**

IaaS is a cloud computing model that provides on-demand access to computing resources, such as servers, storage, and networking. It allows organizations to scale their infrastructure up or down based on their needs.

- ✓ **Continuous Integration/Continuous Deployment (CI/CD)**

CI/CD is a set of practices that automate the building, testing, and deployment of software. It helps to ensure that code changes are integrated and delivered to production quickly and reliably.

- ✓ **Container**

A container is a standardized unit of software that packages up code and its dependencies so that the application runs consistently across different environments. Docker is a popular tool for creating and managing containers.

- ✓ **Node**

A node is a single instance of a running application or process. In a distributed system, multiple nodes can work together to provide a service.

Theoretical activity 1.1.2: Identification of Linux distributions

Tasks:

1: Answer the following questions:

i. What are the Linux distributions?

ii. Explain the components of Linux

2: Write your answers on flipchart/paper, blackboard or whiteboard.

3: Present your findings to the whole class and trainer

4: Ask questions for clarification if necessary

5: Read the key readings 1.1.2

Key readings 1.1.2: Identification of Linux distributions

✓ Description of Linux Distribution

- Definition

- **Linux Server:** A popular open-source NOS known for its flexibility, scalability, and security. It is widely used in web hosting, enterprise servers, and embedded systems.

Linux is an operating system made up of several key components, each playing a specific role in managing the system's hardware, software, and user interactions.

- **Components of Linux**:

1. Kernel

- The core part of the Linux operating system.
- Types of kernels include **monolithic kernels** (Linux is a monolithic kernel) and **microkernels**.
- It serves as a bridge between software applications and the hardware.

2. System Libraries

- These are special programs that provide application programs with access to kernel features without needing to directly communicate with the kernel.
- **GNU C Library (glibc)** is an example that allows interaction with system functionalities like input/output handling, memory management, etc.

3. System Utilities

- Programs responsible for performing specialized, individual-level tasks.
- These utilities are essential for system maintenance, monitoring, and management.
- **Examples**: ps for listing processes, df for displaying disk usage, top for real-time process monitoring.

4. Shell

- The shell acts as an interface between the user and the kernel.
- It processes commands entered by the user and then sends them to the operating system to be executed.
- **Types of Shells**:
  - **Bash (Bourne Again Shell)**: The most common shell.
  - **Zsh, Fish**: Other shell varieties.
- You interact with the shell through the **Terminal**, where you input commands.

5. File System

- Linux organizes files and directories in a hierarchical structure.
- The **root** directory (/) is the top-level directory, and everything branches off from there.
- **Common Directories**:
  - /root: The home directory for the root user.
  - /home: Contains user directories.
  - /bin and /sbin: Essential binaries for users and the system.
  - /etc: Configuration files.
  - /var: Variable data files like logs.
  - /dev: Device files (interfaces for hardware like disks, mice, etc.).

✓ **Common Linux distributions**

- **Ubuntu**

  - **Base:** Debian
  - **Target Users:** Beginners, general desktop users, developers.
  - **Use Case:** Desktop, server, cloud.

    - **Key Features:**

  - User-friendly interface.
  - Extensive documentation and community support.
  - Regular releases (LTS versions are long-term support).
  - **Popular Derivatives:** Kubuntu, Xubuntu, Lubuntu, Ubuntu Server.

- **Debian**

  - **Base:** Independent
  - **Target Users:** Advanced users, servers, stable environments.
  - **Use Case:** Desktop, server.

- **Key Features:**
  - Very stable and secure.
  - Long release cycles.
  - Large package repository (APT package management system).

Fedora

- **Base:** Independent (sponsored by Red Hat).
- **Target Users:** Developers, enthusiasts, cutting-edge technology users.
- **Use Case:** Desktop, development, workstation.

- **Key Features:**
  - Cutting-edge software and frequent updates.
  - Emphasis on open-source software.
  - The foundation for Red Hat Enterprise Linux (RHEL).

CentOS

- **Base:** Red Hat Enterprise Linux (RHEL).
- **Target Users:** Enterprise users, system administrators, servers.
- **Use Case:** Servers, enterprise environments, cloud.

- **Key Features:**
  - Stability and reliability (downstream rebuilds of RHEL).
  - Long support cycles.
  - CentOS Stream offers a rolling-release model.

Arch Linux

- **Base:** Independent
- **Target Users:** Advanced users, enthusiasts.
- **Use Case:** Desktop, highly customizable environments.

- **Key Features:**
  - Rolling release model (always up to date).
  - Minimalistic, highly customizable.
  - Manual installation process (user must configure everything).

- **Popular Derivatives:** Manjaro (user-friendly Arch-based distro).

- **Linux Mint**
  - **Base:** Ubuntu/Debian
  - **Target Users:** Beginners, desktop users.
  - **Use Case:** Desktop.

- **Key Features:**
  - Easy to use with pre-installed software.
  - Cinnamon, MATE, and Xfce desktop environments.
  - Focus on stability and usability.

Practical Activity 1.1.3: Installing Linux Operating system

Task:

1: Read the key reading 1.1.3

2: Referring to the key reading 1.1.3, you are requested to go to the computer lab to install Linux operating system.

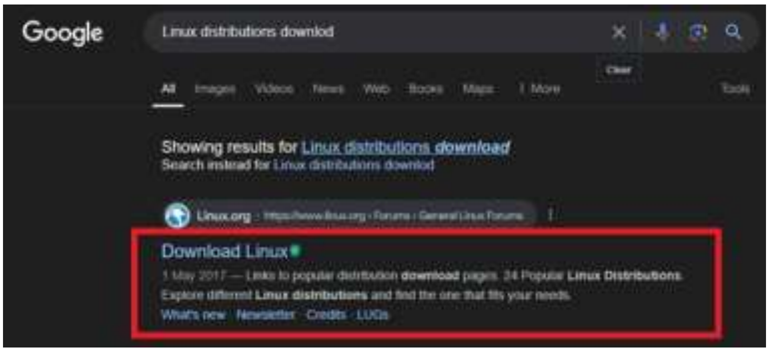
3: Present your work to the trainer and whole class

4: Ask questions for clarification where necessary

Key readings 1.1.3: Installing Linux Operating system

Steps1: Download Linux distribution

1) **Open browser and Visit the Official Linux distribution Website:** https://www.linux.org/pages/download/



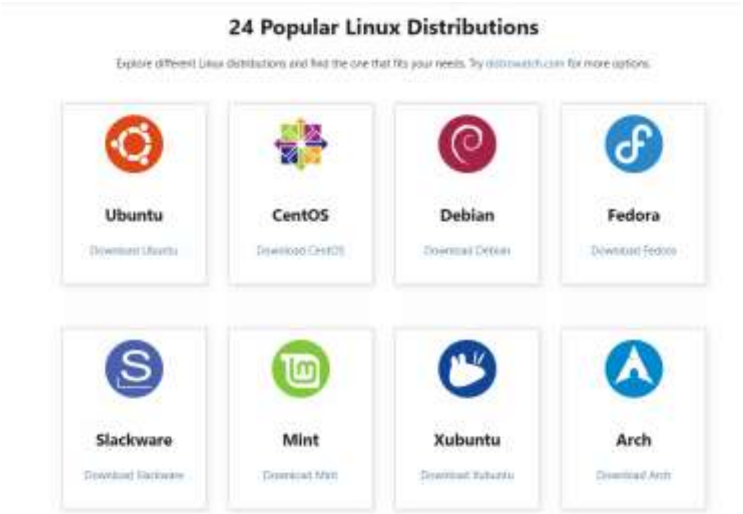
Showing results for Linux distributions download
Search instead for Linux distributions downlod

Linux.org · https://www.linux.org › Forums › General Linux Forums

Download Linux
1 May 2017 — Links to popular distribution download pages. 24 Popular Linux Distributions. Explore different Linux distributions and find the one that fits your needs.
What's new · Newsletter · Credits · LUGs

Link:

2) **Choose Your Edition:** Linux distribution offers various editions like Desktop, Server, and Mate. Select the one that best suits your needs.



24 Popular Linux Distributions

Explore different Linux distributions and find the one that fits your needs. Try distrowatch.com for more options.

Ubuntu
Download Ubuntu

CentOS
Download CentOS

Debian
Download Debian

Fedora
Download Fedora

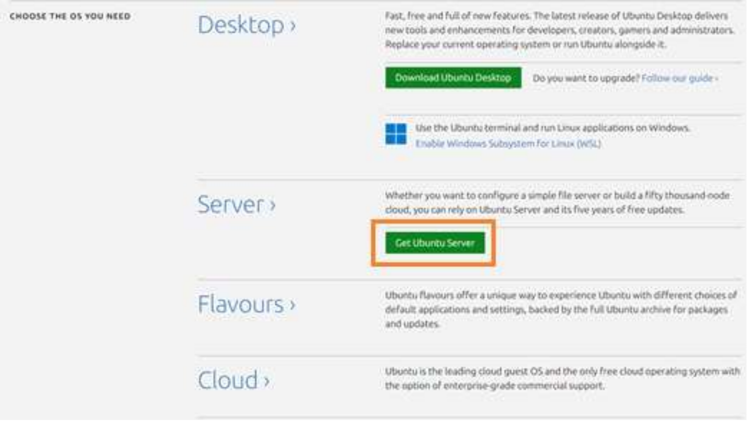
Slackware
Download Slackware

Mint
Download Mint

Xubuntu
Download Xubuntu

Arch
Download Arch

3) Select Linux operations included Desktop, server or cloud



CHOOSE THE OS YOU NEED

Desktop ›

Fast, free and full of new features. The latest release of Ubuntu Desktop delivers new tools and enhancements for developers, creators, gamers and administrators. Replace your current operating system or run Ubuntu alongside it.

Download Ubuntu Desktop   Do you want to upgrade? Follow our guide ›

Use the Ubuntu terminal and run Linux applications on Windows.
Enable Windows Subsystem for Linux (WSL)

Server ›

Whether you want to configure a simple file server or build a fifty thousand-node cloud, you can rely on Ubuntu Server and its five years of free updates.

Get Ubuntu Server

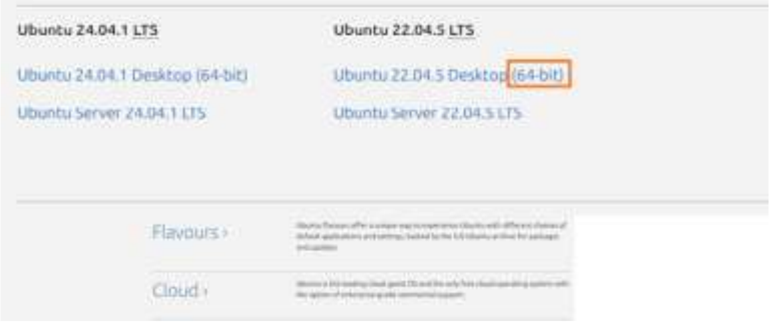
Flavours ›

Ubuntu flavours offer a unique way to experience Ubuntu with different choices of default applications and settings, backed by the full Ubuntu archive for packages and updates.

Cloud ›

Ubuntu is the leading cloud guest OS and the only free cloud operating system with the option of enterprise-grade commercial support.

4) **Select the Architecture:** Choose between 64-bit (x86_64) or 32-bit (i386) based on your hardware. Most modern computers use 64-bit architecture.



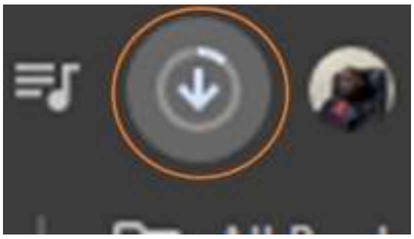
Ubuntu 24.04.1 LTS
Ubuntu 24.04.1 Desktop (64-bit)
Ubuntu Server 24.04.1 LTS

Ubuntu 22.04.5 LTS
Ubuntu 22.04.5 Desktop (64-bit)
Ubuntu Server 22.04.5 LTS

Flavours ›
[illegible]

Cloud ›
[illegible]

5) **Start the Download:** Click the "Download" button to initiate the ISO file download



Once the download is complete, you can use the ISO file to create a bootable USB drive or DVD and install Ubuntu on your computer.

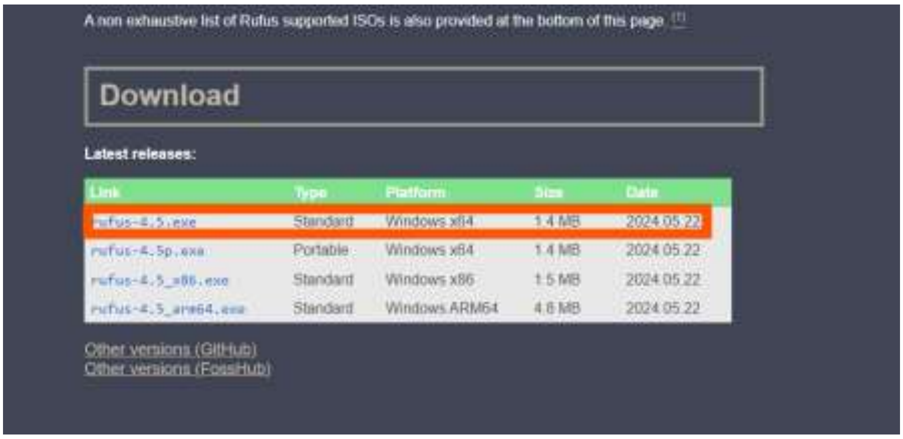
Steps to make a storage device bootable

1. **Prepare Your Storage Device:**

- **Check Size:** Ensure the storage device has greater than 4GB .

2. **Download Bootable Media Creation Tool:**

- **Rufus (Windows):** A popular tool for creating bootable media by clicking on selected link.



A non exhaustive list of Rufus supported ISOs is also provided at the bottom of this page [?]

Download

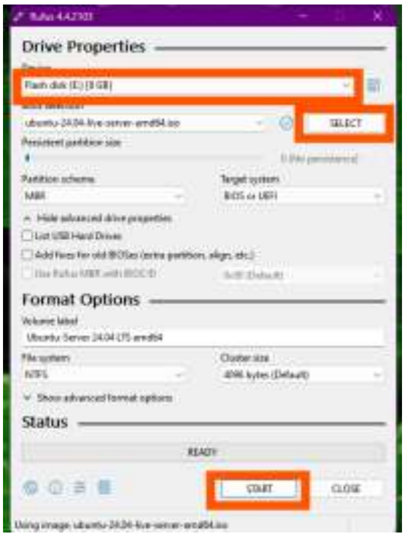
Latest releases:

Link	Type	Platform	Size	Date
rufus-4.5.exe	Standard	Windows x64	1.4 MB	2024.05.22
rufus-4.5p.exe	Portable	Windows x64	1.4 MB	2024.05.22
rufus-4.5_x86.exe	Standard	Windows x86	1.5 MB	2024.05.22
rufus-4.5_arm64.exe	Standard	Windows ARM64	4.8 MB	2024.05.22

Other versions (GitHub)
Other versions (FossHub)

3. **Launch the Tool:**

- Open the downloaded tool, select iso file, choose target devices and start creation process.



Rufus 4.4.2103

Drive Properties

Device

Flash disk (E:) [8 GB]

Boot selection

ubuntu-24.04-live-server-amd64.iso

Persistent partition size

0 (No persistence)

Partition scheme

MBR

Target system

BIOS or UEFI

Hide advanced drive properties

□ List USB Hard Drives

□ Add fixes for old BIOSes (extra partition, align, etc.)

□ Use Rufus MBR with BIOS ID

0x80 (Default)

Format Options

Volume label

Ubuntu-Server 24.04 LTS amd64

File system

NTFS

Cluster size

4096 bytes (Default)

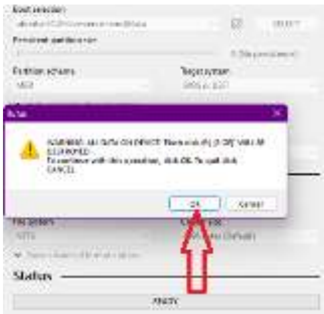
Show advanced format options

Status

READY

Using image: ubuntu-24.04-live-server-amd64.iso

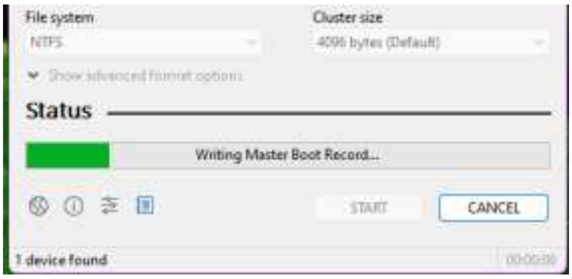
4. format the disk



Rufus

WARNING: ALL DATA ON DEVICE 'Flash disk (E:) [8 GB]' WILL BE DESTROYED.
To continue with this operation, click OK. To quit click CANCEL.

5. wait until a process complete



File system

NTFS

Cluster size

4096 bytes (Default)

Show advanced format options

Status

Writing Master Boot Record...

1 device found

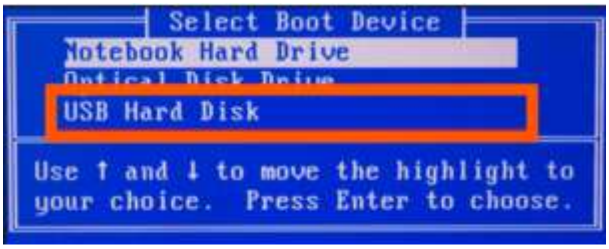
00:00:08

Steps 2: Install Linux from USB

Step 1: Boot from the USB Drive

1. **Insert the USB Drive**:
   - Insert the bootable Ubuntu USB drive into your computer.

2. **Restart Your Computer**:
   - Restart the computer and press the key to access the **boot menu**. The key is usually **F2**, **F10**, **F12**, or **Esc**, depending on your system.
3. **Select USB Drive**:
   - From the boot menu, select the USB drive to boot from.

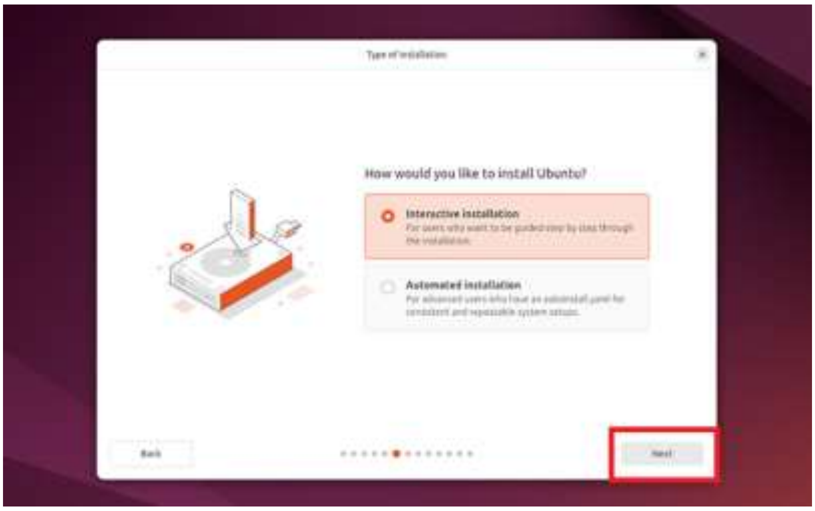


```
Select Boot Device
Notebook Hard Drive
Optical Disk Drive
USB Hard Disk

Use ↑ and ↓ to move the highlight to
your choice.  Press Enter to choose.
```

4. **Start Ubuntu Installer**:
   - Your computer will now boot from the USB drive, and you'll be greeted with the Ubuntu welcome screen.

5. Installation Setup

You will be prompted to choose between **Interactive installation** and **Automated Installation**. The interactive option is the standard route, but more advanced users can use the automated installation option to import a configuration file from a web server to standardise multiple installs and add further customisations.



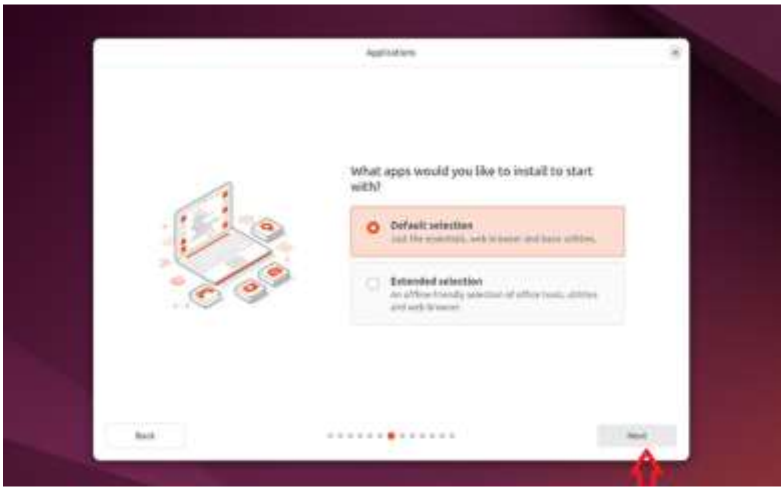
How would you like to install Ubuntu?

● Interactive installation
For users who want to be guided step by step through the installation.

○ Automated installation
For advanced users who have an autoinstall yaml file for consistent and repeatable system setups.

In this tutorial we will remain on the primary route.

You will be prompted to choose between the **Default selection** and **Extended selection** options. The default installation comes with the basic essentials to get started which you can then expand on after install using the App Center. The extended selection contains additional office tools and utilities, useful for offline situations.

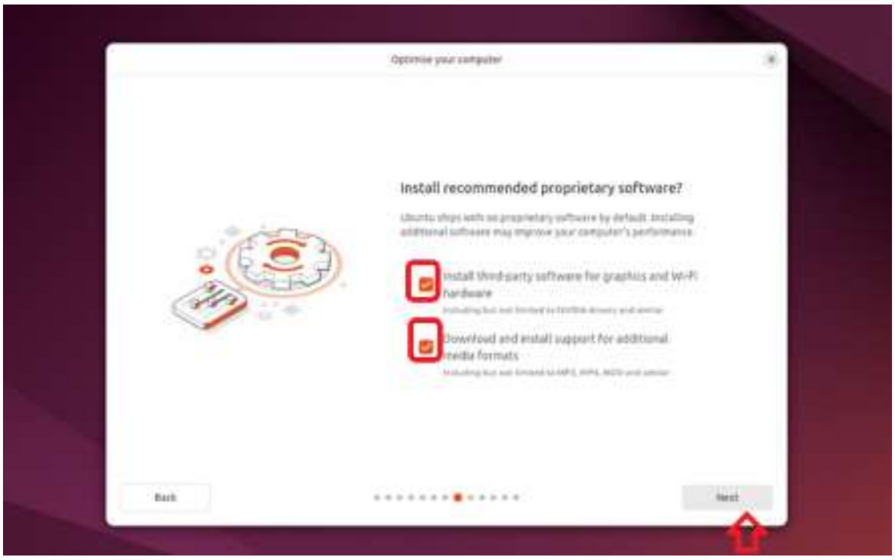


What apps would you like to install to start with?

● Default selection
Just the essentials, web browser and basic utilities.

○ Extended selection
An offline-friendly selection of office tools, utilities and web browser.

In the following screen you will be prompted to install third-party software that may improve device support and performance (for example, Nvidia graphics drivers) and support for additional media formats. It is recommended to check both of these boxes.



Optimise your computer

Install recommended proprietary software?

Ubuntu ships with no proprietary software by default. Installing additional software may improve your computer's performance.

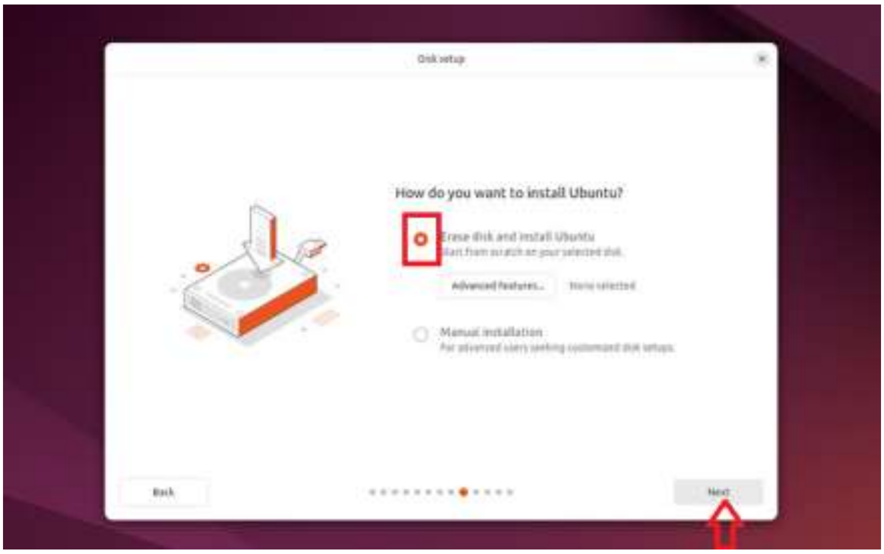
☑ Install third-party software for graphics and Wi-Fi hardware
Including but not limited to NVIDIA drivers and similar

☑ Download and install support for additional media formats
Including but not limited to MP3, MP4, MOV and similar

6. Type of installation

This screen allows you to configure your installation. If you would like Ubuntu to be the only operating system on your hard drive, select **Erase disk and install Ubuntu**.

If your device currently has another operating system installed, you will receive additional options to install Ubuntu alongside that OS rather than replacing it.



Disk setup

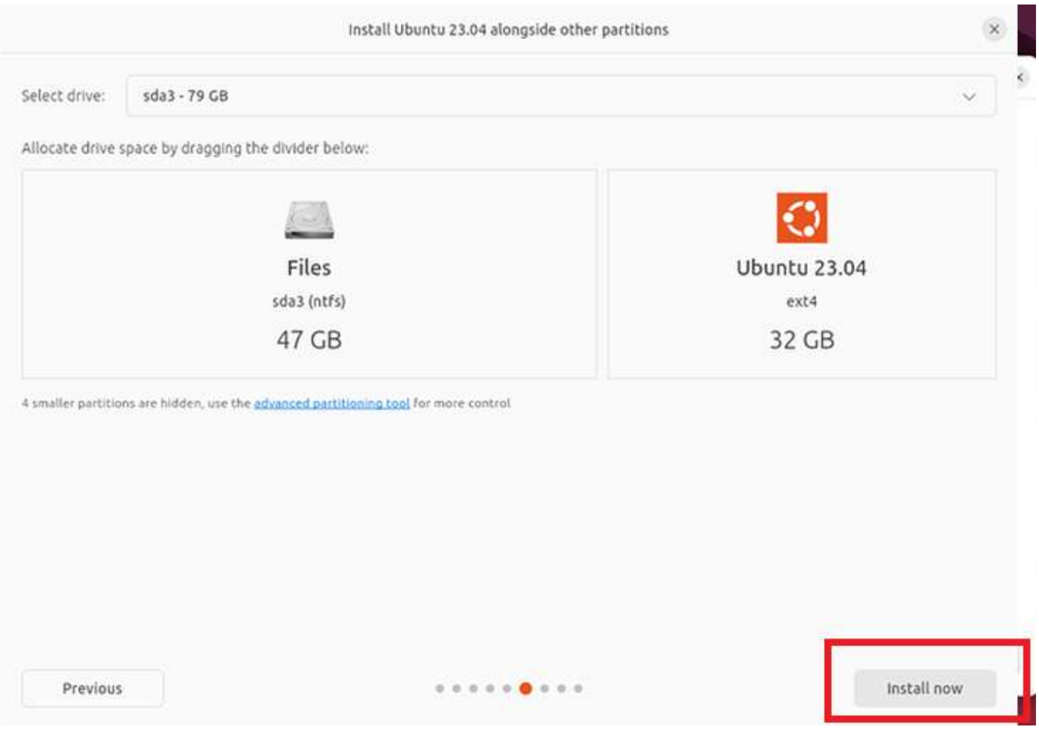
How do you want to install Ubuntu?

● Erase disk and install Ubuntu
Start from scratch on your selected disk.

Advanced features... None selected

○ Manual installation
For advanced users seeking customised disk setups.

Let's take a moment to review all of the above options in detail.



Install Ubuntu 23.04 alongside other partitions

Select drive: sda3 - 79 GB

Allocate drive space by dragging the divider below:

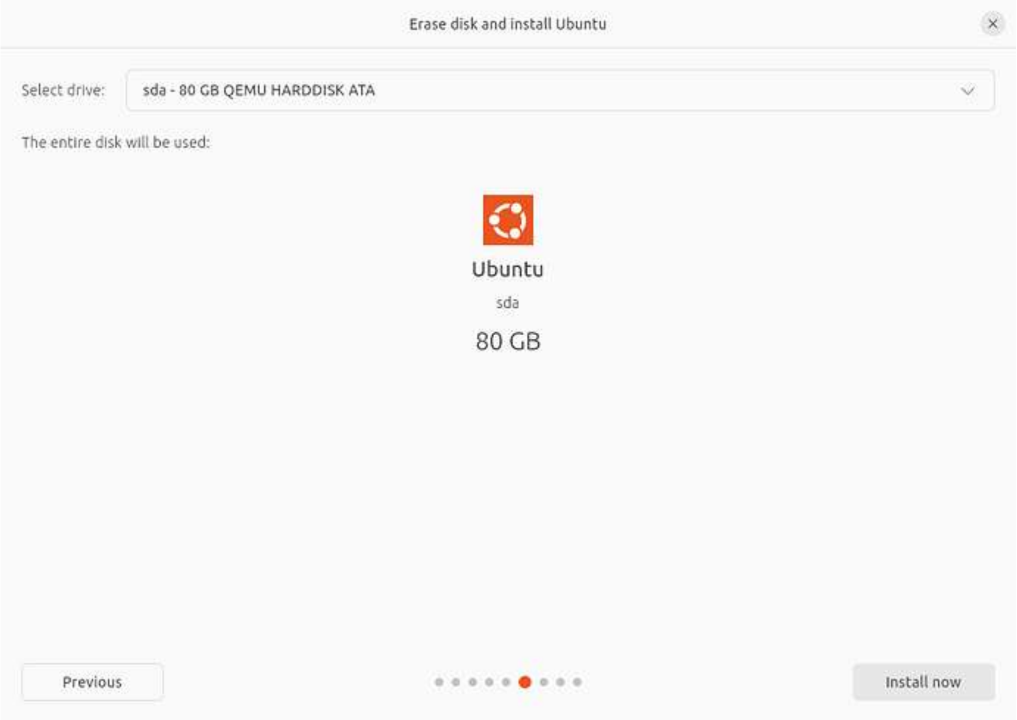
Files
sda3 (ntfs)
47 GB

Ubuntu 23.04
ext4
32 GB

4 smaller partitions are hidden, use the advanced partitioning tool for more control

Erase disk and install Ubuntu

If you select this option Ubuntu will take up the entire disk space on the selected drive.



Erase disk and install Ubuntu

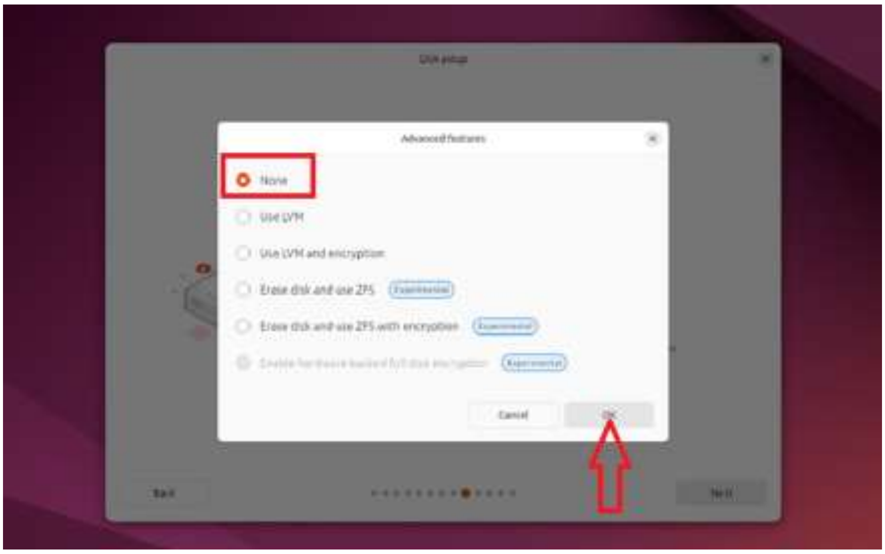
Select drive: sda - 80 GB QEMU HARDDISK ATA

The entire disk will be used:

Ubuntu
sda
80 GB

If your PC has multiple hard drives then this option allows you to install Ubuntu alongside an existing OS as long as they each have their own drive. Take care to ensure that you are selecting the right drive in this instance!

This option also allows you to encrypt your entire drive using LVM, ZFS or using the Trusted Platform Module on the device. To do this open the Advanced features option before proceeding to the above screen and select 'Encrypt the new Ubuntu installation for security'



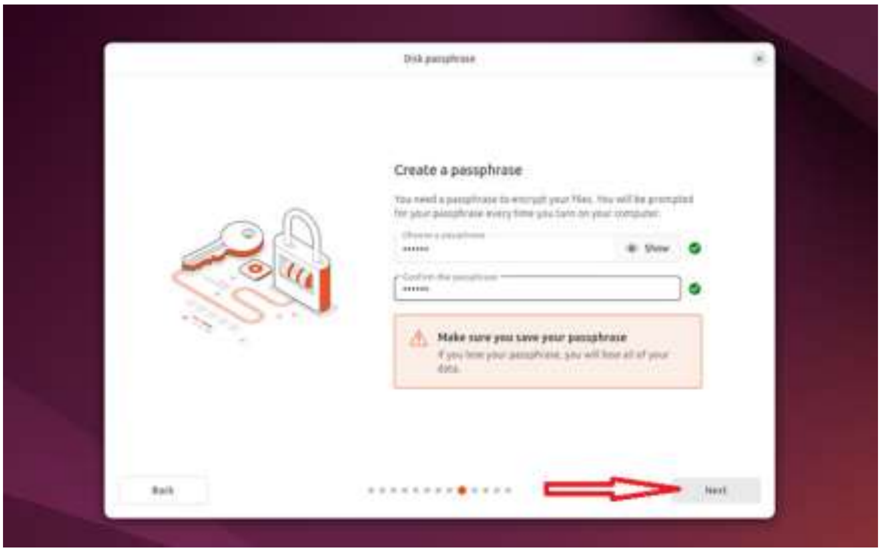
● None
○ Use LVM
○ Use LVM and encryption
○ Erase disk and use ZFS Experimental
○ Erase disk and use ZFS with encryption Experimental
○ Enable hardware-backed full disk encryption Experimental

LVM stands for Logical Volume Management. By using LVM during the setup, it makes it easier to create and manage partitions post installation.

ZFS allows users to create pooled storage volumes that span multiple drives as well as snapshorts and data repair features. It is a powerful option for advanced users.

TPM-backed full disk encryption is a new, highly experimental feature of Ubuntu Desktop that currently supports only the generic kernel. This means that machines that require additional drivers to support webcams or NVIDIA graphics cards will not support this setup until additional features land after release. In addition, certain hardware vendors may have BIOS options enabled that alter the chain of trust. Please do not select this option unless you are comfortable debugging or re-installing in the event of an issue.

If you select either LVM or ZFS based encryption you will be prompted to create a Security key that you will need to enter on boot before logging in with your user credentials.



Create a passphrase

You need a passphrase to encrypt your files. You will be prompted for your passphrase every time you turn on your computer.

Choose a passphrase: ••••••

Confirm the passphrase: ••••••

Make sure you save your passphrase

If you lose your passphrase, you will lose all of your data.

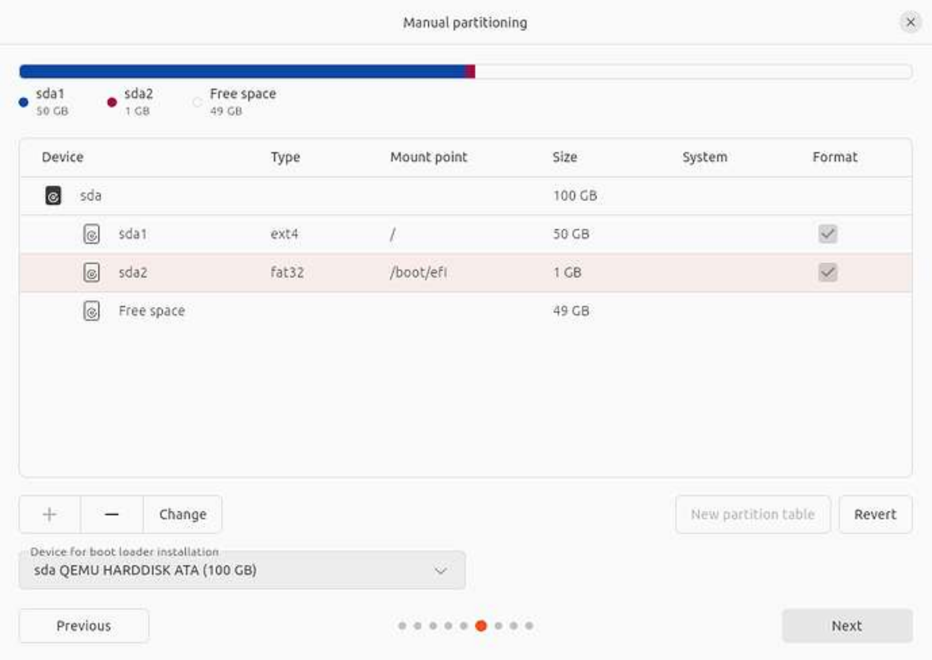
If you are using TPM-based Full Disc Encryption you will be prompted to run the command snap recovery --show-keys after installing to generate a recovery key.

If you select encryption, it is important that you do not lose your security key! Write it down and store it in a safe place outside of your local system. **You will not be able to recover your data without it!**

Manual partitioning

Manual partitioning is designed for advanced users who want to create specific configurations for their use-cases. As such we assume that these users will be comfortable with this interface and will not go into detail during this tutorial on specific setups.

Here users can see all existing drives and partitions and create and manage new partition tables and configurations.



Manual partitioning

● sda1 50 GB ● sda2 1 GB ○ Free space 49 GB

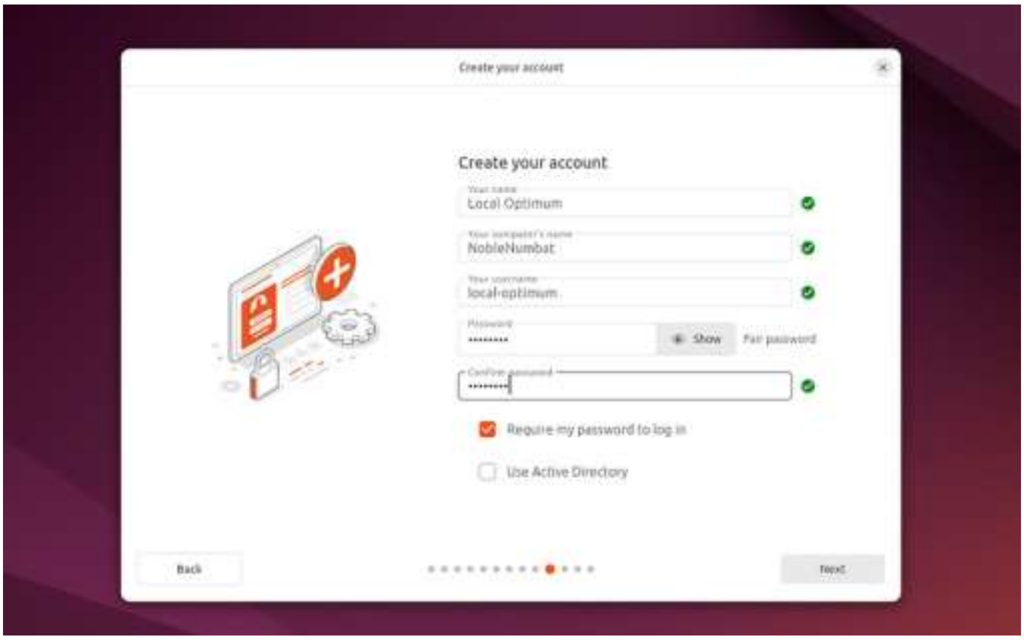
Device	Type	Mount point	Size	System	Format
sda			100 GB		
sda1	ext4	/	50 GB		☑
sda2	fat32	/boot/efi	1 GB		☑
Free space			49 GB		

Device for boot loader installation: sda QEMU HARDDISK ATA (100 GB)

7.Create Your Login Details

On this screen, you will be prompted to enter your name and the name of your computer as it will appear on the network. Finally, you will create a username and a strong password.

You can choose to log in automatically or require a password. If you are using your device whilst travelling, it's recommended to keep "Require my password to log in" enabled.



Create your account

Create your account

Your name: Local Optimum

Your computer's name: NobleNumbat

Your username: local-optimum

Password: ********

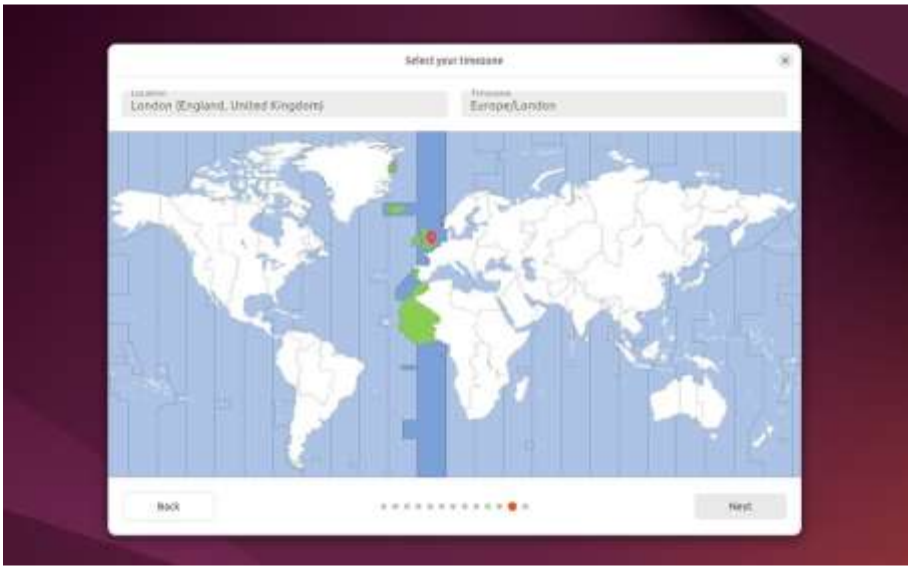
Confirm password: ********

☑ Require my password to log in

□ Use Active Directory

8. Choose your Location

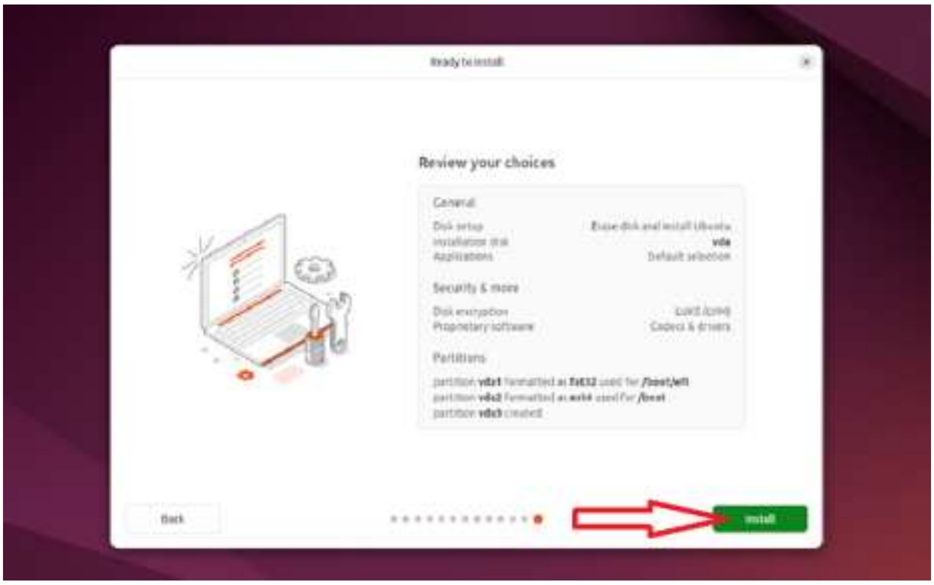
Select your location and timezone from the map screen and click **Continue**. This information will be detected automatically if you are connected to the internet.



Location: London (England, United Kingdom)
Timezone: Europe/London

9. Ready to install

Clicking **Next** will take you to a summary of your installation configuration to give you a chance to confirm your setup before clicking **Install**.



Review your choices

General

Disk setup: Erase disk and install Ubuntu
Installation disk: **vda**
Applications: Default selection

Security & more

Disk encryption: LUKS (LVM)
Proprietary software: Codecs & drivers

Partitions

partition **vda1** formatted as **fat32** used for ***/boot/efi***
partition **vda2** formatted as **ext4** used for ***/boot***
partition **vda3** created

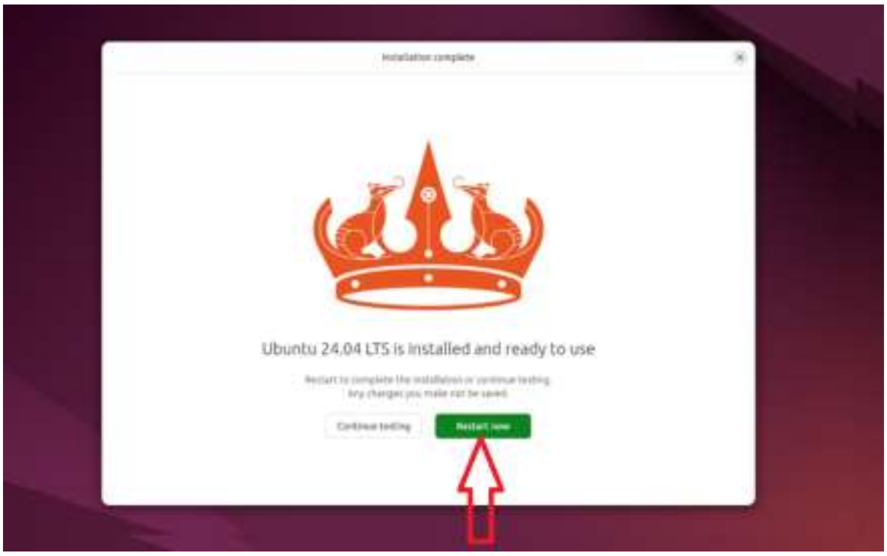
10. Complete the Installation

Sit back and enjoy the slideshow as Ubuntu installs in the background! 🙂

11. Complete the Installation

Sit back and enjoy the slideshow as Ubuntu installs in the background!

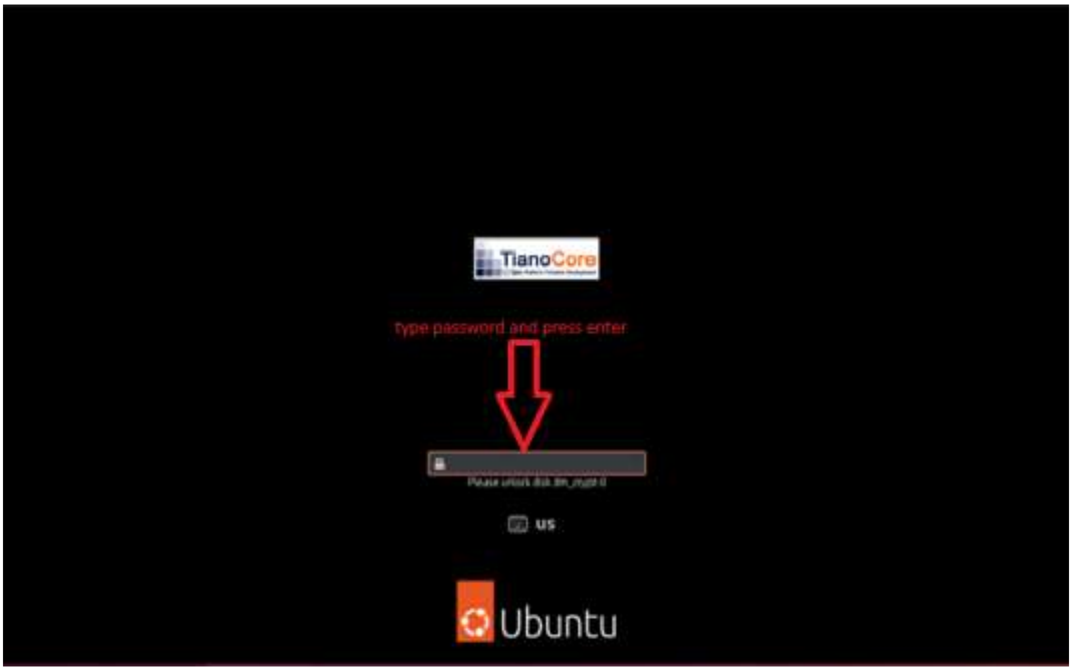
Click restat now to enable login Portal



Ubuntu 24.04 LTS is installed and ready to use

Restart to complete the installation or continue testing.
Any changes you make will not be saved.

Provide cridential to access the ubuntu linux background interface.



Please unlock disk dm_crypt-0

The welcome widget will help you with some additional setup options

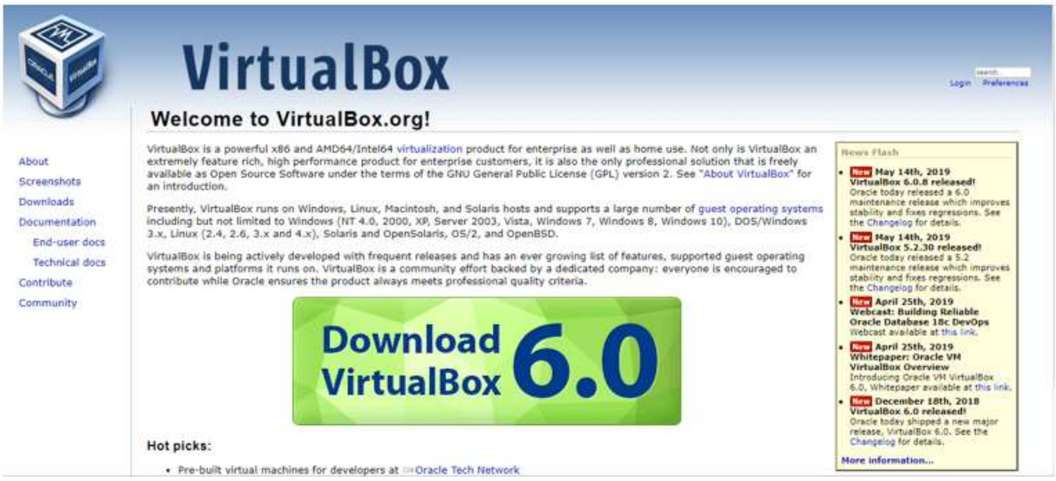
Alternate Method 2: Download Ubuntu on your Virtual Machine

There are several ways to install Ubuntu. If you want to use Ubuntu without making any changes in your Windows system, you can go the virtual machine way. Basically, you install and use this distribution like any regular Windows application. When you just want to try a Linux distro (Ubuntu in this case) for limited use, virtual machines provide a more comfortable option.

Second way

1. Download and install Virtual Box

- Go to the ***official website*** of Oracle Virtual Box and get the latest stable version.



Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Hot picks:

- Pre-built virtual machines for developers at Oracle Tech Network

News Flash

- New May 14th, 2019 **VirtualBox 6.0.8 released!** Oracle today released a 6.0 maintenance release which improves stability and fixes regressions. See the Changelog for details.
- New May 14th, 2019 **VirtualBox 5.2.30 released!** Oracle today released a 5.2 maintenance release which improves stability and fixes regressions. See the Changelog for details.
- New April 25th, 2019 **Webcast: Building Reliable Oracle Database 18c DevOps** Webcast available at this link.
- New April 25th, 2019 **Whitepaper: Oracle VM VirtualBox Overview** Introducing Oracle VM VirtualBox 6.0. Whitepaper available at this link.
- New December 18th, 2018 **VirtualBox 6.0 released!** Oracle today shipped a new major release, VirtualBox 6.0. See the Changelog for details.

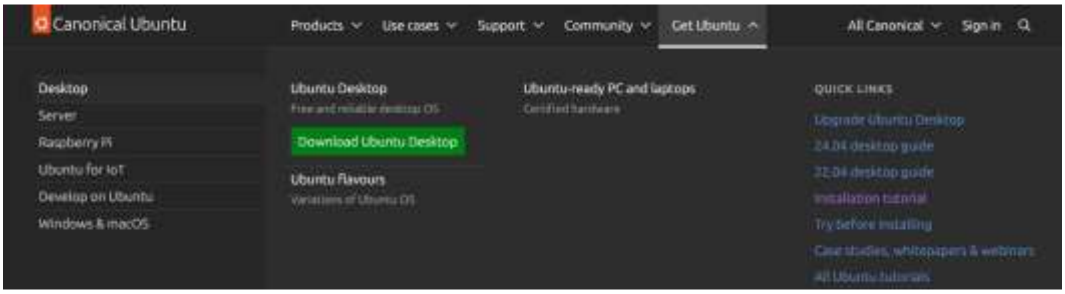
More information...

- Installing Virtual Box is no rocket science. Just double-click on the downloaded .exe file to run it and follow the instructions on the screen. It is like installing regular software on Windows.

1. **Download Linux ISO**

   Open your browser then type Ubuntu.com

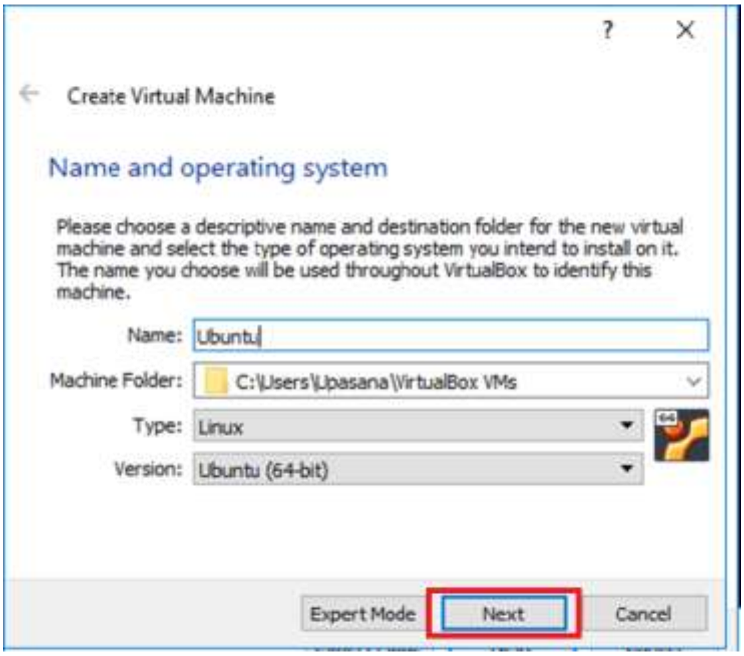
   Next, you need to download the ISO file of the Linux distribution, i.e, Ubuntu. You can get this image from ***this link***.



3. Install Ubuntu using Virtual Box

Now, you have installed Virtual Box and downloaded the ISO for Linux. You are now all set to install Linux in Virtual Box.

- Start Virtual Box, and click on the ***New Symbol*** (it looks like a blue star). Give the virtual OS a relevant name, I'd reckon you give the name of the distro you're going to install – Ubuntu.



Create Virtual Machine

Name and operating system

Please choose a descriptive name and destination folder 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.

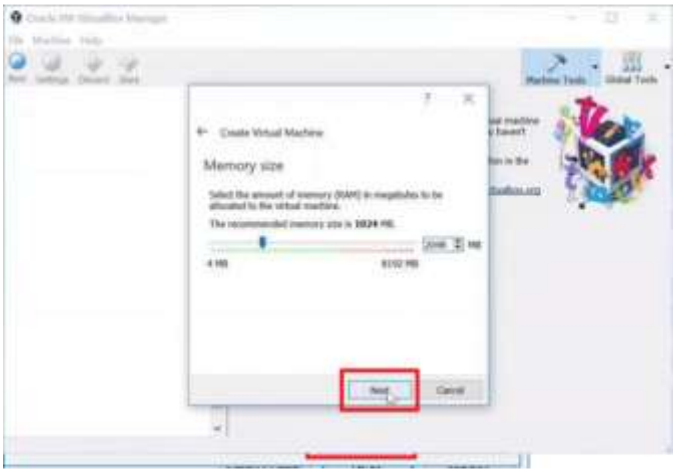
Name: Ubuntu

Machine Folder: C:\Users\Upasana\VirtualBox VMs

Type: Linux

Version: Ubuntu (64-bit)

Allocate RAM to the virtual OS. My system has 8 GB of RAM and I decided to allocate 4 GB of RAM to it. You can use extra RAM if your system has more RAM.



Create Virtual Machine

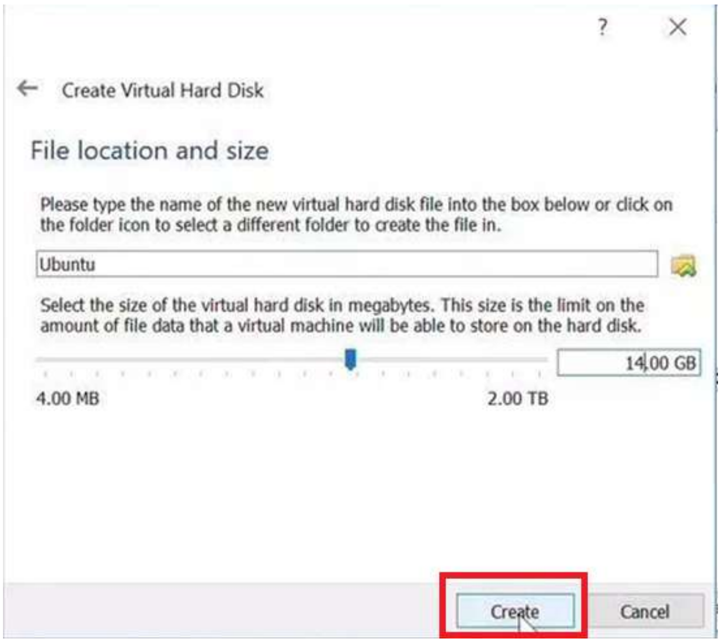
Memory size

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

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

2048 MB

- You can choose either to ***Dynamically allocate*** or a ***Fixed size*** option for creating the virtual hard disk. ***The recommended size is 10 GB.***



Create Virtual Hard Disk

File location and size

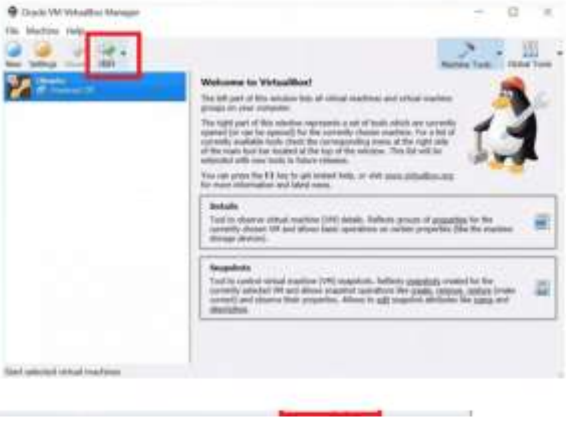
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

Ubuntu

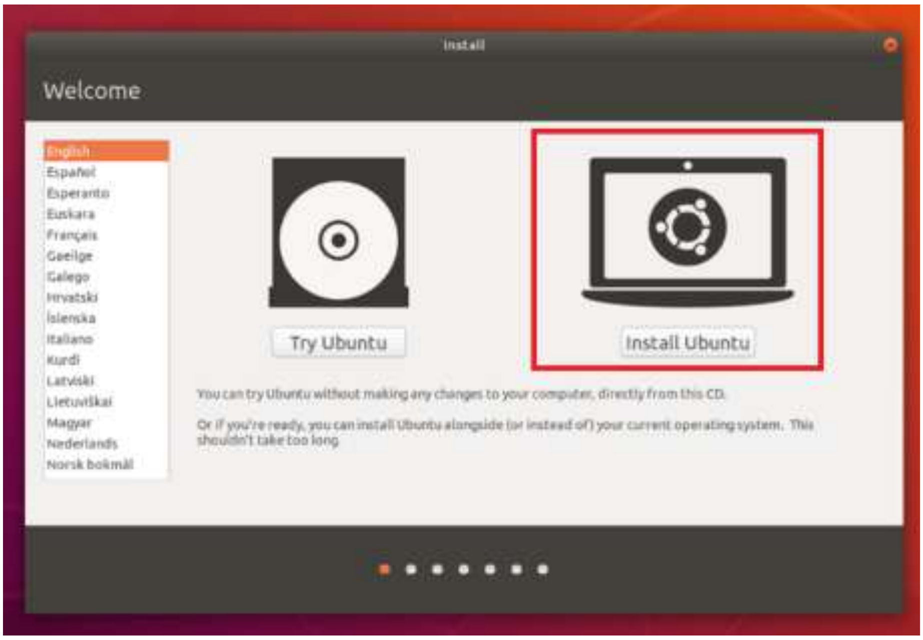
Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

14.00 GB

Once everything is in place, it's time to boot that ISO and install Ubuntu as a virtual operating system. If Virtual Box doesn't detect the Linux ISO, browse to its location by clicking the folder icon as shown in the picture.



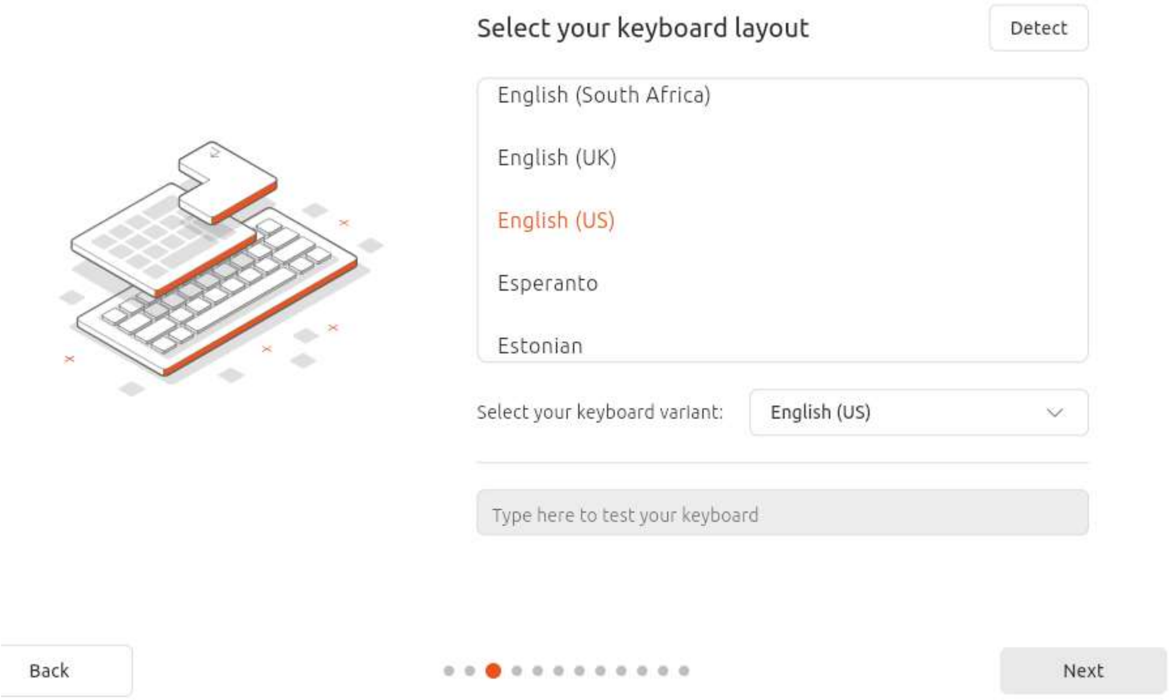
Soon you'll find yourself inside Linux. You'll end up with the option to install it.



You can try Ubuntu without making any changes to your computer, directly from this CD.

Or if you're ready, you can install Ubuntu alongside (or instead of) your current operating system. This shouldn't take too long.

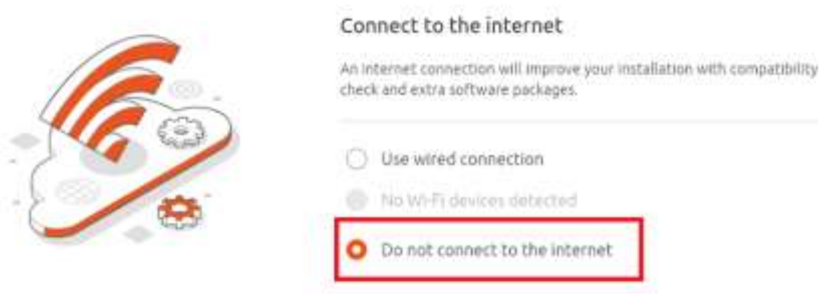
Select your Keyboard



Select your keyboard layout

Select your keyboard variant: English (US)

Chose the option do not connect to the network the click next



Connect to the internet

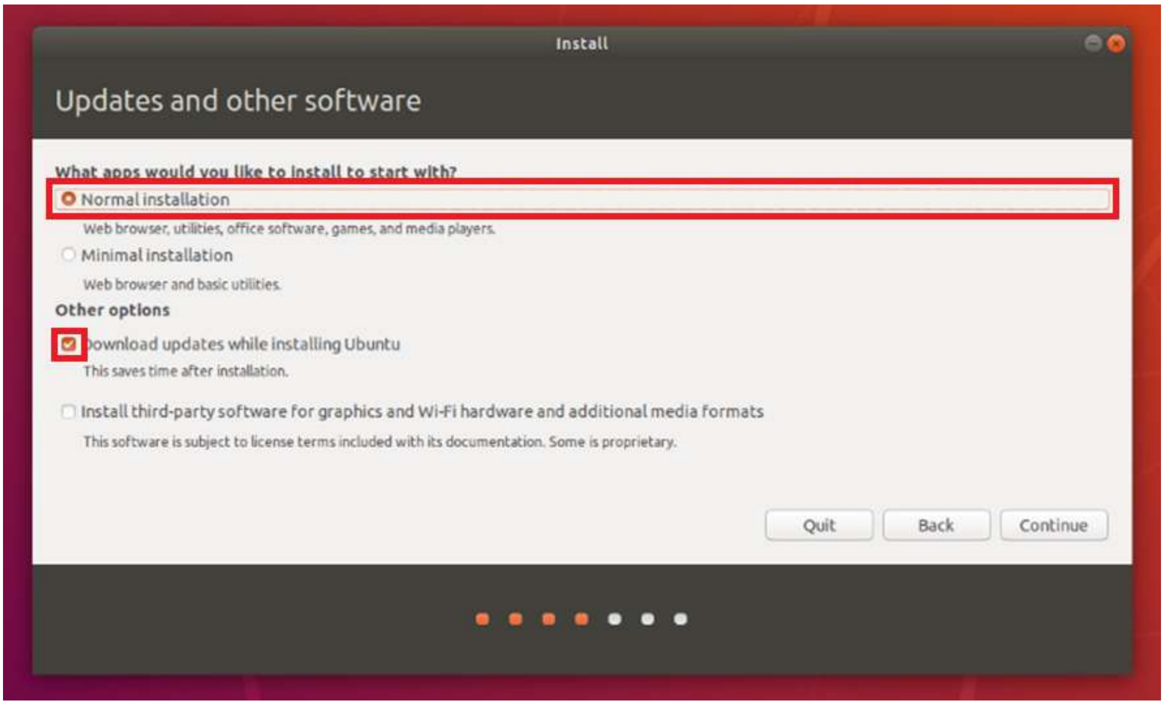
An internet connection will improve your installation with compatibility check and extra software packages.

○ Use wired connection

○ No Wi-Fi devices detected

● Do not connect to the internet

Prepare installation



Install

Updates and other software

What apps would you like to install to start with?

● Normal installation

Web browser, utilities, office software, games, and media players.

○ Minimal installation

Web browser and basic utilities.

Other options

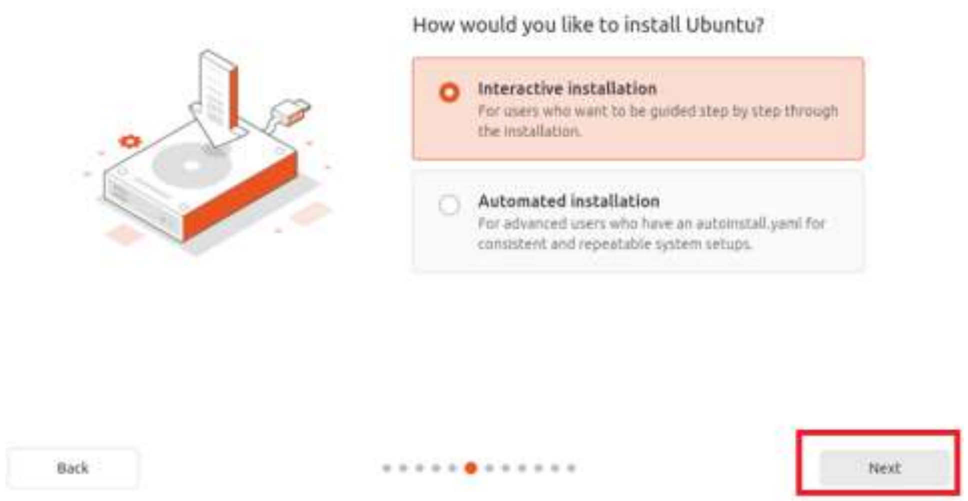
☑ Download updates while installing Ubuntu

This saves time after installation.

□ Install third-party software for graphics and Wi-Fi hardware and additional media formats

This software is subject to license terms included with its documentation. Some is proprietary.

Click continue

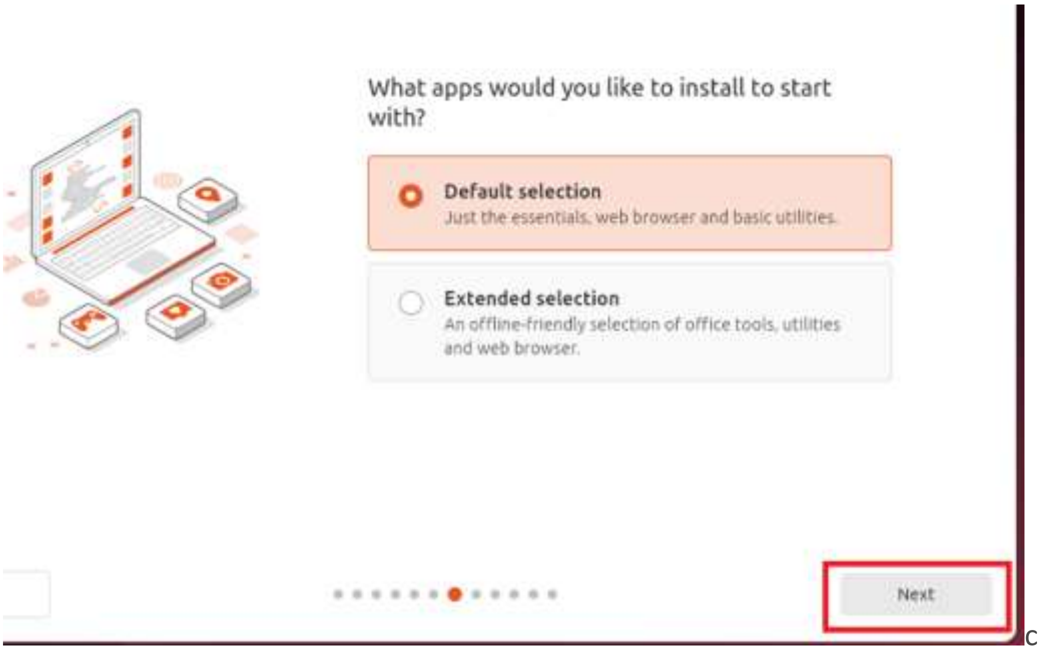


How would you like to install Ubuntu?

● **Interactive installation**
For users who want to be guided step by step through the installation.

○ **Automated installation**
For advanced users who have an autoinstall.yaml for consistent and repeatable system setups.

Click next



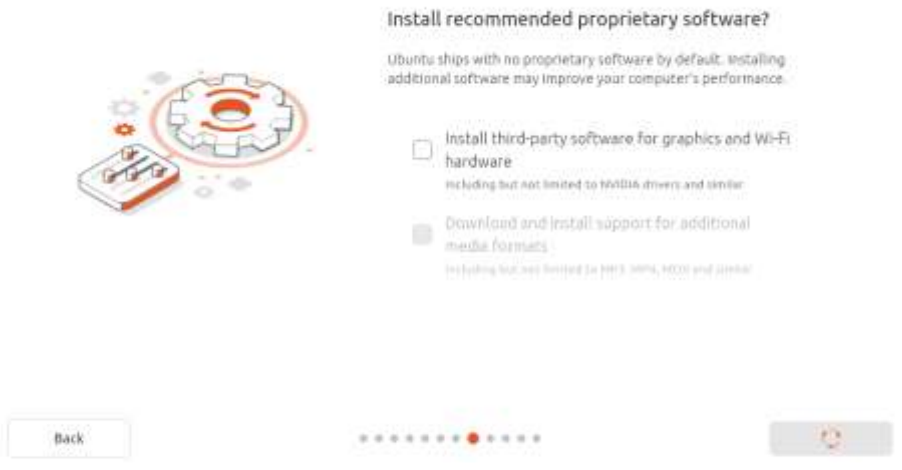
What apps would you like to install to start with?

● **Default selection**
Just the essentials, web browser and basic utilities.

○ **Extended selection**
An offline-friendly selection of office tools, utilities and web browser.

c

Choose the default section then click next



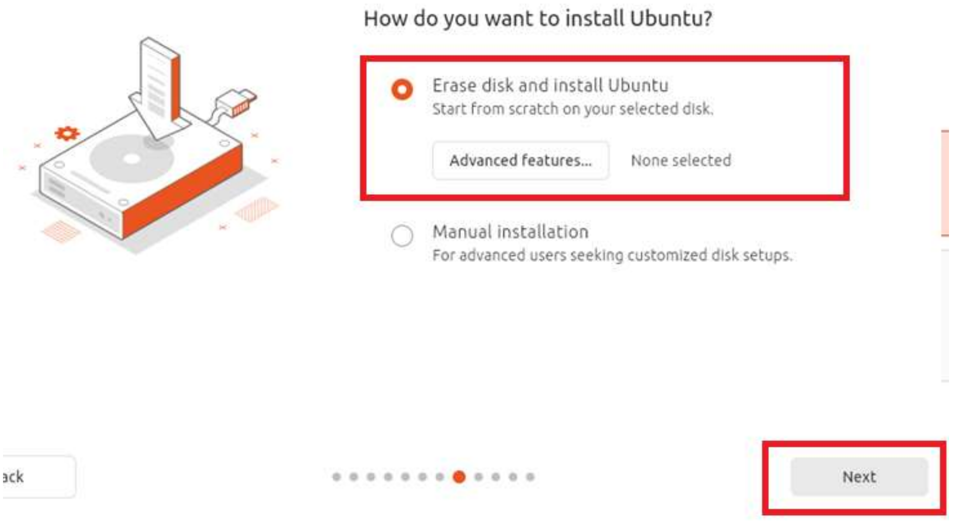
Install recommended proprietary software?

Ubuntu ships with no proprietary software by default. Installing additional software may improve your computer's performance.

☐ Install third-party software for graphics and Wi-Fi hardware
Including but not limited to NVIDIA drivers and similar

☐ Download and install support for additional media formats
Including but not limited to MP3, MP4, HCOI and similar

Click install

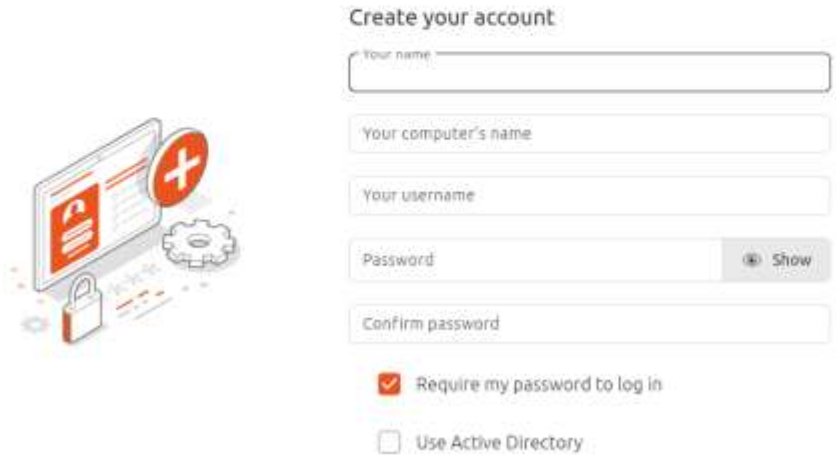


How do you want to install Ubuntu?

● Erase disk and install Ubuntu
Start from scratch on your selected disk.
Advanced features... None selected

○ Manual installation
For advanced users seeking customized disk setups.

Choose eras disk and install Ubuntu then Click next


Create your account

Your name

Your computer's name

Your username

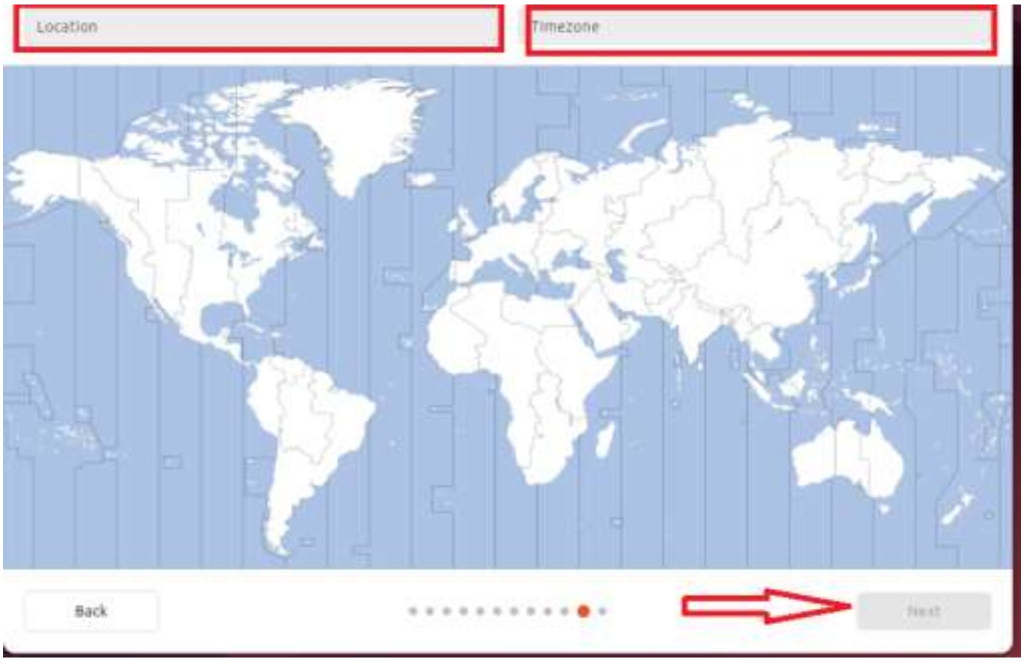
Password

Confirm password

☑ Require my password to log in

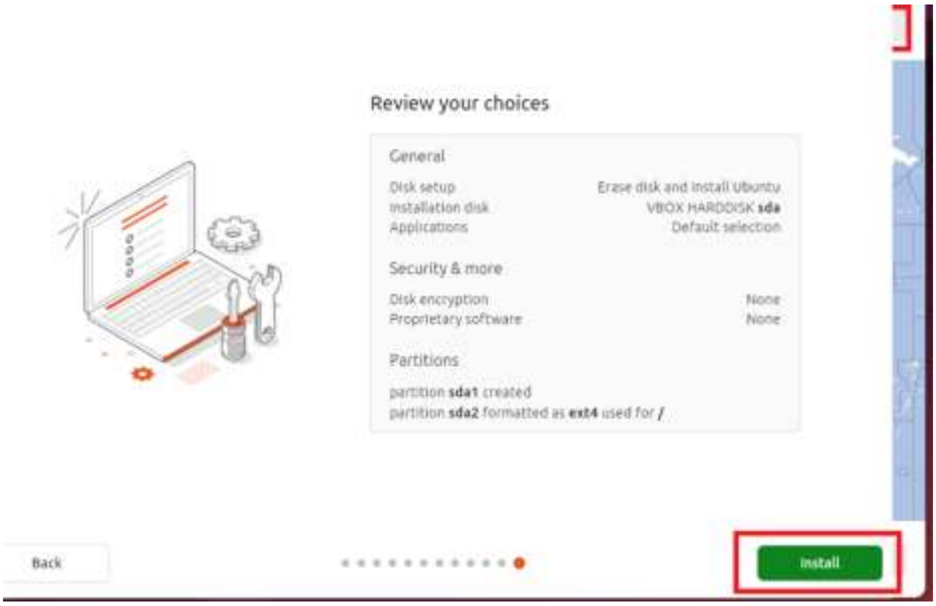
□ Use Active Directory

- Choose whether to **use Active directory** or require may password each time you log in. click next


Location

Timezone

Choose location and time zone then click next



Review your choices

General

Disk setup	Erase disk and install Ubuntu
Installation disk	VBOX HARDDISK **sda**
Applications	Default selection

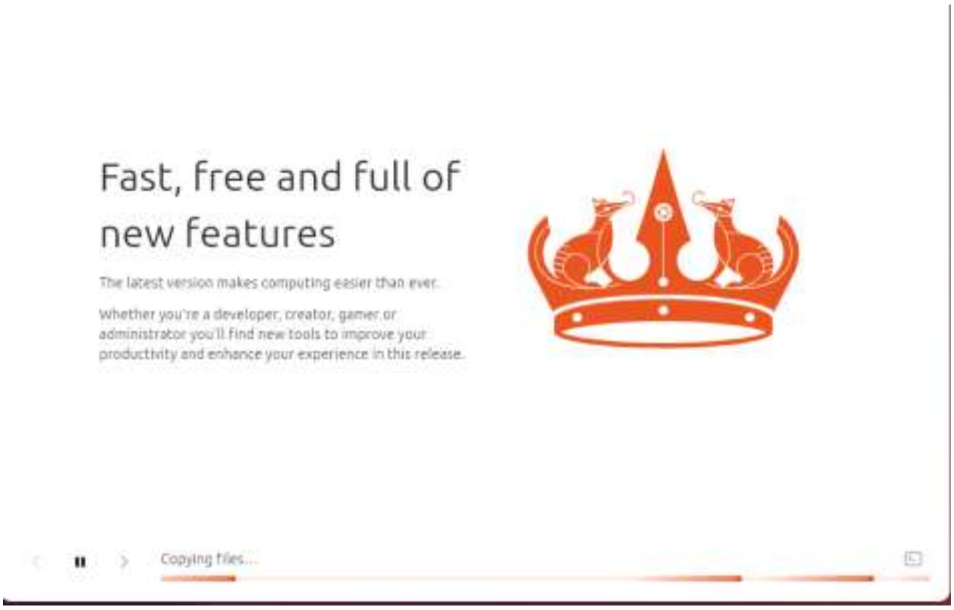
Security & more

Disk encryption	None
Proprietary software	None

Partitions

partition **sda1** created
partition **sda2** formatted as **ext4** used for **/**

Review your choice then click install



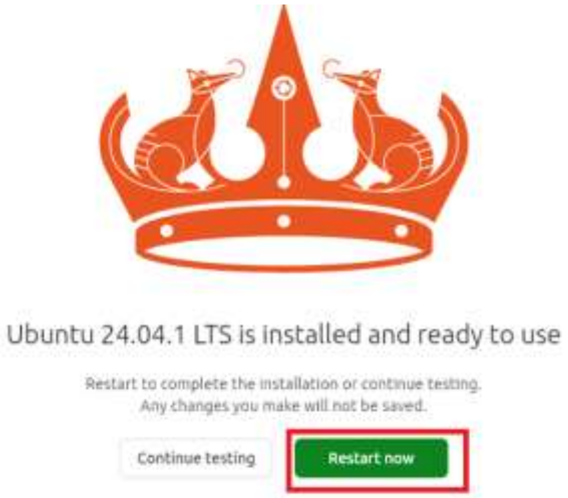
Fast, free and full of new features

The latest version makes computing easier than ever.

Whether you're a developer, creator, gamer or administrator you'll find new tools to improve your productivity and enhance your experience in this release.

Copying files...

Waiting until all files are copied to click next



Ubuntu 24.04.1 LTS is installed and ready to use

Restart to complete the installation or continue testing.
Any changes you make will not be saved.

While installation complete click restart now.

Points to Remember

- There is difference between CI and CD where by Continuous Integration focuses on the early stages of the development process, continuous delivery (CD) takes things a step further, Continuous Integration focuses on the integration and validation of code while continuous delivery (CD) focuses on delivering changes to production or staging environments.

- A container is a standardized unit of software that packages up code and its dependencies so that the application runs consistently across different environments.
- DevOps combines development (Dev) and operations (Ops) to increase the efficiency, speed, and security of software development and delivery compared to traditional processes.
- DevSecOps is an extension of DevOps that incorporates security practices throughout the entire software development lifecycle.
- A Linux distribution is an operating system made from a software collection that includes the Linux kernel and often a package management system. There are the Popular Linux Distributions such as Ubuntu, fedora, Debian, CentOS and more
- Most importance of using ubuntu Linux it making one of the most popular operating systems, especially for developers, IT professionals, and general users. Here are the key reasons why using Ubuntu Linux is important: Free and Open Source, Security and Privacy, Stability and Performance, Software Availability, Developer-Friendly,free and open source.

- **Creating a bootable USB drive is necessary for several tasks, including:** Operating System Installation**,** System Recovery**,** Testing or Running Live Systems**,** Updating Firmware**,** Disk Partitioning or Maintenance**.**

Application of learning 1.1.

An XXX company has a server computer installed with a window server 2012r2, the company needs to upgrade computer with Linux operating system. you are tasked to install Linux operating system in server computer.

Indicative content 1.2: Application of Linux basic commands

Duration: 5 hrs

Theoretical Activity 1.2.1: Description of basic Linux commands

Tasks:

1: Answer the following questions:

I. What are the following Linux basic command?

a) System Information

b) File and Directory Management

c) Text Processing

d) Process Management

e) Package Management

f) User and Group Management

g) System Control

2: Write your answers on flipchart/paper, blackboard or whiteboard.

3: Present your findings to the whole class and trainer

4: Ask questions for clarification if necessary

5: Read the key readings 1.2.1

Key readings 1.2.1.: Description of basic Linux commands

✓ **System Information**

Linux provides various ways to gather detailed information about the system's hardware, software, and performance. Understanding the system's current state is crucial for monitoring, troubleshooting, and system management.

Kernel and OS Information:

The Linux kernel is the core component of the operating system. The kernel version and OS details are essential to know for compatibility with software and hardware. Linux allows you to view the kernel version, distribution, and architecture (e.g., 32-bit or 64-bit).

Common examples of basic Linux commands in kernel information and their uses:

→ **uname –r** Shows the Linux kernel version.

→ **uname -m**: Shows the machine hardware architecture

→ **uname -a**: Displays all available system information (kernel name, version, hardware, etc.)

✚ CPU (Central Processing Unit):

CPU information includes details like the processor type, number of cores, clock speed, and architecture. This helps in understanding the system's computational power and performance potential.

Common examples of basic Linux commands in CPU management and their uses:

→ lscpu – Display CPU architecture information

- **Use**: Shows detailed information about the CPU, including its architecture, number of CPUs, cores, threads, speed, and more.

→ cat /proc/cpuinfo – View CPU details

- **Use**: Displays detailed information about the processor directly from the /proc file system, including model, vendor, and speed.

→ nproc – Show the number of processing units (cores)

- **Use**: Displays the number of available processing units (cores) on the system.

→ mpstat – CPU usage per processor

- **Use**: Displays CPU usage statistics for each processor or core. Requires the sysstat package to be installed

✚ Memory (RAM):

Memory information shows how much RAM is installed and how it is being used (e.g., free, cached, or in use). Knowing memory usage is crucial for assessing system performance, especially when running memory-intensive applications.

Common examples of basic Linux commands in Memory and their uses:

→ free –h Display memory usage

- **Use**: Shows the total amount of free and used memory (RAM and swap) in the system.

→ vmstat – Report virtual memory statistics

- **Use**: Displays information about processes, memory, paging, block I/O, traps, and CPU activity.

→ top Monitor system processes and memory

- **Use**: Shows a dynamic real-time view of running processes, CPU usage, and memory consumption.

✚ Disk and Storage:

Disk usage and storage capacity information show the total space available on storage devices, partitions, and the file system. It also displays how much space is currently in use. This is important for managing storage resources efficiently.

Common examples of basic Linux commands in Disk and storage and their uses:

→ df –h Disk Space Usage

- **Use**: Displays the amount of disk space used and available on file systems.

→ lsblk – List Block Devices

- **Use**: Lists information about all available block devices, such as disks and partitions.

✚ Network Information:

Network details include the system's IP addresses, MAC addresses, and network interface configurations. Knowing the network status helps in troubleshooting connectivity issues and managing network resources.

Common examples of basic Linux commands in Network information and their uses:

→ ifconfig

- **Use**: Displays network interface configuration, including IP addresses, MAC addresses, and network statistics.

→ **ping**

- **Use**: Tests network connectivity to a specified host by sending ICMP echo request packets and measuring the response time.

→ **traceroute**

- **Use**: Displays the route and measures transit delays of packets across a network.

- **Hardware Information:**

Information about the hardware components such as USB devices, PCI cards (e.g., graphic cards), and block devices (disks, drives) is also available. This helps in detecting hardware configurations and troubleshooting hardware-related problems.

Common examples of basic Linux commands in Hardware Information and their uses:

→ **lscpu**

- **Use**: Displays detailed information about the CPU architecture, including the number of CPUs, cores, threads, and clock speed.

→ **lsblk**

- **Use**: Lists all block devices (such as hard drives and partitions) in a tree-like format, showing their mount points and sizes.

→ **sudo dmidecode**

- **Use**: Displays detailed information about the system's hardware, including the BIOS version, manufacturer, memory details, and system board information. This command usually requires root privilege

System Uptime and Load:

Uptime refers to how long the system has been running without a restart, and load average gives an idea of how much work the system is currently handling. Monitoring uptime and load helps in identifying system stability and performance issues.

By collecting and analyzing this information, system administrators and users can optimize performance, troubleshoot hardware/software issues, and ensure the system runs efficiently.

Common examples of basic Linux commands in System Uptime and Load **and their uses:**

→ **uptime**

- **Use**: Displays how long the system has been running, along with the current time, number of users logged in, and the system load averages.

Output Explanation:

- Shows the time the system has been running (e.g., up 5 days, 3 hours).
- Displays the number of users currently logged in.
- Provides load averages for the last 1, 5, and 15 minutes (e.g., load average: 0.25, 0.15, 0.05).

→ **top**

- **Use**: Provides a real-time view of the system's processes, including CPU and memory usage. It also shows system uptime and load averages.

Output Explanation:

- At the top of the interface, you'll see system uptime, number of users, and load averages.
- Below this, the list of running processes is displayed, along with their resource usage.

→ **vmstat**

- **Use**: Reports information about processes, memory, paging, block I/O, traps, and CPU activity.

Output Explanation:

- The first line gives a summary of the overall system status.
- Subsequent lines (updated every 5 seconds in this case) show metrics including CPU load, which can indicate system performance over time.

✓ **File and Directory Management**

Theoretical Overview: File and directory management in an operating system involves creating, modifying, deleting, and navigating through files and directories (folders). These actions are critical for organizing data and working efficiently within the filesystem.

Basic Commands:

- ls – Lists files and directories in the current working directory.
- mkdir <directory_name> – Creates a new directory.
- rm <file_name> – Removes a file; use rm -r <directory_name> to remove a directory and its contents.

✓ **Text Processing**

Theoretical Overview: Text processing involves manipulating text files, searching within them, extracting data, and formatting the output. It's commonly used for system administration tasks, scripting, and data extraction.

Basic Commands:

- cat <file_name> – Displays the content of a file.
- grep <pattern> <file_name> – Searches for a specific pattern or string in a file.
- sed 's/old/new/g' <file_name> – Replaces occurrences of a string within a file.

✓ **Process Management**

Theoretical Overview: Process management includes starting, stopping, and monitoring processes in the operating system. Managing processes is crucial for ensuring system stability and performance.

Basic Commands:

- ps – Displays a list of currently running processes.
- kill <process_id> – Terminates a process by its process ID.

- top – Displays real-time system stats, including processes, memory, and CPU usage.

✓ **Package Management**

Theoretical Overview: Package management refers to installing, updating, and removing software packages in an operating system. Each package typically contains binaries, libraries, and configuration files required by software.

Basic Commands:

- apt-get install <package_name> – Installs a package on Debian-based systems.
- yum remove <package_name> – Removes a package on Red Hat-based systems.
- apt-get update – Updates the list of available packages and their versions.

✓ **User and Group Management**

Theoretical Overview: Managing users and groups ensures that only authorized individuals can access certain parts of the system. It also helps in controlling permissions and resource access across different users.

Basic Commands:

- useradd <username> – Creates a new user.
- passwd <username> – Sets or changes the password for a user.
- groupadd <group_name> – Creates a new group.

✓ **System Control**

Theoretical Overview: System control covers tasks like system rebooting, shutting down, managing services, and viewing system logs. It's essential for maintaining system stability and performance.

Basic Commands:

- systemctl start <service_name> – Starts a service.
- shutdown now – Shuts down the system immediately.
- reboot – Reboots the system.

Practical Activity 1.2.2: Applying Linux basic commands

Task:

1: Read the key Reading 1.2.2

2: Referring to the key reading 1.2.2, you are asked to go to the computer lab to apply Linux basic command.

3: Present your work to the trainer or whole class

4: Ask question for clarification where necessary

5: Perform the task provided in application of Leaning **1.2**

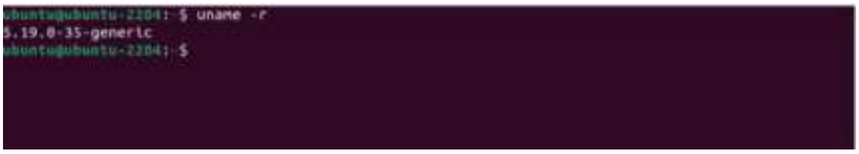
Key readings 1.2.2: application of Linux System information basic commands

✓ **System Information Basic Linux commands**

I. Shows the Linux kernel version.

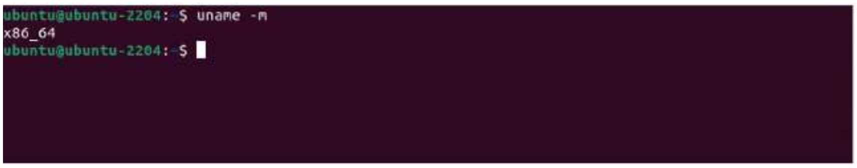
Step 1: Open terminal

Step 2: **uname –r**



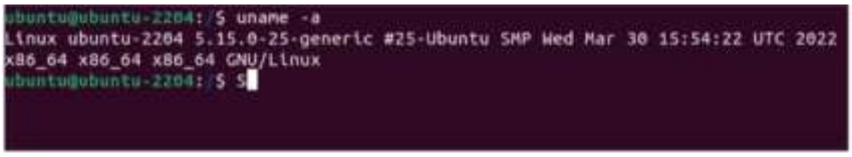
```
ubuntu@ubuntu-2204:~$ uname -r
5.19.0-35-generic
ubuntu@ubuntu-2204:~$
```

uname -m: Shows the machine hardware architecture



```
ubuntu@ubuntu-2204:~$ uname -m
x86_64
ubuntu@ubuntu-2204:~$
```

name -a: Displays all available system information (kernel name, version, hardware, etc.)

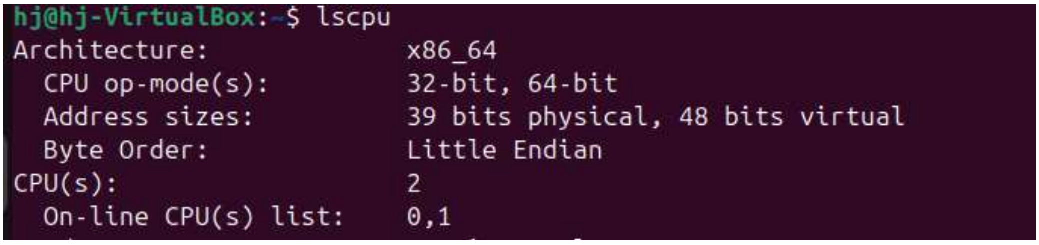


```
ubuntu@ubuntu-2204:~$ uname -a
Linux ubuntu-2204 5.15.0-25-generic #25-Ubuntu SMP Wed Mar 30 15:54:22 UTC 2022
x86_64 x86_64 x86_64 GNU/Linux
ubuntu@ubuntu-2204:~$ $
```

ii. CPU management

lscpu – Display CPU architecture information

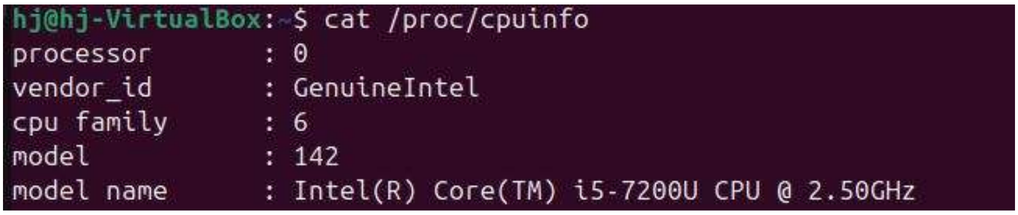
a. **Use**: Shows detailed information about the CPU, including its architecture, number of CPUs, cores, threads, speed, and more.



```
hj@hj-VirtualBox:~$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          39 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   2
  On-line CPU(s) list:    0,1
```

→ cat /proc/cpuinfo – View CPU details

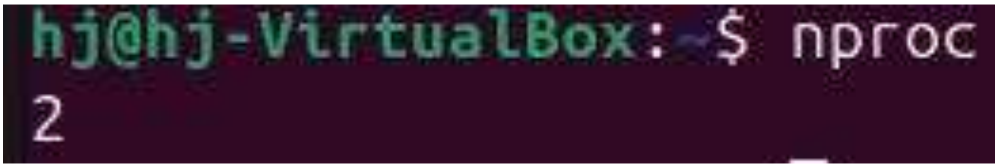
b. **Use**: Displays detailed information about the processor directly from the /proc file system, including model, vendor, and speed.



```
hj@hj-VirtualBox:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 142
model name      : Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
```

→ nproc – Show the number of processing units (cores)

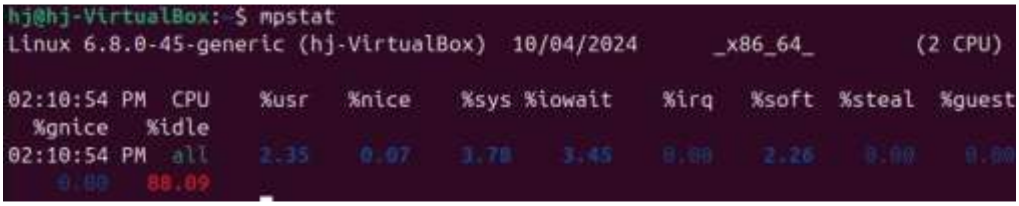
c. **Use**: Displays the number of available processing units (cores) on the system.



```
hj@hj-VirtualBox:~$ nproc
2
```

→ mpstat – CPU usage per processor

d. **Use**: Displays CPU usage statistics for each processor or core. Requires the sysstat package to be installed

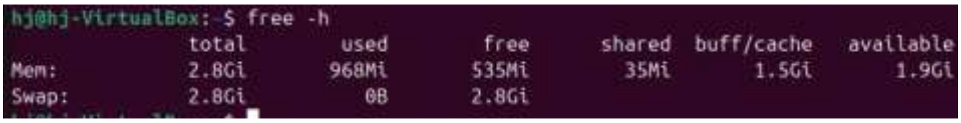


```
hj@hj-VirtualBox:~$ mpstat
Linux 6.8.0-45-generic (hj-VirtualBox)  10/04/2024      _x86_64_        (2 CPU)

02:10:54 PM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest
  %gnice   %idle
02:10:54 PM  all    2.35    0.07    3.78    3.45    0.00    2.26    0.00    0.00
    0.00   88.09
```

e. **Memory management**

free –h Display memory usage

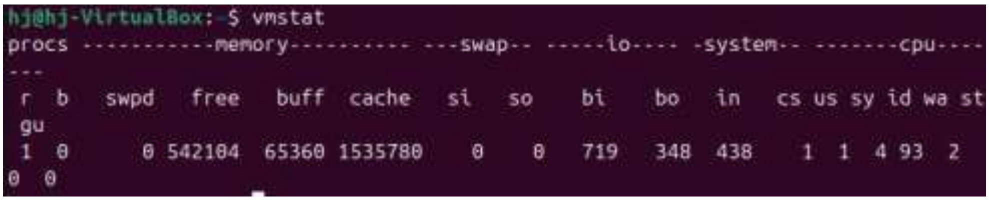
f. **Use**: Shows the total amount of free and used memory (RAM and swap) in the system.



```
hj@hj-VirtualBox:~$ free -h
               total        used        free      shared  buff/cache   available
Mem:           2.8Gi       968Mi       535Mi        35Mi       1.5Gi       1.9Gi
Swap:          2.8Gi          0B       2.8Gi
```

→ **vmstat – Report virtual memory statistics**

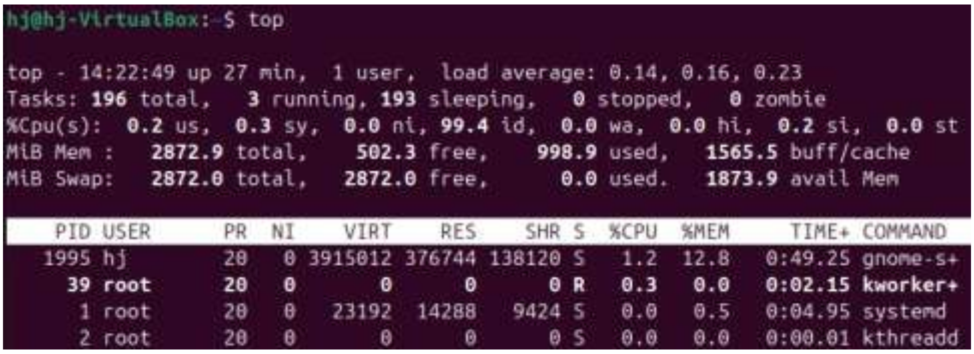
g. **Use**: Displays information about processes, memory, paging, block I/O, traps, and CPU activity.



```
hj@hj-VirtualBox:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- -------cpu----
---
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 gu
 1  0      0 542104  65360 1535780    0    0   719   348  438    1  1  4 93  2
0  0
```

→ **top Monitor system processes and memory**

h. **Use**: Shows a dynamic real-time view of running processes, CPU usage, and memory consumption.

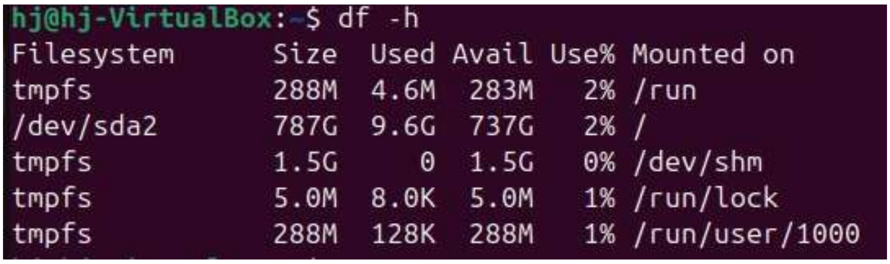


```
hj@hj-VirtualBox:~$ top

top - 14:22:49 up 27 min,  1 user,  load average: 0.14, 0.16, 0.23
Tasks: 196 total,   3 running, 193 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.3 sy,  0.0 ni, 99.4 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   2872.9 total,    502.3 free,    998.9 used,   1565.5 buff/cache
MiB Swap:   2872.0 total,   2872.0 free,      0.0 used.   1873.9 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1995 hj        20   0 3915012 376744 138120 S   1.2  12.8   0:49.25 gnome-s+
     39 root      20   0       0      0      0 R   0.3   0.0   0:02.15 kworker+
      1 root      20   0   23192  14288   9424 S   0.0   0.5   0:04.95 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01 kthreadd
```

i. **Disk and storage:**

→ **df –h Disk Space Usage**

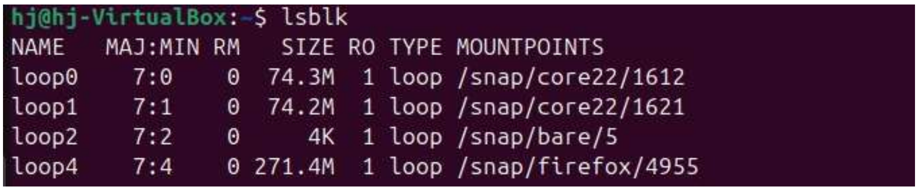
j. **Use**: Displays the amount of disk space used and available on file systems.



```
hj@hj-VirtualBox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           288M  4.6M  283M   2% /run
/dev/sda2       787G  9.6G  737G   2% /
tmpfs           1.5G     0  1.5G   0% /dev/shm
tmpfs           5.0M  8.0K  5.0M   1% /run/lock
tmpfs           288M  128K  288M   1% /run/user/1000
```

→ lsblk – List Block Devices

k. **Use**: Lists information about all available block devices, such as disks and partitions.

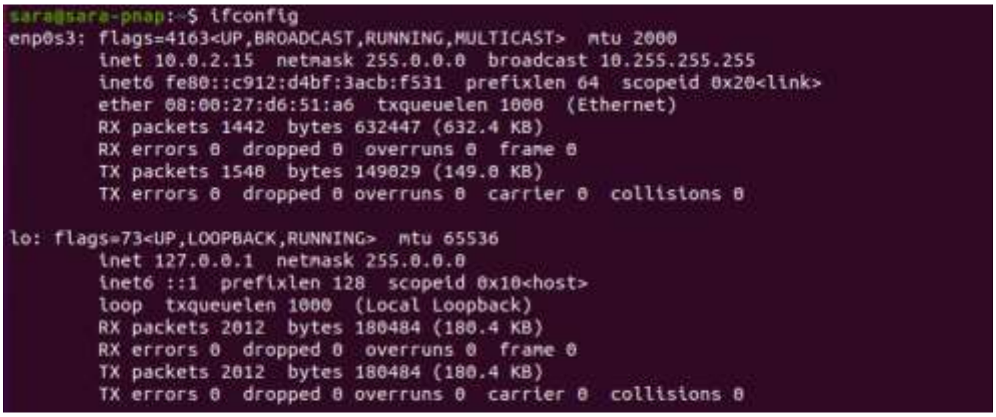


```
hj@hj-VirtualBox:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0    7:0    0  74.3M  1 loop /snap/core22/1612
loop1    7:1    0  74.2M  1 loop /snap/core22/1621
loop2    7:2    0     4K  1 loop /snap/bare/5
loop4    7:4    0 271.4M  1 loop /snap/firefox/4955
```

l. Network information

→ ifconfig

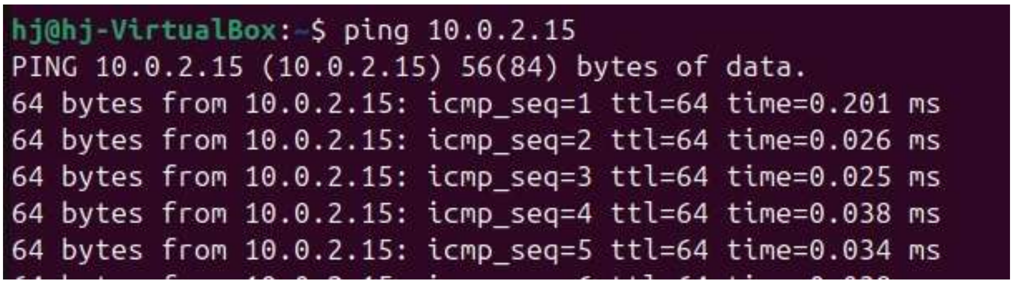
m. **Use**: Displays network interface configuration, including IP addresses, MAC addresses, and network statistics.



```
sara@sara-pnap:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 2000
        inet 10.0.2.15  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::c912:d4bf:3acb:f531  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:d6:51:a6  txqueuelen 1000  (Ethernet)
        RX packets 1442  bytes 632447 (632.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1540  bytes 149029 (149.0 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 2012  bytes 180484 (180.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2012  bytes 180484 (180.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

→ ping

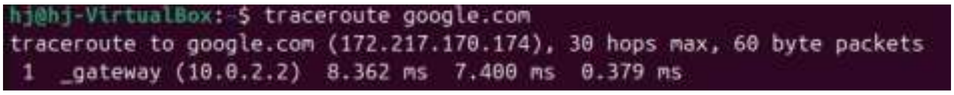
n. **Use**: Tests network connectivity to a specified host by sending ICMP echo request packets and measuring the response time.



```
hj@hj-VirtualBox:~$ ping 10.0.2.15
PING 10.0.2.15 (10.0.2.15) 56(84) bytes of data.
64 bytes from 10.0.2.15: icmp_seq=1 ttl=64 time=0.201 ms
64 bytes from 10.0.2.15: icmp_seq=2 ttl=64 time=0.026 ms
64 bytes from 10.0.2.15: icmp_seq=3 ttl=64 time=0.025 ms
64 bytes from 10.0.2.15: icmp_seq=4 ttl=64 time=0.038 ms
64 bytes from 10.0.2.15: icmp_seq=5 ttl=64 time=0.034 ms
```

→ **traceroute**

o. **Use**: Displays the route and measures transit delays of packets across a network.

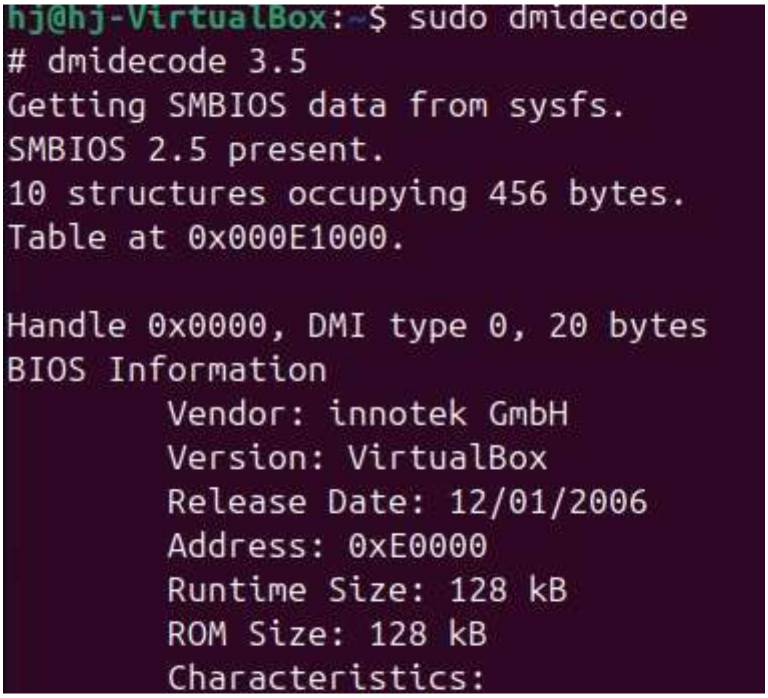


```
hj@hj-VirtualBox:~$ traceroute google.com
traceroute to google.com (172.217.170.174), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  8.362 ms  7.400 ms  0.379 ms
```

p. **Hardware Information and their uses:**

→ **sudo dmidecode**

q. **Use**: Displays detailed information about the system's hardware, including the BIOS version, manufacturer, memory details, and system board information. This command usually requires root privilege

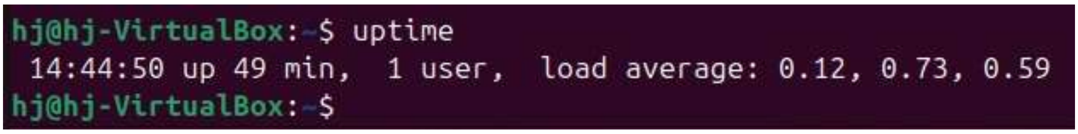


```
hj@hj-VirtualBox:~$ sudo dmidecode
# dmidecode 3.5
Getting SMBIOS data from sysfs.
SMBIOS 2.5 present.
10 structures occupying 456 bytes.
Table at 0x000E1000.

Handle 0x0000, DMI type 0, 20 bytes
BIOS Information
        Vendor: innotek GmbH
        Version: VirtualBox
        Release Date: 12/01/2006
        Address: 0xE0000
        Runtime Size: 128 kB
        ROM Size: 128 kB
        Characteristics:
```

r. **System Uptime and** Load **and their uses:**

→ **uptime**

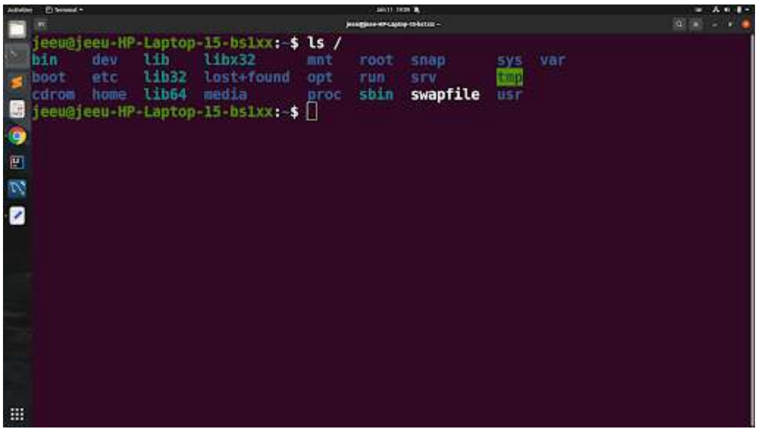
s. **Use**: Displays how long the system has been running, along with the current time, number of users logged in, and the system load averages.



```
hj@hj-VirtualBox:~$ uptime
 14:44:50 up 49 min,  1 user,  load average: 0.12, 0.73, 0.59
hj@hj-VirtualBox:~$
```

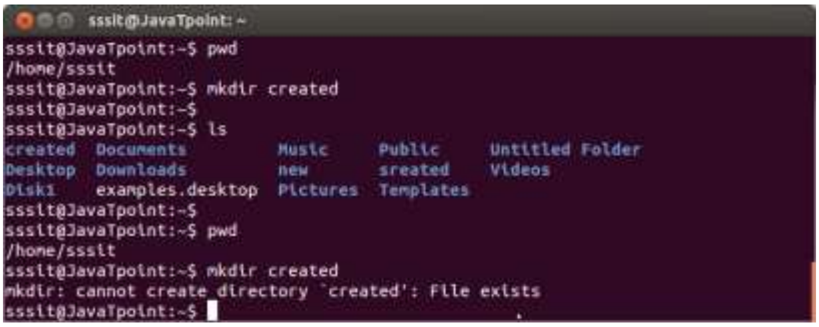
✓ **File and Directory Management**

- ls – Lists files and directories in the current working directory.



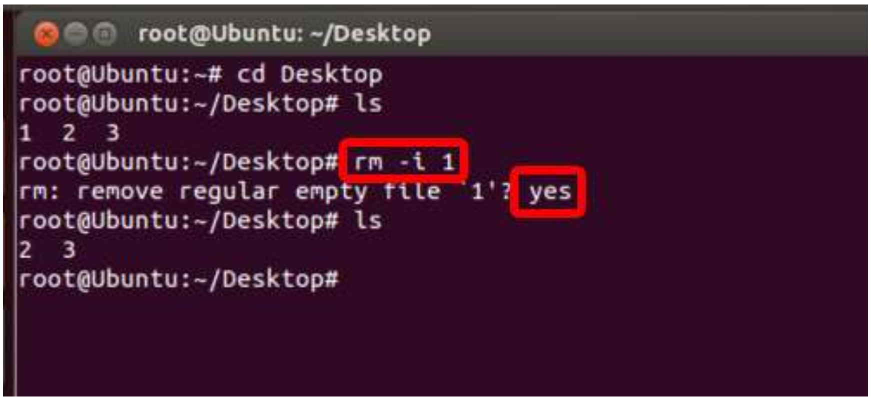
```
jeeu@jeeu-HP-Laptop-15-bs1xx:~$ ls /
bin    dev   lib    libx32      mnt   root  snap      sys  var
boot   etc   lib32  lost+found  opt   run   srv       tmp
cdrom  home  lib64  media       proc  sbin  swapfile  usr
jeeu@jeeu-HP-Laptop-15-bs1xx:~$ 
```

- mkdir <directory_name> – Creates a new directory.



```
sssit@JavaTpoint:~$ pwd
/home/sssit
sssit@JavaTpoint:~$ mkdir created
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ ls
created  Documents          Music     Public     Untitled Folder
Desktop  Downloads          new       sreated    Videos
Disk1    examples.desktop   Pictures  Templates
sssit@JavaTpoint:~$
sssit@JavaTpoint:~$ pwd
/home/sssit
sssit@JavaTpoint:~$ mkdir created
mkdir: cannot create directory `created': File exists
sssit@JavaTpoint:~$ 
```

- rm <file_name> – Removes a file; use rm -r <directory_name> to remove a directory and its contents.

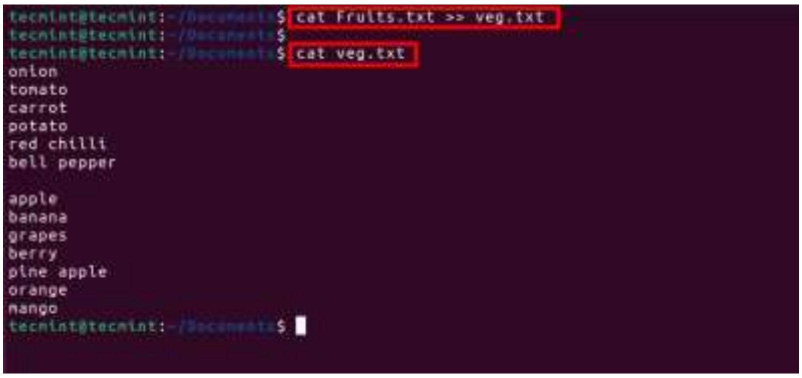


root@Ubuntu: ~/Desktop

```
root@Ubuntu:~# cd Desktop
root@Ubuntu:~/Desktop# ls
1  2  3
root@Ubuntu:~/Desktop# rm -i 1
rm: remove regular empty file '1'? yes
root@Ubuntu:~/Desktop# ls
2  3
root@Ubuntu:~/Desktop#
```

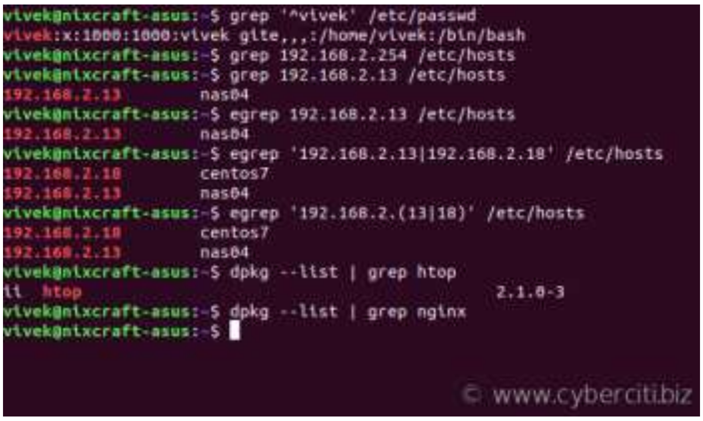
Text processing

cat <file_name> – Displays the content of a file.



```
tecmint@tecmint:~/Documents$ cat Fruits.txt >> veg.txt
tecmint@tecmint:~/Documents$
tecmint@tecmint:~/Documents$ cat veg.txt
onion
tomato
carrot
potato
red chilli
bell pepper

apple
banana
grapes
berry
pine apple
orange
mango
tecmint@tecmint:~/Documents$ █
```

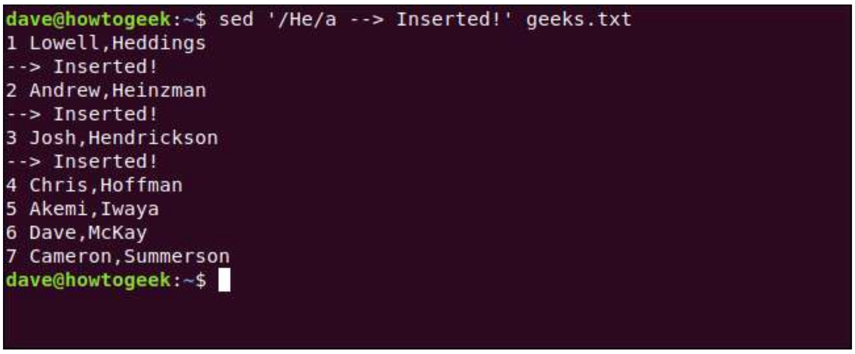
grep <pattern> <file_name> – Searches for a specific pattern or string in a file.



```
vivek@nixcraft-asus:~$ grep '^vivek' /etc/passwd
vivek:x:1000:1000:vivek gite,,,:/home/vivek:/bin/bash
vivek@nixcraft-asus:~$ grep 192.168.2.254 /etc/hosts
vivek@nixcraft-asus:~$ grep 192.168.2.13 /etc/hosts
192.168.2.13        nas04
vivek@nixcraft-asus:~$ egrep 192.168.2.13 /etc/hosts
192.168.2.13        nas04
vivek@nixcraft-asus:~$ egrep '192.168.2.13|192.168.2.18' /etc/hosts
192.168.2.18        centos7
192.168.2.13        nas04
vivek@nixcraft-asus:~$ egrep '192.168.2.(13|18)' /etc/hosts
192.168.2.18        centos7
192.168.2.13        nas04
vivek@nixcraft-asus:~$ dpkg --list | grep htop
ii  htop                                 2.1.0-3
vivek@nixcraft-asus:~$ dpkg --list | grep nginx
vivek@nixcraft-asus:~$ █
```

© www.cyberciti.biz

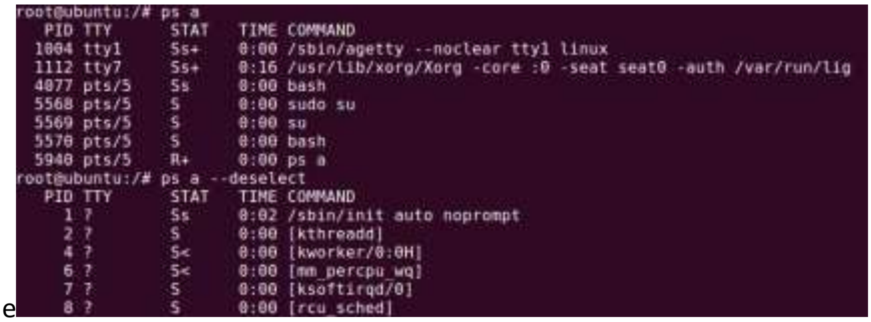
sed 's/old/new/g' <file_name> – Replaces occurrences of a string within a file.



```
dave@howtogeek:~$ sed '/He/a --> Inserted!' geeks.txt
1 Lowell,Heddings
--> Inserted!
2 Andrew,Heinzman
--> Inserted!
3 Josh,Hendrickson
--> Inserted!
4 Chris,Hoffman
5 Akemi,Iwaya
6 Dave,McKay
7 Cameron,Summerson
dave@howtogeek:~$
```

- ✓ **Process management**

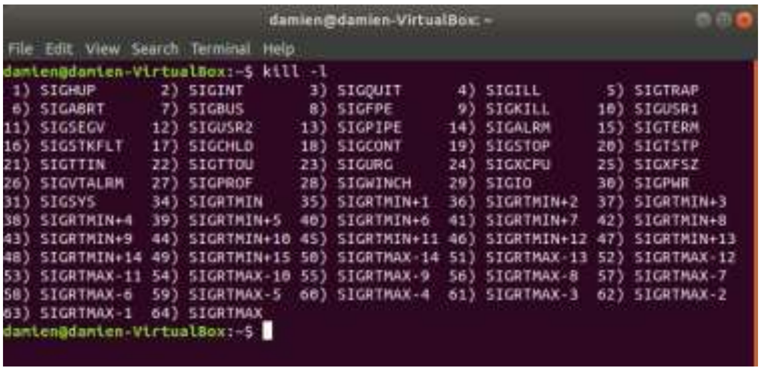
ps – Displays a list of currently running processes.



```
root@ubuntu:/# ps a
  PID TTY      STAT   TIME COMMAND
 1004 tty1     Ss+    0:00 /sbin/agetty --noclear tty1 linux
 1112 tty7     Ss+    0:16 /usr/lib/xorg/Xorg -core :0 -seat seat0 -auth /var/run/lig
 4077 pts/5    Ss     0:00 bash
 5568 pts/5    S      0:00 sudo su
 5569 pts/5    S      0:00 su
 5570 pts/5    S      0:00 bash
 5940 pts/5    R+     0:00 ps a
root@ubuntu:/# ps a --deselect
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:02 /sbin/init auto noprompt
    2 ?        S      0:00 [kthreadd]
    4 ?        S<     0:00 [kworker/0:0H]
    6 ?        S<     0:00 [mm_percpu_wq]
    7 ?        S      0:00 [ksoftirqd/0]
    8 ?        S      0:00 [rcu_sched]
```

e

kill <process_id> – Terminates a process by its process ID.

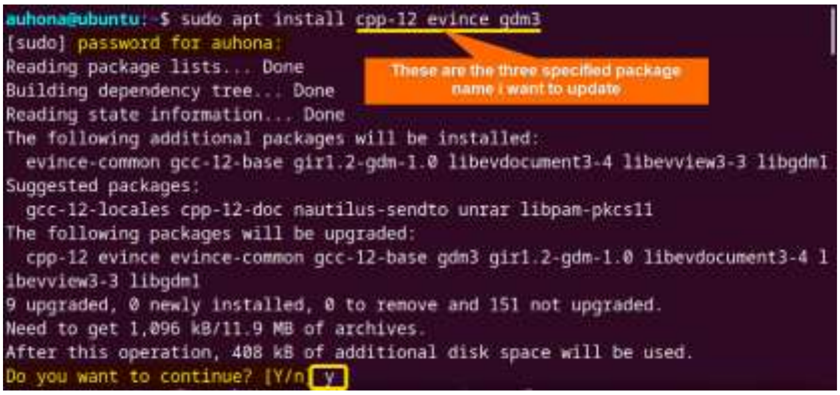

damien@damien-VirtualBox: ~
File Edit View Search Terminal Help

```
damien@damien-VirtualBox:~$ kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL       5) SIGTRAP
 6) SIGABRT      7) SIGBUS       8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
damien@damien-VirtualBox:~$
```

top – Displays real-time system stats, including processes, memory, and CPU usage.

- ✓ **Package management**

apt-get install <package_name> – Installs a package on Debian-based systems.

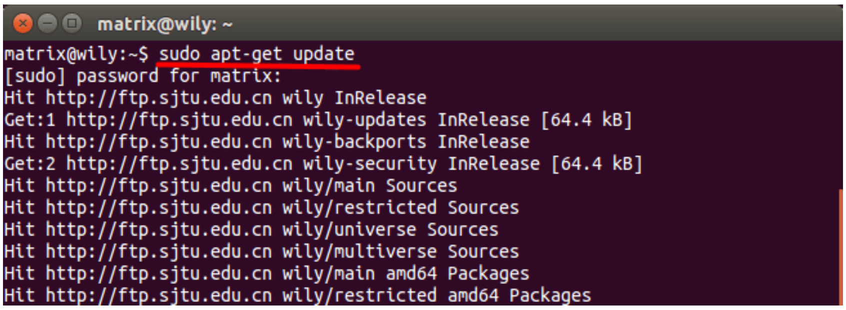


```
auhona@ubuntu:~$ sudo apt install cpp-12 evince gdm3
[sudo] password for auhona:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  evince-common gcc-12-base gir1.2-gdm-1.0 libevdocument3-4 libevview3-3 libgdm1
Suggested packages:
  gcc-12-locales cpp-12-doc nautilus-sendto unrar libpam-pkcs11
The following packages will be upgraded:
  cpp-12 evince evince-common gcc-12-base gdm3 gir1.2-gdm-1.0 libevdocument3-4 l
ibevview3-3 libgdm1
9 upgraded, 0 newly installed, 0 to remove and 151 not upgraded.
Need to get 1,096 kB/11.9 MB of archives.
After this operation, 408 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

These are the three specified package name i want to update

yum remove <package_name> – Removes a package on Red Hat-based systems.

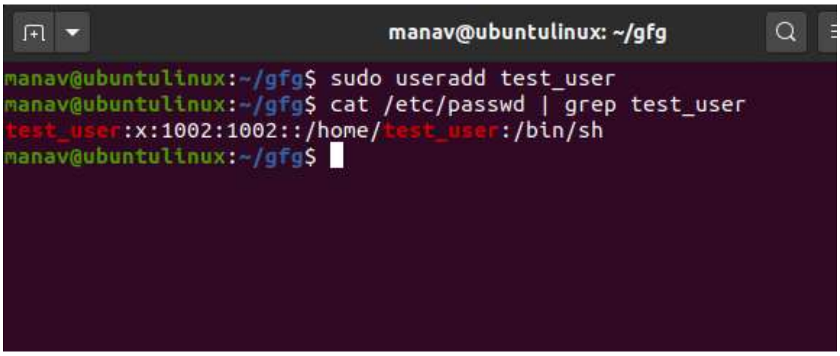
apt-get update – Updates the list of available packages and their versions.



```
matrix@wily:~$ sudo apt-get update
[sudo] password for matrix:
Hit http://ftp.sjtu.edu.cn wily InRelease
Get:1 http://ftp.sjtu.edu.cn wily-updates InRelease [64.4 kB]
Hit http://ftp.sjtu.edu.cn wily-backports InRelease
Get:2 http://ftp.sjtu.edu.cn wily-security InRelease [64.4 kB]
Hit http://ftp.sjtu.edu.cn wily/main Sources
Hit http://ftp.sjtu.edu.cn wily/restricted Sources
Hit http://ftp.sjtu.edu.cn wily/universe Sources
Hit http://ftp.sjtu.edu.cn wily/multiverse Sources
Hit http://ftp.sjtu.edu.cn wily/main amd64 Packages
Hit http://ftp.sjtu.edu.cn wily/restricted amd64 Packages
```

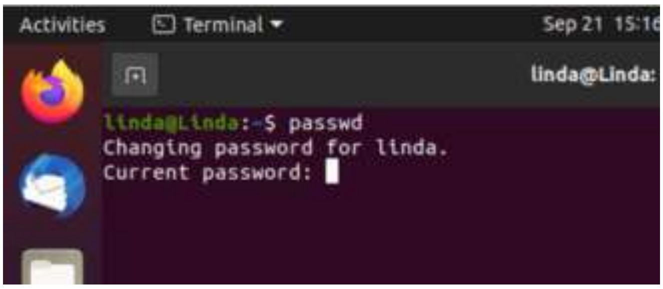
- ✓ **User and group management**

useradd <username> – Creates a new user.



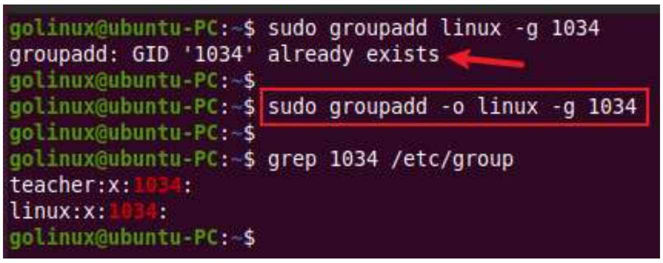
```
manav@ubuntulinux:~/gfg$ sudo useradd test_user
manav@ubuntulinux:~/gfg$ cat /etc/passwd | grep test_user
test_user:x:1002:1002::/home/test_user:/bin/sh
manav@ubuntulinux:~/gfg$
```

passwd <username> – Sets or changes the password for a user.



```
linda@Linda:~$ passwd
Changing password for linda.
Current password:
```

groupadd <group_name> – Creates a new group.



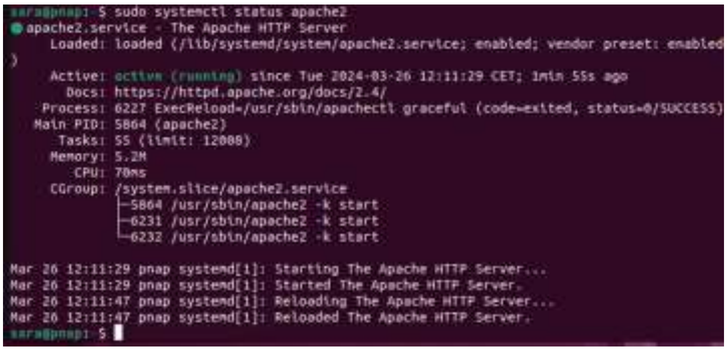
```
golinux@ubuntu-PC:~$ sudo groupadd linux -g 1034
groupadd: GID '1034' already exists
golinux@ubuntu-PC:~$
golinux@ubuntu-PC:~$ sudo groupadd -o linux -g 1034
golinux@ubuntu-PC:~$
golinux@ubuntu-PC:~$ grep 1034 /etc/group
teacher:x:1034:
linux:x:1034:
golinux@ubuntu-PC:~$
```

users: Displays a simple list of logged-in usernames.

```
users
```

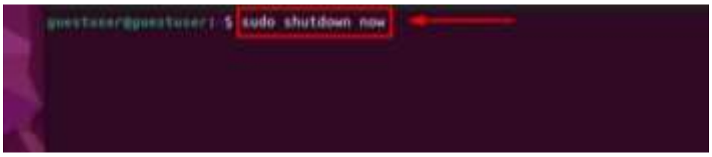
- ✓ **System control**

systemctl start <service_name> – Starts a service.



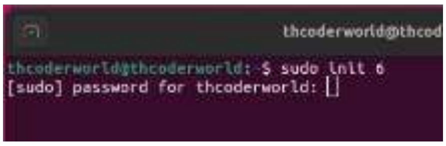
```
sara@pnap:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled
)
     Active: active (running) since Tue 2024-03-26 12:11:29 CET; 1min 55s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 6227 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
   Main PID: 5864 (apache2)
      Tasks: 55 (limit: 12008)
     Memory: 5.2M
        CPU: 70ms
     CGroup: /system.slice/apache2.service
             ├─5864 /usr/sbin/apache2 -k start
             ├─6231 /usr/sbin/apache2 -k start
             └─6232 /usr/sbin/apache2 -k start

Mar 26 12:11:29 pnap systemd[1]: Starting The Apache HTTP Server...
Mar 26 12:11:29 pnap systemd[1]: Started The Apache HTTP Server.
Mar 26 12:11:47 pnap systemd[1]: Reloading The Apache HTTP Server...
Mar 26 12:11:47 pnap systemd[1]: Reloaded The Apache HTTP Server.
sara@pnap:~$ 
```

shutdown now – Shuts down the system immediately.



```
guestuser@guestuser:~$ sudo shutdown now
```

reboot – Reboots the system.



```
thcoderworld@thcoderworld:~$ sudo init 6
[sudo] password for thcoderworld: 
```

Points to Remember

- **Commands** are instructions that you type into a terminal (command-line interface) to perform specific tasks. These commands allow you to interact with the operating system, manage files, processes, users, and system resources, and automate various tasks.
- **Terminal** refers to an interface that allows users to interact with the computer's operating system by typing commands. Originally, a terminal was a physical device used to input and display data (such as early teletype machines or video display terminals). However, in modern systems, a terminal is typically a **software application** that emulates this function.

- Using basic commands in Linux is essential for several reasons, especially in a command-line-driven environment. Here's why they are important Because Linux basic commands are used for Efficiency and Control, Resource Management, Automation, Remote Access, Power and Flexibility, Troubleshooting, Learning Foundation

- **system control commands** provide essential tools for managing services, processes, system status, and user accounts in Linux. They are key to effective system administration.

Application of learning 1.2.

ABC computer network is a newly opened company that needs to perform the following tasks:

a) Gather system information
b) Perform file and directory management
c) Use text processing tools
d) Handle package management
e) Perform user and group management
f) Use system control commands

Now, as a system administrator, you are hired to help the company in the aforementioned tasks.

Indicative content 1.3: Management of server services

Duration: 7 hrs

Theoretical Activity 1.3.1: Description of Server Services

Tasks:

1: Discuss on the following questions:

a. What are the server Services?

b. Differentiate between monitoring and Logging

2: write your answers on flipchart/paper

3: Present your findings to the whole class and trainer

4: Ask questions if necessary

5: Read the key readings 1.3.1

Key readings 1.3.1: Description of key terms

- ✓ **Web**: Refers to the World Wide Web (WWW), a system of interlinked hypertext documents and resources, primarily accessed via web browsers using HTTP/HTTPS protocols. Websites and web applications are part of this.
- ✓ A **web server** is a software or hardware system that serves web pages to clients (typically web browsers) over the internet. It handles requests via HTTP/HTTPS protocols, processes them, and sends back the appropriate response, typically HTML pages, images, or other content. Popular web server software includes Apache, Nginx, and LiteSpeed
- ✓ **Mail**: Refers to electronic mail (email), a method of exchanging digital messages across the internet. Mail servers manage the sending, receiving, and storage of emails, typically using protocols like SMTP, IMAP, or POP3.
- ✓ A **mail server** is a software application that manages the sending, receiving, and storage of emails. It communicates with other mail servers and clients via standard email protocols like **SMTP** (Simple Mail Transfer Protocol), **IMAP** (Internet Message Access Protocol), and **POP3** (Post Office Protocol). It consists of two main parts:

- **Mail Transfer Agent (MTA)**: Responsible for routing and delivering emails (e.g., Postfix, Exim).

- **Mail Delivery Agent (MDA)**: Stores the email in the recipient’s mailbox (e.g., Dovecot).
- **Mail User Agent (MUA)**: The client application users interact with to send and receive emails (e.g., Thunderbird).

✓ **File**: Refers to a block of digital data stored on a system, identified by a name and extension. Files can contain text, images, software, or any other digital content, and can be shared over a network or locally.

A **file** in computing is a collection of data or information stored on a disk and identified by a name. Files can contain anything from text, images, and software programs to system configurations. Each file has attributes like its name, size, type, and permissions that define how it can be accessed and modified.

✓ **SSH (Secure Shell)**: A cryptographic network protocol for securely accessing and managing remote computers over an unsecured network. It encrypts the session to prevent eavesdropping and connection hijacking.

SSH (Secure Shell) in Linux Ubuntu

SSH (Secure Shell) is a cryptographic network protocol used for securely accessing and managing a remote system over an unsecured network. It provides an encrypted connection to the remote system, ensuring that sensitive data such as passwords, commands, and files are protected from eavesdropping.

SSH is commonly used by system administrators for remote management tasks, file transfers, and secure tunneling. On Ubuntu, SSH is typically implemented using **OpenSSH**, which provides both the client and server utilities for establishing and managing SSH

✓ **DNS (Domain Name System)** is a system used to translate human-readable domain names (like www.example.com) into IP addresses (like 192.168.1.1), which computers use to communicate. It acts as the "phonebook" of the internet, allowing users to access websites and services using easy-to-remember names rather than numerical IP addresses.

In Ubuntu, DNS plays a vital role in networking, ensuring that your machine can resolve domain names to IP addresses when accessing the internet or network resources.

- **How DNS Works**

When you type a domain name in a web browser, the system queries a DNS server to get the corresponding IP address. This process involves several steps:

- **Local DNS Cache**: The system first checks its local DNS cache to see if it already knows the IP address.

- **DNS Resolver**: If not found locally, the request is sent to a DNS resolver, usually provided by your ISP or a public DNS service like Google DNS or Cloudflare DNS.
- **Root DNS Server**: If the resolver doesn't know the answer, it queries the root DNS server, which points to the authoritative DNS server for the domain.
- **Authoritative DNS Server**: This server provides the IP address for the domain.

Once the IP address is found, it's returned to the user's machine, and the connection to the website or service is established.

✓ A **proxy server** acts as an intermediary between a client (such as a web browser) and the internet. When a client makes a request to access a resource (e.g., a website), the proxy server processes the request and retrieves the resource on behalf of the client, which can provide several benefits, such as:

- **Anonymity**: The client's IP address is hidden, as the proxy's IP is used.
- **Access Control**: Proxies can filter web traffic and restrict access to certain sites.
- **Content Caching**: Frequently accessed content can be cached to improve load times.
- **Logging and Monitoring**: Proxies can log traffic for security and monitoring purposes.
- **Traffic Optimization**: Proxies can compress data and optimize traffic.

✓ **Monitoring** and **Logging** are essential components of system management, both offering different methods of tracking system performance, issues, and activity. They serve distinct purposes in maintaining the health and security of Linux systems.

- **Logging**

Logging refers to the process of recording detailed events, errors, and messages generated by the operating system, applications, and services. These logs provide granular information about what has happened on the system, helping administrators track down issues, audit events, or understand system behavior.

Key Characteristics of Logging:

- **Event-Based**: Logs capture events as they happen (e.g., user logins, service errors, security violations).
- **Stored as Files**: Logs are usually stored in text files on the system (e.g., /var/log/ directory in Linux).
- **Historical Records**: Logs contain historical data, useful for post-incident analysis or auditing.
- **Granular Information**: Logs capture fine-grained details like error codes, system messages, and service-specific information.

Examples of Logs in Linux:

- **System Logs** (/var/log/syslog or /var/log/messages): General system events like startup messages, hardware issues, etc.
- **Authentication Logs** (/var/log/auth.log or /var/log/secure): Records of authentication attempts, such as user logins.
- **Application Logs** (/var/log/apache2/access.log): Application-specific logs, e.g., web server access logs.
- **Kernel Logs** (/var/log/kern.log): Logs generated by the Linux kernel, useful for debugging hardware issues.

Log Management Tools:

- **rsyslog** or **syslog**: Standard Linux logging systems that handle log messages from different services.

- ✓ **Logrotate**: Automatically manages log file rotation, compression, and deletion.
- ✓ **Monitoring** refers to the process of continuously observing system performance and behavior in real time. It typically involves collecting metrics such as CPU usage, memory consumption, disk I/O, network traffic, and service availability, and alerting administrators when performance thresholds are exceeded or failures occur.

Key Characteristics of Monitoring:

- **Real-Time**: Monitoring focuses on real-time metrics, continuously checking the system for specific performance indicators.
- **Performance-Oriented**: It tracks system health and performance, such as resource usage, response times, and uptime.
- **Alerts and Notifications**: When thresholds are crossed (e.g., CPU usage above 90%), monitoring tools can trigger alerts to notify administrators.

- **Proactive**: Monitoring is often used to detect issues before they cause major failures or to optimize system performance.

✓ **Backup** in Linux refers to the process of copying data from your system to a secure location to ensure data safety in case of accidental deletion, hardware failure, system corruption, or other data loss incidents. It is a critical component of system administration to ensure the continuity and recovery of important data.

There are several strategies, tools, and methods for performing backups in Linux, depending on the requirements (e.g., full backup, incremental backup, file-based, or system-based).

Types of Backups in Linux

- **Full Backup**:
  - **Description**: A complete copy of all the data and files from the system.
  - **Pros**: Provides a complete snapshot of the system, making restoration easier.
  - **Cons**: Time-consuming and requires large storage space.
  - **Incremental Backup**:
  - **Description**: Only backs up the changes made since the last backup (either full or incremental).
  - **Pros**: Saves time and storage space by only copying modified data.
  - **Cons**: Restoration may require multiple backup files (last full backup + all incremental backups).
  - **Differential Backup**:
  - **Description**: Backs up data changed since the last full backup (not incremental).
  - **Pros**: Faster than full backup, and simpler restoration compared to incremental backup.
  - **Cons**: Consumes more storage compared to incremental backups.
  - **Snapshot Backup**:
  - **Description**: A point-in-time copy of the file system or storage device. Often used with LVM or filesystem features like ZFS and Btrfs.
  - **Pros**: Fast and efficient for large-scale file systems.
  - **Cons**: Requires snapshot support on the file system

Backup Methods in Linux

- **Local Backups**

This involves storing backups on the same system or a locally attached device, such as an external hard drive, USB, or another disk partition.

- **Pros**: Easy to set up and restore from.
- **Cons**: If the system hardware fails, both the original data and backups could be lost.

○ **Remote Backups**

Remote backups store data on a different machine or remote storage, ensuring that backups are safe from local hardware failures.

- **Pros**: Provides additional safety by storing backups off-site.
- **Cons**: Requires network bandwidth and may be slower.

○ **Cloud Backups**

Data is stored on a cloud storage service (like Google Drive, Amazon S3, or Dropbox) for redundancy and ease of access.

- **Pros**: Highly secure and accessible from anywhere.
- **Cons**: Requires a reliable internet connection and may incur costs depending on storage needs.

Practical Activity 1.3.2:Configuring Server services

Tasks:

1: Go to the computer lab and referring to the previous theoretical activity 1.3.1 then perform the following task individually:

Configure server services

2: Present the steps to configure server service

3: Referring to the steps presented on task 2 Install configure DNS service

4: Present your work to the trainer and whole class

5: Read the key readings 1.3.2 and ask questions for clarification where necessary

6: Perform the task provided in application of leaning 1.3.

Key readings 1.3.2.: configuration of server services

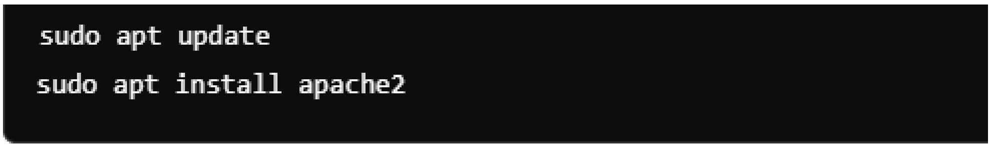
✓ **web services**

To configure a web server, we'll go through the steps for setting up **Apache** or **Nginx**—two popular web servers—on an Ubuntu system. I'll outline both options below.

Steps to install an apache Web Server Configuration

- **Install Apache**

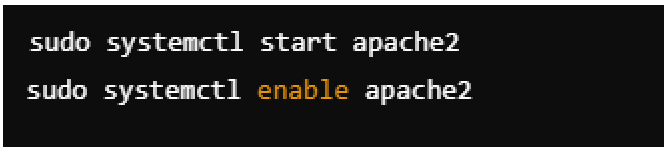
Start by installing Apache using the package manager:



```
sudo apt update
sudo apt install apache2
```

- **Start and Enable Apache**

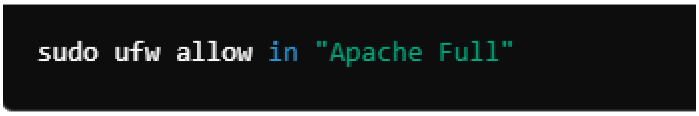
Ensure that the Apache service is running and set to start on boot:



```
sudo systemctl start apache2
sudo systemctl enable apache2
```

- **Firewall Configuration**

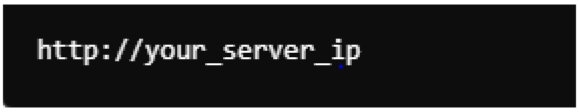
Allow HTTP and HTTPS traffic through the firewall



```
sudo ufw allow in "Apache Full"
```

- **Test Apache Installation**

Check if Apache is working by visiting your server’s IP address:



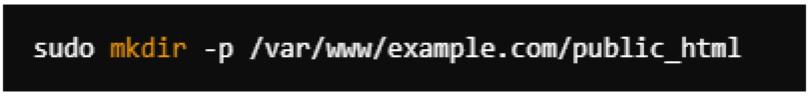
```
http://your_server_ip
```

You should see the default Apache page, confirming that the web server is up and running.

Configure Apache Virtual Hosts

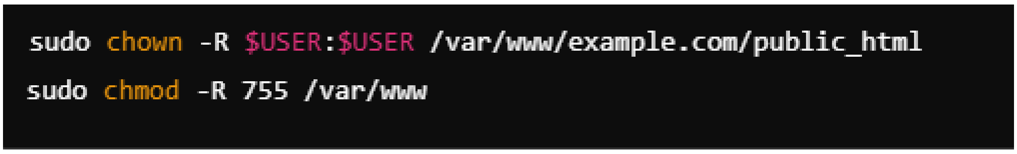
You need to configure virtual hosts to serve multiple websites on a single server.

- **Create a directory for your website:**



```
sudo mkdir -p /var/www/example.com/public_html
```

- Set the correct permissions

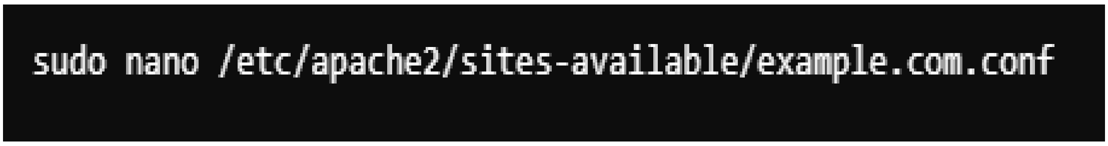


```
sudo chown -R $USER:$USER /var/www/example.com/public_html
sudo chmod -R 755 /var/www
```

- Create a sample index.html file:

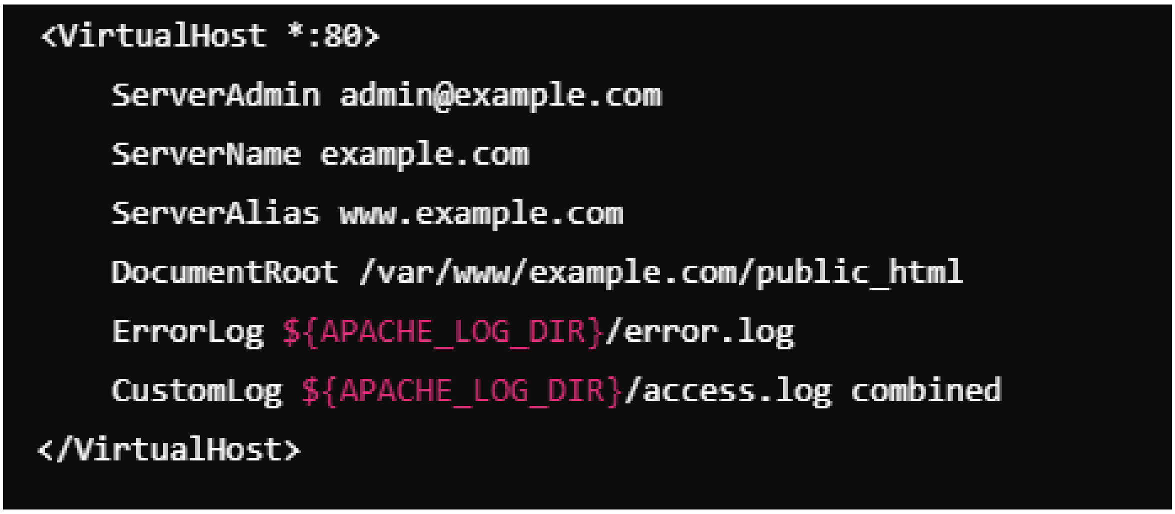
echo "<html><head><title>Welcome to Example.com</title></head><body><h1>Success! Example.com is working!</h1></body></html>" | sudo tee /var/www/example.com/public_html/index.html

- Create a virtual host configuration file



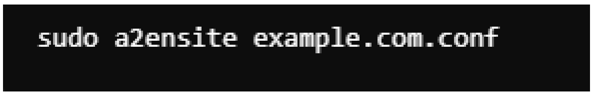
```
sudo nano /etc/apache2/sites-available/example.com.conf
```

Add the following configuration:



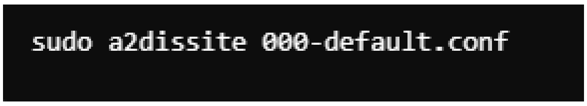
```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com/public_html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

- Enable the new virtual host



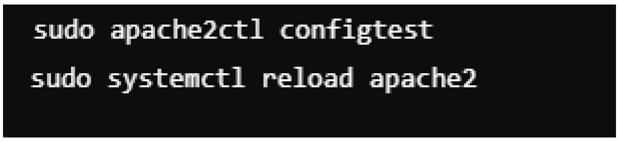
```
sudo a2ensite example.com.conf
```

- Disable the default virtual host (optional)



```
sudo a2dissite 000-default.conf
```

- Test the configuration and reload Apache

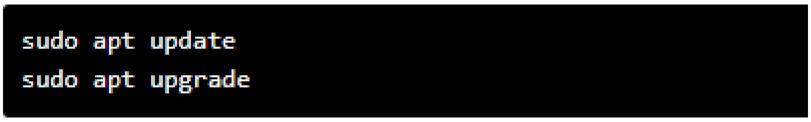


```
sudo apache2ctl configtest
sudo systemctl reload apache2
```

✓ **Configuration of mail services**

Step 1: Update System Packages

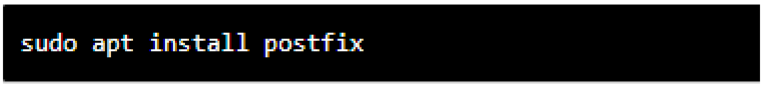
Before installing any new software, it’s essential to ensure that your system is up to date. Open a terminal and execute the following commands:



```
sudo apt update
sudo apt upgrade
```

Step 2: Installing Postfix

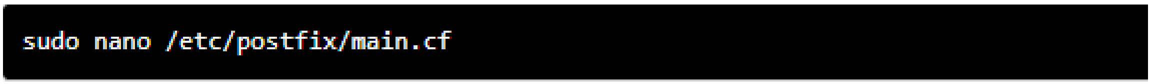
To install Postfix, use the following command:



```
sudo apt install postfix
```

Step 3: Configuring Postfix

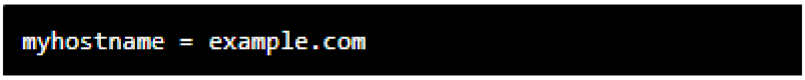
Once Postfix is installed, we need to make a few configurations. Open the main Postfix configuration file using your preferred text editor:



```
sudo nano /etc/postfix/main.cf
```

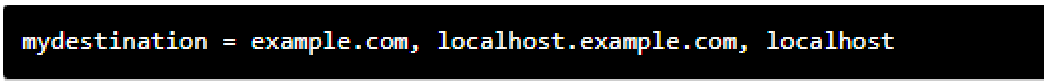
Make the following changes in the configuration file:

- Locate the myhostname parameter and set it to your domain name:



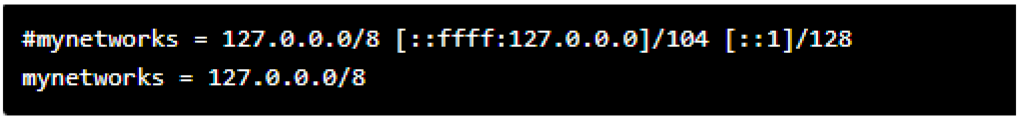
```
myhostname = example.com
```

- Find the mydestination parameter and modify it as follows:



```
mydestination = example.com, localhost.example.com, localhost
```

- Uncomment the mynetworks parameter to allow connections from trusted networks:

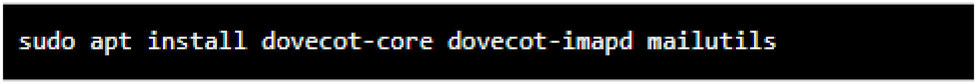


```
#mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mynetworks = 127.0.0.0/8
```

Save the changes and exit the editor.

Step 4: Installing Dovecot

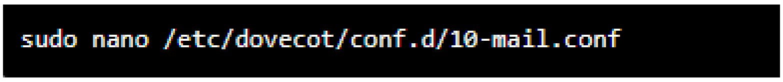
To install Dovecot, execute the following command:



```
sudo apt install dovecot-core dovecot-imapd mailutils
```

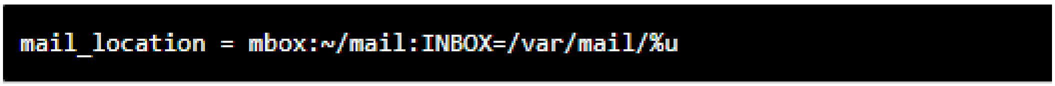
Step 5: Configuring Dovecot

Next, we’ll configure Dovecot to work in conjunction with Postfix. Open the Dovecot configuration file:



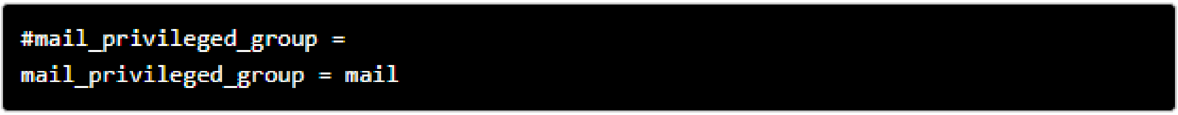
```
sudo nano /etc/dovecot/conf.d/10-mail.conf
```

- Locate the mail_location parameter and set it to the default value:



```
mail_location = mbox:~/mail:INBOX=/var/mail/%u
```

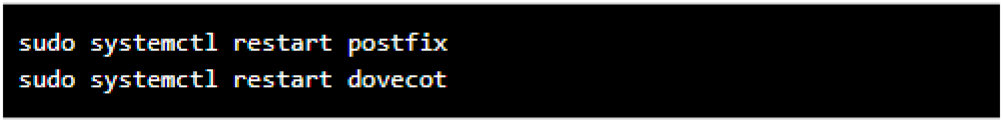
Uncomment the mail_privileged_group parameter and set it to mail



```
#mail_privileged_group =
mail_privileged_group = mail
```

Step 6: Restarting Services

Now, restart both Postfix and Dovecot services to apply the changes made



```
sudo systemctl restart postfix
sudo systemctl restart dovecot
```

Step 7: Testing the Configuration

To ensure that everything is functioning correctly, you can send a test email to an external address using the mail command:

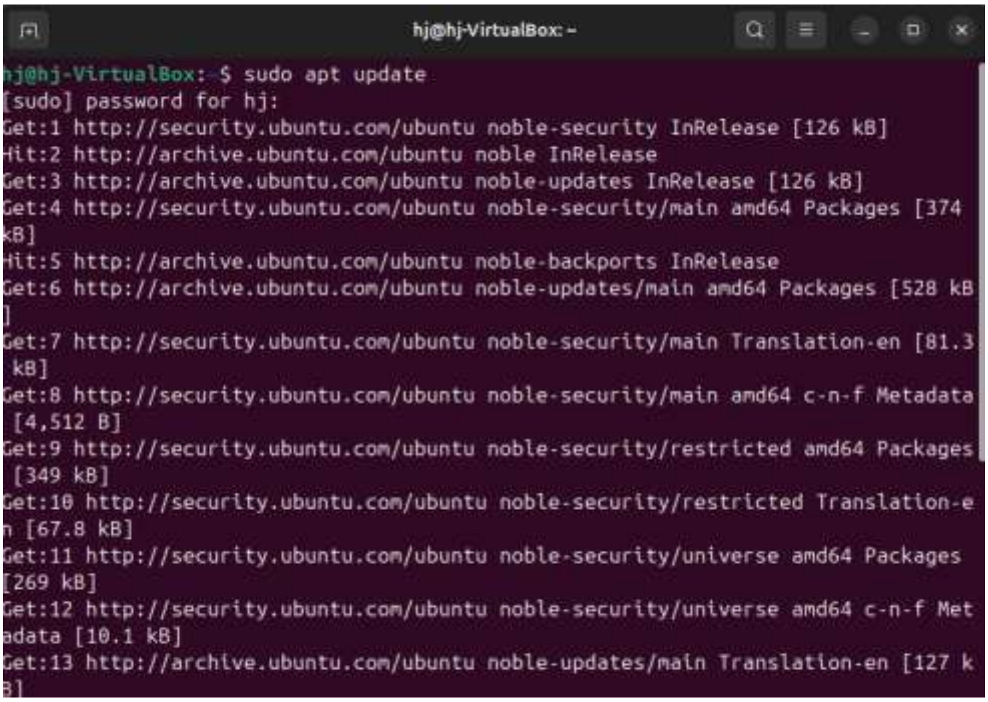
✓ **Configuration of file services**

To configure a file server service in Ubuntu, you'll typically use **Samba** if you want to share files between Linux and Windows systems, or **NFS** if you’re sharing files between Linux systems. Here’s a step-by-step guide for both.

1. Samba (Linux and Windows)

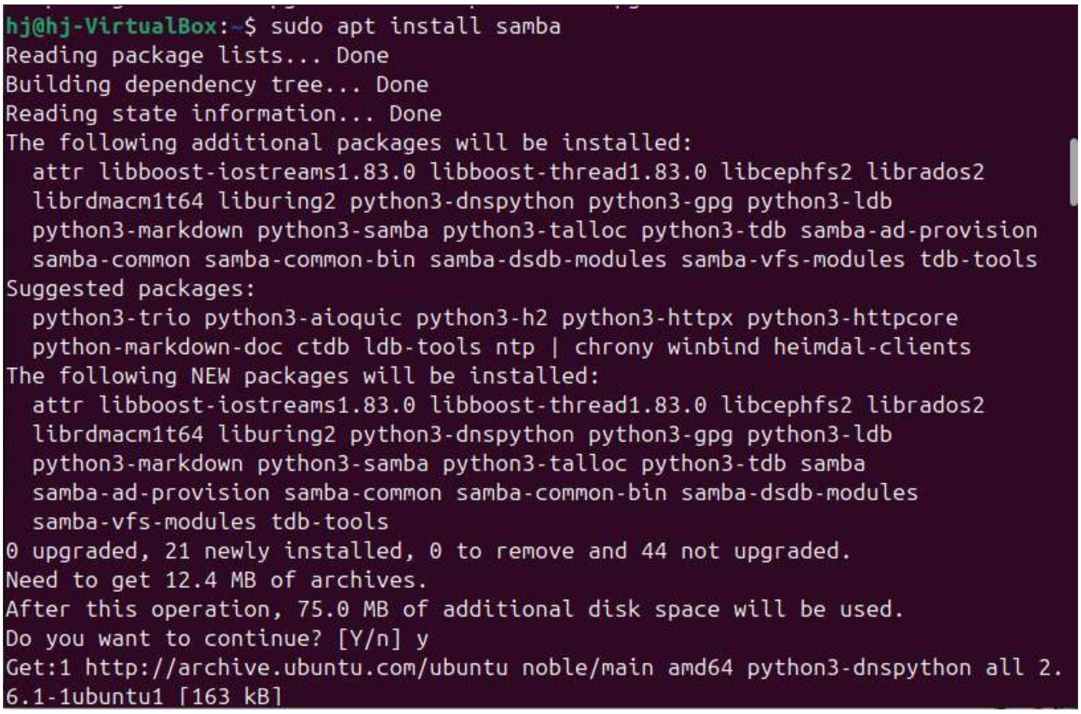
Samba allows Linux to share files with Windows systems

Step 1: Update and install Samba



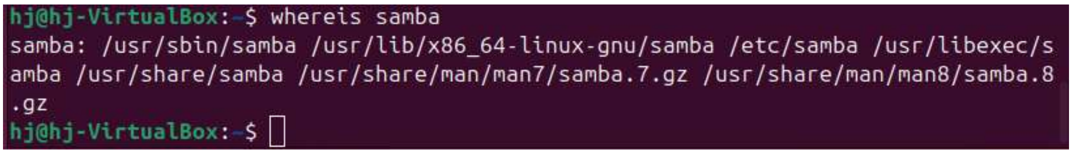
```
hj@hj-VirtualBox:~$ sudo apt update
[sudo] password for hj:
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [374
kB]
Hit:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [528 kB
]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [81.3
 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata
 [4,512 B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages
 [349 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-e
n [67.8 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages
[269 kB]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Met
adata [10.1 kB]
Get:13 http://archive.ubuntu.com/ubuntu noble-updates/main Translation-en [127 k
B]
```

Step 2 install samba by using **sudo apt install samba** command



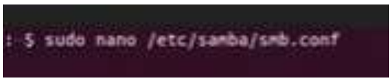
```
hj@hj-VirtualBox:~$ sudo apt install samba
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  attr libboost-iostreams1.83.0 libboost-thread1.83.0 libcephfs2 librados2
  librdmacm1t64 liburing2 python3-dnspython python3-gpg python3-ldb
  python3-markdown python3-samba python3-talloc python3-tdb samba-ad-provision
  samba-common samba-common-bin samba-dsdb-modules samba-vfs-modules tdb-tools
Suggested packages:
  python3-trio python3-aioquic python3-h2 python3-httpx python3-httpcore
  python-markdown-doc ctdb ldb-tools ntp | chrony winbind heimdal-clients
The following NEW packages will be installed:
  attr libboost-iostreams1.83.0 libboost-thread1.83.0 libcephfs2 librados2
  librdmacm1t64 liburing2 python3-dnspython python3-gpg python3-ldb
  python3-markdown python3-samba python3-talloc python3-tdb samba
  samba-ad-provision samba-common samba-common-bin samba-dsdb-modules
  samba-vfs-modules tdb-tools
0 upgraded, 21 newly installed, 0 to remove and 44 not upgraded.
Need to get 12.4 MB of archives.
After this operation, 75.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble/main amd64 python3-dnspython all 2.
6.1-1ubuntu1 [163 kB]
```

Step 3: check where Samba is installed successful



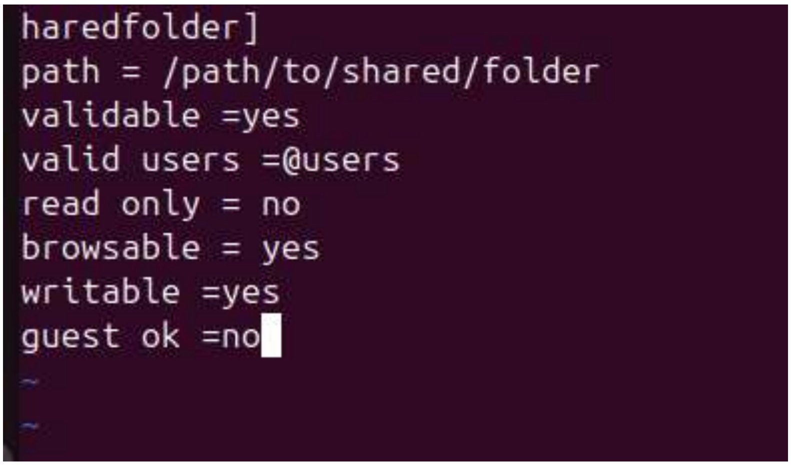
```
hj@hj-VirtualBox:~$ whereis samba
samba: /usr/sbin/samba /usr/lib/x86_64-linux-gnu/samba /etc/samba /usr/libexec/s
amba /usr/share/samba /usr/share/man/man7/samba.7.gz /usr/share/man/man8/samba.8
.gz
hj@hj-VirtualBox:~$ 
```

Edit configuration file for samba



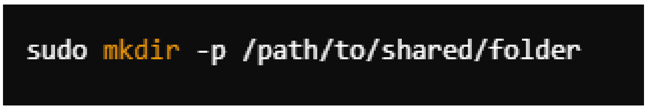
```
:~$ sudo nano /etc/samba/smb.conf
```

Add a new share configuration at the end of the file. For example



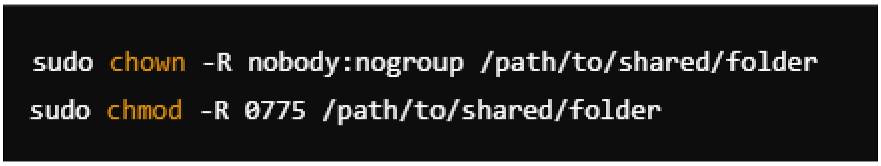
```
haredfolder]
path = /path/to/shared/folder
validable =yes
valid users =@users
read only = no
browsable = yes
writable =yes
guest ok =no
~
~
```

Step 3: Create a shared directory



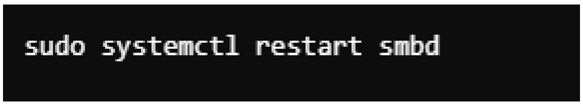
```
sudo mkdir -p /path/to/shared/folder
```

Set the proper permissions



```
sudo chown -R nobody:nogroup /path/to/shared/folder
sudo chmod -R 0775 /path/to/shared/folder
```

tep 5: Restart Samba service



```
sudo systemctl restart smbd
```

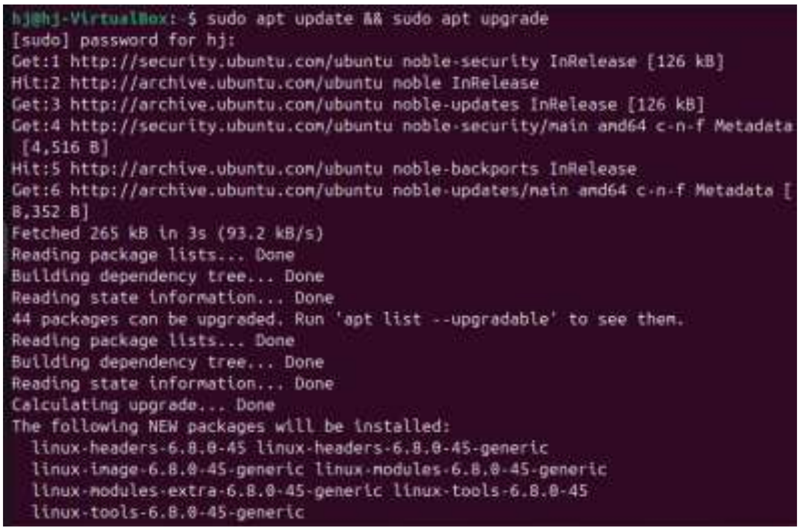
- ✓ **configuration of Network service**

Steps to configure network server service

Step 1: update the system

Before starting any installation or configuration, it's a good idea to update the system to ensure all packages are up-to-date.

sudo apt update && sudo apt upgrade



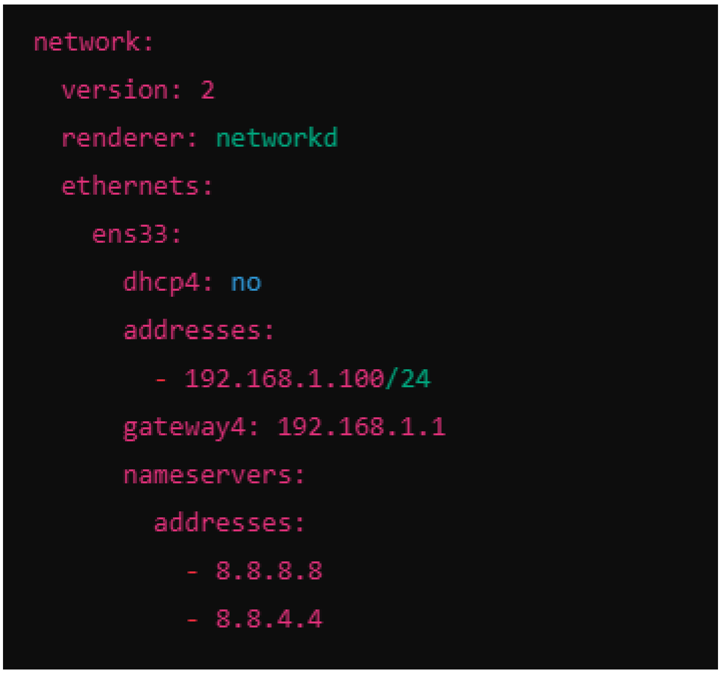
```
hj@hj-VirtualBox:~$ sudo apt update && sudo apt upgrade
[sudo] password for hj:
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:2 http://archive.ubuntu.com/ubuntu noble InRelease
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata
 [4,516 B]
Hit:5 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Get:6 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 c-n-f Metadata [
8,352 B]
Fetched 265 kB in 3s (93.2 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
44 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following NEW packages will be installed:
  linux-headers-6.8.0-45 linux-headers-6.8.0-45-generic
  linux-image-6.8.0-45-generic linux-modules-6.8.0-45-generic
  linux-modules-extra-6.8.0-45-generic linux-tools-6.8.0-45
  linux-tools-6.8.0-45-generic
```

2. **Configure Network Interfaces**

Ubuntu uses netplan for network configuration. You may need to set a static IP address or configure your network interface for dynamic IP (DHCP) depending on your needs.

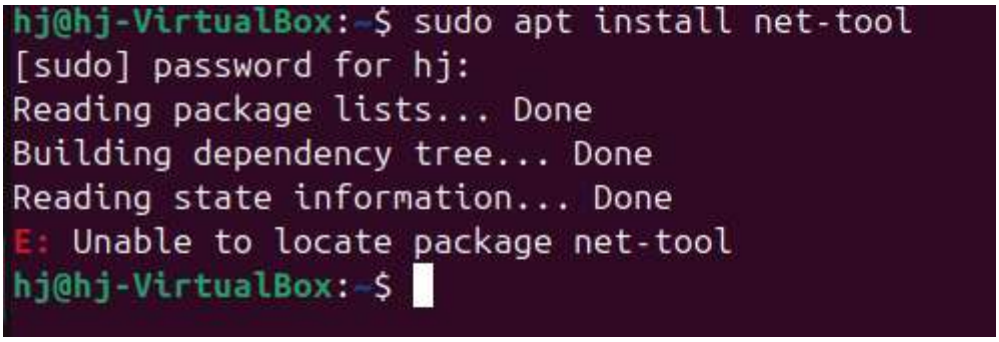
To configure a network interface, edit the Netplan configuration file, usually located in /etc/netplan/ (e.g., 50-cloud-init.yaml).

sudo nano /etc/netplan/50-cloud-init.yaml



```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens33:
      dhcp4: no
      addresses:
        - 192.168.1.100/24
      gateway4: 192.168.1.1
      nameservers:
        addresses:
          - 8.8.8.8
          - 8.8.4.4
```

Install network tool

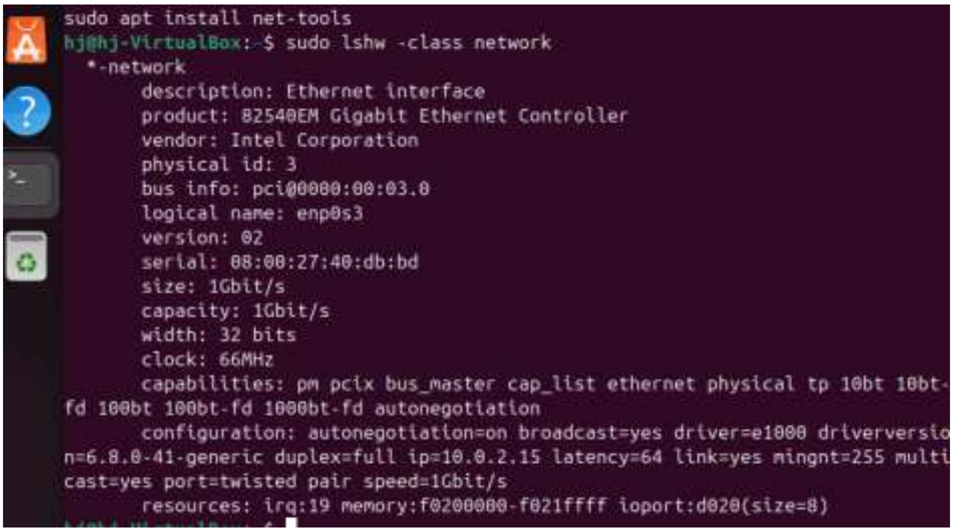


```
hj@hj-VirtualBox:~$ sudo apt install net-tool
[sudo] password for hj:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package net-tool
hj@hj-VirtualBox:~$ 
```

Check ip address

```
sudo lshw -class network
```

With the **lshw** command, you can define all the network interfaces you need for configuring networks on Ubuntu. Below you will see an example command. This command will display bus information, driver details, and all its supported capabilities as a single Ethernet interface:

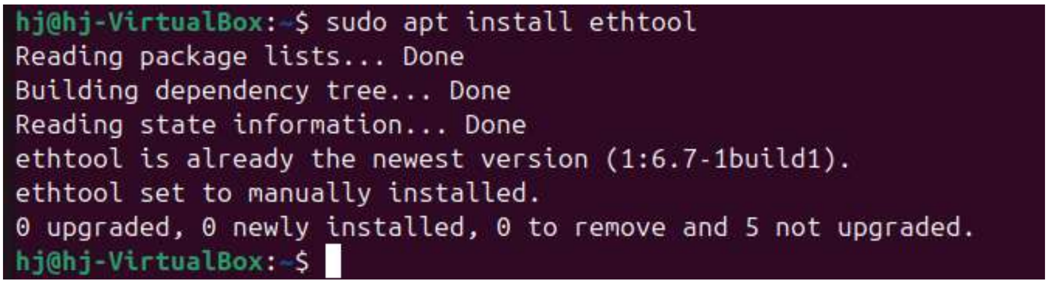


```
sudo apt install net-tools
hj@hj-VirtualBox:~$ sudo lshw -class network
  *-network
       description: Ethernet interface
       product: 82540EM Gigabit Ethernet Controller
       vendor: Intel Corporation
       physical id: 3
       bus info: pci@0000:00:03.0
       logical name: enp0s3
       version: 02
       serial: 08:00:27:40:db:bd
       size: 1Gbit/s
       capacity: 1Gbit/s
       width: 32 bits
       clock: 66MHz
       capabilities: pm pcix bus_master cap_list ethernet physical tp 10bt 10bt-
fd 100bt 100bt-fd 1000bt-fd autonegotiation
       configuration: autonegotiation=on broadcast=yes driver=e1000 driverversio
n=6.8.0-41-generic duplex=full ip=10.0.2.15 latency=64 link=yes mingnt=255 multi
cast=yes port=twisted pair speed=1Gbit/s
       resources: irq:19 memory:f0200000-f021ffff ioport:d020(size=8)
```

Configuring Ethernet Interface on Ubuntu 18.04 and Earlier

With the **ethtool** program, you can view settings such as auto-negotiation, duplex mode, and port speed. If ethtool is not installed on your server, you can install it using the following command:

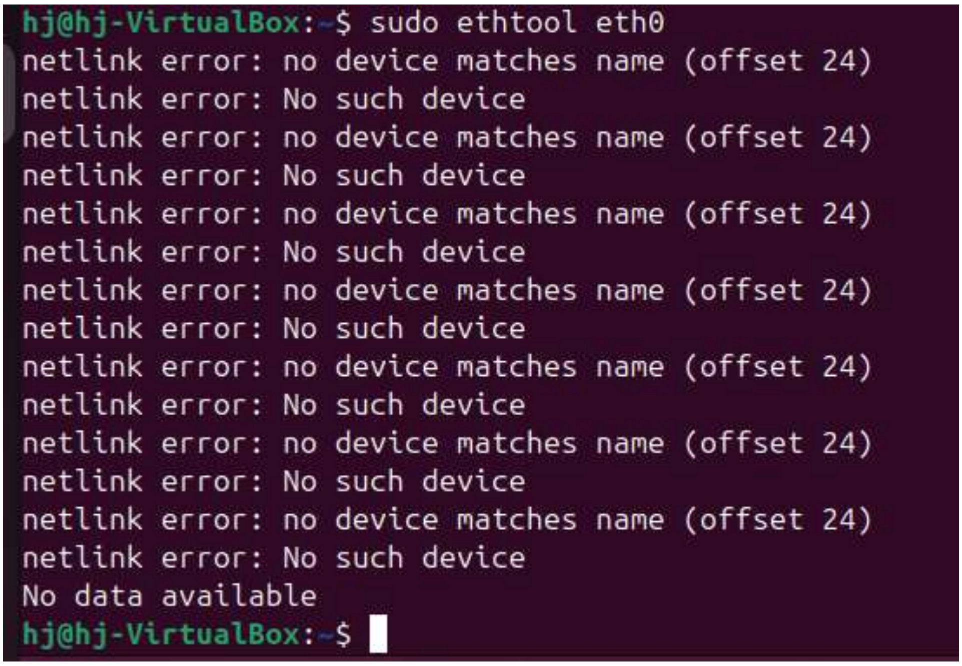
```
sudo apt install ethtool
```



```
hj@hj-VirtualBox:~$ sudo apt install ethtool
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ethtool is already the newest version (1:6.7-1build1).
ethtool set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
hj@hj-VirtualBox:~$
```

After the installation is complete, you can see the sample output for **eth0**:

```
sudo ethtool eth0
```



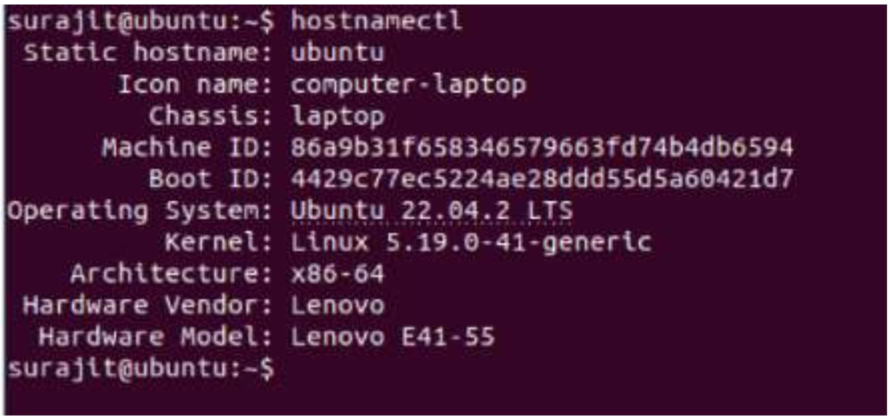
```
hj@hj-VirtualBox:~$ sudo ethtool eth0
netlink error: no device matches name (offset 24)
netlink error: No such device
netlink error: no device matches name (offset 24)
netlink error: No such device
netlink error: no device matches name (offset 24)
netlink error: No such device
netlink error: no device matches name (offset 24)
netlink error: No such device
netlink error: no device matches name (offset 24)
netlink error: No such device
netlink error: no device matches name (offset 24)
netlink error: No such device
netlink error: no device matches name (offset 24)
netlink error: No such device
No data available
hj@hj-VirtualBox:~$
```

- ✓ **DNS configuration steps**

Step 1: Determine your Lunix Distribution and their operation.

For determining the Linux distribution installed in the system, we will use the **hostnamectl** command.

Just open a terminal window and run the command.

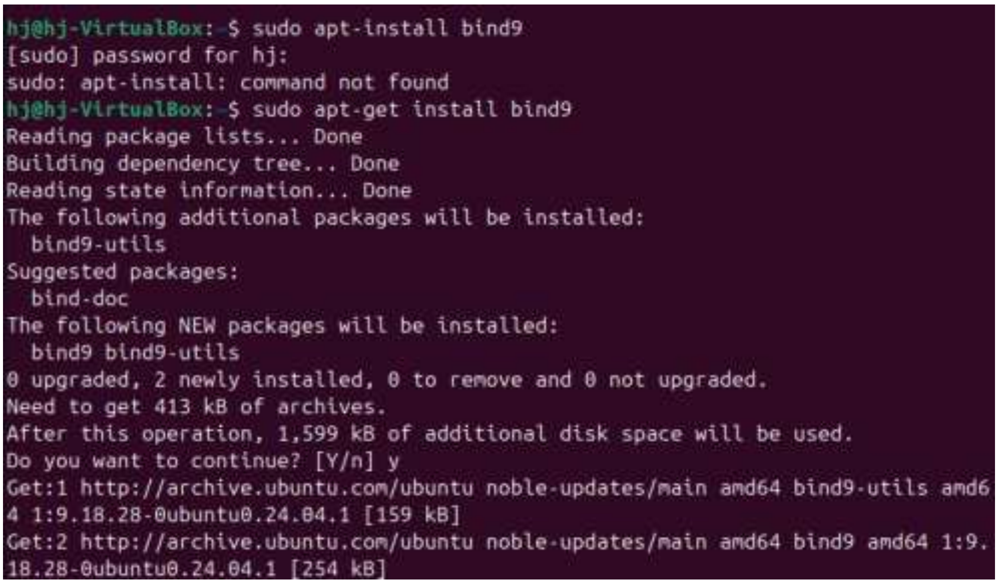


```
surajit@ubuntu:~$ hostnamectl
 Static hostname: ubuntu
       Icon name: computer-laptop
         Chassis: laptop
      Machine ID: 86a9b31f658346579663fd74b4db6594
         Boot ID: 4429c77ec5224ae28ddd55d5a60421d7
Operating System: Ubuntu 22.04.2 LTS
          Kernel: Linux 5.19.0-41-generic
    Architecture: x86-64
 Hardware Vendor: Lenovo
  Hardware Model: Lenovo E41-55
surajit@ubuntu:~$
```

Step - 2: Install DNS Server Software

Once you've determined your Linux distribution, the next step is to install the DNS server software. **Bind (Berkeley Internet Name Domain**). To install Bind, open the terminal and use the appropriate package manager command for your distribution. For example, on Ubuntu, you can use the below-mentioned command.

sudo apt-get install bind9



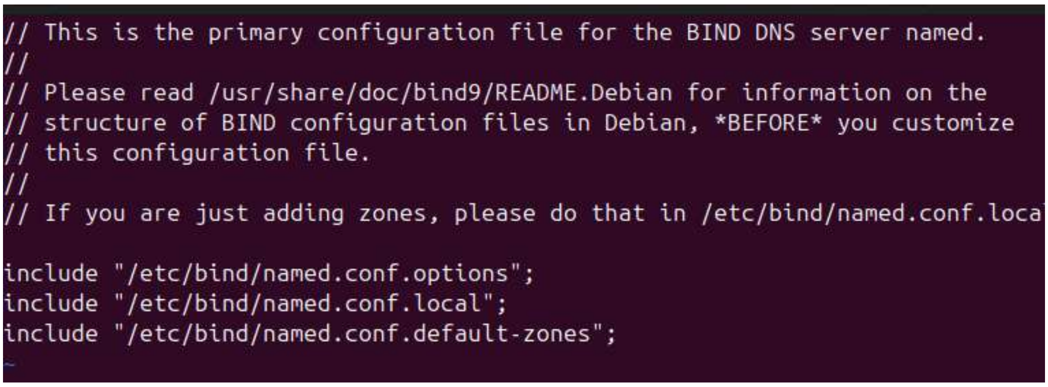
```
hj@hj-VirtualBox:~$ sudo apt-install bind9
[sudo] password for hj:
sudo: apt-install: command not found
hj@hj-VirtualBox:~$ sudo apt-get install bind9
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bind9-utils
Suggested packages:
  bind-doc
The following NEW packages will be installed:
  bind9 bind9-utils
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 413 kB of archives.
After this operation, 1,599 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 bind9-utils amd6
4 1:9.18.28-0ubuntu0.24.04.1 [159 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 bind9 amd64 1:9.
18.28-0ubuntu0.24.04.1 [254 kB]
```

During installation system ask you to type Y/No to continue install addition type Y

Step - 3: Configure Bind DNS Server

After installing Bind, you need to configure it according to your requirements. The main configuration file for Bind is generally located at /etc/bind/named.conf. Open this file with a text editor of your choice.

To open the file using nano

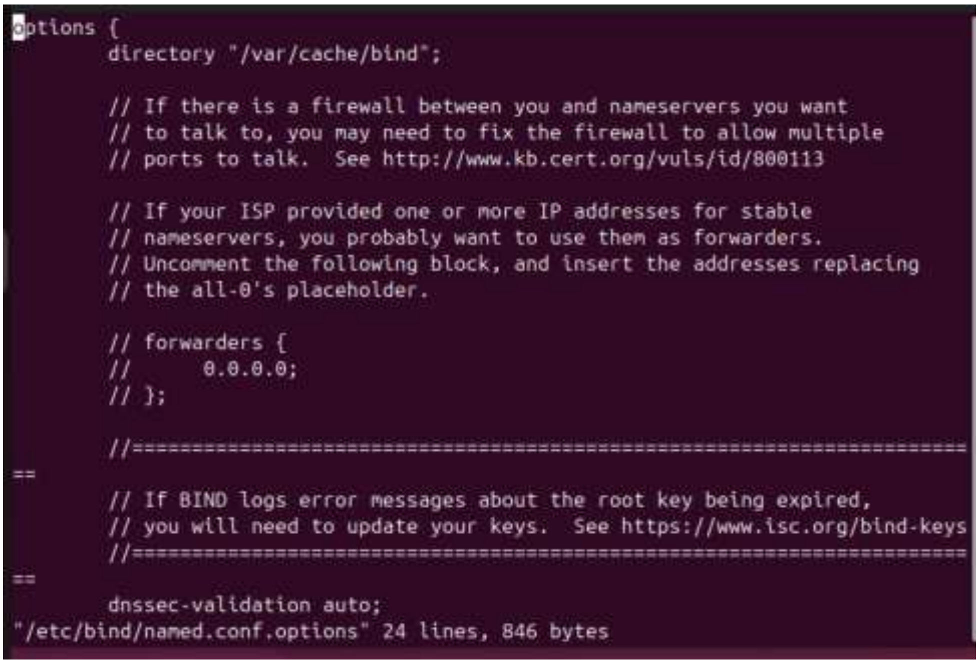


```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.loca

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
~
```

These files contain additional settings for options, local zones, and default zones. You can edit them as needed.

1. Configure Options (Edit named.conf.options)

Here’s a basic example of global options:



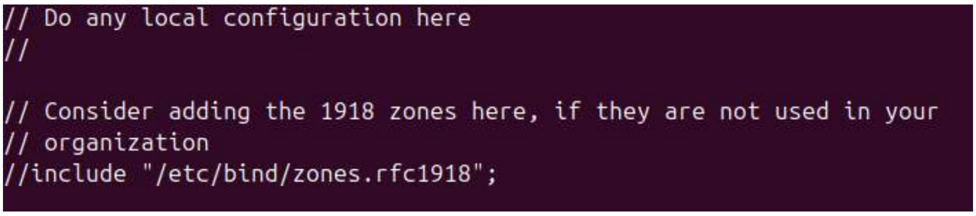
```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        // forwarders {
        //      0.0.0.0;
        // };

        //=====================================================================
==
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //=====================================================================
==
        dnssec-validation auto;
"/etc/bind/named.conf.options" 24 lines, 846 bytes
```

4. Configure Local Zones (Edit named.conf.local)

You need to define your zones (forward and reverse) in the named.conf.local file.

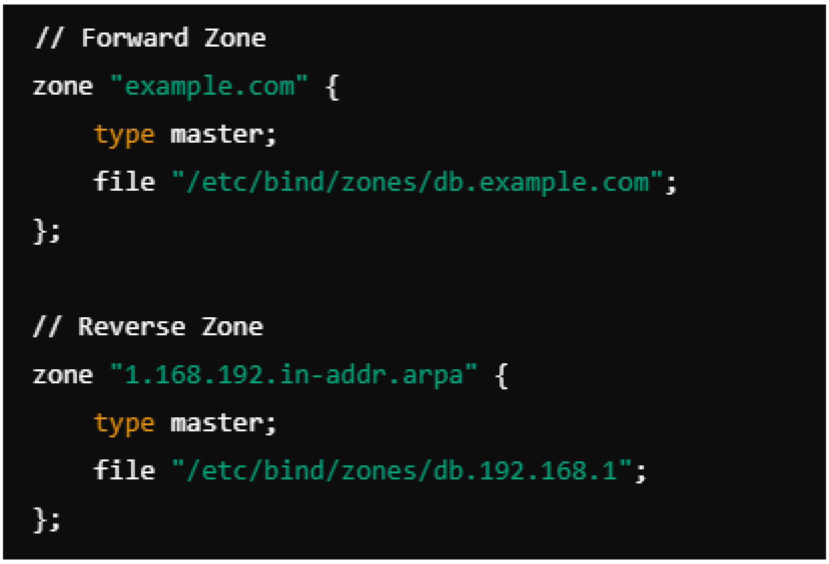
Open it:

sudo nano /etc/bind/named.conf.local



```
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
```

Example of adding a forward and reverse zone:

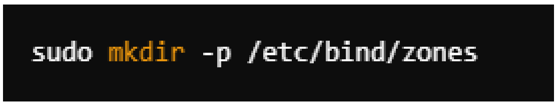


```
// Forward Zone
zone "example.com" {
    type master;
    file "/etc/bind/zones/db.example.com";
};

// Reverse Zone
zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/db.192.168.1";
};
```

5. Create Zone Files

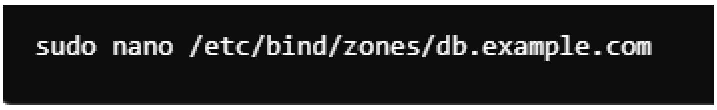
After defining the zones, you need to create the corresponding zone files. These are usually stored in /etc/bind/zones/.

1. **Create the directory for zone files** (if it doesn't exist):



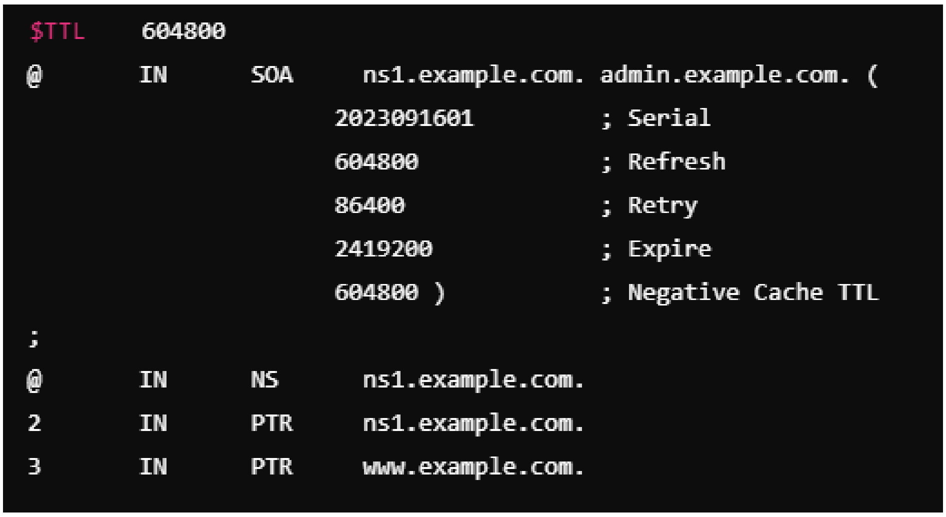
```
sudo mkdir -p /etc/bind/zones
```

Create the zone file for the forward zone:



```
sudo nano /etc/bind/zones/db.example.com
```

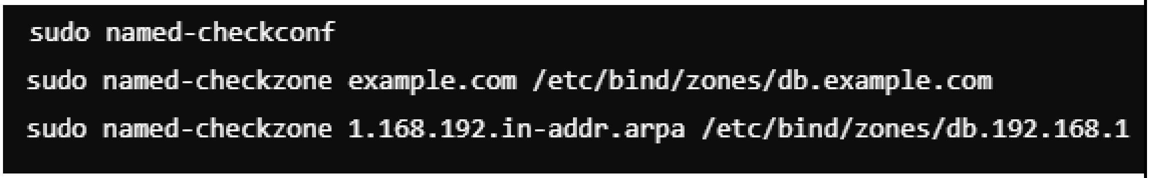
Add content like



```
$TTL    604800
@       IN      SOA     ns1.example.com. admin.example.com. (
                      2023091601         ; Serial
                      604800             ; Refresh
                      86400              ; Retry
                      2419200            ; Expire
                      604800 )           ; Negative Cache TTL
;
@       IN      NS      ns1.example.com.
2       IN      PTR     ns1.example.com.
3       IN      PTR     www.example.com.
```

6. Check Syntax for Errors

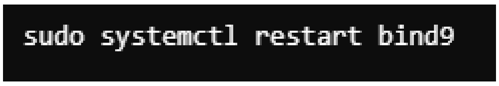
Before restarting BIND, verify the configuration for syntax errors:



```
sudo named-checkconf
sudo named-checkzone example.com /etc/bind/zones/db.example.com
sudo named-checkzone 1.168.192.in-addr.arpa /etc/bind/zones/db.192.168.1
```

7. Restart BIND

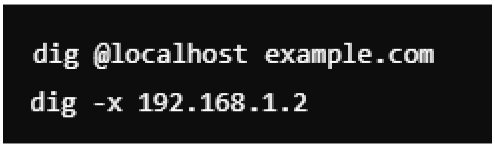
Finally, restart the BIND service to apply the changes:



```
sudo systemctl restart bind9
```

8. Test the DNS Server

Test your DNS server to ensure everything is working properly



```
dig @localhost example.com
dig -x 192.168.1.2
```

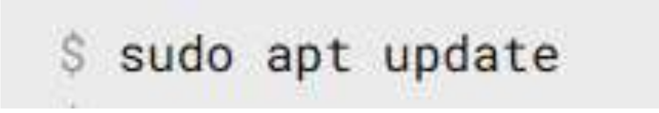
✓ **Configuration of SSH Server Service**

To configure an SSH server on Ubuntu, follow these steps:

Step 1: Install the SSH Server

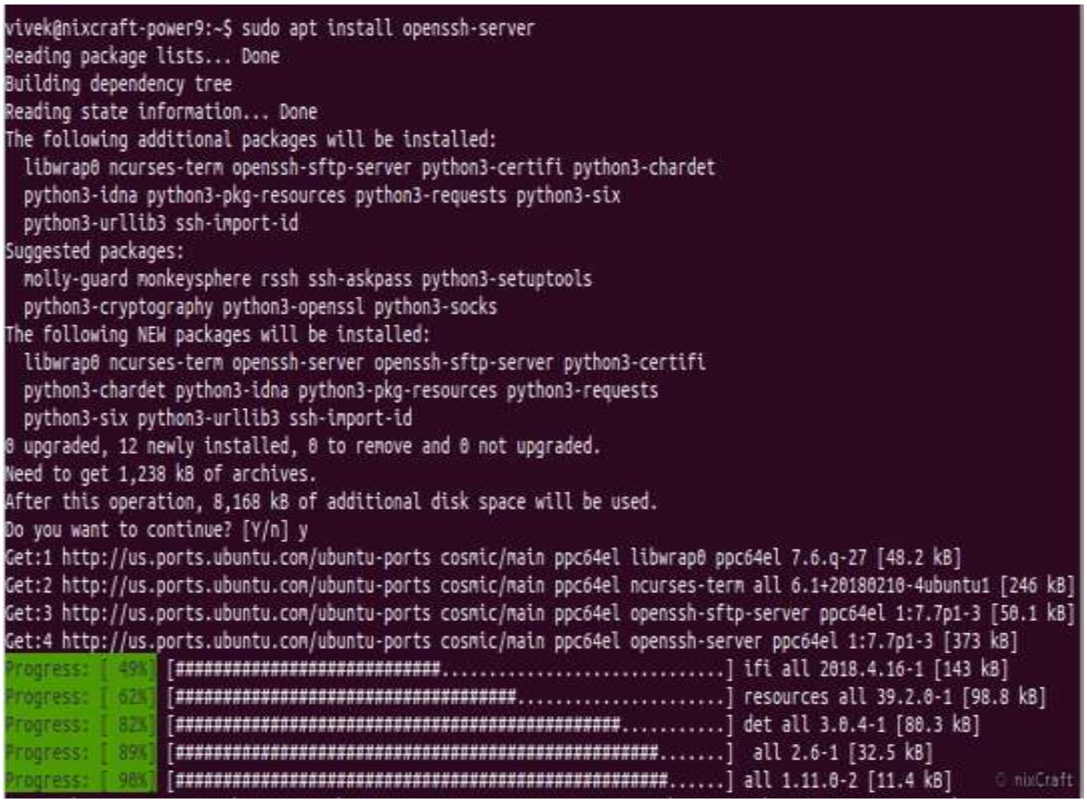
First, install the OpenSSH server package, which provides the SSH service.

sudo apt update



```
$ sudo apt update
```

sudo apt install openssh-server

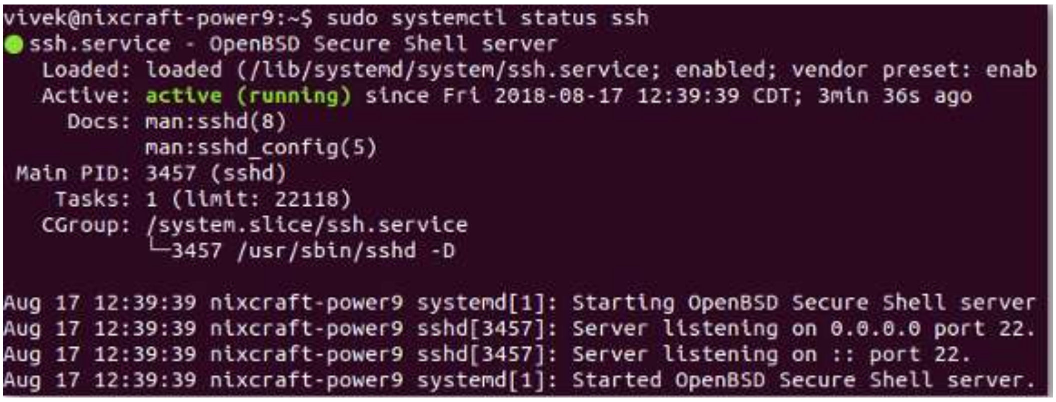


```
vivek@nixcraft-power9:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libwrap0 ncurses-term openssh-sftp-server python3-certifi python3-chardet
  python3-idna python3-pkg-resources python3-requests python3-six
  python3-urllib3 ssh-import-id
Suggested packages:
  molly-guard monkeysphere rssh ssh-askpass python3-setuptools
  python3-cryptography python3-openssl python3-socks
The following NEW packages will be installed:
  libwrap0 ncurses-term openssh-server openssh-sftp-server python3-certifi
  python3-chardet python3-idna python3-pkg-resources python3-requests
  python3-six python3-urllib3 ssh-import-id
0 upgraded, 12 newly installed, 0 to remove and 0 not upgraded.
Need to get 1,238 kB of archives.
After this operation, 8,168 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.ports.ubuntu.com/ubuntu-ports cosmic/main ppc64el libwrap0 ppc64el 7.6.q-27 [48.2 kB]
Get:2 http://us.ports.ubuntu.com/ubuntu-ports cosmic/main ppc64el ncurses-term all 6.1+20180210-4ubuntu1 [246 kB]
Get:3 http://us.ports.ubuntu.com/ubuntu-ports cosmic/main ppc64el openssh-sftp-server ppc64el 1:7.7p1-3 [50.1 kB]
Get:4 http://us.ports.ubuntu.com/ubuntu-ports cosmic/main ppc64el openssh-server ppc64el 1:7.7p1-3 [373 kB]
Progress: [ 49%] [############################.............................] ifi all 2018.4.16-1 [143 kB]
Progress: [ 62%] [####################################.....................] resources all 39.2.0-1 [98.8 kB]
Progress: [ 82%] [##############################################...........] det all 3.0.4-1 [80.3 kB]
Progress: [ 89%] [##################################################.......]  all 2.6-1 [32.5 kB]
Progress: [ 90%] [###################################################......] all 1.11.0-2 [11.4 kB]
```

Step 2: Verify SSH Service Status

After installation, the SSH service should start automatically. To check its status, run:

sudo systemctl status ssh



```
vivek@nixcraft-power9:~$ sudo systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enab
   Active: active (running) since Fri 2018-08-17 12:39:39 CDT; 3min 36s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
 Main PID: 3457 (sshd)
    Tasks: 1 (limit: 22118)
   CGroup: /system.slice/ssh.service
           └─3457 /usr/sbin/sshd -D

Aug 17 12:39:39 nixcraft-power9 systemd[1]: Starting OpenBSD Secure Shell server
Aug 17 12:39:39 nixcraft-power9 sshd[3457]: Server listening on 0.0.0.0 port 22.
Aug 17 12:39:39 nixcraft-power9 sshd[3457]: Server listening on :: port 22.
Aug 17 12:39:39 nixcraft-power9 systemd[1]: Started OpenBSD Secure Shell server.
```

Step 3: Configure SSH Settings

SSH configuration is stored in the /etc/ssh/sshd_config file. To make any changes, open the configuration file with a text editor like Vi

sudo vi /etc/ssh/sshd_config

Step 4: Change default SSH port (optional): You can change the default port (from 22) for added security:

Port 2222

Permit or disable root login (recommended to disable):

PermitRootLogin no

Restrict SSH access to certain users (optional)

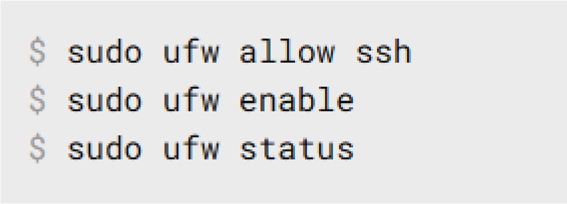
AllowUsers username

Restart the SSH Service

sudo systemctl restart ssh

Step 5: Configure Firewall (if applicable)

If a firewall is active on your Ubuntu server, you will need to allow SSH traffic. By default, SSH listens on port 22. If you've changed the port, modify the command accordingly.



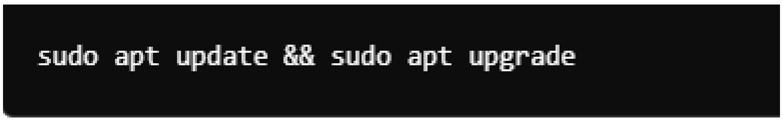
```
$ sudo ufw allow ssh
$ sudo ufw enable
$ sudo ufw status
```

- ✓ **Configuration of proxy Server Service**

Steps to configure Proxy server service

Step 1: Update the System

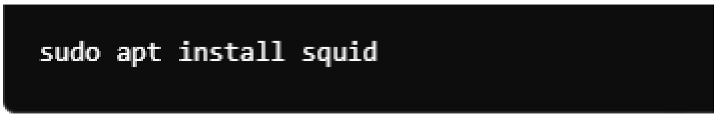
Before starting, ensure your system is up to date.



```
sudo apt update && sudo apt upgrade
```

Step 2. Install Squid Proxy Server

To install Squid, use the following command:

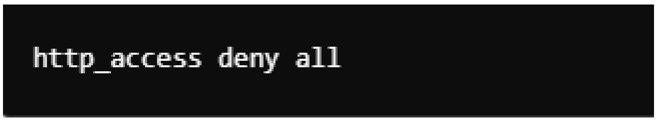


```
sudo apt install squid
```

After installation, the Squid configuration file will be located at /etc/squid/squid.conf

Step 3. Configure Squid

The main configuration file for Squid is /etc/squid/squid.conf. You will need to edit this file to configure your proxy server.

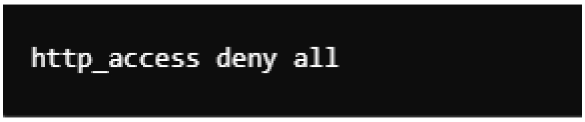


```
http_access deny all
```

Allowing Access

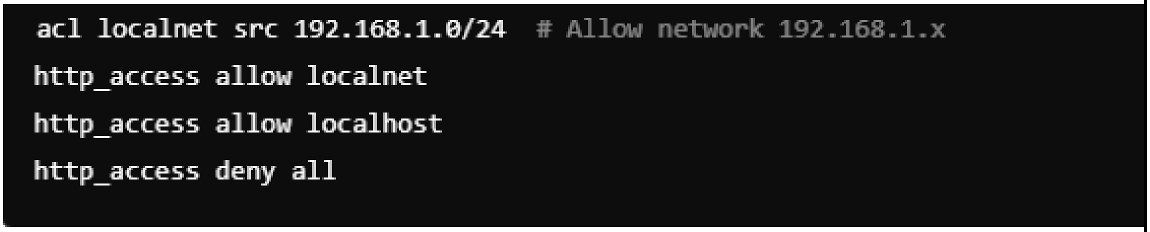
By default, Squid denies all incoming traffic for security purposes. You need to define access rules by configuring **Access Control Lists (ACLs)**.

In the squid.conf file, search for the following lines:



```
http_access deny all
```

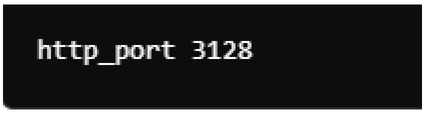
Replace it with



```
acl localnet src 192.168.1.0/24  # Allow network 192.168.1.x
http_access allow localnet
http_access allow localhost
http_access deny all
```

b. Setting a Proxy Port (Optional)

By default, Squid listens on port 3128. You can change the port by editing this line in the configuration file

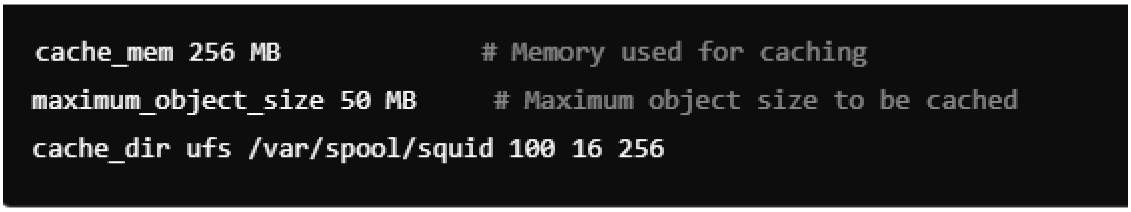


```
http_port 3128
```

4. Configure Proxy Caching (Optional)

One of Squid's core features is caching web content. You can configure Squid to cache frequently accessed content by adjusting cache settings in /etc/squid/squid.conf.

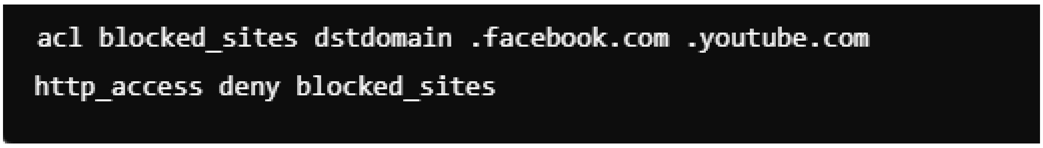
For example:



```
cache_mem 256 MB              # Memory used for caching
maximum_object_size 50 MB     # Maximum object size to be cached
cache_dir ufs /var/spool/squid 100 16 256
```

5. Restrict Access by IP or Time (Optional)

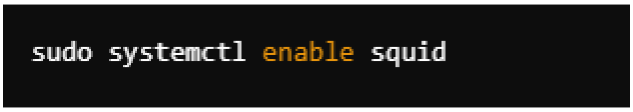
You can restrict access to certain websites or set up access schedules. For instance, to block specific websites



```
acl blocked_sites dstdomain .facebook.com .youtube.com
http_access deny blocked_sites
```

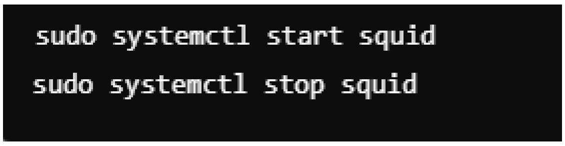
6. Configure Squid to Start on Boot

Ensure Squid starts automatically when the system boots:



```
sudo systemctl enable squid
```

You can also manually start and stop Squid with:



```
sudo systemctl start squid
sudo systemctl stop squid
```

Points to Remember

- Server service is a software component that runs on a server (a computer that hosts websites and web applications) and provides a specific functionality or service to other computers or devices connected to the network.
- There are many types of server services depending on the functionality or service they provide. Some of the most common used are Web Server Service, Database server service, Mail server service, FTP Server Service,
- While managing server there are different Services require system administrator to install and manage in order the server operate smoothly such as network, web, DNS, SSH, proxy and more.

Application of learning 1.3.

RTDX Company has recently expanded its infrastructure to accommodate a growing number of users and services. As the system administrator, you are tasked with setting up and configuring essential services across multiple Linux-based servers to ensure the smooth operation of internal and external communications.

Learning outcome 1 end assessment

Theoretical assessment

Q1. Circle the letter corresponding with the correct answer.

i. The main objective of DevOps is:

a) To deliver software in isolated silos
b) To enhance collaboration between development and operations
C) To slow down the release process
D) To remove automation in development

ii. CI/CD stand for:

a) Continuous Input/Continuous Delivery
b) Continuous Integration/Continuous Development
c) Continuous Improvement/Continuous Deployment
d) Continuous Integration/Continuous Deployment

iii. One of the following statements describes "Infrastructure as Code" (IaC):

a) Managing infrastructure manually
b) Using code to define and manage infrastructure
c) Using spreadsheets to manage servers
d) Hiring additional staff for server management

iv. Which of the following is a version control system widely used in DevOps?
A) Git
B) Docker
C) Slack
D) Kubernetes

Q2. Match the commands **(column B)** with their function **(column C).** by writing the letter corresponding with the right answer in provided blank space **(column A)**

Answer (Column A)	**Commands (Column B)**	**Functions (column C)**
1………..	1. Cd.	A) is used to install any package
2……….	2. sudo shutdown now	B) to change directory

3………	3.mkdir	C) used to list the contents of a directory.
4………	4.Cat	D) is command used to shuts down the system immediately
5……….	5. Sudo	E) to create directory
6……….	6. Cp	F) Restarting System Services
7………	7. PWD	G) to display the root path
8………	8. sudo systemctl restart service name	I) is used to run commands with superuser (root) privileges. It stands for "SuperUser DO" and allows authorized users to execute administrative tasks without logging in as the root user.
9……..	9. sudo apt install	J) is command used to copy files and directories
		I) Linux is used to **change the file permissions**
		H) is command used to shuts down the system immediately

Practical assessment

Rwanda network Ltd is a new Company hosted the web application, this company is located at Muhanga District. It has a Network system that is Implemented with windows Server Management, it needs to upgrade from windows server to Linux server in order to manage different services. you are requested to install Linux Server, configure Web server services and Test server functionalities.

References

DevOps glossary: 78 basic DevOps. (2023, July 12). Retrieved April 27, 2024, from Its Vit: https://itsvit.com/blog/devops-glossary-78-basic-devops-terms-in-simple-words/

Fortinet. (2023, April). DevOps Security. Retrieved April 27, 2024, from Fortinet: https://www.fortinet.com/resources/cyberglossary/devops-security

GitLab. (2024). What is DevOps. Retrieved April 27, 2024, from About Git Lab: https://about.gitlab.com/topics/devops/

HALL, T. (2023, April). DevOps metrics: Why, what, and how to measure success in DevOps. Retrieved April 27, 2024, from Atlassian: https://www.atlassian.com/devops/frameworks/devops-metrics

Humble, J. &. (2010). Continous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional.

Learning Outcome 2: Deploy System

Indicative contents

2.1 Preparation of deployment

2.2 Use Continuous delivery and Configure of containerization

2.3 Perform Migration

Key Competencies for Learning Outcome 2: Deploy system

Knowledge	Skills	Attitudes
● Description of deployment key terms ● Identification of evolution and Importance of DevOps ● Description of Advantages and Disadvantages of DevOps ● Description of DevOps Technologies ● Description of DevOps principles ● Description of DevOps lifecycle ● Identification of technologies used in system to be deployed ● Identification of data migration ● Description continuous integration and continuous delivery ● Identification of container	● Installing of dependencies ● Configuring CI/CD pipeline ● Implementing data migration pipeline.	● Being collaborative while preparing or deploying software ● Being flexible while preparing deployment ● Being decision make while handling errors.

Duration: 30 hrs

Learning outcome 2 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Describe clearly the deployment key terms based on the system to be deployed
2. Describe properly the evolutions, advantages and disadvantages of DevOps based on the system to be deployed.
3. Describe properly lifecycle used in continuous Integration and continuous deployment
4. Describe correctly technologies and principles of DevOps based on the system to be deployed.
5. Identify clearly the containerization based on scalability of application
6. Identify properly data migration based on system environment
7. Install properly the required dependencies based on system environment
8. Configure properly Container based System environment.
9. Configure continuous integration and continuous pipeline deployment based on area of automation
10. Create correctly data migration based on system environment.

Resources

Equipment	**Tools**	**Materials**
- Computer - server	- Docker setup - Git setup - GitHub account - Linux OS - VS code - Amazon account	- Internet - Electricity

Indicative content 2.1: Preparation of deployment environnement

Duration: 5 hrs

Theoretical Activity 2.1.1: Description of key terms

Tasks:

1: You are asked to answer the following questions:

i. Give the meaning of the following terms:

a. Deployment
b. Build agent
c. Containerization
d. Docker
e. Kubernetes
f. Jargon
g. Dependence

ii. Give DevOps evolutions with their advantages and disadvantages.

iii. Identify the principles of DevOps

iv. Identify phases of DevOps lifecycle

v. What are the tools and technologies used in system deployment?

2: Provide your answers on paper, flipchart, blackboard or whiteboard

3: Present your answers to whole class trainees and trainer

4: Ask questions for clarifications if necessary

5: Read the key readings 2.1.1

Key readings 2.1.1.: Description of deployment environment

✓ **Definition of key terms**

- **Deployment** in DevOps refers to the process of releasing a new version of an application or software to a production environment. It's the final stage of the software development lifecycle (SDLC) where the application becomes available to end-users.
- **Build Agent** in DevOps is a machine or service responsible for executing the automation tasks defined in the build pipeline, such as compiling code, running tests, packaging applications, and deploying software. It is an essential

component of **Continuous Integration (CI)** and **Continuous Delivery (CD)** processes, enabling the automation of various stages in the software development lifecycle.

- **Containerization** is a type of virtualization that allows developers to package an application with all its dependencies and ship it as one unit. Containers are isolated from each other, so they can run on the same host without affecting one another. This makes them ideal for micro services, or small, modular applications that can be deployed independently.

There are two main types of containers:

-Linux containers These types of containers use features built into the Linux kernel to isolate processes from each other. Linux containers are more popular in DevOps because they're lighter weight and easier to work with.

-Windows containers This container type uses the Windows Server Container feature to isolate processes from each other.

- **Docker is** an open-source tool used for packaging applications and their dependencies into containers. It also provides a way to manage and orchestrate **containerized applications**, allowing users to easily deploy and manage their applications using Docker tools such as Docker Swarm.
  In the context of **DevOps**, Docker plays a crucial role in supporting the principles of Continuous Integration (CI), Continuous Delivery (CD), automation, and infrastructure as code.

- **Kubernetes** is often abbreviated as **K8s**, is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications.

- **Jargon** refers to the specialized terminology and language that is commonly used within the field to describe processes, tools, methodologies, and concepts specific to DevOps practices.
- **Dependence** in DevOps, **dependencies** refer to any external software, libraries, services, or resources that an application or system relies on to function correctly.

Types of Dependencies in DevOps

1. **Software Dependencies**:
   - These are external libraries or packages that an application needs to run. For example, a Python application may depend on specific versions of libraries like requests or numPy.
2. **Infrastructure Dependencies**:
   - These include the underlying infrastructure needed by an application, such as databases, storage systems, networks, and cloud services. For instance, an application might depend on a database like MySQL or an external service like AWS S3.
3. **Service Dependencies**:
   - Many modern applications, especially in microservices architectures, rely on other services or APIs to function. For example, a payment service might depend on external authentication or messaging services.
4. **Environmental Dependencies**:
   - These include configurations specific to the environment in which the application runs, such as environment variables, OS versions, or hardware-specific features.

✓ **Evolution of DevOps and its importance**

DevOps has transformed the software development and operations landscape by integrating development and operations teams to streamline software delivery. Here is a summary of how DevOps evolved and its significance:

1. Traditional Software Development (Pre-DevOps Era)

Waterfall Model: Software development followed a linear, siloed approach. Developers and operations teams worked independently, leading to inefficiencies.

Challenges: Long development cycles, delayed feedback, and deployment bottlenecks. Developers wrote code without considering how it would be maintained in production, while operations struggled with stability issues in deployments.

2. Agile Methodology

Agile Revolution (Early 2000s): Agile introduced faster, iterative development cycles, improving collaboration between development and business teams. However, operations remained separate, causing bottlenecks in deployment and infrastructure management.

Importance of agile revolution in DevOps:

DevOps and agile practices encourage teams to implement improvements, collaborate, and reduce bottlenecks. This allows both methodologies to speed up software development while maintaining quality.

Benefits of Agile:

- Satisfied customers. By involving customers in the development process, Agile teams keep them in the loop and show that they value their opinion.
- Improved quality.
- Adaptability.
- Predictability.
- Reduced risk.
- Better communication.

3. The Emergence of DevOps (Late 2000s)

Birth of DevOps (2009): The term "DevOps" was coined to address the gap between development and operations. The goal was to bring them together to improve collaboration, automate deployment, and reduce conflicts.

Importance of the Emergence of DevOps (Late 2000s)

- **Bridging the Gap Between Development and Operations**:

Collaboration: DevOps fostered collaboration between development and operations teams, breaking down silos. This integration led to a shared understanding of goals and responsibilities, reducing friction and misunderstandings.

- **Faster Delivery Cycles**:

Continuous Integration/Continuous Delivery (CI/CD): DevOps practices emphasized automation of the software delivery process, allowing for frequent releases and quicker time-to-market.

- **Improved Quality and Reliability**:

Automated Testing: DevOps introduced automated testing early in the development process, which helps catch bugs before they reach production. This leads to higher quality software and reduced rollback rates.

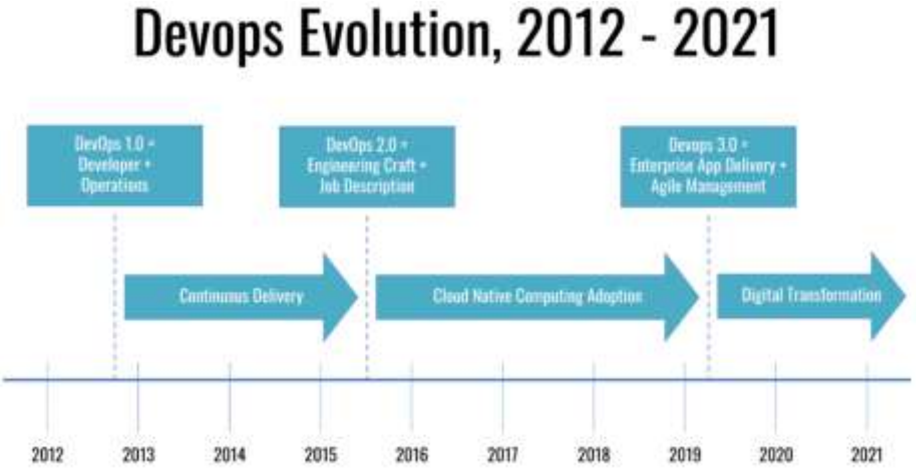
Stable Environments: Consistent environments (through containerization and infrastructure as code) reduced the discrepancies between development, testing, and production, minimizing deployment issues.

- **Increased Efficiency Through Automation**:

Automation of Repetitive Tasks: DevOps encouraged the automation of manual tasks such as builds, tests, and deployments, which reduced human error and saved time.

Resource Optimization: Automation enabled better utilization of resources, allowing teams to allocate more time to innovation and less time on repetitive tasks.

4. Modern DevOps (2010s and Beyond)


Devops Evolution, 2012 - 2021
DevOps 1.0 = Developer + Operations
DevOps 2.0 = Engineering Craft + Job Description
Devops 3.0 = Enterprise App Delivery + Agile Managment
Continuous Delivery
Cloud Native Computing Adoption
Digital Transformation
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

Importance of modern DevOps

- **Faster Software Delivery**

Continuous Integration (CI) / Continuous Delivery (CD): Modern DevOps automates the process of integrating and delivering code, allowing for faster, smaller, and more frequent releases. This enables companies to respond quickly to market demands and customer feedback.

Shorter Release Cycles: Frequent deployments reduce the time between development and production, helping organizations to stay competitive.

- **Increased Collaboration and Communication**

Breaking Down Silos: DevOps encourages collaboration between development, operations, and other teams (e.g., security). This shared responsibility fosters a culture of teamwork, improving problem-solving and decision-making.

Aligned Goals: By aligning development and operations with shared objectives, organizations achieve smoother workflows and reduce conflicts between teams.

- **Scalability and Flexibility**

Containerization (Docker): Containers allow applications to run consistently across multiple environments, simplifying the deployment process and enabling easy scaling of applications as needed.

- **Security Integration (DevSecOps)**

Security as a Core Part of DevOps: In Modern DevOps, security practices are integrated throughout the development lifecycle (DevSecOps). This ensures that security is built into the software from the start, reducing vulnerabilities and enhancing overall security.

Automated Security Checks: Security tools are automated to scan code, identify vulnerabilities, and ensure compliance without slowing down the development process.

✓ **Advantages and disadvantages of DevOps**

Advantages

- Faster development and deployment of applications.
- Faster response to the market changes to improve business growth.
- Business profit is increased as there is a decrease in software delivery time and transportation costs.
- Improves customer experience and satisfaction.
- Simplifies collaboration as all the tools are placed in the cloud for customers to access.
- Leads to better team engagement and productivity due to collective responsibility.

- ✓ **Disadvantages**

- Less availability of DevOps professionals.
- Infrastructure cost is high for setting by DevOps environment.
- Lack of DevOps knowledge can lead to problems in the continuous integration of automation projects

- ✓ **Description of DevOps principles**

1. Collaboration and Communication

- DevOps emphasizes breaking down silos between teams. Developers, operations, and other stakeholders work together closely throughout the software development lifecycle (SDLC).

2. Automation

- One of the core principles is to automate as much of the process as possible, from code integration, testing, deployment, to monitoring. Tools like Jenkins, Ansible, and Docker help automate manual tasks to increase efficiency and reduce errors.

3. Continuous Integration (CI) and Continuous Delivery (CD)

- Developers frequently merge their code changes into a central repository where automated builds and tests are run. This ensures code changes are tested early and often, reducing the risk of integration issues.

- CD ensures that code is always in a deployable state. It focuses on automating the delivery of code changes to production, or at least to a staging environment, ensuring frequent, reliable releases.

4. Infrastructure as Code (IaC)

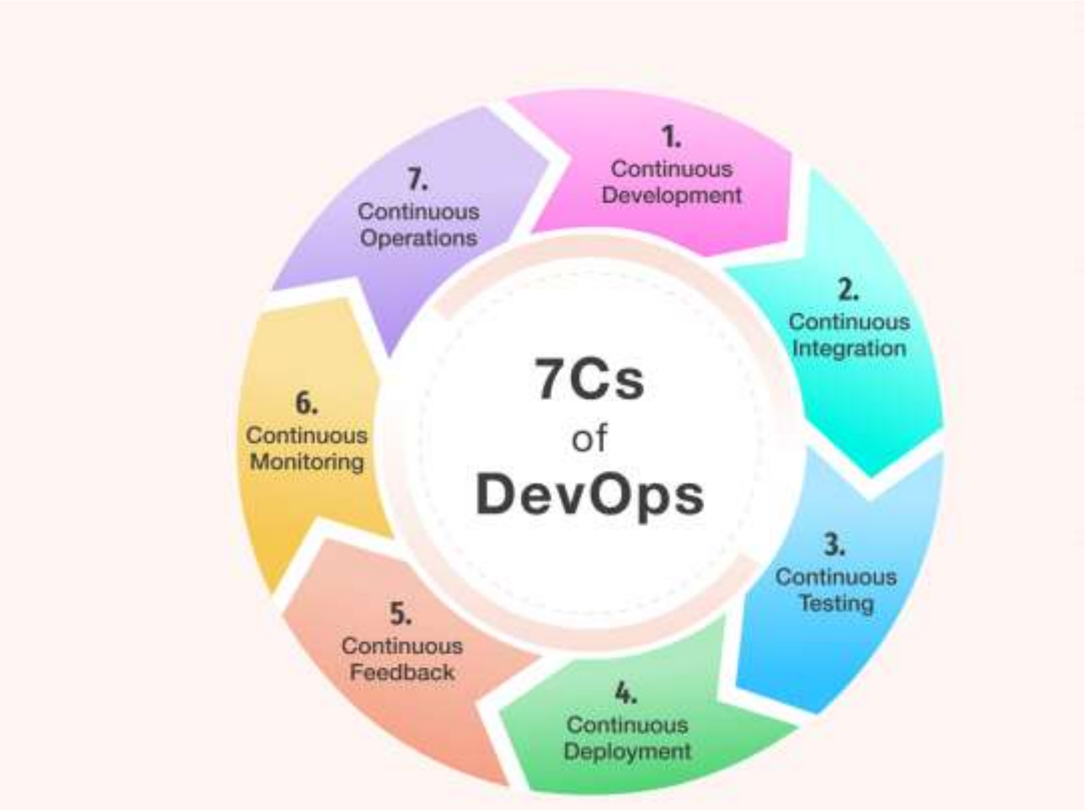
- This principle treats infrastructure (servers, networks, etc.) as code. Using tools like Terraform or AWS CloudFormation, teams can version control and automate the provisioning and management of infrastructure.

5. Monitoring and Logging

- Constant monitoring of applications and infrastructure is crucial to ensure high availability, reliability, and performance. Tools like Prometheus, Grafana, and ELK Stack (Elasticsearch, Logstash, and Kibana) allow for real-time monitoring and logging.

✓ **Description of DevOps lifecycle**

DevOps Lifecycle is a set of practices that automate the process of software development, testing, and deployment. It emphasizes collaboration between development and operations teams to deliver software faster and more reliably.



The key 7C phases in the DevOps lifecycle

- **Continuous Development:** This involves ongoing development of software features and bug fixes in small, incremental batches.
- **Continuous Integration:** This practice involves merging code changes from multiple developers into a shared repository on a frequent basis and running automated tests to ensure code quality.
- **Continuous Testing:** This involves running automated tests throughout the development process to identify and fix defects early.
- **Continuous Deployment:** This practice involves automatically deploying code changes to production environments after they pass automated tests.
- **Continuous Feedback:** This involves gathering feedback from users, stakeholders, and the development team to continuously improve the software.
- **Continuous Monitoring:** This involves monitoring the performance and health of applications in production environments to identify and address issues quickly.

- **Continuous Operations:** This involves managing and maintaining IT infrastructure and applications to ensure they are running smoothly and efficiently.

✓ **Description of DevOps Technologies and tools.**

DevOps, a combination of "development" and "operations," is a set of practices and tools that aim to bridge the gap between software development and IT operations teams. This collaboration enhances efficiency, communication, and the overall software delivery lifecycle.

Key DevOps Technologies

1. **Version Control Systems (VCS):**

   - **Git:** The most popular VCS, offering distributed version control, branching, and merging capabilities.
   - **SVN (Subversion):** A centralized VCS with a simpler structure, suitable for smaller teams.

2. **Continuous Integration (CI) Tools:**

   - **Jenkins:** A highly customizable open-source CI server that automates the build, test, and deployment processes.
   - **GitLab CI/CD:** A built-in CI/CD pipeline in GitLab, offering seamless integration with the Git repository.

3. **Continuous Delivery (CD) Tools:**

   - **Ansible:** A configuration management tool that automates infrastructure provisioning and application deployment.
   - **Chef:** A configuration management tool that uses a domain-specific language (DSL) to define infrastructure and application configurations.

4. **Containerization:**

   - **Docker:** The most widely used containerization platform, allowing applications to be packaged with their dependencies into containers.
   - **Kubernetes:** An open-source container orchestration platform that manages containerized applications across multiple hosts.

5. **Configuration Management Tools:**

    - **Terraform:** An infrastructure as code (IaC) tool that allows you to define and manage infrastructure resources using a declarative language.
    - **CloudFormation:** AWS's IaC tool that uses templates to define and manage cloud resources.

6. **Monitoring and Logging Tools:**

    - **Prometheus:** An open-source monitoring system that collects metrics from various sources.
    - **Grafana:** A visualization platform that can be used to create dashboards and graphs from Prometheus data.
    - **ELK Stack (Elasticsearch, Logstash, Kibana):** A popular logging and analytics stack for collecting, processing, and visualizing log data.

7. **Collaboration and Communication Tools:**

    - **Slack:** A popular team messaging app that can be integrated with various DevOps tools.
    - **Jira:** A project management and issue tracking tool commonly used in DevOps teams.

Benefits of DevOps Technologies

- **Faster Time to Market:** By automating processes and reducing manual tasks, DevOps can accelerate software delivery.
- **Improved Quality:** Automated testing and continuous monitoring help identify and fix issues early in the development process.
- **Increased Reliability:** DevOps practices focus on building reliable and scalable systems.
- **Enhanced Collaboration:** DevOps fosters collaboration between development and operations teams, leading to better communication and problem-solving.

Practical Activity 2.1.2: Selection of technology and tools

Task:

1: Read the key reading 2.1.2

2: Refer to the theoretical activity 2.1.2 you are asked to select deployment technology and tools.

3: Present your work to trainer and whole class

4: Ask any clarification to the trainer or colleagues

Key readings 2.1.2 Selection of technology and tools

Step 1: analyse to available technologies and tools based on features

I. vision control System

a) Git tool

1. Features
   - ❖ Distributed System
   - ❖ Staging Area
   - ❖ Lightweight and Fast
   - ❖ Commit History
   - ❖ Collaboration Support
2. Required for installation
   - ❖ Local Repository
   - ❖ Remote Repository (e.g., GitHub, GitLab)
   - ❖ Text Editor (for commits)
   - ❖ Disk Space depend on the following factors:

-**Small Repositories** (code only, minimal history): A few MB (1–100 MB). -**Medium Repositories** (moderate history and some larger files): 100 MB–1 GB. -**Large Repositories** (extensive history, large binaries, or media): 1 GB and above.

b) Subversion Tool

1. Features
   - ❖ Atomic Commits
   - ❖ Versioning of Directories
   - ❖ Efficient Handling of Binary Files
   - ❖ File Locking Mechanism

- ❖ Branching and Tagging
- ❖ Merge Tracking
- ❖ Conflict Resolution

2. Required for installation

- ❖ Server (Apache HTTPD or SVN Server)
- ❖ Client Software (TortoiseSVN, CLI tools)
- ❖ Repository Storage (Berkley DB or FSFS)
- ❖ Network Protocols (HTTP, HTTPS, SVN, SVN+SSH)
- ❖ Authentication Mechanism (LDAP, Basic Auth, SSL/TLS)

II. Containerization

a) Docker tools

1. Features
   - ❖ Lightweight Containers
   - ❖ Isolation
   - ❖ Portability
   - ❖ Layered Filesystem (UnionFS)
   - ❖ Container Orchestration
   - ❖ Version Control and Tagging
   - ❖ Container Registry (Docker Hub)
2. Required tools
   - ❖ Host Operating System (Linux, Windows, macOS)
   - ❖ Docker Engine Installation
   - ❖ Hardware Requirements (CPU=(single-core,64-bit architecture), RAM=(**2 GB** of system memory)
   - ❖ Root/Administrative Privileges
   - ❖ Networking Configuration
   - ❖ Container Images

b) Kubernetes

1. Features
   - Automated Deployment
   - Scaling (Horizontal Scaling)
   - Self-Healing
   - Service Discovery and Load Balancing
   - Automated Rollouts and Rollback
2. Required tools
   - YAML or JSON Configuration Files
   - Nodes (Master and Worker Nodes)
   - Cluster Networking (e.g., CNI)
   - ETCD for Cluster State Management

III. CD

a) Ansible

1. Features

- Agentless Architecture
- Declarative Playbooks (YAML)
- Inventory Management
- Idempotency
- Configuration Management

2. Required tools

- Ansible Installed on Control Node
- Python Installed on Target Machines
- YAML for Playbooks
- SSH Access to Target Nodes
- Access to CI Tool (e.g., Jenkins, GitLab CI)

b) Chef

1. Features
   - Automated Deployment

- Pipeline Orchestration
- Version Control Integration
- Rollback Capabilities
- Environment Configuration Management

2. Required tools

- Chief Installed on Deployment Nodes
- Access to Artifact Repository
- Environment-Specific Configuration
- Pipeline Definition Files
- Integration with Source Control (Git)

IV. CI

a) **GitLab CI/CD**

1. Features

- Pipeline Automation
- Auto DevOps
- Parallel Execution
- Pipeline as Code (.gitlab-ci.yml)
- Job Artifacts

2. Required tools

- GitLab Runner Installed
- .gitlab-ci.yml Configuration File
- Git Repository
- Access to CI/CD Variables
- Build Server or Cloud Infrastructure

b)Jenkins

1. Features

- Pipeline as Code
- Plugin Extensibility
- Distributed Builds
- Parallel Execution

- Declarative and Scripted Pipelines

2. Required for installation

- Jenkins Installed on Server
- Java Runtime Environment (JRE)
- Access to Source Code Repositories
- Network Configuration for Agents and Nodes
- Sufficient System Resources (CPU, Memory, Disk)
- Build Tools (e.g., Maven, Gradle)

V. Configuration Management

a)Terraform

HashiCorp Terraform

1. Features
   - Infrastructure as Code (IaC)
   - Declarative Configuration Language (HCL)
   - State Management
   - Immutable Infrastructure
   - Multi-Cloud Support
   - Modular Configuration
2. Required tools
   - Terraform Installed on Local/System
   - Provider Credentials for Cloud Platforms
   - HCL-based Configuration Files
   - Backend for State Storage
   - Version Control System (e.g., Git)

b)cloudformation

CloudFormation

3. Features
   - Infrastructure as Code (IaC)
   - Declarative Templates (YAML/JSON)
   - Stack Management
   - Automatic Rollback

- ❖ Stack Updates and Drift Detection
- ❖ Cross-Stack References

4. Required for installation
    - ❖ WS Account
    - ❖ CloudFormation Templates (YAML/JSON)
    - ❖ AWS CLI/Management Console Access
    - ❖ IAM Permissions
    - ❖ Defined AWS Resources

Step 2: compare available technologies and tools based criteria

Step 3: Select appropriate technologies and tools based on your needs

Practical Activity 2.1.3: Installing of Dependency

Task:

1: Read the key readings 2.1.3

2: Referring to the Key readings 2.1.3**,** you are requested to go to the computer lab to install Node.js and react.js dependencies.

3: Present your works to the trainer and classmate.

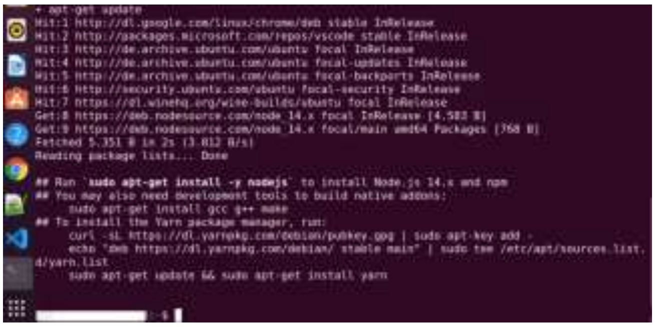
4: Ask a question for clarification where necessary.

Key readings 2.1.3 Installation of Dependency

Steps to install node js and react js dependencies.

Step 1: Select Framework based on software development Technology.
Example: Node js, Django, Laralavel Backend and Vue js, React Js in frontend
Step 2: installation of Framework.
download node js using CLI

```
→sudo apt-get update
→sudo apt-get install –y nodejs
```



```
+ apt-get update
Hit:1 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:2 http://packages.microsoft.com/repos/vscode stable InRelease
Hit:3 http://de.archive.ubuntu.com/ubuntu focal InRelease
Hit:4 http://de.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:5 http://de.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:6 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:7 https://dl.winehq.org/wine-builds/ubuntu focal InRelease
Get:8 https://deb.nodesource.com/node_14.x focal InRelease [4,583 B]
Get:9 https://deb.nodesource.com/node_14.x focal/main amd64 Packages [768 B]
Fetched 5,351 B in 2s (3,812 B/s)
Reading package lists... Done

## Run `sudo apt-get install -y nodejs` to install Node.js 14.x and npm
## You may also need development tools to build native addons:
     sudo apt-get install gcc g++ make
## To install the Yarn package manager, run:
     curl -sL https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
     echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.
d/yarn.list
     sudo apt-get update && sudo apt-get install yarn

:~$
```

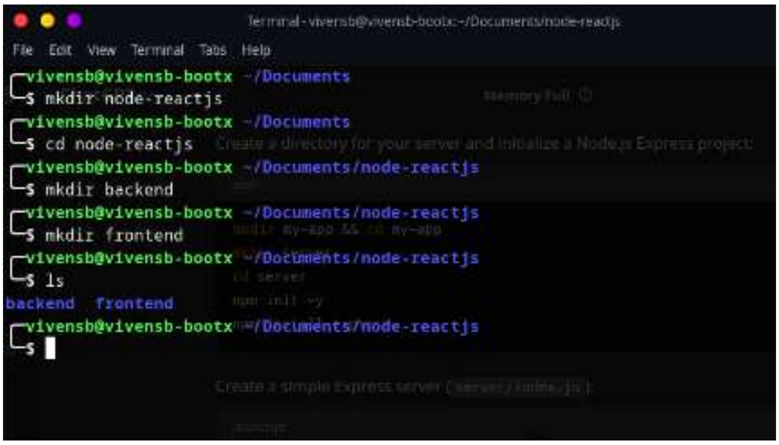
Step 3: Creation of folder used to create it owner dependencies for backend and frontend.

Create folder for backend that contain dependency

→mkdir node-reactjs

→cd node-reactjs

→mkdir backend && mkdir frontend



Terminal - vivensb@vivensb-bootx:~/Documents/node-reactjs

```
┌vivensb@vivensb-bootx ~/Documents
└$ mkdir node-reactjs
┌vivensb@vivensb-bootx ~/Documents
└$ cd node-reactjs
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ mkdir backend
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ mkdir frontend
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ ls
backend  frontend
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ ▌
```

Step 4: installation of Dependencies on backend(node js).

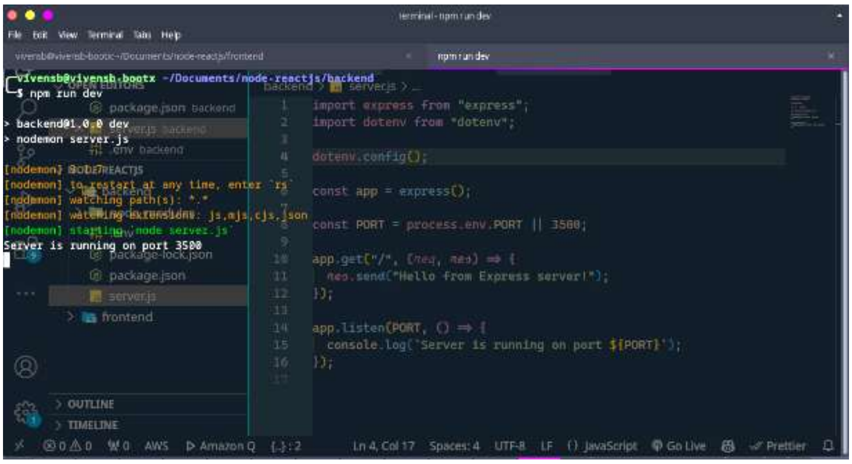
→npm init –y

→npm install express

→npm i dotenv

→npm i nodemon –D

→npm run dev



terminal - npm run dev
File Edit View Terminal Tabs Help
vivensb@vivensb-bootx:~/Documents/node-reactjs/frontend
npm run dev

```
┌vivensb@vivensb-bootx ~/Documents/node-reactjs/backend
└$ npm run dev

> backend@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port 3500
```

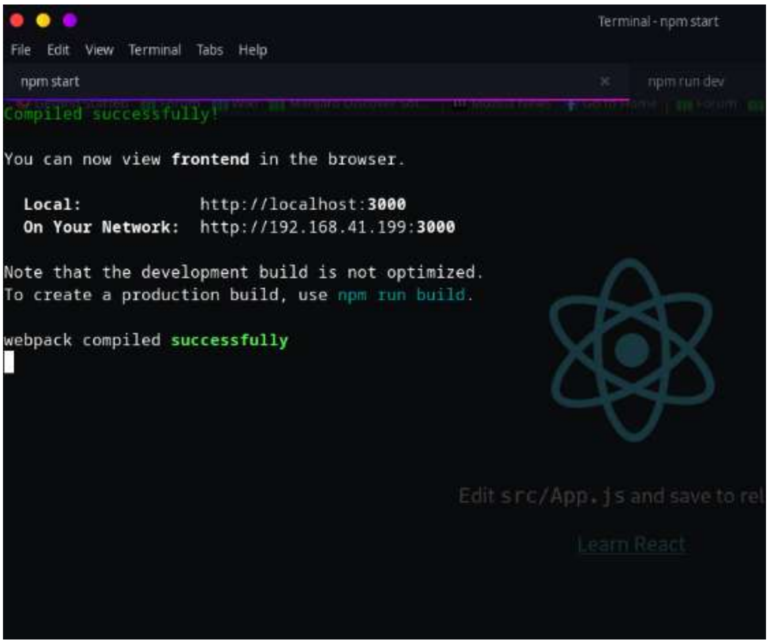
```
import express from "express";
import dotenv from "dotenv";

dotenv.config();

const app = express();

const PORT = process.env.PORT || 3500;

app.get("/", (req, res) => {
  res.send("Hello from Express server!");
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Step 4: installation of Dependencies on frontend(react js).

→cd frontend

→ npx create-react-app

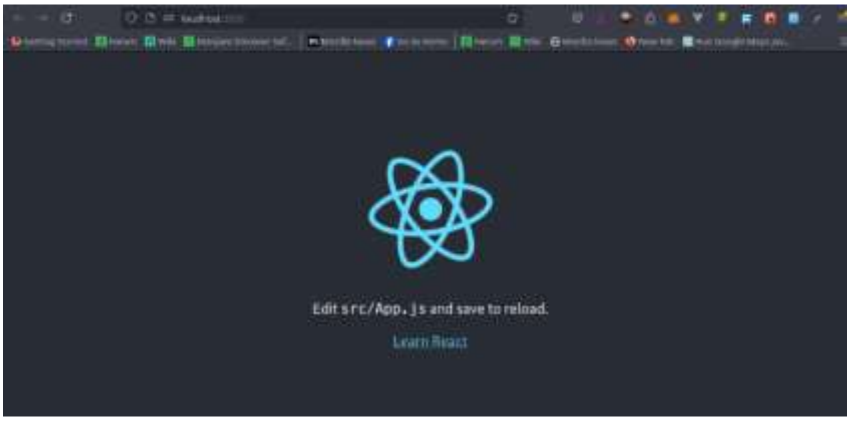
→npm start



Terminal - npm start
File Edit View Terminal Tabs Help
npm start
npm run dev
Edit src/App.js and save to rel
Learn React

```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.41.199:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Step 5: Run server of frontend (reactjs)


Edit src/App.js and save to reload.

Points to Remember

- Before preparing the deployment environment you must describe the tools and technologies.
- DevOps life cycle include the different phases such as Continuous development, Continuous integration, Continuous Testing, Continuous monitoring, Continuous feedback, Continuous deployment and Continuous operation
- While selecting the technologies and tools, you must understand these points: Version Control (Git, SVN) for managing code changes, CI/CD Tools (Jenkins, GitLab CI/CD) to automate testing and deployment, Containerization (Docker, Kubernetes) for packaging and orchestrating applications, Configuration Management (Ansible, Terraform) to automate infrastructure, Monitoring (Prometheus, Grafana) and Logging (ELK Stack) for system insights and Collaboration Tools (Slack, Jira) for communication and project management.
- To install dependencies in a project or system, the following requirements are typically needed: Package Manager, Dependency Manifest File, Internet Access, Correct Software Version, Sufficient Permissions, System Dependencies, Virtual Environment (Optional but recommended), Configuration Settings (Optional).

Application of learning 2.1.

Imagine that XYZ DevOps Company need an IT technician who can manage server hold node.js and react.js application. You are tasked to help the company select tools based on appropriate technology and install dependencies for both backend and frontend.

Indicative content 2.2: Use continuous delivery and configure containerization.

Duration: 15 hrs

Theoretical Activity 2.2.1: Description of Continuous Delivery tools and containerization.

Tasks:

Step 1: Answer the following questions:

i. Discuss on deployment orchestration and CI server

ii. Describe the steps performed on the following CI:

a) Server configuration

b) Set up Automated build and testing

c) Check code Quality

d) Artefact management

e) Integration with version control

iii. Describe Continuous Integration and Continuous Deployment Pipeline

iv. Explain the term "containerization" technology

Step 2: Write the Answers on paper, flipchart, blackboard or whiteboard

Step 3: Present their answers to the trainer and classmate

Step 4: Ask question for clarification if necessary

Step 5: Read the key readings 2.2.1

Key readings 2.2.1.: Use Continuous delivery

✓ **Selection CD tools**

- **Deployment orchestration**

DevOps orchestration tools are software that streamline and synchronize various DevOps processes. These tools automate tasks, manage complex workflows, and integrate different stages of the software development lifecycle, from coding and building to testing and deployment.

When choosing Deployment orchestration tools for a Node.js project, consider the following popular options based on your needs:

a. Docker

- **Overview**: A containerization platform that allows you to bundle your Node.js application and its dependencies into a single container.

- **Features**:
  - Portability across environments (local, staging, production).
  - Isolated, consistent environments for testing and deployment.
  - Works with Kubernetes for orchestration.

b. Kubernetes

- **Overview**: An open-source platform designed for automating deployment, scaling, and managing containerized applications.
- **Features**:
  - Self-healing, load balancing, and scaling of containers.
  - Integration with Docker.
  - Works well with micro services architectures.

c. AWS Elastic Beanstalk

- **Overview**: A fully managed service that handles deployment, scaling, and monitoring for your applications.
- **Features**:
  - Simple orchestration with minimal configuration.
  - Automatically handles capacity provisioning, load balancing, and health monitoring.
  - Good for small- to medium-sized Node.js apps.

d. Google Kubernetes Engine (GKE)

- **Overview**: A managed Kubernetes service on Google Cloud Platform.
- **Features**:
  - Managed Kubernetes environment.
  - Scalability and reliability for large projects.
  - Integration with Google Cloud services.

e. AWS Lambda

- **Overview**: A serverless platform that automatically manages the infrastructure to run your code.
- **Features**:
  - No need to manage servers.
  - Pay-per-use model (only for the resources your Node.js application consumes).

  - Ideal for small applications or microservices.

- **CI server**

When selecting a CI (Continuous Integration) server for your Node.js project, you have several great options, each with its strengths depending on your project needs, team size, and infrastructure. Here are some popular CI servers you can consider:

1. GitHub Actions

- **Pros**:
  - Deep integration with GitHub repositories.
  - Easy to set up for Node.js projects with pre-built templates.
  - Matrix builds, scheduled workflows, and customizable actions.
  - Free tier available for public repositories.
- **Cons**:
  - Limited free usage for private repositories.
  - Less flexible than self-hosted CI tools for complex pipelines.

2. Jenkins

- **Pros**:
  - Extremely customizable and open-source.
  - Large ecosystem of plugins, including ones specific to Node.js.
  - Can be hosted on your own servers, giving more control.
  - Flexible and works for both small and large projects.
- **Cons**:
  - Steeper learning curve and more time-consuming to set up.
  - Requires more maintenance for self-hosted instances.

3. CircleCI

- **Pros**:
  - Easy integration with GitHub, Bitbucket, and Docker.
  - Pre-built Docker images for Node.js projects.
  - Easy to scale and fast build times.
  - Great insights and reports.
- **Cons**:
  - Limited free tier, can get expensive for larger teams.
  - Steep learning curve for complex workflows.

4. Travis CI

- **Pros**:
  - Simple setup, especially with GitHub.
  - Free for open-source projects.
  - Node.js support out of the box.
  - Excellent for small to medium projects.
- **Cons**:
  - Private repository builds are limited and may require a paid plan.
  - Fewer customization options compared to Jenkins or GitLab CI.

5. GitLab CI/CD

- **Pros**:
  - Built-in with GitLab repositories.
  - Powerful and flexible pipelines.
  - Free for most GitLab.com users, including private repositories.
  - Self-hosted version available.
- **Cons**:
  - Hosting it yourself can be resource-intensive.
  - Steeper learning curve for beginners.

6. Bitbucket Pipelines

- **Pros**:
  - Tight integration with Bitbucket.
  - Easy to configure and get started with YAML files.
  - Built-in support for Docker containers and Node.js.
  - Simple pricing model with a free tier.
- **Cons**:
  - Limited to Bitbucket users.
  - Not as feature-rich as Jenkins or GitLab CI.

7. Azure Pipelines

- **Pros**:
  - Cross-platform support (Windows, macOS, Linux).
  - Integrates well with GitHub and Azure DevOps.
  - Highly scalable and customizable.
  - Free tier available for small teams.

- **Cons**:
  - Azure-specific integrations may not be useful for non-Azure users.
  - Can be complex to set up advanced workflows.

8. Drone CI

- **Pros**:
  - Lightweight and easy to self-host.
  - Works well with Docker and containerized applications.
  - Supports Node.js and many other languages out of the box.
  - Simple and clean UI.
- **Cons**:
  - Fewer plugins and integrations than Jenkins or CircleCI.
  - Can be harder to scale for larger teams.

9. Bamboo (Atlassian)

- **Pros**:
  - Deep integration with other Atlassian tools (JIRA, Bitbucket).
  - Highly customizable.
  - Good scalability for growing teams.
- **Cons**:
  - Paid product, no free tier.
  - Can be more complex to set up than cloud CI options.

10. Buddy

- **Pros**:
  - Very easy to use, drag-and-drop interface.
  - Supports Docker and Kubernetes integrations.
  - Free tier available.
  - Great UI and dashboard.
- **Cons**:
  - Less flexible compared to Jenkins or GitLab CI.
  - Can be limiting for highly complex workflow

✓ **Performing Continuous integration (CI)**

- **Configure server**

Server configuration refers to the process of setting up a server to host and manage applications. The configuration includes installing necessary dependencies, setting environment variables, managing network protocols, and ensuring security. Well-configured servers provide the backbone for the reliability and security of applications.

- **Set up Automated Builds**

Automated builds are part of a CI/CD pipeline where the code is compiled, tested, and prepared for deployment automatically when changes are pushed to the source code repository. This automation speeds up the software development cycle while ensuring high-quality code.

- **Automated Testing: Ensuring Code Quality and Reliability**

Automated testing is a key component of any CI/CD pipeline, ensuring that code changes do not introduce errors and that functionality remains intact as the software evolves.

Test Automation Frameworks: Tools like Jest, Mocha, or JUnit (depending on the programming language) can be integrated into the pipeline to run unit, integration, and end-to-end tests. These tests are triggered every time code is pushed or a build is triggered, catching bugs early.

Testing practices typically involve the following stages:

- **Unit testing**: validates individual units of code, such as a function, so it works as expected
- **Integration testing**: ensures several pieces of code can work together without unintended consequences
- **End-to-end testing**: validates that the application meets the user’s expectations
- **Exploratory testing**: takes an unstructured approach to reviewing numerous areas of an application from the user perspective, to uncover functional or visual issues
- **Code Quality Checks: Ensuring Maintainability and Best Practices**

Ensuring code quality is essential for maintainable, scalable, and high-performance software. Automated tools can check for style consistency, adherence to best practices, and the presence of potential bugs.

- **Artifact Management: Versioning and Storing Build Outputs**

Artifacts are the outputs of the build process—whether binaries, container images, or packaged code ready for deployment. Effective management of these artifacts is critical for ensuring traceability, rollback capability, and consistency across environments.

- **Artifact Repositories**: Tools like JFrog Artifactory, Nexus Repository, or AWS S3 are used to store and version artifacts. By keeping a history of all artifacts, teams can easily roll back to previous versions if needed, ensuring stability in production.

- **Integration with Version Control: Foundation for Collaboration and Automation**

Version control systems (VCS) like Git (e.g., GitHub, GitLab, Bitbucket) are the backbone of modern collaborative software development. They enable teams to work on multiple features simultaneously and integrate their work in an organized manner.

- **Automated Builds Triggered by VCS**: Integration between the CI system and version control ensures that every commit or pull request automatically triggers an automated build and testing process. This guarantees that code changes are continuously validated.
- **Pull Requests and Code Reviews**: Version control systems allow for code reviews via pull requests, a crucial step in ensuring code quality. Automated testing and linting tools can provide real-time feedback on code during the review process.
- **Branching Strategies**: Popular strategies such as GitFlow, trunk-based development, or feature branching help teams organize code in ways that support continuous integration. Version control integration ensures that different branches are built, tested, and merged efficiently.

- **The Interconnected Workflow of CI/CD: A Holistic View**

All the components discussed—automated builds, automated testing, code quality checks, artifact management, and version control integration—come together to form a cohesive CI/CD pipeline. This interconnected workflow ensures that every change is thoroughly tested, properly packaged, and consistently deployed across environments.

A CI pipeline automates the process of building, testing, and validating new code whenever a developer commits changes to the shared repository. This process is key to preventing integration issues and maintaining a stable codebase.

- **Configure CI pipeline**

A CI pipeline consists of a sequence of steps that automatically run whenever code is pushed to a version control system like Git. These steps typically include code compilation, unit testing, code quality checks, and artifact creation.

- **Goal**: The main objective of a CI pipeline is to detect and resolve integration issues as early as possible, improving the overall quality of the software and accelerating delivery.

Stages of a CI pipeline

1. Build

This is the stage at which an application is compiled. In a CI pipeline, automation is used to trigger a build cycle once a developer commits code changes to the main repository. From here, a CI tool will query a project's build tools to begin compiling the application in a clean staging environment.

2. Test

After an application is compiled, a series of tests are run to ensure the code works and there are no bugs or critical issues. Organizations will configure their CI tool to automate these tests to begin once the application compiles. Unit tests, which are a way of testing small units of code, are a popular type of test at this stage due to the ease it takes to write them and their low cost to run and maintain. The goal at this stage of the CI pipeline is to ensure that the codebase passes inspection.

3. Package

Next, an organization will begin packaging its code. In a CI pipeline, these steps are automated and triggered once unit testing is complete. The way each organization will package its code will depend on the programming language and production environment.

- ✓ **Continuous deployment (CD)**

Continuous Deployment (CD) Is a software development practice where code changes are automatically built, tested, and prepared for a release to production. It extends Continuous Integration (CI) by ensuring that the software is always in a deployable state, allowing for the frequent and reliable release of updates to end users.

The goal of Continuous Delivery is to ensure that code can be safely deployed to production at any time, reducing the time between writing new code and delivering it to users. While Continuous Integration focuses on integrating and testing code changes, CD takes this a step further by automating the entire release process.

How Continuous Delivery Works

- **Developer Makes Code Changes**: Developers make changes to the codebase and commit their changes to a shared repository.
- **Continuous Integration (CI)**: The CI system (e.g., Jenkins, GitHub Actions, GitLab CI) picks up the change, runs automated tests, and builds the application. If the code passes all the tests, the build is marked as successful.
- **Automated Testing & Staging**: Once the CI process completes, additional tests (e.g., integration, performance, security tests) are run. The application is also deployed to staging environments that mimic production, allowing for further testing under production-like conditions.
- **Deployment Pipeline**: The CD pipeline automates the preparation of the application for release, which could include packaging it into containers (e.g., Docker), creating artifacts(e.g., .jar files, .deb packages), or uploading assets to cloud platforms.
- **Approval for Production**: After passing all tests and deployment readiness checks, the system waits for approval (if configured). In the case of fully automated CD systems, the deployment may proceed directly to production.
- **Deploy to Production**: The application is released to production, ensuring the latest code is available to end users.

Benefits of Continuous Delivery

- **Faster Time to Market**
    - CD enables the delivery of features and updates much faster by reducing bottlenecks in the release process. Teams can release small, incremental changes frequently.
- **Improved Quality and Reliability**

- Continuous testing and deployment pipelines ensure that every change is thoroughly tested, reducing the chances of deploying broken code to production.

- **Reduced Risk of Deployment**
  - Since CD encourages small and frequent releases, each deployment is less risky and easier to manage. If something goes wrong, it's easier to roll back or fix.
- **Better Collaboration and Transparency**
  - With CD, the entire team can see the current state of the system at any time, improving collaboration between developers, testers, and operations teams.
- **Higher Developer Productivity**
  - By automating repetitive tasks like testing, packaging, and deployment, CD allows developers to focus more on writing code and building new features.

In the realm of DevOps and agile development, effective deployment strategies are crucial for maintaining software quality and ensuring rapid delivery. This theory explores how the integration of deployment scripts, Infrastructure as Code (IaC), orchestration tools, automated rollbacks, and continuous delivery (CD) pipelines can optimize the software deployment process.

- **Deployment Scripts**

Deployment scripts automate the process of deploying applications to various environments (development, staging, production). These scripts streamline repetitive tasks, reducing human error and ensuring consistency across deployments. By codifying the deployment process, teams can manage application versions efficiently and deploy changes with confidence.

- **Infrastructure as Code (IaC)**

IaC allows infrastructure to be provisioned and managed using code, enabling teams to maintain consistency and version control over their environments. Tools like Terraform and AWS CloudFormation enable the automated setup and configuration of servers, networks, and other infrastructure components, ensuring that environments are reproducible and scalable.

- **Deployment Orchestration Tools**

Deployment orchestration tools (e.g., Kubernetes, Ansible) automate and coordinate complex deployment processes across multiple servers and environments. They manage the deployment of applications, ensure proper resource allocation, and handle scaling, thereby simplifying the management of microservices and distributed systems.

- **Automated Rollback**

Automated rollback mechanisms are essential for maintaining application stability in case of deployment failures. By automatically reverting to the last known stable version, teams can minimize downtime and quickly recover from issues. This capability enhances reliability and boosts user confidence in new releases.

- **Configure Continuous Delivery Pipeline**

A well-configured CD pipeline integrates all deployment processes, from code commit to production. It automates testing, builds, and deployments, ensuring that each change is validated and deployed seamlessly. By incorporating automated quality checks and feedback loops, teams can deliver features faster and with fewer bugs.

Example Workflow of Continuous Delivery (CD)

- **Developer Commits Code** → Triggered CI pipeline → Code is tested (unit tests, etc.).
- **Build Artifacts** are created automatically (e.g., a Docker image).
- **Integration Tests** are run to ensure the code works with other components.
- The code is **deployed to a staging environment** for further testing (performance, security, etc.).
- Once approved, the code can be **released to production**.

Popular Tools for Continuous Delivery

- **Jenkins**: Open-source automation server for building, testing, and deploying code.
- **GitLab CI/CD**: Integrated tool for GitLab repositories with built-in CI/CD pipelines.
- **GitHub Actions**: CI/CD automation built into GitHub repositories.
- **CircleCI**: Fast, modern CI/CD pipeline system with Docker support.
- **Travis CI**: A cloud-based CI/CD service with easy GitHub integration

- **Configuration of container**
  - ✓ **Identification of Containerization Tools**

Containerization tools like Docker and Kubernetes revolutionize software deployment by encapsulating applications and their dependencies into lightweight, portable containers. These tools ensure consistency across environments, from development to production, by isolating applications from underlying infrastructure.

✓ **Setup Docker**

Docker is a leading containerization platform. Setting up Docker involves installing the Docker engine on the host system, enabling the creation and management of containers. Docker allows applications to run consistently, regardless of the environment.

✓ **Build Docker Images**

Docker images serve as the blueprint for containers. Building Docker images involves defining the application's environment, dependencies, and configurations using a Dockerfile. This ensures consistency and versioning for deployment.

✓ **Store Docker Images**

After building images, they are stored in repositories, such as Docker Hub or private registries. This enables easy sharing, versioning, and pulling of images for deployment across different environments.

✓ **Implement Continuous Integration (CI)**

CI integrates changes continuously into a shared codebase and automatically triggers builds and tests. Tools like Jenkins, GitHub Actions, and GitLab CI automate the process, ensuring every change is validated and prepared for deployment. By integrating Docker, CI pipelines can automatically build, test, and deploy Docker containers efficiently.

This theory emphasizes that combining containerization with continuous integration ensures scalable, reliable, and consistent software delivery across multiple environments.

Practical Activity 2.2.2: Configuring server based on software deployment.

Task:

1: Read key reading 2.2.2.

2: Referring to the key reading 2.2.2, you are asked to go to the computer Lab to Develop the frontend, backend scripts, test the scripts, configure containerisation, configure CI/CD using version control and configure server

3: Present your work to trainer and whole class

4: Ask question for more clarifications

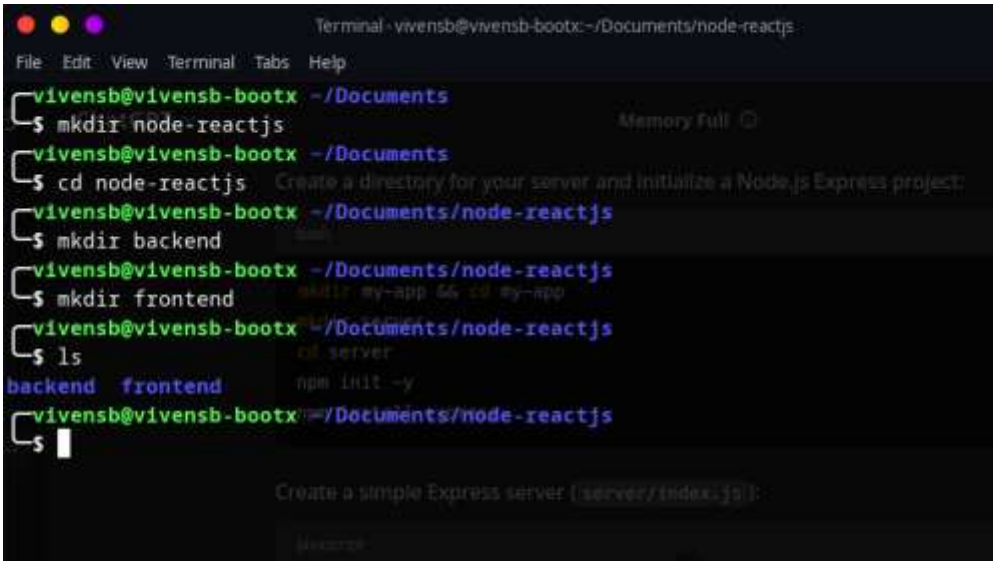
Key readings 2.2.2: Configuring server based on software deployment.

Steps for Configuring server based on software deployment are as follows:

i. Develop you script using Node.js for backend and react.js for frontend

Step 1: **Set Up the Project**

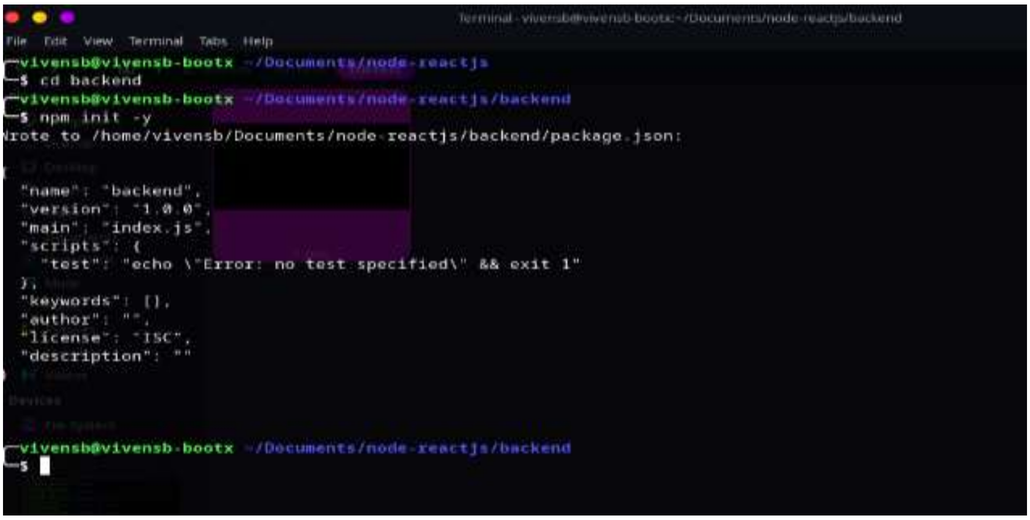
- **Check Node.js**: Ensure Node.js and npm are installed.
- **Initialize Backend (Node.js)**:
  - Create a new directory for the project:
    → **mkdir my-app && cd my-app**


Terminal - vivensb@vivensb-bootx:~/Documents/node-reactjs
File Edit View Terminal Tabs Help

```
┌vivensb@vivensb-bootx ~/Documents
└$ mkdir node-reactjs
┌vivensb@vivensb-bootx ~/Documents
└$ cd node-reactjs
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ mkdir backend
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ mkdir frontend
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ ls
backend  frontend
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ 
```

- Initialize Node.js:

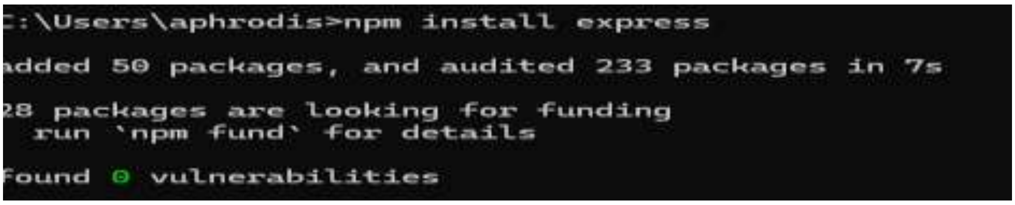
→ **npm init –y**


Terminal - vivensb@vivensb-bootx:~/Documents/node-reactjs/backend
File Edit View Terminal Tabs Help

```
┌vivensb@vivensb-bootx ~/Documents/node-reactjs
└$ cd backend
┌vivensb@vivensb-bootx ~/Documents/node-reactjs/backend
└$ npm init -y
Wrote to /home/vivensb/Documents/node-reactjs/backend/package.json:
{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

┌vivensb@vivensb-bootx ~/Documents/node-reactjs/backend
└$ 
```

- install necessary packages (e.g., Express.js):

→ **npm install express**

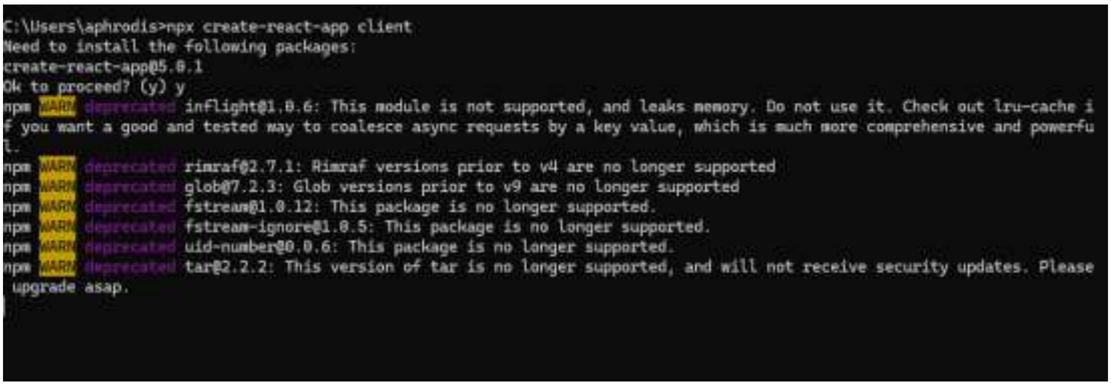


```
C:\Users\aphrodis>npm install express

added 50 packages, and audited 233 packages in 7s

28 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Step 2: Initialize Frontend (React.js):

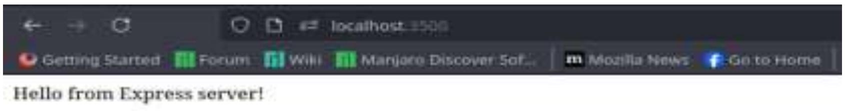
- Create a React app:

→npx create-react-app client



```
C:\Users\aphrodis>npx create-react-app client
Need to install the following packages:
create-react-app@5.0.1
Ok to proceed? (y) y
npm WARN deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use it. Check out lru-cache i
f you want a good and tested way to coalesce async requests by a key value, which is much more comprehensive and powerfu
l.
npm WARN deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm WARN deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm WARN deprecated fstream@1.0.12: This package is no longer supported.
npm WARN deprecated fstream-ignore@1.0.5: This package is no longer supported.
npm WARN deprecated uid-number@0.0.6: This package is no longer supported.
npm WARN deprecated tar@2.2.2: This version of tar is no longer supported, and will not receive security updates. Please
 upgrade asap.
```

i. Testing the developed script for check code quality

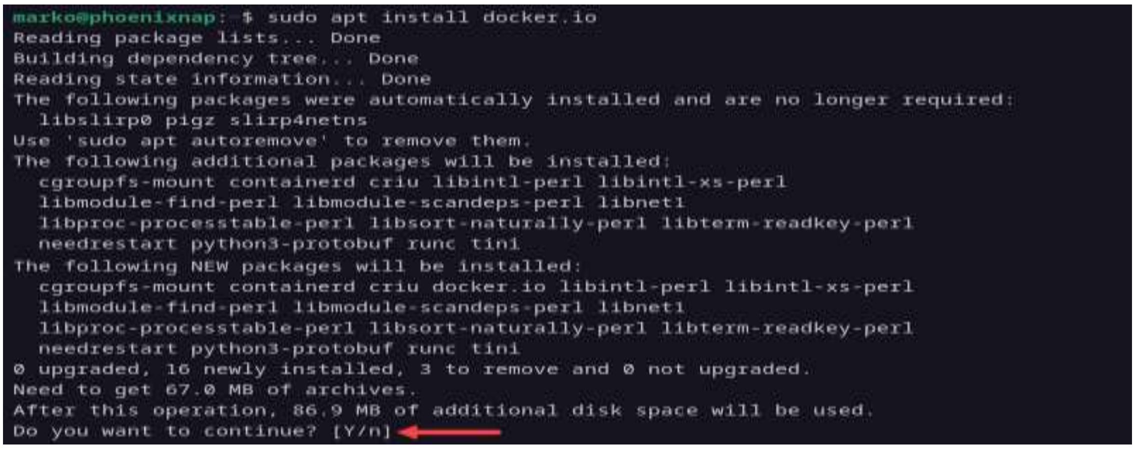


Hello from Express server!

ii. Configure containerisation technology with appropriate tool

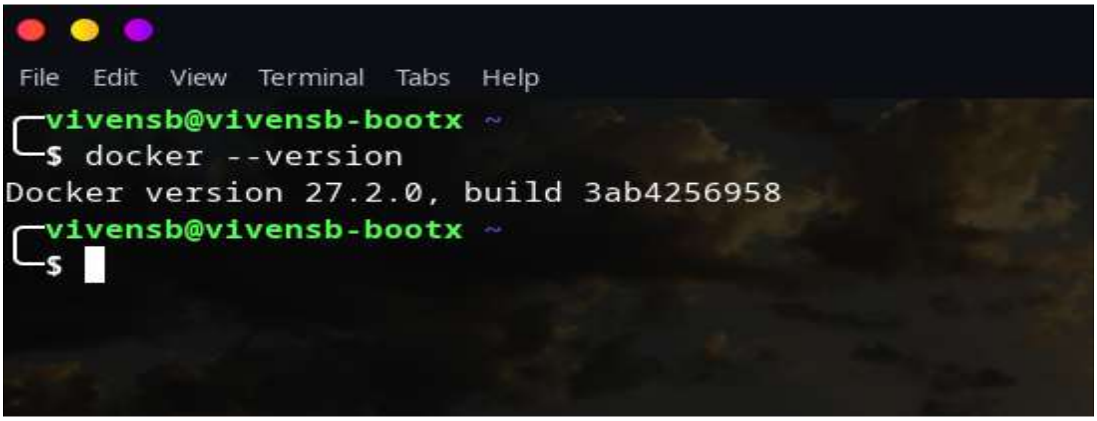
- **Tool**: Docker
- **Configuration Steps**:

1. **Install Docker**: Install Docker on your local machine or server



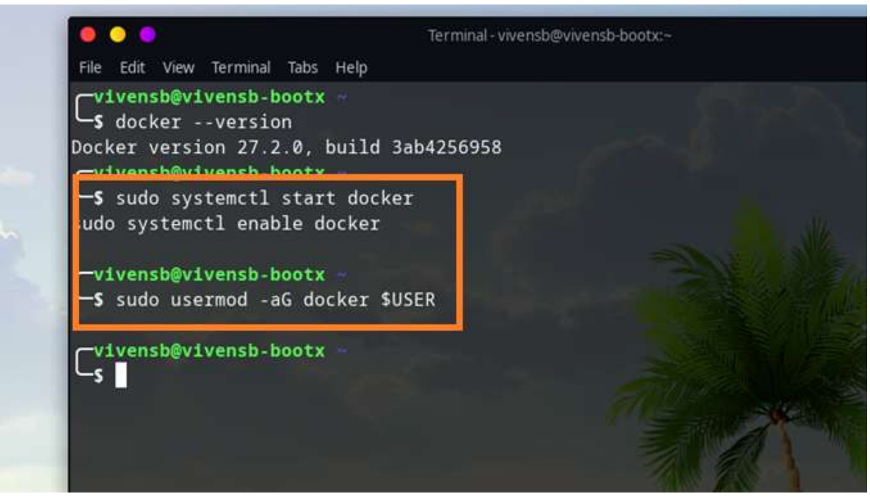
```
marko@phoenixnap:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libslirp0 pigz slirp4netns
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  cgroupfs-mount containerd criu libintl-perl libintl-xs-perl
  libmodule-find-perl libmodule-scandeps-perl libnetl
  libproc-processtable-perl libsort-naturally-perl libterm-readkey-perl
  needrestart python3-protobuf runc tini
The following NEW packages will be installed:
  cgroupfs-mount containerd criu docker.io libintl-perl libintl-xs-perl
  libmodule-find-perl libmodule-scandeps-perl libnetl
  libproc-processtable-perl libsort-naturally-perl libterm-readkey-perl
  needrestart python3-protobuf runc tini
0 upgraded, 16 newly installed, 3 to remove and 0 not upgraded.
Need to get 67.0 MB of archives.
After this operation, 86.9 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

2. check containerization version



```
┌vivensb@vivensb-bootx ~
└$ docker --version
Docker version 27.2.0, build 3ab4256958
┌vivensb@vivensb-bootx ~
└$ 
```

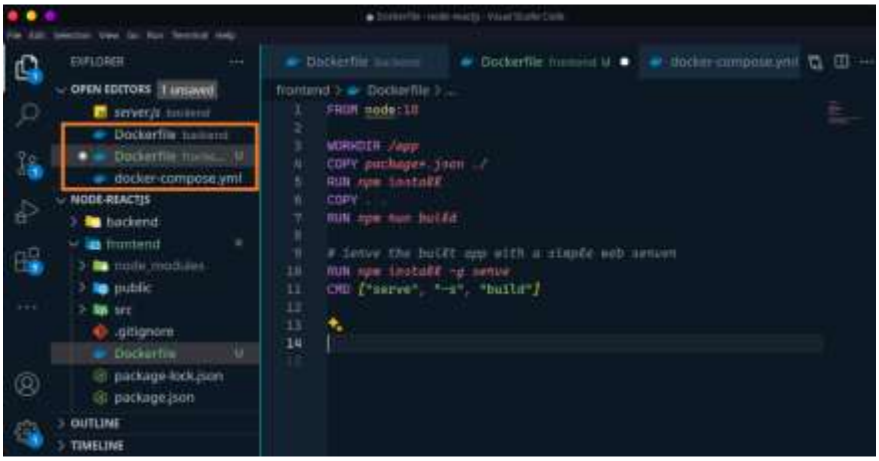
3. Restart Docker Engine



Terminal - vivensb@vivensb-bootx:~
File Edit View Terminal Tabs Help

```
┌vivensb@vivensb-bootx ~
└$ docker --version
Docker version 27.2.0, build 3ab4256958
┌vivensb@vivensb-bootx ~
└$ sudo systemctl start docker
udo systemctl enable docker

┌vivensb@vivensb-bootx ~
└$ sudo usermod -aG docker $USER

┌vivensb@vivensb-bootx ~
└$ 
```

2. **Create a Docker file and Docker image**: Define how to build your container.



```
FROM node:18

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# Serve the built app with a simple web server
RUN npm install -g serve
CMD ["serve", "-s", "build"]
```

1. Dockerfile for Node.js (Backend)

The Node.js Dockerfile will handle the server-side API logic and any interaction with databases or other services.

Same information Docker file from backend must contain the following:

```
FROM node:18

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .

EXPOSE 3500
CMD ["node", "server.js"]
```

2. Dockerfile for React.js (Frontend)

The React.js Dockerfile will handle building the frontend assets, which can later be served by a web server (e.g., NGINX) or directly as static files.

```
FROM node:18

WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
```

```
RUN npm run build

# Serve the built app with a simple web server
RUN npm install -g serve
CMD ["serve", "-s", "build"]
```

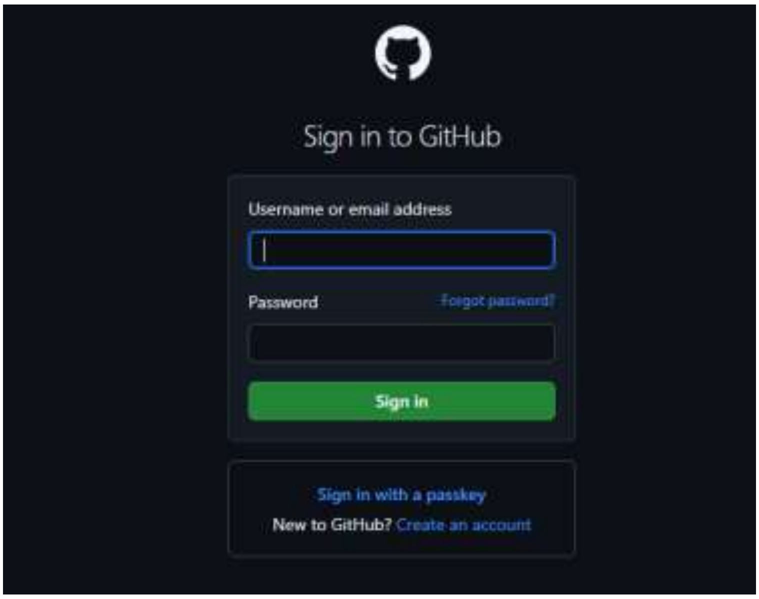
3. docker-compose.yml

A docker-compose.yml file defines the services, networks, and volumes that will be used by Docker to build and run multiple containers. It allows you to orchestrate your multi-container Docker applications (like having separate containers for your **Node.js** backend and **React.js** frontend) in an easy-to-manage way.

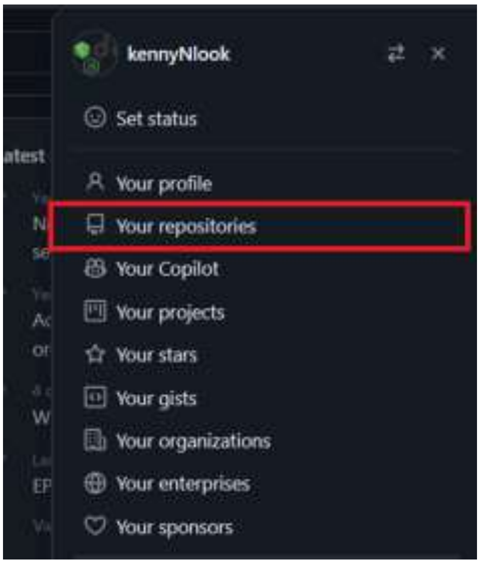
Let's break down the key components and the kind of information that the docker-compose.yml file typically contains:

```
version: '3'
services:
 server:
  build: ./backend
  ports:
   - "4000:4500"
 client:
  build: ./frontend
  ports:
   - "8080:3000"
```

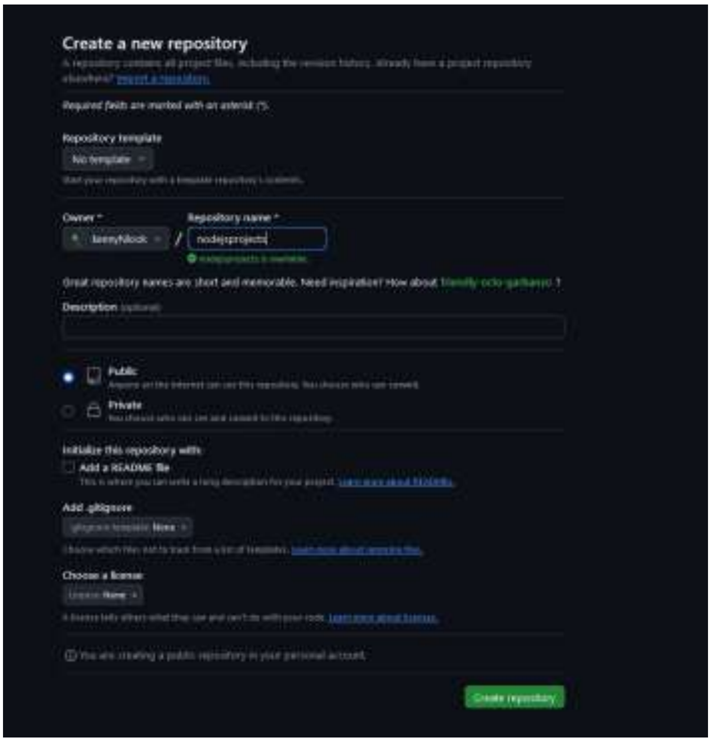
By using Version control technologies push the developed script to the remote repository
Step 1: creation of repository
1.open GitHub by using credentials



2.navigate to Create repository options



3.fill and choose required options



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * lannyNlock / Repository name * nodejsprojects

nodejsprojects is available.

Great repository names are short and memorable. Need inspiration? How about friendly-octo-garbanzo ?

Description (optional)

● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:

□ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

Choose a license

License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

Step 2: Initialize Git (if not already initialized)

If your project directory isn't already a Git repository, you need to initialize Git.

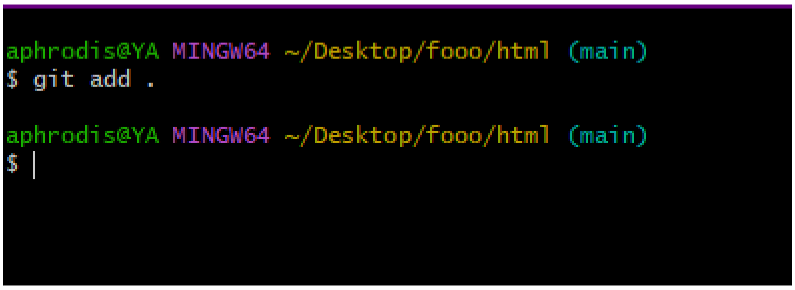
Open your terminal in the project directory and run:

→ git init

2. Add your files to staging

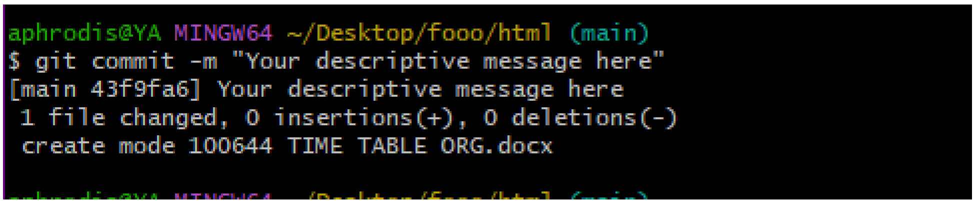
Add the files you want to push to the staging area.

→ git add .



```
aphrodis@YA MINGW64 ~/Desktop/fooo/html (main)
$ git add .

aphrodis@YA MINGW64 ~/Desktop/fooo/html (main)
$ |
```

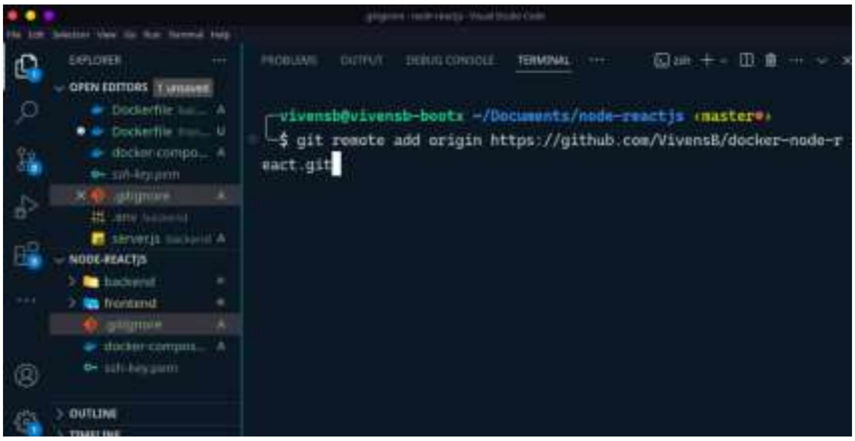
→ git commit -m "Your descriptive message here"



```
aphrodis@YA MINGW64 ~/Desktop/fooo/html (main)
$ git commit -m "Your descriptive message here"
[main 43f9fa6] Your descriptive message here
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 TIME TABLE ORG.docx

aphrodis@YA MINGW64 ~/Desktop/fooo/html (main)
```

→ git remote add origin https://github.com/kennyNlook/nodejsprojects.git

→ git push -u origin main



```
┌vivensb@vivensb-bootx ~/Documents/node-reactjs ‹master●›
└$ git remote add origin https://github.com/VivensB/docker-node-r
eact.git
```

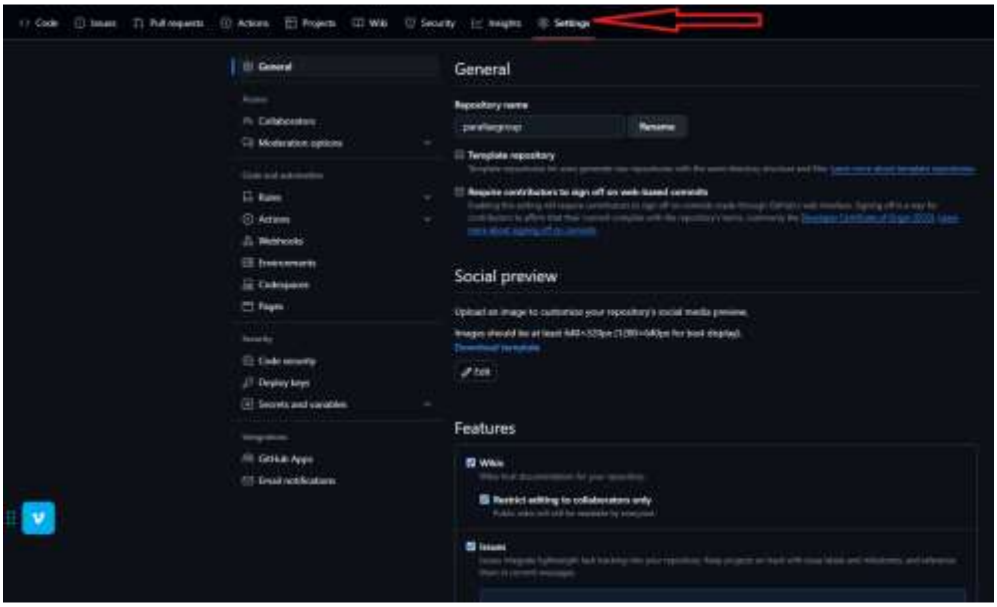
iv. Configure continuous integration and Continuous deployment

Step 1. Create a GitHub Repository

- If you don't have one yet, create a new repository on GitHub.

Step 2. Create a Workflow File

- Navigate to your repository on GitHub.
- Click on setting tab to set permissions



General

Repository name: parallelgroup

□ Template repository

□ Require contributors to sign off on web-based commits

Social preview

Upload an image to customize your repository's social media preview.

Images should be at least 640×320px (1280×640px for best display).

Download template

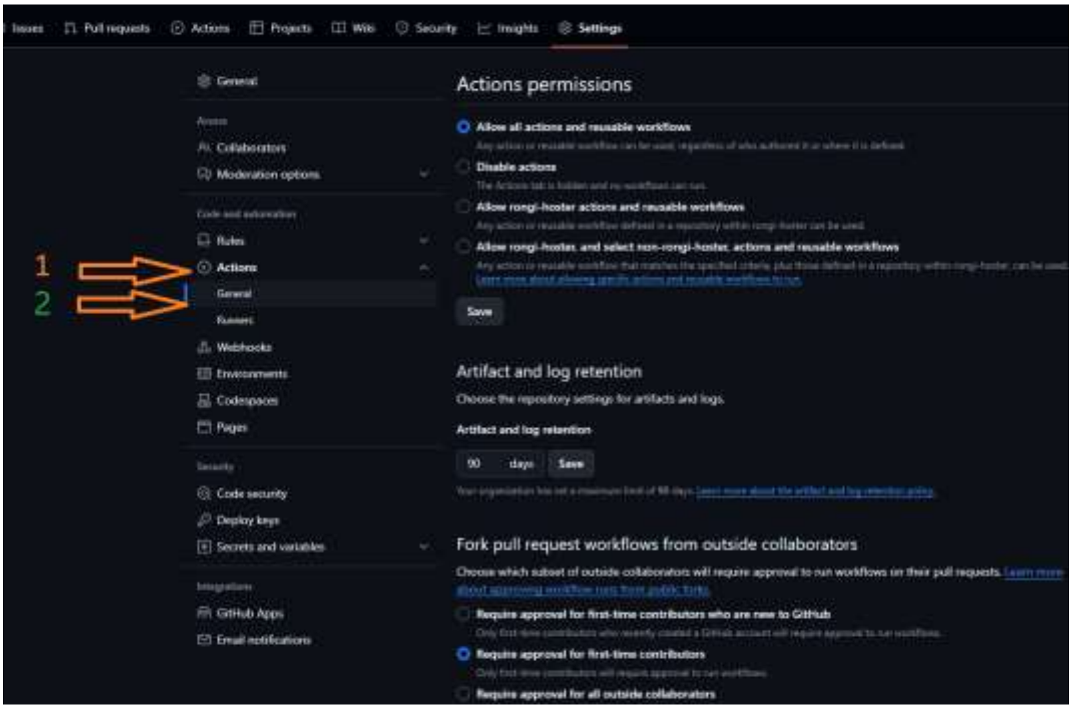
Features

☑ Wikis

☑ Restrict editing to collaborators only

☑ Issues

Click on action tab then choose general option



Actions permissions

● Allow all actions and reusable workflows

○ Disable actions

○ Allow rongi-hoster actions and reusable workflows

○ Allow rongi-hoster, and select non-rongi-hoster, actions and reusable workflows

Artifact and log retention

Choose the repository settings for artifacts and logs.

Artifact and log retention: 90 days

Fork pull request workflows from outside collaborators

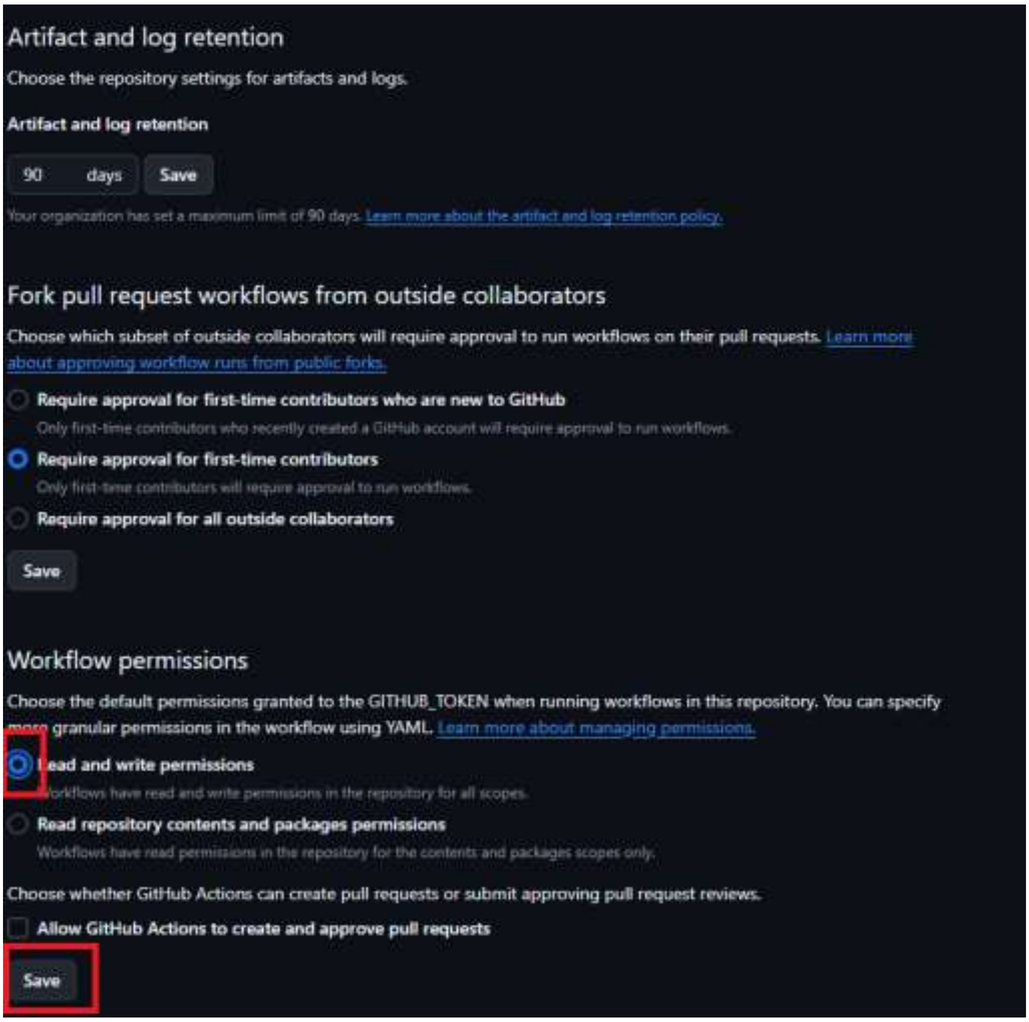
Choose which subset of outside collaborators will require approval to run workflows on their pull requests. Learn more about approving workflow runs from public forks.

○ Require approval for first-time contributors who are new to GitHub

● Require approval for first-time contributors

○ Require approval for all outside collaborators

Enable the permissions



Artifact and log retention

Choose the repository settings for artifacts and logs.

Artifact and log retention

90 days **Save**

Your organization has set a maximum limit of 90 days. Learn more about the artifact and log retention policy.

Fork pull request workflows from outside collaborators

Choose which subset of outside collaborators will require approval to run workflows on their pull requests. Learn more about approving workflow runs from public forks.

○ **Require approval for first-time contributors who are new to GitHub**
Only first-time contributors who recently created a GitHub account will require approval to run workflows.

● **Require approval for first-time contributors**
Only first-time contributors will require approval to run workflows.

○ **Require approval for all outside collaborators**

Save

Workflow permissions

Choose the default permissions granted to the GITHUB_TOKEN when running workflows in this repository. You can specify more granular permissions in the workflow using YAML. Learn more about managing permissions.

● **Read and write permissions**
Workflows have read and write permissions in the repository for all scopes.

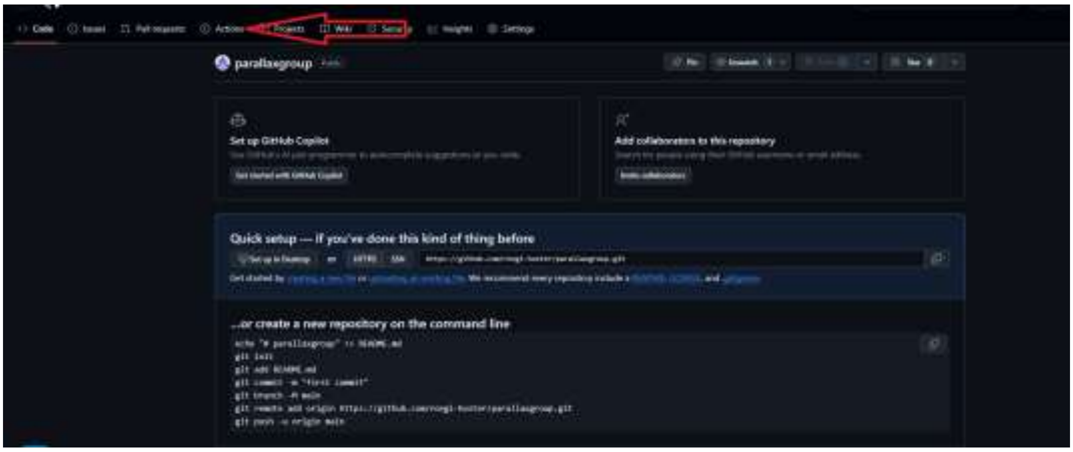
○ **Read repository contents and packages permissions**
Workflows have read permissions in the repository for the contents and packages scopes only.

Choose whether GitHub Actions can create pull requests or submit approving pull request reviews.

□ **Allow GitHub Actions to create and approve pull requests**

Save

- Go to the "Actions" tab.



Set up GitHub Copilot

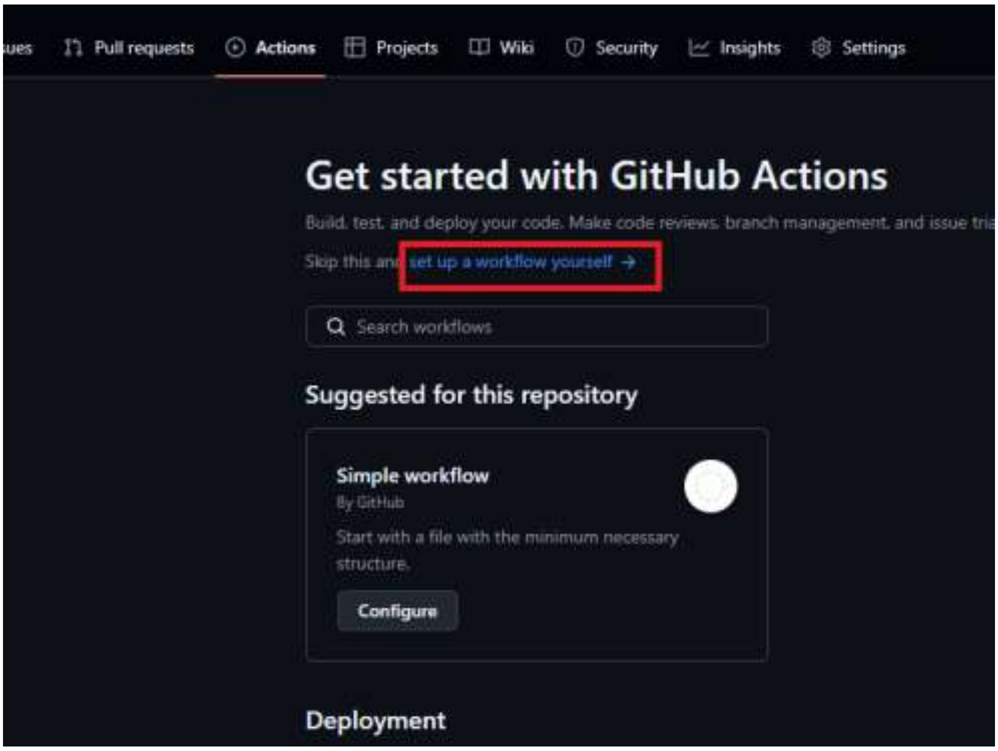
Add collaborators to this repository

Quick setup — if you've done this kind of thing before

...or create a new repository on the command line

```
echo "# parallaxgroup" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/[illegible]/parallaxgroup.git
git push -u origin main
```

- Click on "New workflow" to create a new workflow.



Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue tria

Skip this and set up a workflow yourself →

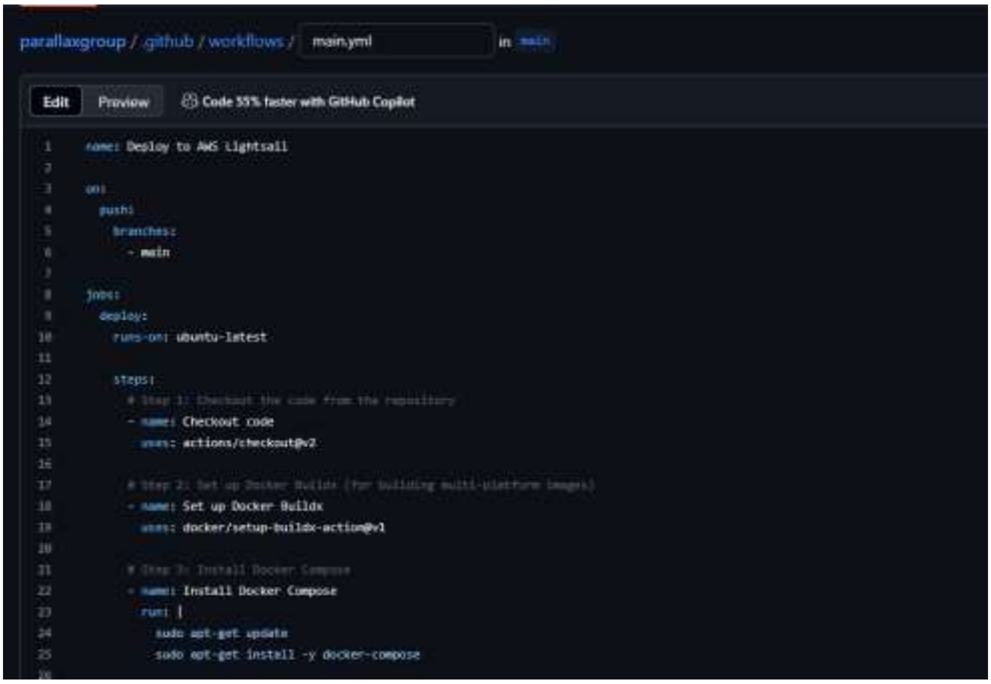
Suggested for this repository

Simple workflow
By GitHub
Start with a file with the minimum necessary structure.

Deployment

Step 3. Define the Workflow

- Create a new file in the .github/workflows/ directory (e.g., ci.yml).


parallaxgroup / .github / workflows / main.yml in main
Edit
Preview
Code 55% faster with GitHub Copilot

```
name: Deploy to AWS Lightsail

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      # Step 1: Checkout the code from the repository
      - name: Checkout code
        uses: actions/checkout@v2

      # Step 2: Set up Docker Buildx (for building multi-platform images)
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v1

      # Step 3: Install Docker Compose
      - name: Install Docker Compose
        run: |
          sudo apt-get update
          sudo apt-get install -y docker-compose
```

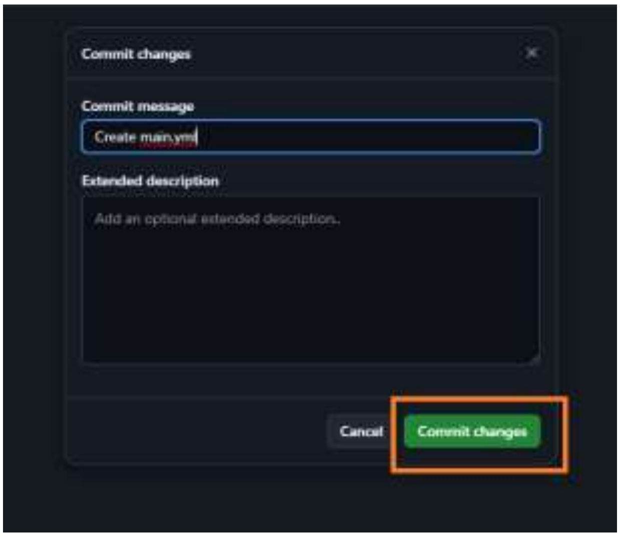
- Use the following basic structure:

This file must contain the following:

```
name: CI

on:
  push:
    branches:
      - main  # Change to your main branch if different
  pull_request:
    branches:
      - main  # Change to your main branch if different

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Check out code
        uses: actions/checkout@v2

      - name: Set up Node.js (or other environment)
        uses: actions/setup-node@v2
        with:
          node-version: '14'  # Specify your version

      - name: Install dependencies
        run: npm install  # Change command based on your project

      - name: Run tests
        run: npm test  # Change command based on your project
```

Step 4. Commit the Workflow

- Save the file and commit it to your repository. This will trigger the CI workflow.



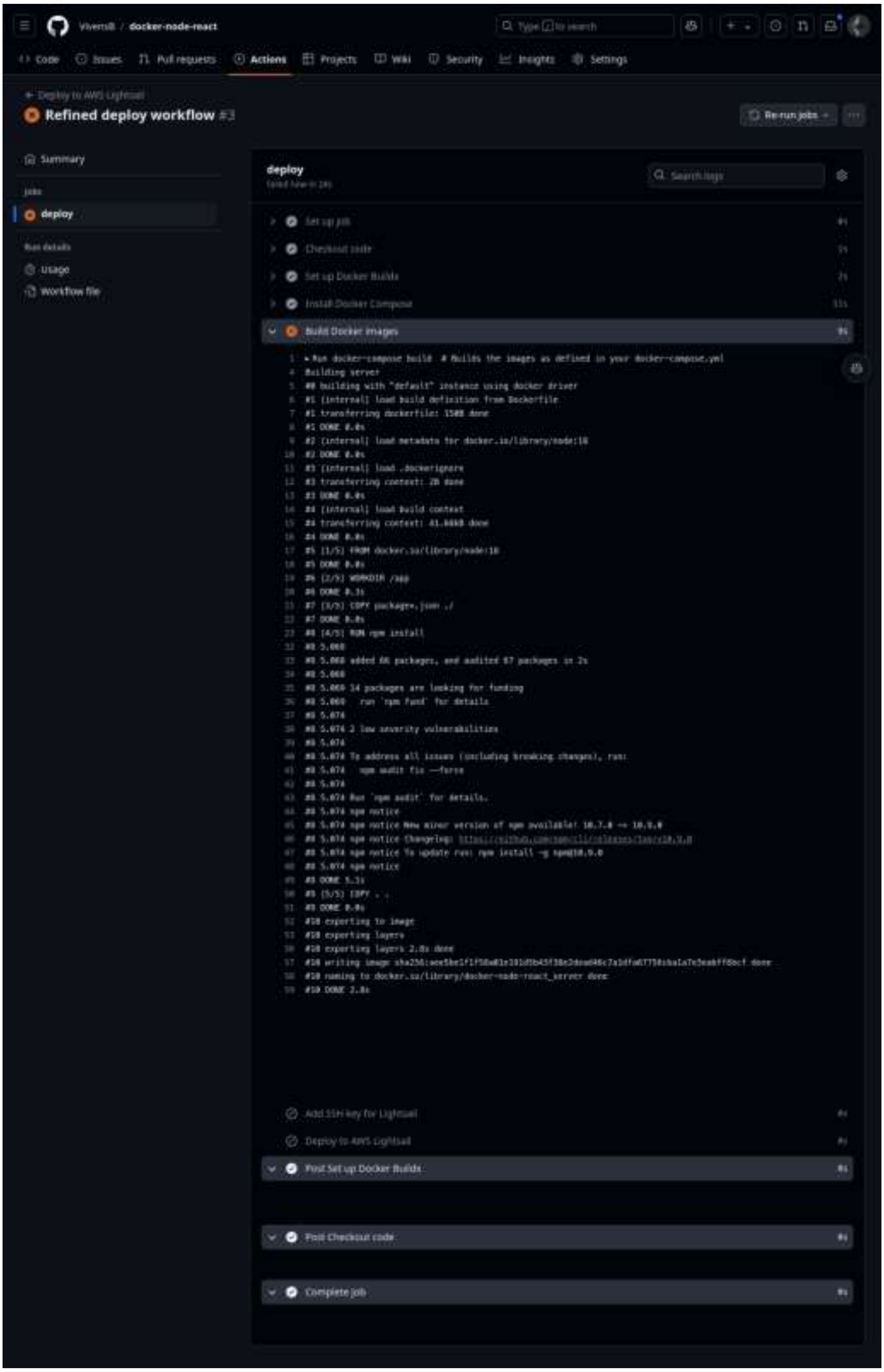
Commit changes

Commit message: Create main.yml

Extended description: Add an optional extended description.

Step 5. Monitor Workflow Runs

- Go to the "Actions" tab in your repository to see the status of your workflow runs.



```
▸ Run docker-compose build  # Builds the images as defined in your docker-compose.yml
Building server
#0 building with "default" instance using docker driver
#1 [internal] load build definition from Dockerfile
#1 transferring dockerfile: 150B done
#1 DONE 0.0s
#2 [internal] load metadata for docker.io/library/node:18
#2 DONE 0.0s
#3 [internal] load .dockerignore
#3 transferring context: 2B done
#3 DONE 0.0s
#4 [internal] load build context
#4 transferring context: 41.66kB done
#4 DONE 0.0s
#5 [1/5] FROM docker.io/library/node:18
#5 DONE 0.0s
#6 [2/5] WORKDIR /app
#6 DONE 0.3s
#7 [3/5] COPY package*.json ./
#7 DONE 0.0s
#8 [4/5] RUN npm install
#8 5.068
#8 5.068 added 66 packages, and audited 67 packages in 2s
#8 5.068
#8 5.069 14 packages are looking for funding
#8 5.069   run `npm fund` for details
#8 5.074
#8 5.074 2 low severity vulnerabilities
#8 5.074
#8 5.074 To address all issues (including breaking changes), run:
#8 5.074   npm audit fix --force
#8 5.074
#8 5.074 Run `npm audit` for details.
#8 5.074 npm notice
#8 5.074 npm notice New minor version of npm available! 10.7.0 -> 10.9.0
#8 5.074 npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.9.0
#8 5.074 npm notice To update run: npm install -g npm@10.9.0
#8 5.074 npm notice
#8 DONE 5.3s
#9 [5/5] COPY . .
#9 DONE 0.0s
#10 exporting to image
#10 exporting layers
#10 exporting layers 2.8s done
#10 writing image sha256:[illegible] done
#10 naming to docker.io/library/docker-node-react_server done
#10 DONE 2.8s
```

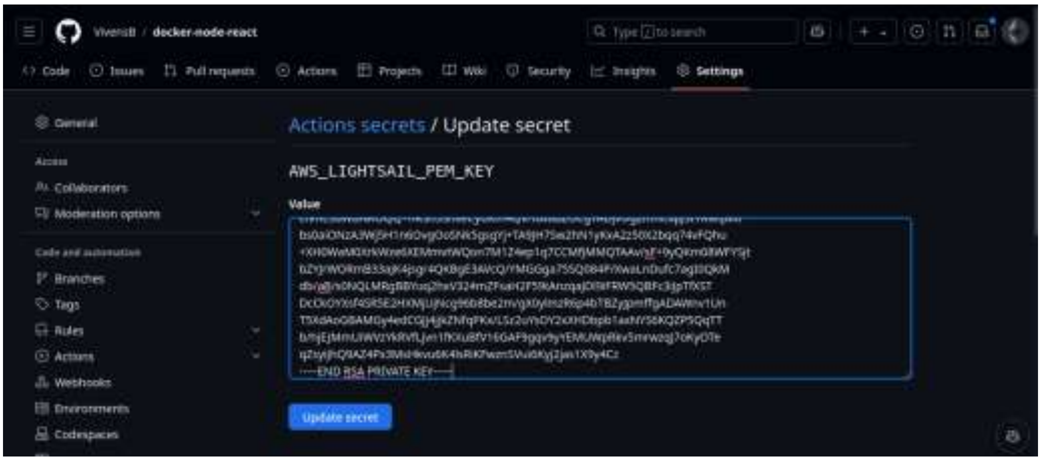
- You can view logs for each step to diagnose any issues.

Step 6. Customize Further

- You can add more jobs, use caching, or set up notifications as needed.
- Adjust triggers under the on: section to specify when you want the CI to run (e.g., on schedule, specific events).

Step 7. (Optional) Use Secrets for Sensitive Data

- If your build requires API keys or passwords, go to your repository settings and add them under "Secrets." Reference them in your workflow file using ${{ secrets.YOUR_SECRET_NAME }}.



Actions secrets / Update secret

AWS_LIGHTSAIL_PEM_KEY

Value

```
[illegible]
-----END RSA PRIVATE KEY-----
```

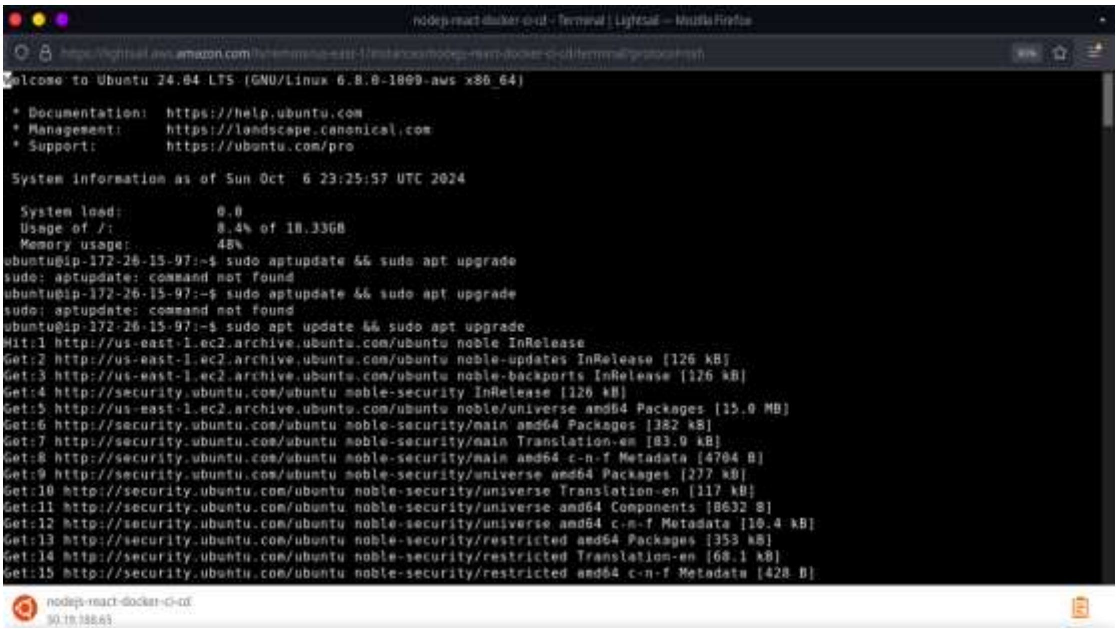
Step 8. (Optional) Integrate with Other CI Services

- If you prefer another CI tool (like Travis CI, CircleCI, etc.), follow their setup instructions and link it to your GitHub repository.

Configure server

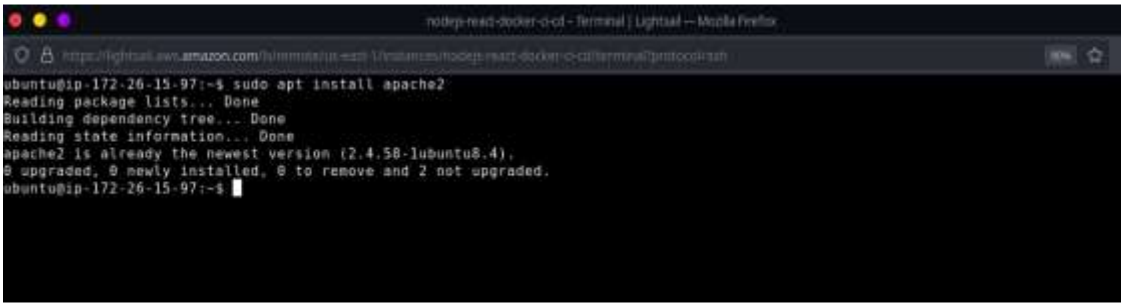
The configuration server in AWS is likely a server that manages and distributes configuration data for applications running on AWS. It can be used to store and manage various types of configuration settings, such as:**Application properties:** Values that are specific to an application, such as database connection strings, API keys, and endpoint URLs.

Step 1: Update and upgrade the server


nodejs-react-docker-ci-cd - Terminal | Lightsail — Mozilla Firefox
https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/nodejs-react-docker-ci-cd/terminal?protocol=ssh
nodejs-react-docker-ci-cd
50.19.188.65

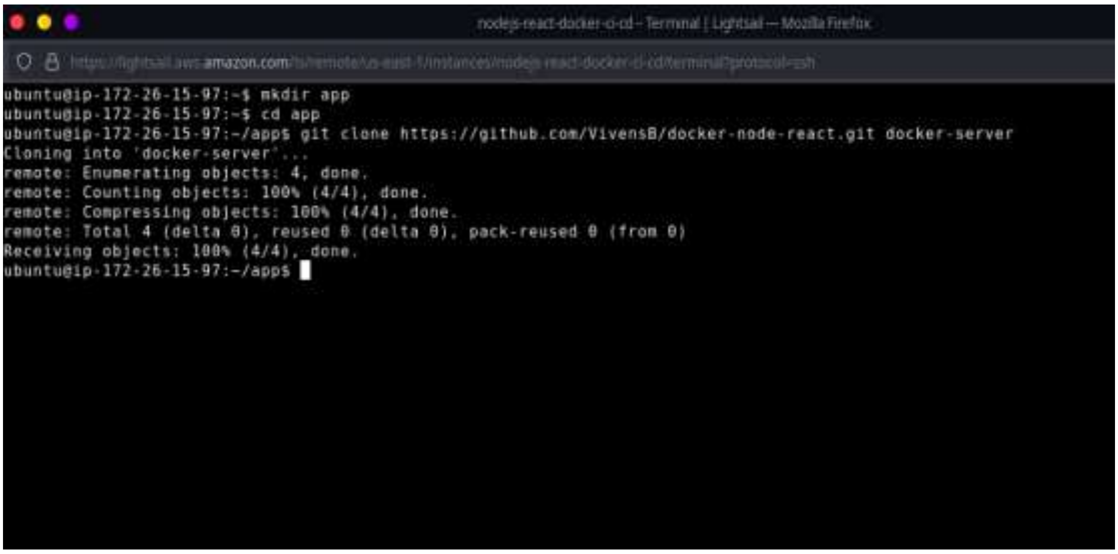
```
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-1009-aws x86_64)

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

 System information as of Sun Oct  6 23:25:57 UTC 2024

  System load:           0.0
  Usage of /:            8.4% of 18.33GB
  Memory usage:          48%
ubuntu@ip-172-26-15-97:~$ sudo aptupdate && sudo apt upgrade
sudo: aptupdate: command not found
ubuntu@ip-172-26-15-97:~$ sudo aptupdate && sudo apt upgrade
sudo: aptupdate: command not found
ubuntu@ip-172-26-15-97:~$ sudo apt update && sudo apt upgrade
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble/universe amd64 Packages [15.0 MB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [382 kB]
Get:7 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [83.9 kB]
Get:8 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [4704 B]
Get:9 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [277 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [117 kB]
Get:11 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [8632 B]
Get:12 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [10.4 kB]
Get:13 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [353 kB]
Get:14 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [68.1 kB]
Get:15 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 c-n-f Metadata [428 B]
```

Step 2: Installation of Apaches


nodejs-react-docker-ci-cd - Terminal | Lightsail — Mozilla Firefox
https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/nodejs-react-docker-ci-cd/terminal?protocol=ssh

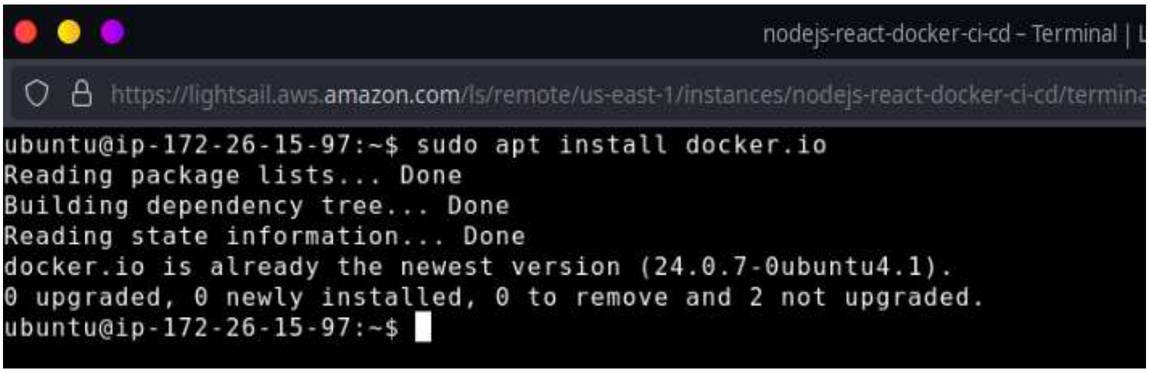
```
ubuntu@ip-172-26-15-97:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.4).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
ubuntu@ip-172-26-15-97:~$ 
```

Step 3: Installation of Dependencies in server


nodejs-react-docker-ci-cd - Terminal | Lightsail — Mozilla Firefox
https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/nodejs-react-docker-ci-cd/terminal?protocol=ssh

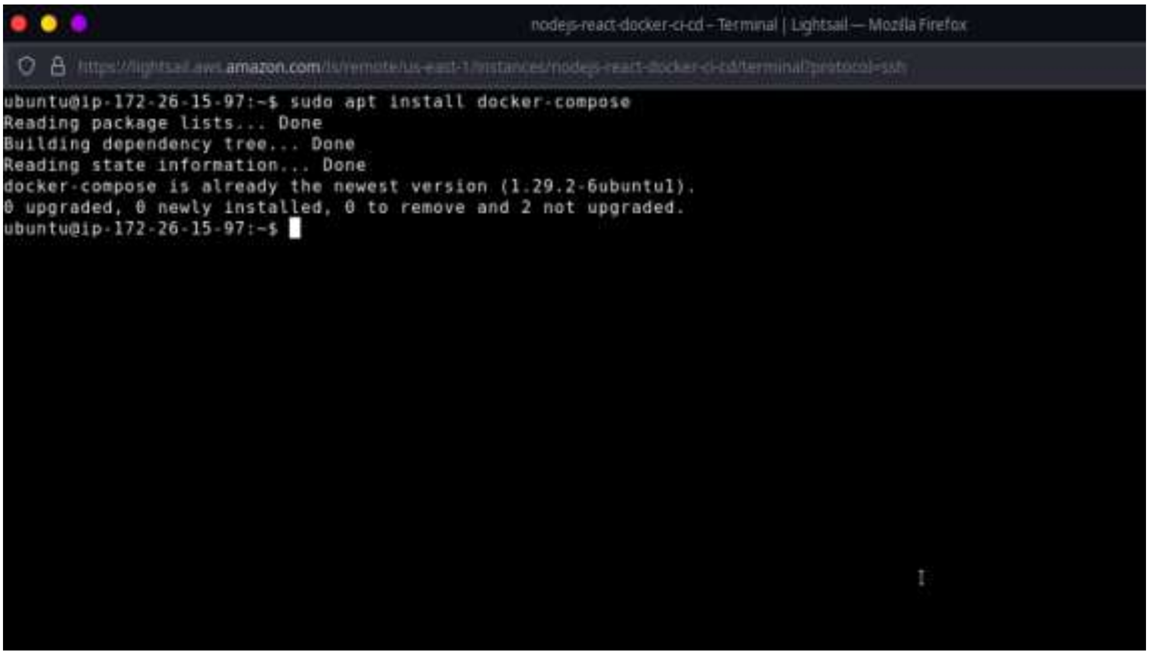
```
ubuntu@ip-172-26-15-97:~$ mkdir app
ubuntu@ip-172-26-15-97:~$ cd app
ubuntu@ip-172-26-15-97:~/apps git clone https://github.com/VivensB/docker-node-react.git docker-server
Cloning into 'docker-server'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.
ubuntu@ip-172-26-15-97:~/apps 
```

Step 4: Installation of docker.io


nodejs-react-docker-ci-cd – Terminal | L
https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/nodejs-react-docker-ci-cd/termina

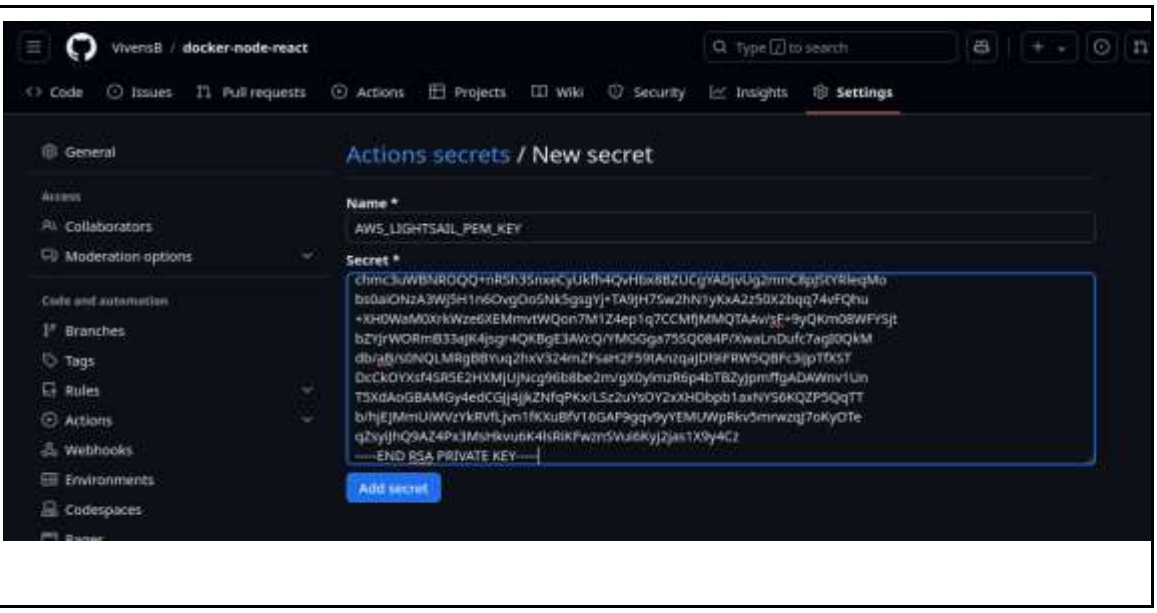
```
ubuntu@ip-172-26-15-97:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker.io is already the newest version (24.0.7-0ubuntu4.1).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
ubuntu@ip-172-26-15-97:~$ 
```

Step5:Install docker composer


nodejs-react-docker-ci-cd – Terminal | Lightsail — Mozilla Firefox
https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/nodejs-react-docker-ci-cd/terminal?protocol=ssh

```
ubuntu@ip-172-26-15-97:~$ sudo apt install docker-compose
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
docker-compose is already the newest version (1.29.2-6ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
ubuntu@ip-172-26-15-97:~$ 
```

. Setup automated build for using secrete key configure on CI/CD



Actions secrets / New secret

Name *

AWS_LIGHTSAIL_PEM_KEY

Secret *

```
chmc3uWBNROQQ+nR5h3SnxeCyUkfh4QvHbx8BZUCgYADjvUg2mnC8pjStYRleqMo
bs0aiONzA3WjSH1n6OvgOo5NkSgsgYj+TA9jH7Sw2hN1yKxA2z50X2bqq74vFQhu
+XH0WaM0XrkWze6XEMmvtWQon7M1Z4ep1q7CCMfjMMQTAAw/sE+9yQKm0BWFYSjt
bZYjrWORmB33ajK4jsgr4QKBgE3AWcQ/YMGGga75SQ084P/XwaLnDufc7ag0QkM
db/aB/s0NQLMRgBBYuq2hxV324mZFsaH2F59tAnzqaJDl9iFRW5QBFc3jpTfXST
DcCkOYXsf4SR5E2HXMjUjNcg96b8be2m/gX0ylmzR6p4bTBZyjpmffgADAWnv1Un
T5XdAoGBAMGy4edCGjj4jjkZNfqPKx/LSz2uYsOY2xXHObpb1axNYS6KQZP5QqTT
b/hjEJMmUiWVzYkRVfLjvn1fKXuBfV16GAP9gqv9yYEMUWpRkv5mrwzqj7oKyOTe
qZsyIJhQ9AZ4Px3MsHkvu6K4lsRiKFwznSVu6Kyj2jas1X9y4Cz
-----END RSA PRIVATE KEY-----
```

Points to Remember

- There is difference between CI and CD where CD automates the process of preparing code changes for deployment, ensuring that the application is always in a deployable state while CI involves the automatic merging and testing of code changes (usually multiple times a day) to ensure that the codebase is always in a working state.
- While developing and deploying software, remember to select containerization technology that help he/ she to enable applications to run consistently across different environments without compatibility issues, providing isolated environments that prevent conflicts and enhance security.
- The most popular version control we must use according key features is GitHub where the developer uses as platform that allows them to create, store, manage and share their code

Application of learning 2.2.

A university offers a course on **Modern Web Development**, where students are learning to build full-stack applications using Node.js (for the backend) and React.js (for the frontend). as full-stack you are tasked to set up project, configure Containerization, and automate deployment using CI/CD pipelines and configure server.

Indicative content 2.3: Perform migration

Duration: 10 hrs

Theoretical Activity 2.3.1: Identification of data migration

Tasks:

1: Answer the following questions

i. What are the best practices for data migration to ensure a smooth and efficient process?

ii. How do you select the right tools and technology for a successful data migration project?

2: Provide your answers on paper, flipchart, blackboard or whiteboard

3: Present your answers to classmate and trainer

4: Ask questions for clarifications if necessary

5: Read the Key reading 2.3.1

Key readings 2.3.1: Identification of data migration

✓ **Identifying Data Migration Best Practices**

Data migration is the process of transferring data from one system to another. It's a critical task that requires careful planning and execution to ensure data integrity and minimize disruptions. Here are some best practices to follow:

- **Define Clear Objectives:** Clearly articulate the goals of the migration. This includes identifying the source and target systems, the data to be migrated, and the desired outcomes.
- **Assess Data Quality:** Evaluate the quality of the data in the source system. Identify and address any inconsistencies, errors, or missing data before migration.
- **Create a Detailed Plan:** Develop a comprehensive migration plan that outlines the steps involved, timelines, responsibilities, and contingency plans.

- **Test Thoroughly:** Conduct rigorous testing to ensure the migrated data is accurate, complete, and compatible with the target system. Use both unit testing and integration testing to validate the migration process.
- **Implement Data Validation:** Establish data validation rules to verify the accuracy and consistency of the data during and after migration.
- **Ensure Security:** Protect sensitive data during the migration process by implementing appropriate security measures, such as encryption and access controls.
- **Provide Training:** Train users on the new system and data structures to minimize disruptions and ensure smooth adoption.
- **Have a Rollback Plan:** Develop a contingency plan to revert to the original state if issues arise during or after the migration.

✓ **Selecting the Right Tools & Technology**

The choice of tools and technology for data migration depends on several factors, including the size and complexity of the data, the source and target systems, and the desired level of automation. Here are some key considerations:

1. **Data Extraction Tools:** Choose tools that can efficiently extract data from the source system, handling various data formats and structures.
2. **Data Transformation Tools:** Select tools that can transform data to match the requirements of the target system, including data cleansing, validation, and conversion.
3. **Data Loading Tools:** Opt for tools that can load data into the target system efficiently and reliably, handling large datasets and different database types.
4. **Integration Platforms:** Consider using integration platforms to automate and manage the entire migration process, providing a centralized view and control over data movement.
5. **Cloud-Based Solutions:** Explore cloud-based data migration services that offer scalability, flexibility, and reduced infrastructure costs.
6. **Open-Source Tools:** Evaluate open-source options for cost-effective solutions, but consider factors like support, community, and maturity.

Creating a data migration pipeline involves several key steps that ensure data is efficiently transferred from a source system to a target system while maintaining its integrity and quality. Here's a high-level overview of the steps involved:

1. Planning & Requirement Gathering

- **Understand Data Source and Target**: Analyze both the source and target databases or systems to understand the data types, structures, formats, and any transformation needs.
- **Define Objectives**: Establish the purpose of the migration and the expected outcome, such as improving performance, storage, or system upgrades.
- **Data Mapping**: Map fields and data types from the source to the target system, accounting for any schema differences.

2. Data Extraction

- **Extract Data from Source**: Use appropriate tools or custom scripts to extract data from the source system.
- **Scheduling**: If the system needs to remain active during the migration, schedule extraction during low-usage hours or in batches to minimize downtime.

3. Data Transformation

- **Cleanse Data**: Perform data cleaning to remove duplicates, incorrect, or incomplete records.
- **Apply Business Rules**: Transform data to match the target schema, format, or specific business requirements. For example, changing date formats or converting data types.
- **Data Validation**: Ensure data quality by validating formats, integrity, and structure.

4. Data Loading

- **Load Data into Target**: Load the transformed data into the target system. This can be done in batches or as a complete load, depending on the size and complexity of the data.
- **Handle Errors**: Implement error handling mechanisms to capture and resolve any issues that arise during the loading process.

5. Testing and Validation

- **Data Integrity Checks**: Validate that the data was migrated correctly by comparing sample data from the source and target.
- **Functional Testing**: Test the application with the migrated data to ensure the system works as expected with the new dataset.

- **Reconciliation**: Perform data reconciliation to ensure the completeness of the migration by comparing counts, totals, or hashes between the source and target.

6. Monitoring and Optimization

- **Monitor Performance**: Continuously monitor the migration process to optimize load times and ensure minimal downtime.
- **Error Tracking**: Track any errors or data discrepancies in real-time for quick troubleshooting.

7. Post-Migration Tasks

- **Final Validation**: Perform a final validation of the migrated data and run reports to confirm accuracy.
- **System Cutover**: Once everything is confirmed, switch the system to use the new migrated data.
- **Data Backup**: Ensure that all the migrated data and source system data is backed up for future reference.
- **Documentation**: Document the entire pipeline, including the tools used, transformation rules, and validation tests for future audits or migrations.

Practical Activity 2.3.2: Creation of Data migration pipeline

Task:

1: Read key reading 2.3.2.

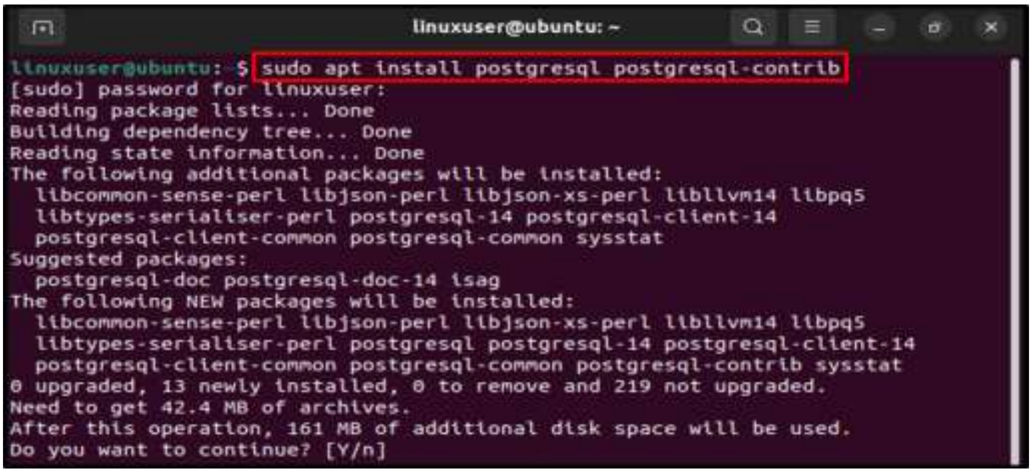
i. Referring to the key reading 2.3.2, You are requested to go to the computer Lab to install database management system and create data migration to new changes of database schema.

2.Present your work to trainer and classmate

3: Ask questions for clarifications if necessary

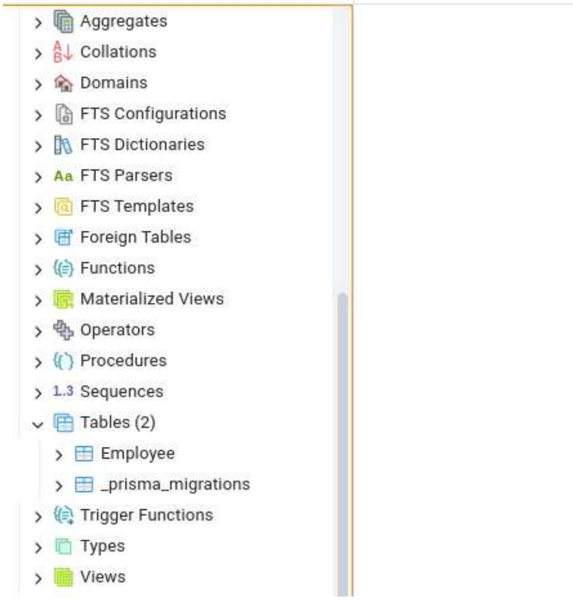
Key readings 2.3.2.: Creating of Data migration pipeline

Step 1: install postgress from official website or using command.

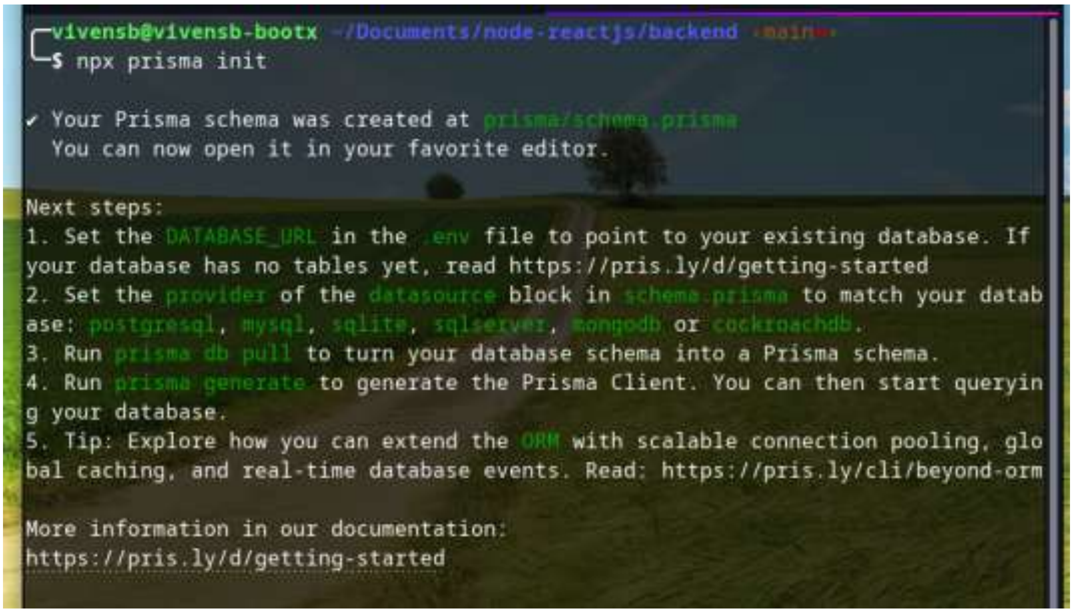


```
linuxuser@ubuntu:~$ sudo apt install postgresql postgresql-contrib
[sudo] password for linuxuser:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5
  libtypes-serialiser-perl postgresql-14 postgresql-client-14
  postgresql-client-common postgresql-common sysstat
Suggested packages:
  postgresql-doc postgresql-doc-14 isag
The following NEW packages will be installed:
  libcommon-sense-perl libjson-perl libjson-xs-perl libllvm14 libpq5
  libtypes-serialiser-perl postgresql postgresql-14 postgresql-client-14
  postgresql-client-common postgresql-common postgresql-contrib sysstat
0 upgraded, 13 newly installed, 0 to remove and 219 not upgraded.
Need to get 42.4 MB of archives.
After this operation, 161 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Step 2: creation of Database and table in databases using postgres as DBMS in local databases

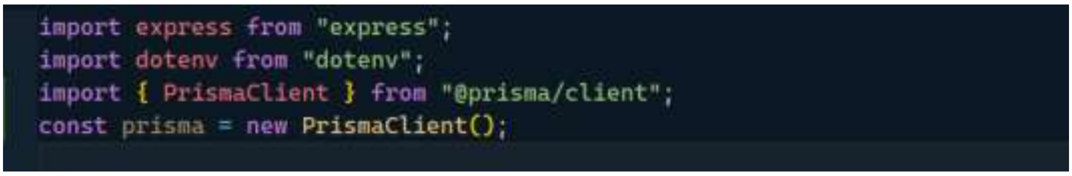


Step 3: Install and start prisma that facilitate to connect database node js application



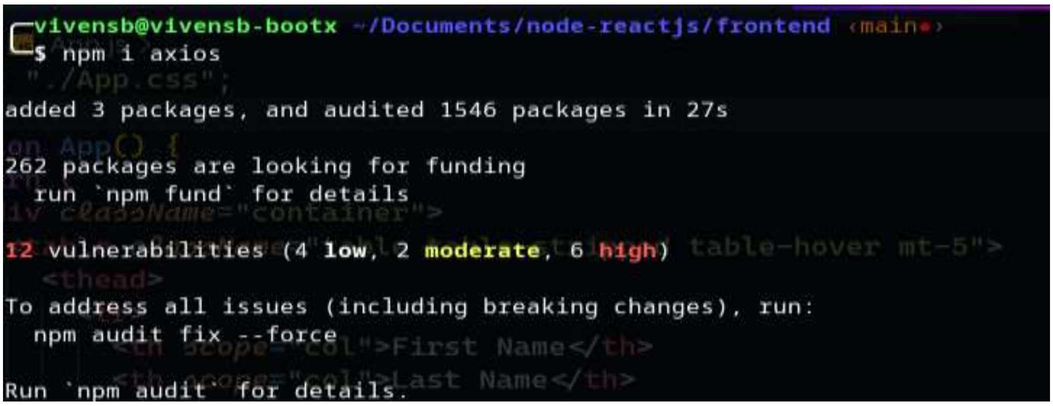
```
┌vivensb@vivensb-bootx ~/Documents/node-reactjs/backend ‹main›
└$ npx prisma init

✔ Your Prisma schema was created at prisma/schema.prisma
  You can now open it in your favorite editor.

Next steps:
1. Set the DATABASE_URL in the .env file to point to your existing database. If
your database has no tables yet, read https://pris.ly/d/getting-started
2. Set the provider of the datasource block in schema.prisma to match your datab
ase: postgresql, mysql, sqlite, sqlserver, mongodb or cockroachdb.
3. Run prisma db pull to turn your database schema into a Prisma schema.
4. Run prisma generate to generate the Prisma Client. You can then start queryin
g your database.
5. Tip: Explore how you can extend the ORM with scalable connection pooling, glo
bal caching, and real-time database events. Read: https://pris.ly/cli/beyond-orm

More information in our documentation:
https://pris.ly/d/getting-started
```

Step 4: import prisma in node js application



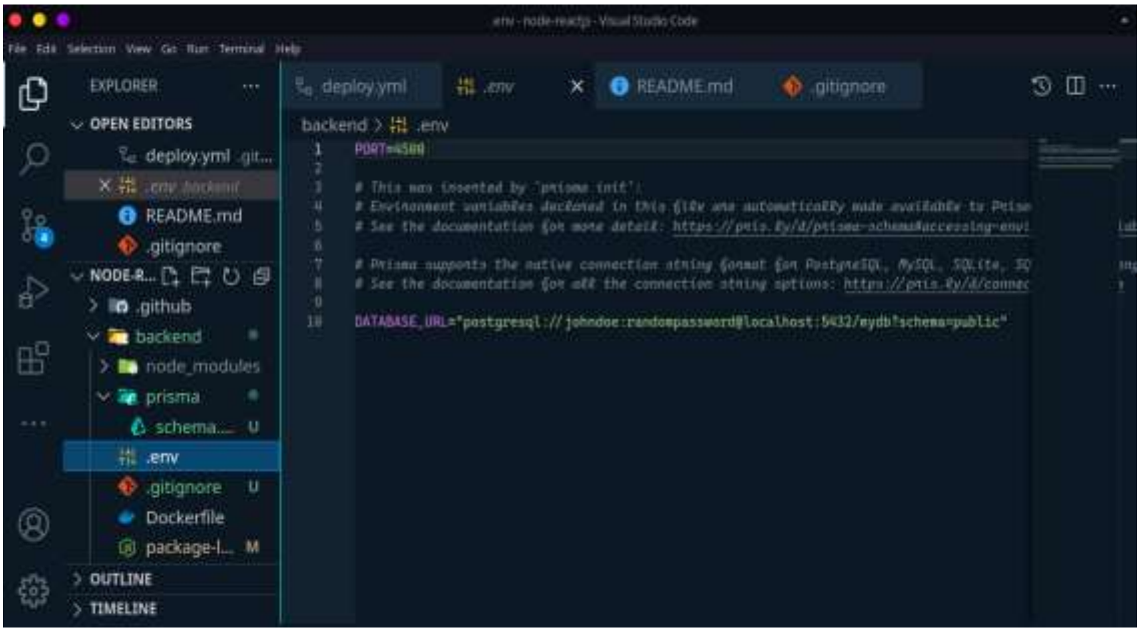
```
import express from "express";
import dotenv from "dotenv";
import { PrismaClient } from "@prisma/client";
const prisma = new PrismaClient();
```

Step 5: install axios that request post or get in database



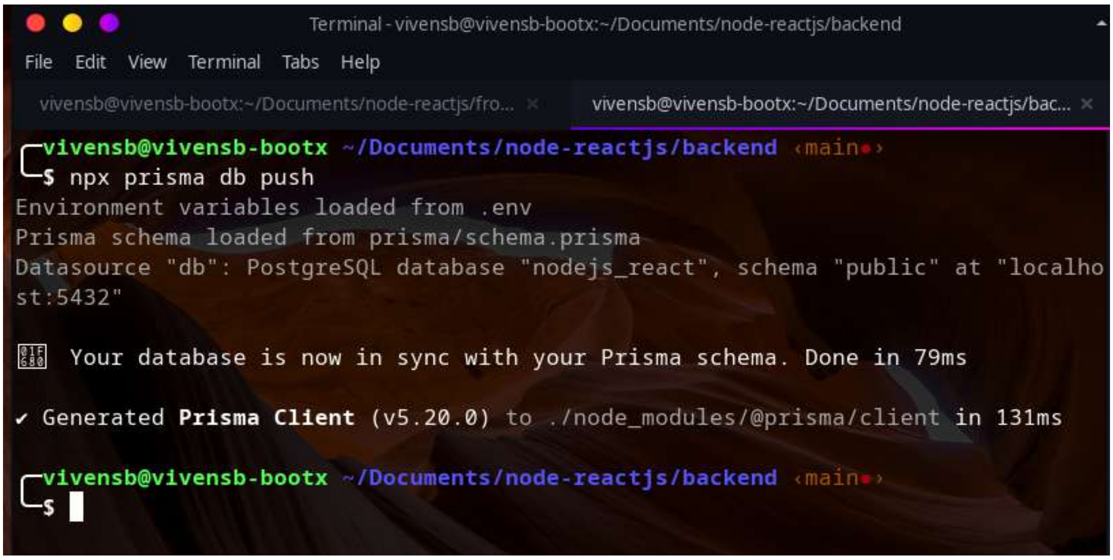
```
┌vivensb@vivensb-bootx ~/Documents/node-reactjs/frontend ‹main›
└$ npm i axios

added 3 packages, and audited 1546 packages in 27s

262 packages are looking for funding
  run `npm fund` for details

12 vulnerabilities (4 low, 2 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Step 6: change the env path of databases where will store data



```
PORT=4500

# This was inserted by `prisma init`:
# Environment variables declared in this file are automatically made available to Prism
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-envi

# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQ
# See the documentation for all the connection string options: https://pris.ly/d/connec

DATABASE_URL="postgresql://johndoe:randompassword@localhost:5432/mydb?schema=public"
```

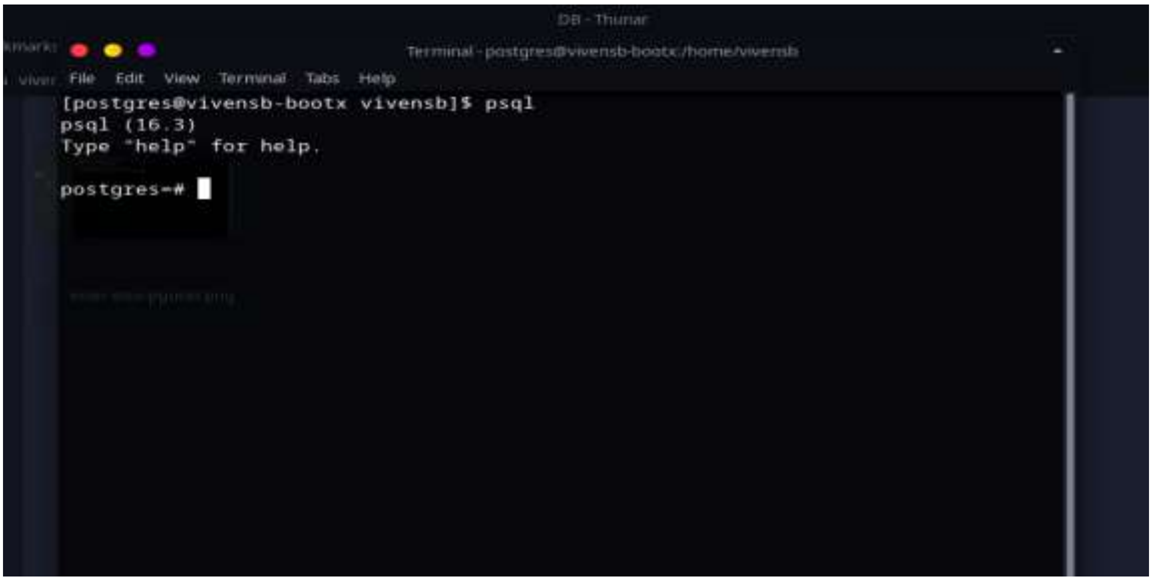
7. make migrate of database schema



```
┌vivensb@vivensb-bootx ~/Documents/node-reactjs/backend ‹main●›
└$ npx prisma db push
Environment variables loaded from .env
Prisma schema loaded from prisma/schema.prisma
Datasource "db": PostgreSQL database "nodejs_react", schema "public" at "localho
st:5432"

🚀  Your database is now in sync with your Prisma schema. Done in 79ms

✔ Generated Prisma Client (v5.20.0) to ./node_modules/@prisma/client in 131ms

┌vivensb@vivensb-bootx ~/Documents/node-reactjs/backend ‹main●›
└$ █
```

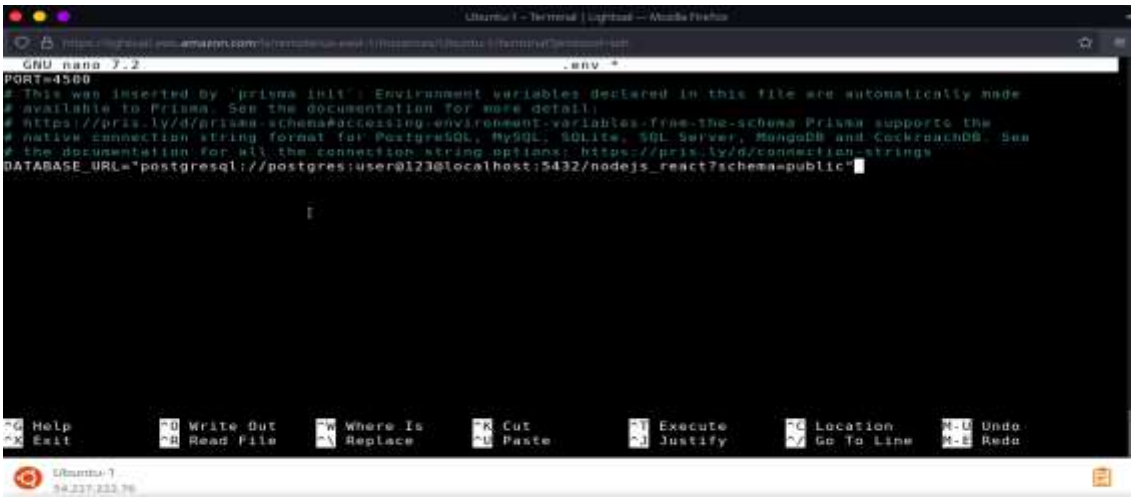
Terminal - vivensb@vivensb-bootx:~/Documents/node-reactjs/backend
File Edit View Terminal Tabs Help
vivensb@vivensb-bootx:~/Documents/node-reactjs/fro...
vivensb@vivensb-bootx:~/Documents/node-reactjs/bac...

Step 8: Configuration of Database In remote server



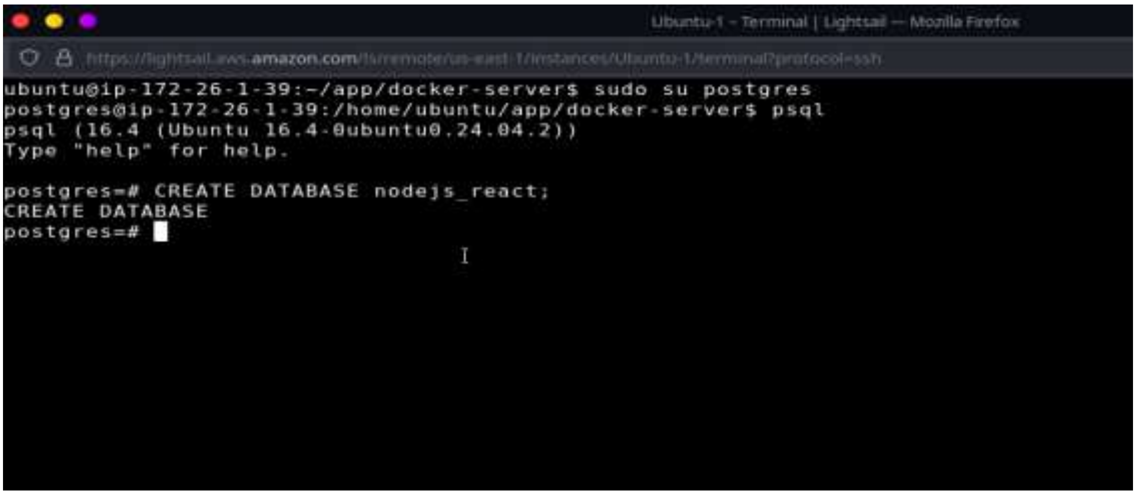
```
[postgres@vivensb-bootx vivensb]$ psql
psql (16.3)
Type "help" for help.

postgres=#
```

Step 9:configure postgress path from local env in server



```
  GNU nano 7.2                              .env *
PORT=4500
# This was inserted by `prisma init`: Environment variables declared in this file are automatically made
# available to Prisma. See the documentation for more detail:
# https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema Prisma supports the
# native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB and CockroachDB. See
# the documentation for all the connection string options: https://pris.ly/d/connection-strings
DATABASE_URL="postgresql://postgres:user@123@localhost:5432/nodejs_react?schema=public"
```

Step 10: create database in server and table has the same name with local


Ubuntu-1 - Terminal | Lightsail — Mozilla Firefox
https://lightsail.aws.amazon.com/ls/remote/us-east-1/instances/Ubuntu-1/terminal?protocol=ssh

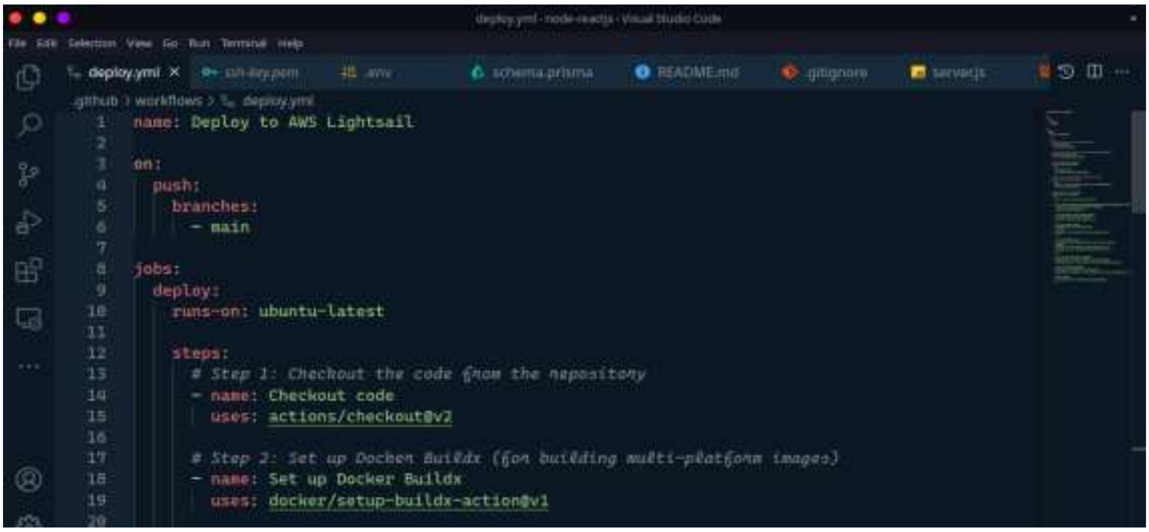
```
ubuntu@ip-172-26-1-39:~/app/docker-server$ sudo su postgres
postgres@ip-172-26-1-39:/home/ubuntu/app/docker-server$ psql
psql (16.4 (Ubuntu 16.4-0ubuntu0.24.04.2))
Type "help" for help.

postgres=# CREATE DATABASE nodejs_react;
CREATE DATABASE
postgres=#
```

After install all requirement needed

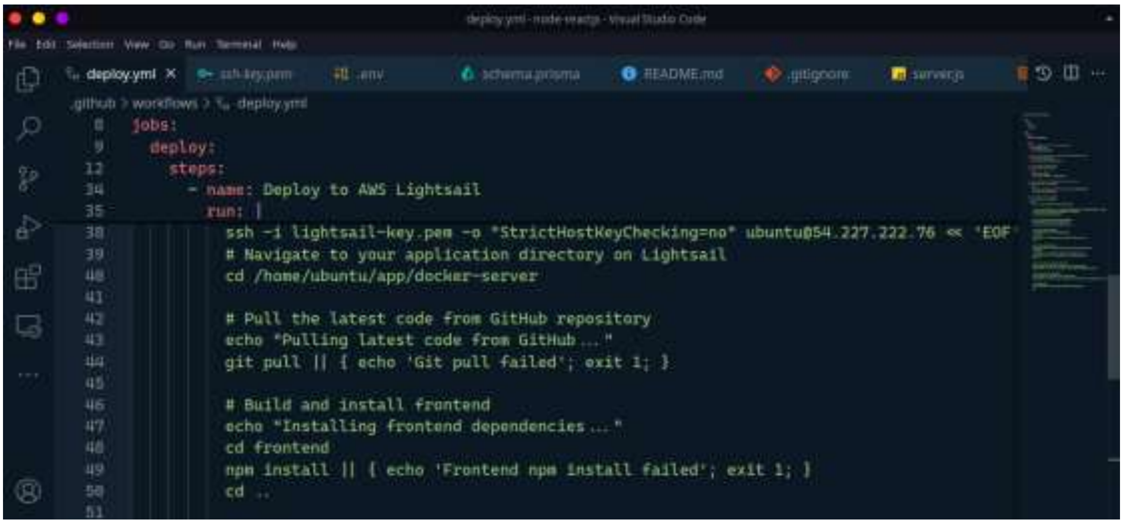
Implement Continuous Integration

This implementation of Continuous Integration provide the data from local server to migrate data to github and provide it to AWS server.

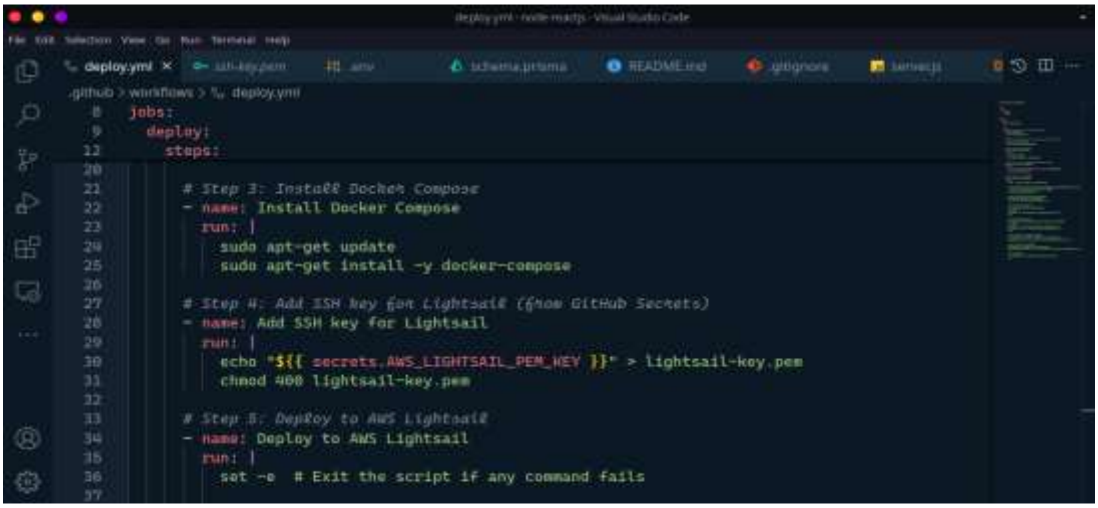
Step 11: configure CI/CD


deploy.yml - node-reactjs - Visual Studio Code
File Edit Selection View Go Run Terminal Help
deploy.yml
ssh-key.pem
.env
schema.prisma
README.md
.gitignore
server.js
.github > workflows > deploy.yml

```yaml
name: Deploy to AWS Lightsail

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      # Step 1: Checkout the code from the repository
      - name: Checkout code
        uses: actions/checkout@v2

      # Step 2: Set up Docker Buildx (for building multi-platform images)
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v1
```

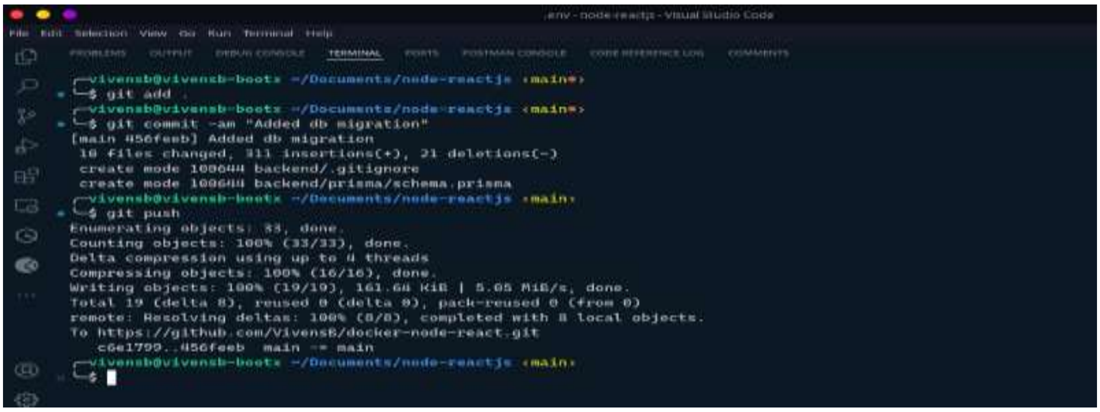


```
jobs:
  deploy:
    steps:
      - name: Deploy to AWS Lightsail
        run: |
          ssh -i lightsail-key.pem -o "StrictHostKeyChecking=no" ubuntu@54.227.222.76 << 'EOF'
          # Navigate to your application directory on Lightsail
          cd /home/ubuntu/app/docker-server

          # Pull the latest code from GitHub repository
          echo "Pulling latest code from GitHub..."
          git pull || { echo 'Git pull failed'; exit 1; }

          # Build and install frontend
          echo "Installing frontend dependencies..."
          cd frontend
          npm install || { echo 'Frontend npm install failed'; exit 1; }
          cd ..
```

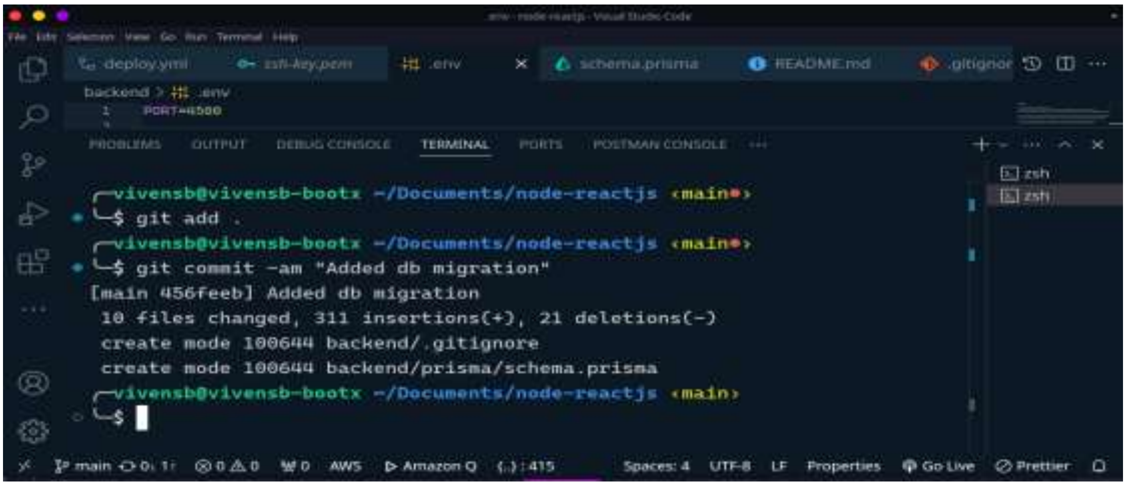


```
jobs:
  deploy:
    steps:

      # Step 3: Install Docker Compose
      - name: Install Docker Compose
        run: |
          sudo apt-get update
          sudo apt-get install -y docker-compose

      # Step 4: Add SSH key for Lightsail (from GitHub Secrets)
      - name: Add SSH key for Lightsail
        run: |
          echo "${{ secrets.AWS_LIGHTSAIL_PEM_KEY }}" > lightsail-key.pem
          chmod 400 lightsail-key.pem

      # Step 5: Deploy to AWS Lightsail
      - name: Deploy to AWS Lightsail
        run: |
          set -e  # Exit the script if any command fails
```

Step 12: commit and push new change from local server

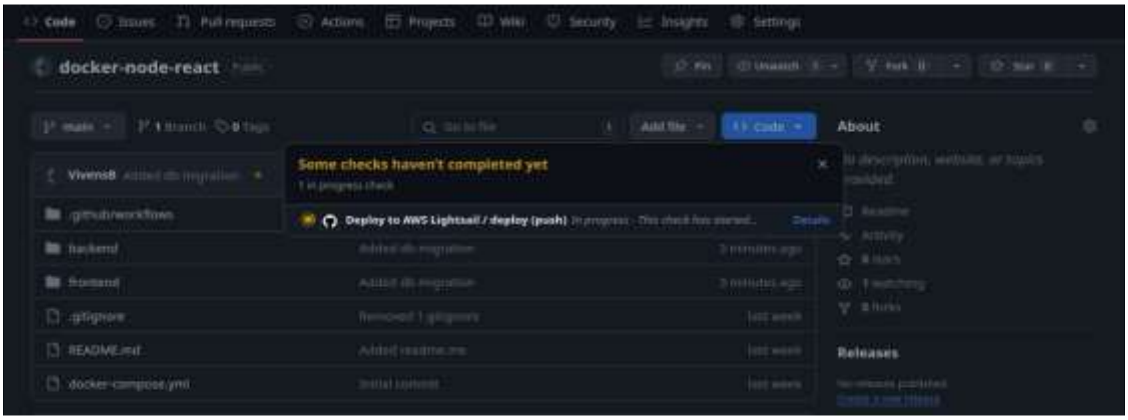


```
┌─vivensb@vivensb-bootx ~/Documents/node-reactjs ‹main*›
└─$ git add .
┌─vivensb@vivensb-bootx ~/Documents/node-reactjs ‹main*›
└─$ git commit -am "Added db migration"
[main 456feeb] Added db migration
 10 files changed, 311 insertions(+), 21 deletions(-)
 create mode 100644 backend/.gitignore
 create mode 100644 backend/prisma/schema.prisma
┌─vivensb@vivensb-bootx ~/Documents/node-reactjs ‹main›
└─$ git push
Enumerating objects: 33, done.
Counting objects: 100% (33/33), done.
Delta compression using up to 4 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (19/19), 161.64 KiB | 5.05 MiB/s, done.
Total 19 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 8 local objects.
To https://github.com/VivensB/docker-node-react.git
   c6e1799..456feeb  main -> main
┌─vivensb@vivensb-bootx ~/Documents/node-reactjs ‹main›
└─$ ▌
```



```
┌─vivensb@vivensb-bootx ~/Documents/node-reactjs ‹main●›
└─$ git add .
┌─vivensb@vivensb-bootx ~/Documents/node-reactjs ‹main●›
└─$ git commit -am "Added db migration"
[main 456feeb] Added db migration
 10 files changed, 311 insertions(+), 21 deletions(-)
 create mode 100644 backend/.gitignore
 create mode 100644 backend/prisma/schema.prisma
┌─vivensb@vivensb-bootx ~/Documents/node-reactjs ‹main›
└─$ █
```

Step 13: CI-CD-job-started-deployment



Some checks haven't completed yet
1 in progress check
Deploy to AWS Lightsail / deploy (push) In progress — This check has started...

Points to Remember

- Data Migration Deployment is a critical phase in DevOps where data is transferred from an existing system to a remote system according the changes of database schema by Define the migration goal, identify source and target systems, assess data quality, create details plan, test thoroughly and training users to new system.
- While creating data migration remember that data are transfer from local repository to remote repository by using version control and then data migrate from remote repository to remote server.

Application of learning 2.3.

BNC university is upgrading its Learning Management System (LMS) to a new, more modern platform. The existing LMS holds years of data, including student information, course applications, grades, and feedback. You are tasked with migrating all this data to the new LMS while maintaining data integrity and push the change to remote server.

Learning outcome 2 end assessment

Theoretical assessment

Q1. Circle the correct answer on the following questions:

i. Which of the following is a tool commonly used for Continuous Deployment (CD)?

   a. Jenkins
   b. Ansible
   c. SonarQube
   d. Git

ii. **What is the primary role of a Continuous Integration (CI) server?**

   a. Manage code repositories
   b. Automate the building and testing of code
   c. Orchestrate server deployments
   d. Monitor system uptime

iii. Which of the following tasks is related to Continuous Integration (CI)?

   a. Implementing automated rollback
   b. Developing deployment scripts
   c. Setting up automated builds
   d. Using infrastructure as code (IaC)

iv. Which tool is best suited for deployment orchestration in Continuous Delivery (CD)?

   a. GitLab CI
   b. Kubernetes
   c. Jenkins
   d. Travis CI

v. What does Infrastructure as Code (IaC) allow you to do?

   a. Automate deployments using scripts
   b. Manage code quality
   c. Track software versions
   d. Create CI pipelines

vi. Which of the following is NOT a task in Continuous Delivery (CD)?

   a. Configure the CI pipeline
   b. Perform code quality checks
   c. Implement automated rollback

d. Use a deployment orchestration tool

vii. Which of the following steps would you take when implementing a Continuous Integration (CI) pipeline?

a. A Configure CD pipeline
b. Set up an automated build process
c. Develop deployment scripts
d. Use deployment orchestration tools

viii. Which feature of Continuous Deployment (CD) helps in minimizing disruptions and errors during deployment.

a. Automated rollback
b. Infrastructure as Code (IaC)
c. Code quality checks
d. Artifact management

Q2. Read carefully and then answer the following questions

i. Describe CI and CD tools.
ii. What are the benefits of adopting DevOps approach?
iii. How does DevOps promote collaboration between development and operation teams?
iv. What are the challenges that organisation might face when implementing DevOps?

Practical assessment

Imagine you are part of a development team with responsibilities for building and deploying a scalable web application. As full-stack developer you are requested to step up project, containerize application, deploy application by using CI/CD pipeline and perform data migration to a new database system.

END

References

N, V., Batra, A., & Bhandari, V. (2023, April). *What are the best practices for data migration in your DevOps pipeline?* Retrieved April 27, 2024, from Linkedin: https://www.linkedin.com/advice/3/what-best-practices-data-migration-your-devops-ade8e

Poulton, N. (2017). *Docker Deep Dive.* Nigel Poulton.

REHKOPF, M. (2024, April). *Continuous integration tools*. Retrieved April 27, 2024, from Atlassian: https://www.atlassian.com/continuous-delivery/continuous-integration/tools

Simplilearn. (2023, April). *Docker And Containers Explained | Containerization Explained | Docker Tutorial | Simplilearn*. Retrieved April 27, 2024, from Simpli Learn - Yooutube video: https://www.youtube.com/watch?app=desktop&v=A0g7I4A6GN4

DevOps glossary: 78 basic DevOps. (2023, July 12). Retrieved April 27, 2024, from Its Vit: https://itsvit.com/blog/devops-glossary-78-basic-devops-terms-in-simple-words/

Fortinet. (2023, April). *DevOps Security*. Retrieved April 27, 2024, from Fortinet: https://www.fortinet.com/resources/cyberglossary/devops-security

GitLab. (2024). *What is DevOps*. Retrieved April 27, 2024, from About Git Lab: https://about.gitlab.com/topics/devops/

Learning Outcome 3: Implement Monitoring

Indicative contents

3.1 Preparation of monitoring tools in DevOps environment

3.2 Analysis of Performance Metrics and Feedback Data

3.3 Documentation of monitoring report

Key Competencies for Learning Outcome 3: Implement Monitoring

Knowledge	**Skills**	**Attitudes**
• Identification of monitoring tools. • Identification of Performance metric and feedback • Description of report •	• Installing monitoring tools • Utilizing Monitoring Tools • Writing a Report	• Being patient and persistence while analysing data • Being problem-solving Mindset while analysing data • Paying attention to details while analysing data

Duration: 10 hrs

Learning outcome 3 objectives:

By the end of the learning outcome, the trainees will be able to:

1. Identify correctly monitoring tools according to their respective documentation.

2.Identify clearly the performance metrics and feedback data are routinely analysed in accordance with system requirements

3.Describe correctly the documented report based on the work done

4.Install accurately the monitoring tool based on respective documentation

5.Generate properly the report documented based on analysed data

Resources

Equipment	Tools	Materials
• Computer • Server • Printer	• IDE • Browsers • DBMS	• Paper • Internet • electricity

Indicative content 3.1: Preparation of monitoring Tools in DevOps environment

Duration: 4 hrs

Theoretical Activity 3.1.1: identification of monitoring tools in DevOps environment

Tasks:

1: You are asked to answer the following questions:

i. Give the benefits of DevOps Monitoring

ii. List the importance of Monitoring Tools

iii. Identify the following monitoring tools types

a) Application tools
b) Networking Tools
c) Infrastructure tools

2: Write your answers on paper, flipchart, blackboard or whiteboard

3: Present your answers to classmate and trainer

4: Ask questions for clarifications if necessary

5: Read the key readings 3.1.1

Key readings 3.1.1: Preparation of monitoring tools in DevOps environment

Definition of Monitoring Tools.

Monitoring tools are software applications or platforms designed to track, collect, and analyze data related to the performance, health, and functionality of systems, networks, applications, and infrastructure. They provide real-time metrics and logs, which are essential for identifying potential issues, optimizing resources, and ensuring that services are running smoothly.

✓ **Benefits of DevOps Monitoring**

1. **Enhanced Performance**:
   - Continuous monitoring allows teams to identify and resolve performance issues in real-time, ensuring that applications run smoothly and efficiently.
   - It helps in optimizing resource utilization, leading to improved application performance and user satisfaction.
2. **Faster Incident Response**:
   - Automated monitoring tools can quickly detect anomalies and alert teams, allowing for faster incident response and resolution.
   - This reduces downtime and minimizes the impact of incidents on end users.
3. **Proactive Issue Resolution**:
   - Monitoring provides insights into system behavior, helping teams anticipate potential problems before they escalate.
   - By analyzing trends and patterns, teams can implement proactive measures to prevent outages.
4. **Collaboration and Communication**:
   - DevOps emphasizes collaboration between development and operations teams. Monitoring fosters this by providing shared visibility into system performance and issues.
   - Teams can communicate effectively about incidents, improvements, and system health.
5. **Data-Driven Decisions**:
   - Monitoring tools collect vast amounts of data, which can be analyzed to inform strategic decisions.
   - Teams can leverage insights to optimize processes, allocate resources effectively, and improve overall system architecture.
6. **Improved User Experience**:
   - By maintaining high system availability and performance, monitoring directly contributes to a better user experience.
   - Monitoring helps ensure that user-facing services are reliable and perform well under varying loads.
7. **Continuous Improvement**:
   - Regular monitoring and feedback loops allow teams to assess the impact of changes and continuously improve their processes and systems.
   - This aligns with the DevOps philosophy of iterative development and continuous delivery.

8. **Compliance and Security**:
   - Monitoring can help identify security breaches and compliance issues by providing visibility into system access and changes.
   - It enables teams to enforce security policies and maintain compliance with industry regulations.

✓ **Importance of Monitoring Tools**

1. **Real-Time Visibility**:
   - Monitoring tools provide real-time insights into system performance, application health, and infrastructure status.
   - This visibility is crucial for identifying issues quickly and ensuring system reliability.
2. **Centralized Dashboard**:
   - Most monitoring tools offer centralized dashboards that consolidate metrics from various sources, making it easier for teams to track performance and incidents in one place.
   - This simplifies the monitoring process and enhances situational awareness.
3. **Automation and Efficiency**:
   - Monitoring tools automate the process of data collection and alerting, reducing the need for manual intervention and allowing teams to focus on strategic tasks.
   - This increases operational efficiency and reduces the likelihood of human error.
4. **Scalability**:
   - As organizations grow, monitoring tools can scale to accommodate increased complexity in infrastructure and applications.
   - This ensures that monitoring remains effective even as systems evolve.
5. **Integration with CI/CD Pipelines**:
   - Monitoring tools can integrate with Continuous Integration/Continuous Deployment (CI/CD) pipelines to provide feedback on application performance during the development cycle.
   - This facilitates rapid iteration and deployment of high-quality software.
6. **Historical Analysis and Reporting**:
   - Monitoring tools often include features for historical data analysis, enabling teams to track performance over time and identify trends.
   - This data is valuable for post-incident reviews and long-term strategic planning.

7. **Alerting and Notification**:
    - Monitoring tools provide customizable alerting mechanisms that notify teams of critical issues based on predefined thresholds.
    - This ensures that teams are informed of potential problems before they impact users.
8. **Collaboration Features**:
    - Many monitoring tools offer collaboration features that allow teams to share insights, comment on incidents, and manage tasks collectively.
    - This fosters a culture of collaboration and enhances incident response efforts.

✓ **Identification of monitoring tools types**

Here's a detailed identification of the types of monitoring tools, categorized into application tools, networking tools, and infrastructure tools:

1. Application Monitoring Tools

Application monitoring tools focus on tracking the performance and behavior of software applications. They help ensure that applications run smoothly and provide insights into user interactions. Common types include:

- **Application Performance Monitoring (APM)**:
    - **Examples**: New Relic, AppDynamics, Dynatrace
    - **Functionality**: Monitors application performance metrics, transaction tracing, and user experiences. They help identify bottlenecks, errors, and response times.
- **Log Management Tools**:
    - **Examples**: Splunk, ELK Stack (Elasticsearch, Logstash, Kibana), Graylog
    - **Functionality**: Collect, analyze, and visualize log data from applications to detect anomalies and troubleshoot issues.
- **User Experience Monitoring**:
    - **Examples**: Google Analytics, Hotjar, FullStory
    - **Functionality**: Analyzes user interactions with web applications to improve user experience through behavior tracking and heatmaps.
- **Synthetic Monitoring**:
    - **Examples**: Pingdom, Uptrends, Site24x7
    - **Functionality**: Simulates user transactions to monitor application performance from different locations and devices.

2. Networking Monitoring Tools

Networking monitoring tools help organizations track network performance and security. They provide visibility into network traffic, device status, and overall health.

Key types include:

- **Network Performance Monitoring (NPM)**:
  - **Examples**: SolarWinds Network Performance Monitor, PRTG Network Monitor, Nagios
  - **Functionality**: Monitors network traffic, device performance, and identifies issues like bottlenecks or latency.
- **Network Security Monitoring (NSM)**:
  - **Examples**: Snort, Suricata, Cisco Stealthwatch
  - **Functionality**: Monitors network traffic for security threats, detects anomalies, and analyzes intrusion attempts.
- **Bandwidth Monitoring Tools**:
  - **Examples**: NetFlow Analyzer, Wireshark, ManageEngine NetFlow Analyzer
  - **Functionality**: Tracks bandwidth usage across the network, identifies top talkers, and analyzes traffic patterns.
- **Configuration Management Tools**:
  - **Examples**: Ansible, Puppet, Chef
  - **Functionality**: Monitors and manages network device configurations, ensuring compliance with policies and reducing configuration errors.

3. Infrastructure Monitoring Tools

Infrastructure monitoring tools focus on the health and performance of the underlying infrastructure, including servers, databases, and cloud resources. They ensure that hardware and services operate optimally. Common types include:

- **Server Monitoring Tools**:
  - **Examples**: Zabbix, Datadog, Nagios
  - **Functionality**: Monitors server health metrics like CPU usage, memory consumption, disk I/O, and uptime.
- **Cloud Monitoring Tools**:
  - **Examples**: AWS CloudWatch, Azure Monitor, Google Cloud Operations Suite
  - **Functionality**: Monitors cloud resources and services, providing insights into performance, costs, and scalability.
- **Database Monitoring Tools**:
  - **Examples**: SolarWinds Database Performance Analyzer, Redgate SQL Monitor, Datadog Database Monitoring
  - **Functionality**: Monitors database performance, queries, and resource usage to optimize efficiency and prevent downtime.
- **Container Monitoring Tools**:
  - **Examples**: Prometheus, Grafana, Sysdig

- **Functionality**: Monitors containerized applications and orchestration platforms (like Kubernetes), providing insights into resource utilization and performance.

✓ **Installation of monitoring tools**

Installing monitoring tools can vary significantly based on the specific tool you choose and the environment (cloud, on-premises, hybrid) where you intend to deploy it. Below is a general guide for installing various types of monitoring tools, including application, networking, and infrastructure monitoring tools.

Practical Activity 3.1.2: Installing monitoring tools

Task:

1: Read the key readings 3.1.2

2. Referring to the key readings 3.1.2 you are requested to go to the computer lab to install the monitoring tools.

3: Present your work to classmate and trainer

4: Ask questions for clarifications if necessary

Key readings 3.1.2: Analysis of Performance Metrics and Feedback Data

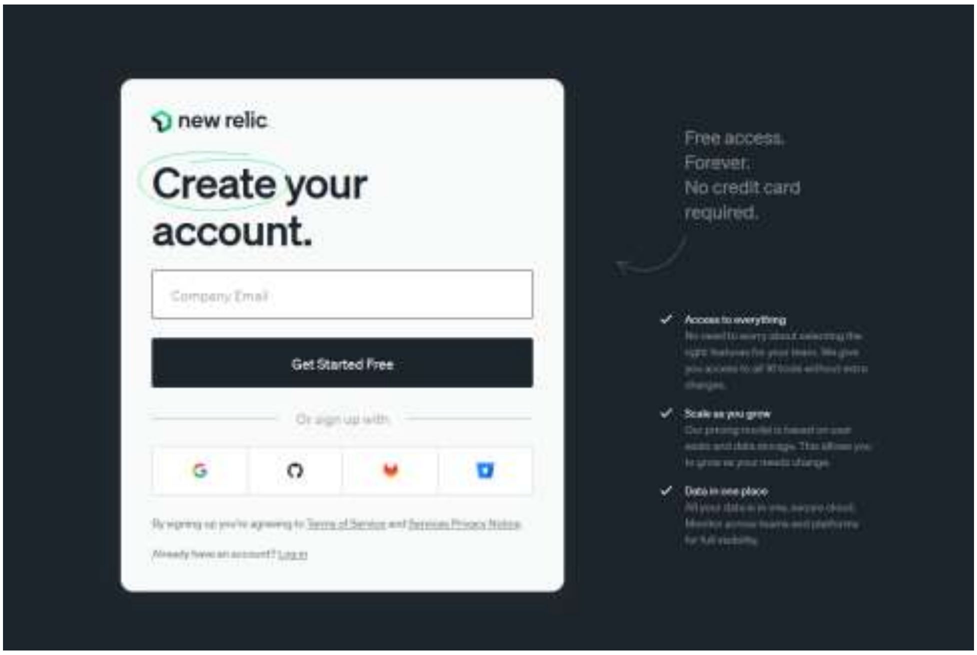
General Steps for Installation monitoring tools

1. **Prerequisites**:
   - Check the system requirements of the monitoring tool (CPU, RAM, disk space).
   - Ensure you have administrative access to the system or environment where you will install the tool.
   - Verify that necessary network configurations (firewall rules, proxy settings) are in place.
2. **Installation Steps**:
   - Follow the specific installation instructions provided in the documentation for the tool. Below are examples for different types of monitoring tools.

Application Monitoring Tool Installation Examples

Example 1: New Relic APM

1. **Sign Up**: Create a New Relic account.



new relic

Create your account.

Company Email

Get Started Free

Or sign up with

By signing up you're agreeing to Terms of Service and Services Privacy Notice.

Already have an account? Log in

Free access.
Forever.
No credit card required.

✓ Access to everything
No need to worry about selecting the right features for your team. We give you access to all 30 tools without extra charges.

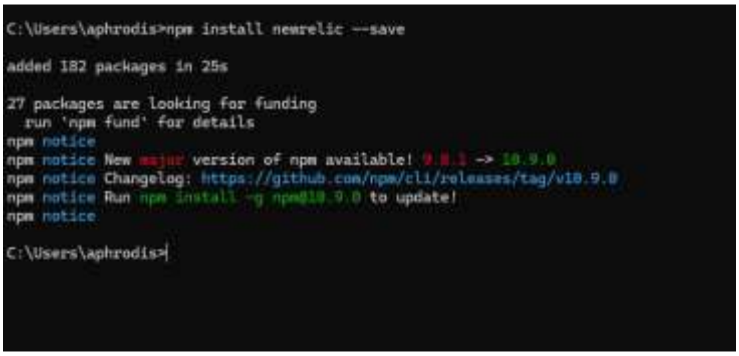
✓ Scale as you grow
Our pricing model is based on user seats and data storage. This allows you to grow as your needs change.

✓ Data in one place
All your data in one, secure cloud. Monitor across teams and platforms for full visibility.

2. **Install Agent**:

   - For Node.js applications, run:

→ **npm install newrelic –save**



```
C:\Users\aphrodis>npm install newrelic --save

added 182 packages in 25s

27 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 9.8.1 -> 10.9.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v10.9.0
npm notice Run npm install -g npm@10.9.0 to update!
npm notice

C:\Users\aphrodis>
```

   - For Python applications, run:

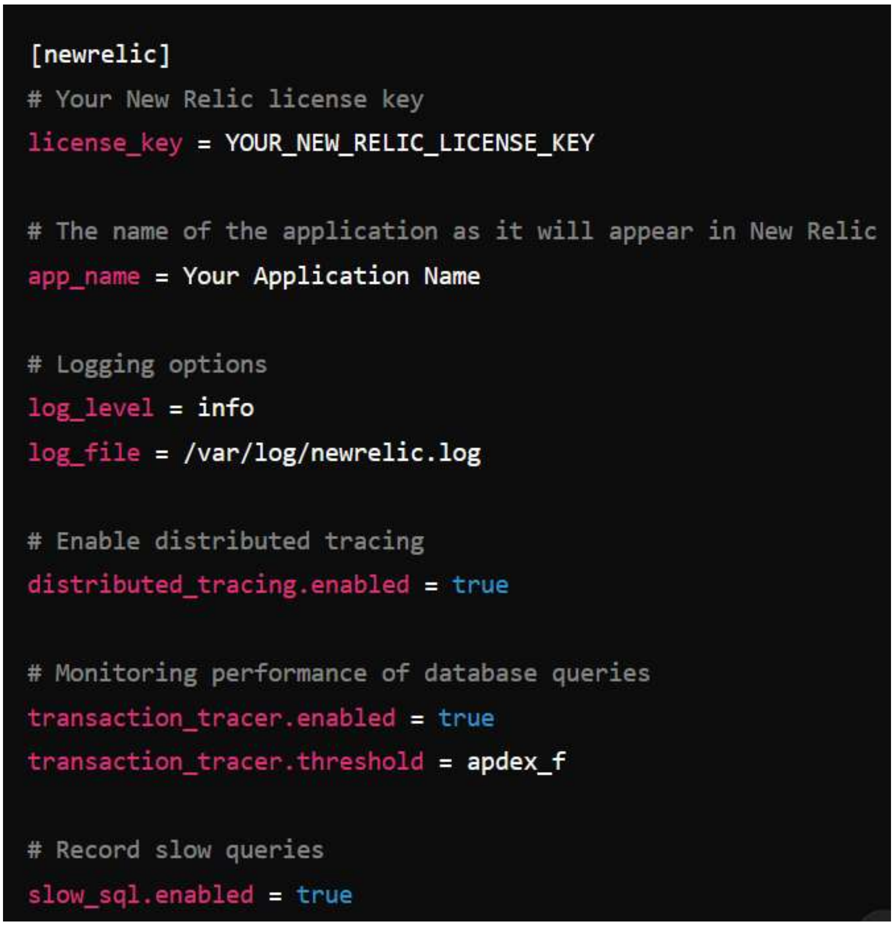
→ **pip install newrelic**

3. Configuration:

   - Create a configuration file (e.g., newrelic.ini) using the key provided in your New Relic account.

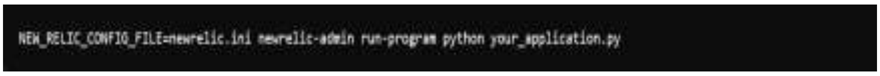
You can create a New Relic configuration file in your project root, such as newrelic.ini. This file is used to configure New Relic's APM (Application Performance Monitoring) for your application.

Here’s a basic template for a newrelic.ini file:



```
[newrelic]
# Your New Relic license key
license_key = YOUR_NEW_RELIC_LICENSE_KEY

# The name of the application as it will appear in New Relic
app_name = Your Application Name

# Logging options
log_level = info
log_file = /var/log/newrelic.log

# Enable distributed tracing
distributed_tracing.enabled = true

# Monitoring performance of database queries
transaction_tracer.enabled = true
transaction_tracer.threshold = apdex_f

# Record slow queries
slow_sql.enabled = true
```

- ○ Add the configuration file to your application.



```
NEW_RELIC_CONFIG_FILE=newrelic.ini newrelic-admin run-program python your_application.py
```

Application of learning 3.1.

Ketty is leading a team in a DevOps-driven project for a SaaS application. The team is focused on optimizing the application’s performance, improving deployment frequency, and reducing incident response times. As a DevOps Engineer, you are requested to install the required monitoring tool.

Indicative content 3.2: Analysis of performance metrics and feedback data

Duration: 4 hrs

Theoretical Activity 3.2.1: Identification of Performance metric and feedback

Tasks:

1: You are asked to answer the following questions:

i. List down significance of data analysis

ii. Give and explain types of data in DevOps

iii. Give the process of installation monitoring Tools

iv. Differentiate the following data in DevOps

a) Regular Review
b) Root Cause Analysis
c) Actionable Insights
d) Feedback Loop Integration

2: Write your answers on paper, flipchart, blackboard or whiteboard

3: Present your answers to trainer or classmate

4: Ask questions for clarifications if necessary

5: Read the Key reading 3.1.2

Key readings 3.1.1. Identification of Performance metric and feedback

✓ **Introduce performance metrics and Feedback Data**

Performance metrics and feedback data are essential tools for assessing and improving performance in various domains, including business, education, and technology. Here's an overview of both concepts:

Performance Metrics

Definition: Performance metrics are quantifiable measures used to evaluate the efficiency, effectiveness, and quality of a particular process, team, or individual. These metrics help organizations set goals, track progress, and make data-driven decisions.

Types of Performance Metrics:

1. **Financial Metrics**:
   - Revenue growth rate
   - Profit margin
   - Return on investment (ROI)
2. **Operational Metrics**:
   - Cycle time (time taken to complete a process)
   - Efficiency ratio (output/input)
   - Inventory turnover
3. **Customer Metrics**:
   - Customer satisfaction score (CSAT)
   - Net Promoter Score (NPS)
   - Customer retention rate
4. **Employee Performance Metrics**:
   - Productivity (output per employee)
   - Employee engagement score
   - Turnover rate
5. **Project Management Metrics**:
   - Schedule variance (difference between planned and actual project timelines)
   - Cost variance (difference between planned and actual project costs)
   - Quality metrics (defect rates, customer complaints)

Feedback Data

Definition: Feedback data refers to information collected from stakeholders (customers, employees, partners, etc.) about their experiences, opinions, and suggestions related to products, services, or processes. This data is crucial for continuous improvement.

Sources of Feedback Data:

1. **Surveys and Questionnaires**:
   - Customer satisfaction surveys
   - Employee engagement surveys
   - Exit interviews
2. **Direct Feedback**:
   - Customer complaints and compliments
   - Performance reviews and one-on-one meetings
3. **Observational Data**:
   - User interactions with products (e.g., usability tests)
   - Employee behavior in the workplace
4. **Social Media and Online Reviews**:
   - Comments and ratings on platforms like Yelp, Google Reviews, etc.
   - Engagement metrics on social media posts

Integrating Performance Metrics and Feedback Data

Integrating performance metrics with feedback data can provide a more comprehensive understanding of performance and areas for improvement. For example:

- **Identify Gaps**: Metrics can highlight performance gaps, while feedback can reveal the underlying reasons for those gaps.
- **Continuous Improvement**: Regularly assessing metrics alongside feedback allows for timely adjustments and enhancements to processes, products, or services.
- **Data-Driven Decision Making**: Combining quantitative metrics with qualitative feedback enables organizations to make more informed decisions, aligning strategies with actual stakeholder needs and experiences.

✓ **Describe significance of Data Analysis**

Data analysis plays a crucial role in various sectors, including business, healthcare, education, and technology. Here are several key aspects of its significance:

1. Informed Decision-Making

Data analysis helps organizations make informed decisions based on empirical evidence rather than intuition. By interpreting data trends and patterns, decision-makers can evaluate the potential outcomes of their choices and reduce uncertainty.

2. Identifying Trends and Patterns

Through data analysis, organizations can identify trends and patterns that might not be immediately obvious. This insight allows them to adapt strategies, forecast future outcomes, and anticipate market changes.

3. Enhancing Operational Efficiency

Data analysis can uncover inefficiencies in processes, workflows, or supply chains. By identifying bottlenecks and areas for improvement, organizations can streamline operations, reduce costs, and enhance overall efficiency.

4. Improving Customer Experience

Analyzing customer data helps organizations understand preferences, behaviors, and needs. This information allows for personalized marketing, tailored services, and improved customer interactions, ultimately leading to higher customer satisfaction and loyalty.

5. Risk Management

Data analysis aids in identifying potential risks and vulnerabilities within an organization. By analyzing historical data, companies can develop risk mitigation strategies, prepare for potential crises, and make proactive adjustments to their operations.

6. Driving Innovation

Insights derived from data can inspire innovation by revealing new opportunities for product development or service enhancement. Organizations can leverage data to explore uncharted markets, refine existing offerings, or create entirely new solutions.

7. Performance Measurement

Data analysis provides key performance indicators (KPIs) that enable organizations to measure success against predefined goals. By continuously monitoring performance metrics, organizations can assess their progress and make necessary adjustments.

8. Competitive Advantage

Organizations that effectively analyze data can gain a competitive edge over rivals. By leveraging data insights, they can respond more quickly to market changes, understand consumer behavior better, and optimize their offerings to meet customer demands.

9. Facilitating Collaboration

Data analysis fosters a culture of collaboration by providing a common platform for discussion and decision-making. When teams share data insights, they can align their objectives and work together more effectively towards shared goals.

10. Supporting Research and Development

In fields like healthcare, finance, and technology, data analysis supports research and development efforts. By analyzing large datasets, researchers can uncover new findings, test hypotheses, and validate scientific theories.

- ✓ **Describe types of data in DevOps**

In DevOps, data is critical for monitoring, optimizing, and automating workflows across the software development lifecycle. Different types of data are used to assess the performance of applications, infrastructure, and teams, enabling continuous integration, continuous delivery (CI/CD), and continuous improvement. Below are the major types of data in DevOps:

1. Application Performance Data

This type of data focuses on how applications perform in real-time, including aspects like speed, responsiveness, and reliability.

- **Metrics**:
    - Response time (latency)

  - Throughput (requests per second)
  - Error rates (number of failed requests)
  - Availability (uptime)
- **Tools**:
  - Application Performance Monitoring (APM) tools like New Relic, Datadog, or Dynatrace are used to collect and analyze this data.

2. Infrastructure Data

This refers to data that is collected from the underlying hardware, virtual machines, containers, and cloud services that host the applications.

- **Metrics**:
  - CPU usage
  - Memory consumption
  - Disk I/O (input/output)
  - Network bandwidth and latency
- **Tools**:
  - Monitoring tools like Prometheus, Nagios, or AWS CloudWatch collect and report on infrastructure metrics.

3. Log Data

Log data is generated by applications, servers, and other infrastructure components. Logs contain detailed information about the operations and events occurring within the system.

- **Examples**:
  - Error logs
  - Access logs
  - Event logs (for security or debugging)
- **Tools**:
  - Log management tools such as ELK Stack (Elasticsearch, Logstash, Kibana), Splunk, or Fluentd are used to centralize, search, and analyze log data.

4. Build and Deployment Data

In DevOps, CI/CD pipelines automate the process of building, testing, and deploying applications. Data generated from these pipelines helps in ensuring software is being developed and released efficiently.

- **Metrics**:
  - Build success/failure rates
  - Build times
  - Deployment frequency
  - Mean time to recovery (MTTR) after a failed deployment
- **Tools**:
  - CI/CD tools like Jenkins, CircleCI, GitLab CI, and Azure Pipelines provide data on build and deployment performance.

5. Version Control Data

This type of data tracks code changes and how teams collaborate during the software development process.

- **Metrics**:
  - Number of commits
  - Code change velocity
  - Pull request (PR) approval time
  - Merge conflicts
- **Tools**:
  - Version control systems like Git (GitHub, GitLab, Bitbucket) provide insights into code contributions, code reviews, and collaboration efficiency.

6. Security Data

Security is a crucial part of DevOps, often referred to as DevSecOps. Security data helps identify vulnerabilities, security breaches, and compliance issues.

- **Metrics**:
  - Number of vulnerabilities detected

  - Security patching time
  - Failed security tests
  - Audit logs (user access, system changes)
- **Tools**:
  - Tools like Snyk, OWASP ZAP, and HashiCorp Vault help analyze and secure DevOps pipelines by tracking vulnerabilities and managing secrets.

7. Incident and Alerting Data

This type of data is related to the management of incidents and the notification systems that trigger alerts when something goes wrong.

- **Metrics**:
  - Incident response time
  - Number of incidents per week/month
  - Alert frequency and accuracy
  - Mean time to resolve (MTTR)
- **Tools**:
  - Alerting systems like PagerDuty, OpsGenie, or VictorOps notify teams when specific metrics or logs indicate potential issues.

8. User Experience Data

User experience (UX) data focuses on how end-users interact with the application, providing insights into usability and satisfaction.

- **Metrics**:
  - Page load times
  - User session data
  - Bounce rates (users leaving the app after a short interaction)
  - Conversion rates (percentage of users taking desired actions)
- **Tools**:
  - Tools like Google Analytics, Hotjar, or Mixpanel provide insights into user behavior and application usability.

9. Automation Data

Automation is a core aspect of DevOps, especially in CI/CD pipelines and infrastructure management. Data from automated systems is critical for assessing the effectiveness of automation efforts.

- **Metrics**:
  - Number of automated jobs
  - Automation success rates
  - Time saved through automation
  - Number of manual interventions needed
- **Tools**:
  - Automation tools like Ansible, Terraform, or Chef provide data on how effectively automated tasks are executed.

10. Collaboration and Workflow Data

In a DevOps environment, efficient collaboration between teams (development, operations, security, etc.) is critical. Data related to communication and workflows helps improve team collaboration and efficiency.

- **Metrics**:
  - Number of tasks completed within a sprint
  - Collaboration frequency (e.g., via Slack, Jira)
  - Cycle time (time it takes from idea to implementation)
  - Bottlenecks in the workflow
- **Tools**:
  - Project management and collaboration tools like Jira, Trello, or Microsoft Teams capture data on team communication and task management.

11. Business and Financial Data

This data is used to measure the overall business impact of DevOps initiatives, such as how well development and operational efficiencies translate into revenue, cost savings, or customer retention.

- **Metrics**:
  - Return on investment (ROI) for DevOps initiatives

- Customer acquisition cost (CAC)
- Customer lifetime value (CLTV)
- Revenue growth related to faster releases and better uptime

- **Tools**:
  - Business intelligence platforms like Tableau, Power BI, or Looker help visualize and analyze the financial impact of DevOps

✓ **Utilizing Monitoring Tools**

Utilizing monitoring tools is a key practice in DevOps to ensure that applications, systems, and infrastructure are performing optimally. Monitoring tools help detect issues in real-time, ensure system reliability, and provide insights for continuous improvement. Below is a comprehensive guide on how to effectively utilize monitoring tools in a DevOps environment:

1. Types of Monitoring Tools

DevOps monitoring can be categorized into different types depending on the focus area:

- **Infrastructure Monitoring**: Monitors the underlying hardware, virtual machines, containers, and networks.
  - **Examples**: Nagios, Prometheus, Zabbix, AWS CloudWatch.
- **Application Performance Monitoring (APM)**: Focuses on the performance and health of applications.
  - **Examples**: New Relic, Datadog, Dynatrace, AppDynamics.
- **Log Monitoring**: Tracks log files generated by applications and infrastructure for troubleshooting and auditing.
  - **Examples**: ELK Stack (Elasticsearch, Logstash, Kibana), Splunk, Fluentd.
- **Network Monitoring**: Monitors the performance, traffic, and latency in the network.
  - **Examples**: SolarWinds, Wireshark, PRTG Network Monitor.
- **Security Monitoring**: Focuses on identifying security threats and ensuring system integrity.
  - **Examples**: Snyk, Qualys, HashiCorp Vault, OSSEC.

2. Key Metrics to Monitor

Monitoring tools help collect critical metrics that inform system health and performance. Here are some important metrics:

- **System Metrics**:
  - **CPU Usage**: Monitors how much of the CPU is being used.
  - **Memory Usage**: Tracks memory allocation and usage patterns.
  - **Disk I/O**: Monitors the input/output operations on the disk.
  - **Network Latency**: Measures delays in network data transmission.
- **Application Metrics**:
  - **Response Time**: The time it takes for an application to respond to a request.
  - **Error Rates**: The frequency of errors or failed requests.
  - **Request Throughput**: The number of requests processed per second.
  - **Transaction Time**: Time taken for a specific transaction to complete.
- **Security Metrics**:
  - **Failed Login Attempts**: Tracks the number of failed authentication attempts.
  - **Vulnerability Detection**: Measures the number and severity of security vulnerabilities.
  - **Security Patch Application**: Tracks how quickly patches are applied to systems.
- **Business Metrics**:
  - **Uptime/Availability**: Measures the percentage of time a system or application is operational.
  - **MTTR (Mean Time to Recovery)**: The average time it takes to recover from a failure.
  - **MTBF (Mean Time Between Failures)**: The average time between system failures.

3. Implementing Monitoring Tools in DevOps

To effectively use monitoring tools in DevOps, follow these steps:

a. Identify Key Metrics and KPIs

Determine which metrics are most critical for your infrastructure, applications, and business goals. Collaborate with stakeholders, including developers, operations, and business teams, to define key performance indicators (KPIs) such as system uptime, response time, or error rates.

b. Choose the Right Monitoring Tool

Select monitoring tools that fit your environment and objectives. For example:

- If you’re using containers, **Prometheus** with **Grafana** might be suitable.
- For cloud infrastructure, **AWS CloudWatch** is a native solution for AWS environments.
- For real-time log analysis, **ELK Stack** (Elasticsearch, Logstash, Kibana) is a popular choice.

c. Implement Continuous Monitoring

Set up continuous monitoring for both applications and infrastructure, with a focus on automated, real-time alerts. This ensures that your team is immediately notified of any performance issues, such as high CPU usage or application errors.

d. Set Up Alerting Systems

Use alerting systems that notify the right people at the right time. Configure thresholds for important metrics, and integrate alerts with communication platforms such as Slack, Microsoft Teams, or incident management systems like PagerDuty or OpsGenie.

e. Visualize Data

Use dashboards and visualization tools to make data more understandable. **Grafana**, for example, provides customizable dashboards that can display metrics from multiple sources like Prometheus, Elasticsearch, or Datadog. Visualizing data in real-time helps teams spot anomalies quickly.

f. Establish Baselines

Create performance baselines to understand normal behavior for your systems. This makes it easier to spot deviations or unusual activity, which might indicate an issue.

For instance, if the average CPU usage for an application is 30%, but it suddenly jumps to 90%, the monitoring tool can trigger an alert.

g. Perform Log Aggregation and Analysis

Aggregate logs from various services, applications, and infrastructure into a centralized location for easier analysis and troubleshooting. **Log management tools** like the ELK Stack or Splunk provide real-time insights into system behaviors and error tracking.

h. Enable Auto-Remediation

Some monitoring tools allow for automated remediation actions. For example, if a system exceeds a CPU usage threshold, the monitoring system could automatically spin up new instances or restart services to prevent failures.

4. Best Practices for Utilizing Monitoring Tools

- **Start Small, Scale Gradually**: Begin by monitoring the most critical parts of your system, then expand monitoring as needed. Don't overwhelm your team with too much data at once.
- **Set Clear Alert Thresholds**: Avoid alert fatigue by carefully setting thresholds that are meaningful and actionable. Alerts should trigger only when action is needed.
- **Perform Regular Audits and Updates**: Regularly review and update your monitoring setup, especially as your systems evolve. Ensure that monitoring configurations are up-to-date with any infrastructure changes.
- **Monitor the Full DevOps Pipeline**: Don’t just monitor the production environment—track the entire CI/CD pipeline. This includes monitoring build and deployment processes, test environments, and staging environments to catch issues earlier in the development cycle.
- **Correlation Between Metrics**: Look for correlations between different metrics to get deeper insights. For example, an increase in error rates may correlate with higher memory usage, helping you troubleshoot root causes faster.
- **Automate Reporting**: Use automated reports to track long-term performance trends. These reports can help identify issues that may not be obvious in real-time monitoring, such as gradual performance degradation.
- **Security Monitoring**: Integrate security monitoring into your DevOps practices (DevSecOps). This ensures continuous monitoring of vulnerabilities and security risks alongside performance metrics.

5. Popular Monitoring Tools in DevOps

- **Prometheus**: An open-source monitoring tool designed for reliability and scalability, often used with **Grafana** for visualization.
- **Datadog**: A comprehensive monitoring platform for cloud applications, infrastructure, logs, and user experience.
- **Nagios**: A widely used monitoring solution for infrastructure, offering robust alerting and reporting capabilities.
- **New Relic**: A powerful APM tool that helps monitor application performance, transactions, and errors.
- **ELK Stack (Elasticsearch, Logstash, Kibana)**: A popular log aggregation and analysis tool for real-time log monitoring and visualization.

✓ **Analysing Data in DevOps**

Analyzing data in DevOps is crucial for continuous improvement, faster issue resolution, and optimizing performance across the entire software development and operations lifecycle. It involves regular data reviews, identifying the root causes of issues, deriving actionable insights, and integrating feedback loops for continuous feedback and improvement. Below is a detailed explanation of each step:

1. Regular Review

Regular review of collected data ensures that DevOps teams are continuously monitoring performance, identifying trends, and catching potential problems before they escalate.

Key Activities:

- **Scheduled Data Audits**: Regularly audit data from key performance indicators (KPIs) such as infrastructure performance, build and deployment data, and application health. This helps identify bottlenecks or anomalies early.
- **Dashboard Monitoring**: Utilize dashboards for real-time visualization of metrics (e.g., Grafana, Datadog). Dashboards provide a consolidated view of data from various systems and help monitor performance at a glance.
- **Trend Analysis**: Identify long-term patterns in metrics such as response times, error rates, or system load. Regular reviews help track if performance is improving or degrading over time.

- **Security Reviews**: Continuously monitor security metrics to ensure compliance and prevent vulnerabilities. This includes reviewing access logs, security patches, and vulnerabilities.

Benefits:

- Early detection of issues
- Consistent performance tracking
- Informed decision-making based on historical data

2. Root Cause Analysis (RCA)

Root Cause Analysis (RCA) is the process of identifying the underlying cause of an issue or failure within the system. This helps prevent recurring problems by addressing the core issue rather than just the symptoms.

Key Steps:

- **Identify the Problem**: Begin by defining the problem clearly. For example, if a service is experiencing downtime, define the extent of the downtime, when it started, and what systems are affected.
- **Collect Data**: Gather all relevant data, including system logs, performance metrics, and user reports. Monitoring tools like ELK Stack or Splunk are useful for reviewing logs and pinpointing when and where the issue occurred.
- **Analyze Metrics**: Look for deviations or anomalies in key metrics, such as high CPU usage, memory leaks, network latency spikes, or unexpected error logs, to narrow down the cause of the issue.
- **Conduct the “5 Whys”**: Use the "5 Whys" technique by asking “why” repeatedly until the fundamental cause of the issue is identified. This can help dig deeper into the problem.
- **Post-Mortem**: After resolving the issue, conduct a post-mortem to document what went wrong, why, and how the problem was fixed. This is crucial for learning and preventing future incidents.

Benefits:

- Prevents recurring issues
- Improves system reliability and stability
- Increases efficiency by reducing firefighting

3. Actionable Insights

Actionable insights are derived from data analysis and can be directly applied to improve performance, address problems, and optimize workflows.

Key Approaches:

- **Analyze Performance Metrics**: Derive insights from key metrics such as application response time, infrastructure utilization, and error rates. For example, if latency is increasing, an insight might be to scale up resources or optimize the codebase.
- **Use Predictive Analytics**: Predictive analytics can forecast future issues by analyzing historical data. Machine learning tools can analyze trends and predict potential problems like server outages or performance degradation, enabling proactive measures.
- **Optimization Suggestions**: Actionable insights should suggest specific improvements. For example, if deployment times are increasing, analyze the pipeline to identify inefficiencies and propose steps to streamline the process.
- **Cost Optimization**: Insights from cloud infrastructure monitoring (e.g., AWS CloudWatch) can help reduce unnecessary resource utilization, optimizing costs by scaling down unused resources or switching to reserved instances.

Benefits:

- Enables informed decision-making
- Focuses on improvements that add immediate value
- Drives continuous improvement across the CI/CD pipeline

4. Feedback Loop Integration

In DevOps, the feedback loop refers to the continuous cycle of feedback from various stakeholders (developers, operations teams, and customers) to inform future decisions and improvements.

Key Activities:

- **Internal Feedback**: Use data from automated tests, monitoring tools, and alerts to provide feedback to development and operations teams. For instance, test failures in the CI/CD pipeline can give immediate feedback to developers to fix broken code.

- **Customer Feedback**: Incorporate user feedback on application performance and user experience (UX) through tools like customer satisfaction surveys, bug reports, or user behavior data (e.g., Google Analytics).
- **Rapid Feedback in CI/CD**: A key goal in DevOps is rapid feedback from code deployment and integration. The faster teams get feedback, the sooner they can address issues. Automated testing, monitoring, and alerting ensure immediate feedback when something goes wrong.
- **Continuous Improvement Cycle**: Integrate feedback into future development cycles to continuously improve the system. Each feedback iteration should result in improved system performance, better user experience, and optimized workflows.
- **Automated Feedback**: Automate feedback collection through monitoring systems, log management, and testing pipelines, ensuring that all data is automatically fed back into the system for review and analysis.

Benefits:

- Shortens the development cycle
- Promotes a culture of continuous improvement
- Enhances system quality and user satisfaction

Integration of All Steps for Maximum Efficiency

1. **Regular Review**: Continuous reviews of infrastructure, application, and security data provide baseline insights into system performance.
2. **Root Cause Analysis**: When issues arise, conduct RCA to identify the underlying cause, preventing future occurrences of the same problem.
3. **Actionable Insights**: After identifying root causes and performance trends, generate actionable insights that lead to concrete improvements in development practices, system performance, and operational efficiency.
4. **Feedback Loop Integration**: Close the loop by feeding the insights and improvements back into the DevOps pipeline, leading to faster, more reliable deployments and a more resilient system.

Practical Activity 3.2.2: Analysing the performance metric s and feedback data

Task:

1: You are asked to go to the computer Lab to perform the following task:

i. Go to the computer lab to analyse performance metrics and feedback data using New relic tool.

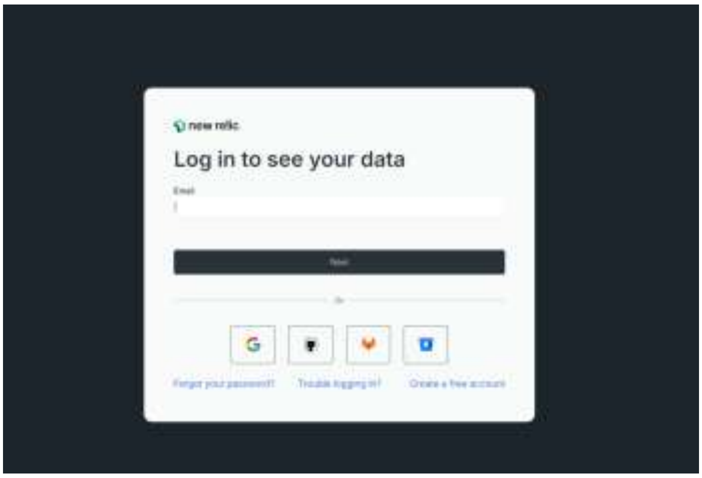
2.Present your work to trainer and classmate

3: Ask questions for clarifications if necessary

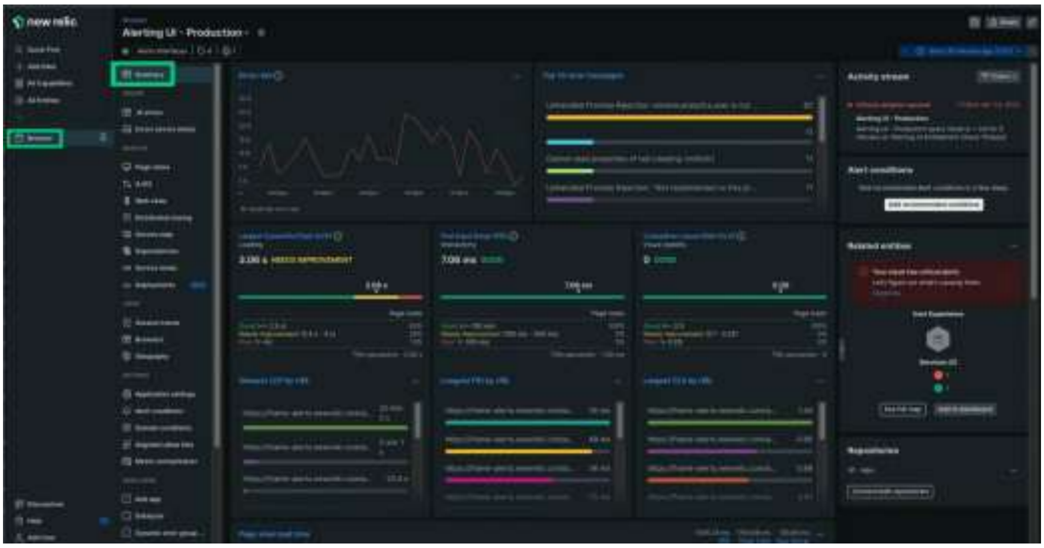
Key readings 3.2.2 Analysing of performance and feedback data

- **Steps involved to analyse performance and feedback data**

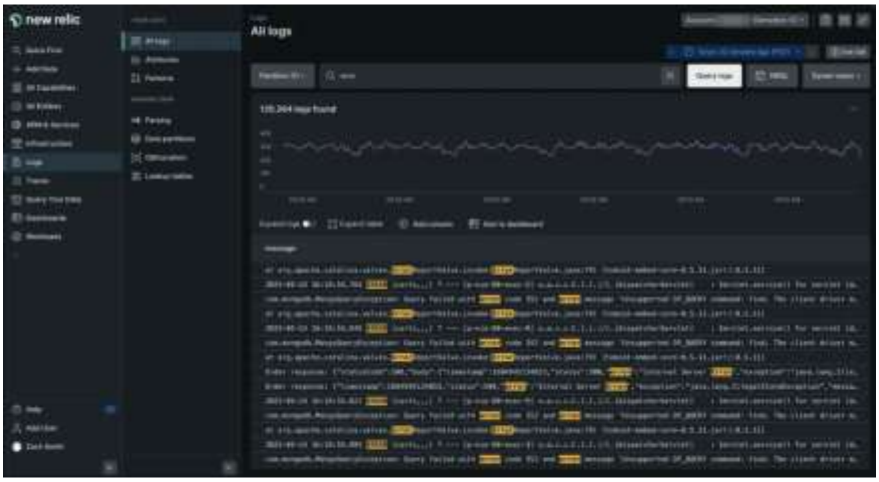
Step 1:log in the new relic tools



Step 2: select a performance needed to analyse



Step 3: Analyse logs of developed system



Points to Remember

- Before performing any monitoring activity, you must understand the different types of monitoring tools such as networking monitoring tools, application monitoring tool and infrastructure monitoring tools.
- When installing a monitoring tool in a DevOps environment, first download tool then after follow the installation steps until tool will completed.
- Describe the type of data such as log data, infrastructure data, application performance data and build and deployment data, version control data, security data user experience data, automation data, collaboration and workflow data and business and financial data.

- While writing there are many Key Points respect for Comprehensive Work Report
- Apart from using commands during installation, there are other monitoring tools which can be downloaded from the official manufacture website and then be used during installation: these include washark, prometheus, grafana, datadog, nagios, zabbix and ELK stack.

Application of learning 3.2.

RFTX Ltd company has a hosted web application on online server for managing its transactions. It needs a competent IT Technician capable to analyse the data in DevOps. As DevOps Engineer, you are requested to check performance metrics and feedback data.

Indicative content 3.3: Documentation of monitoring report

Duration: 2 hrs

Theoretical Activity 3.3.1: Identification of monitoring report

Tasks:

1: You are asked to answer the following questions:

i. What is purpose of monitoring report

ii. Define the following items concepts

a) Executive Summary

b) Key Metrics

c) Report findings

d) Trends Analysis

e) Alerts and Incidents

f) Action Items

g) Optimization or remediation.

h) Conclusion

2: Provide your answers on paper, flipchart, blackboard or whiteboard

3: Present your answers to whole class trainees and trainer

4: Ask questions for clarifications if necessary

5: Read the Key reading 3.3.1

Key readings 3.3.1: Identification of monitoring report

- **Monitoring Report for DevOps Operations**

✓ **Executive Summary**

In this report, we summarize the current state of the DevOps monitoring system and its impact on operational efficiency. Our monitoring covers the entire pipeline from code commits to deployment and production environments using various tools to track system health, performance, and availability. Continuous monitoring is critical for early detection of incidents, minimizing downtime, and ensuring the reliability and scalability of our infrastructure.

We present an overview of key metrics, findings, trends, and incidents from the monitoring data collected over the reporting period, highlighting areas for improvement and success.

✓ **Key Metrics**

Monitoring metrics provide visibility into the performance and stability of the DevOps pipeline and production systems. Here are the primary key metrics tracked:

- **CPU Utilization:** Indicates system load across servers and critical infrastructure.
  Normal Range: 30%-70%.
  Anomalies: Spikes above 85% detected in high-traffic periods.

- **Memory Usage:** Monitors the usage of system memory across instances.
  Normal Range: 40%-75%.
  Anomalies: Consistent high usage in specific microservices.

- **Disk I/O:** Measures the rate at which data is read from or written to storage.
  Normal Range: 100-500 MB/s.
- Anomalies: Occasional lags during deployment processes.

- **Application Latency:** Tracks the time taken for requests to be processed by the system.
  Normal Range: 100-300 ms.
  Anomalies: Increased latency detected post-deployment.

- **Error Rates:** Keeps track of HTTP error responses (4xx, 5xx) and internal system errors.
  Normal Threshold: Less than 1% of total requests.
  Anomalies: Noticeable increase during scaling events.

- **Deployment Frequency:** Number of releases within a defined period.
  Average: 10 deployments/week.
  Stability: 90% success rate.

✓ **Report Findings**

The monitoring systems indicate that overall system health is within acceptable limits, but several areas show room for optimization:

Infrastructure Performance: Generally stable, though occasional resource saturation (CPU, memory) was observed during peak hours.
Service Latency: Latency issues surfaced during periods of high traffic, particularly following updates to key microservices.
Error Rate Spikes: Error rates spiked during certain deployments, suggesting potential issues with CI/CD pipelines or application-level bugs.
Alert Management: Response to alerts has been timely, minimizing downtime.

✓ **Trends Analysis**

Scaling and Performance:
Horizontal scaling has proven effective in managing increased traffic load, though some inefficiencies were observed in automatic scaling mechanisms, leading to delayed resource provisioning.

- **Error Rates Over Time**:

Error rates have been trending downward, suggesting that recent code changes and infrastructure updates have improved stability. However, occasional spikes indicate room for better automated testing and pre-deployment checks.

- **Deployment Impact:**

System performance tends to dip immediately following deployments. This suggests a need to refine blue-green or canary deployment strategies to minimize user impact during updates.

- **System Uptime:**

Uptime has been consistently above 99.9%, demonstrating strong reliability. However, there are isolated incidents where downtime occurred due to unanticipated failures in the monitoring system itself.

✓ **Alerts and Incidents**

During the reporting period, several alerts were triggered:

- **CPU Utilization Alert (Incident #X123):**

Triggered on October 5th, when CPU usage spiked to 95% for several hours due to a poorly optimized database query. The issue was resolved after patching the query.

- **Memory Leak Incident (Incident #X145):**

Detected on October 9th in one of the microservices, leading to an emergency deployment rollback. The leak caused the service to crash under heavy load.

- **Network Latency Alert (Incident #X158):**

On October 12th, network latency increased significantly during a code deployment. Root Cause analysis revealed a configuration error in the load balancer.

✓ **Action Items**

Based on the analysis of metrics and findings, several action items have been identified for improving the performance and stability of our DevOps pipeline and infrastructure:

1. **Improve Automated Scaling Mechanisms**:
   - Investigate and optimize the auto-scaling policies to ensure a faster response to peak traffic, especially during deployment windows.
   - Responsible Team: Infrastructure Team.
   - Timeline: 2 weeks.
2. **Optimize CI/CD Pipeline**:
   - Review the continuous integration pipeline to reduce the deployment time and minimize errors during release cycles. Focus on introducing better testing automation and validation steps before pushing updates to production.
   - Responsible Team: DevOps Team.
   - Timeline: 1 month.
3. **Enhance Error Monitoring and Resolution**:
   - Implement more robust logging and error tracking tools (e.g., Datadog, Sentry) to improve visibility into errors and reduce the time to identify and resolve them.
   - Responsible Team: Development & QA Teams.
   - Timeline: Ongoing.
4. **Database Query Optimization**:
   - Review and optimize database queries, particularly those causing high CPU usage and long processing times.
   - Responsible Team: Database Admins.
   - Timeline: 3 weeks.
5. **Refine Deployment Strategy**:
   - Move towards a more granular deployment approach using blue-green deployments or canary releases to minimize the impact on end-users during upgrades.
   - Responsible Team: Release Management.
   - Timeline: 1 month.

- ✓ **Optimization or Remediation**

After analyzing the report findings, several remediation measures have been suggested to optimize the current systems:

1. **Memory Leak Resolution**:
   - Implement tools for automated memory leak detection during the development and testing phase. Monitor microservices more closely, focusing on memory consumption patterns.
   - *Outcome*: Memory-related incidents should decrease.
2. **Load Balancer Configuration**:
   - Correct the load balancer configurations to handle traffic more efficiently, reducing network latency during deployment phases.
   - *Outcome*: Lower latency and improved response times during traffic surges.
3. **Improving Application Performance**:
   - Profiling and optimizing slow application services and API endpoints that are contributing to latency. Use tools like New Relic or AppDynamics to pinpoint bottlenecks.
   - *Outcome*: Enhanced performance, reduced load times.
4. **Refining Alert Thresholds**:
   - Review and adjust monitoring alert thresholds to better align with acceptable performance limits, reducing false positives while still catching critical issues early.
   - *Outcome*: More actionable alerts, faster incident response.

- ✓ **Conclusion**

The monitoring report has shown that while the overall health of the DevOps pipeline and infrastructure is stable, there are a few key areas requiring immediate attention and optimization. Action items such as refining auto-scaling mechanisms, improving the CI/CD process, and better error tracking will ensure higher efficiency, faster response times, and improved user experience.

Continuous improvements in monitoring, especially around memory consumption, CPU usage, and system latency, will help maintain system uptime and scalability. Ongoing efforts to automate remediation and improve root cause analysis will reduce downtime and optimize future deployments.

- ✓ **Appendix**

The appendix contains detailed charts, graphs, and raw data from the monitoring tools. These visual representations help illustrate the findings and trends mentioned in the report.

1. **CPU Usage Graph**:
    - Displays CPU usage across all servers for the past month, highlighting periods of high load and anomalies.
2. **Memory Consumption Chart**:
    - A breakdown of memory usage by service, showing patterns over time and during peak traffic.
3. **Disk I/O Report**:
    - Detailed analysis of disk read/write speeds, demonstrating the impact of deployments on disk performance.
4. **Error Rate Trends**:
    - A graph displaying the error rate trends over the last few weeks, including spikes during deployment windows.
5. **Application Latency Statistics**:
    - Visual representation of average latency per service, showing improvements and occasional degradations after each deployment.
6. **Raw Data**:
    - Logs, resource usage metrics, and incident alerts as collected by monitoring tools like Prometheus, Grafana, and CloudWatch. Raw data is available for review by technical teams to further investigate specific incidents.

Practical Activity 3.3.2: writing a report

Task:

1: You are asked to go to the computer Lab to perform the following task:

After data analysis, migration and deployment of software project to online server, you are requested to create a report of work done.

2: Provide your answers on paper, flipchart, blackboard or whiteboard

3: Present your answers to whole class trainees and trainer

4: Ask questions for clarifications if necessary

5: Read the Key reading 3.3.2

Key readings 3.3.2 writing a report

- **Steps for writing a report**

Step 1. Executive Summary

This report assesses the current state of DevOps practices at XYZ Enterprises, focusing on areas such as continuous integration (CI), continuous delivery (CD), infrastructure automation, and security. The objective is to evaluate pipeline efficiency, identify bottlenecks, and recommend improvements to enhance deployment speed, reliability, and overall productivity.

Step 2. Objective and Scope

Objective: To assess and improve the efficiency of the DevOps workflow, focusing on key areas of the CI/CD pipeline, infrastructure management, and automation.

Scope:

- Continuous Integration and Continuous Delivery (CI/CD)
- Automation and Infrastructure as Code (IaC)
- Monitoring and Logging
- DevSecOps (security integration in DevOps)

Step 3. Data Collection

Data was gathered from key tools used in the current DevOps processes:

- **Source Control**: GitLab
- **CI/CD**: Jenkins, GitLab CI
- **Containerization**: Docker, Kubernetes
- **Cloud Infrastructure**: AWS (EC2, RDS, Lambda)
- **Monitoring**: Prometheus, Grafana
- **Security Tools**: Snyk, Aqua Security
- **Logging**: ELK Stack (Elasticsearch, Logstash, Kibana)

Data collection took place over a 3-month period (July to September 2024) and was supported by performance logs, monitoring dashboards, and interviews with the development, operations, and security teams.

Step 4. Key Performance Indicators (KPIs)

The following KPIs were tracked to evaluate the effectiveness of the DevOps pipeline:

1. **Deployment Frequency**: How often code is deployed to production.
2. **Lead Time for Changes**: The time taken from code commit to production deployment.
3. **Mean Time to Recovery (MTTR)**: Average time to recover from a failure or incident.
4. **Change Failure Rate**: Percentage of deployments that result in failure or rollback.
5. **Infrastructure Costs**: Cloud usage and cost efficiency for production workloads.

Step 5. Current State Assessment

5.1 Continuous Integration and Continuous Delivery (CI/CD)

- **Deployment Frequency**: On average, deployments occur **3 times per week**, which is aligned with industry standards for an organization of XYZ's size. However, the team reports **occasional rollback issues** due to insufficient pre-deployment testing.
- **Lead Time for Changes**: The lead time is **36 hours** from code commit to deployment, primarily due to manual approval processes and limited test automation coverage.
- **Change Failure Rate**: The failure rate is **10%**, which is higher than the target of 5%, due to intermittent issues in the integration tests and environment mismatches between staging and production.

5.2 Infrastructure and Automation

- **Infrastructure Management**: AWS infrastructure is heavily utilized but exhibits some inefficiencies. Resource utilization reports indicate **over-provisioned EC2 instances** running at low capacity, increasing operational costs by 15%.
- **Kubernetes Setup**: Kubernetes clusters are in place but are missing autoscaling configurations. Some applications experience **resource contention** under high load due to lack of proper resource quotas.
- **Infrastructure as Code (IaC)**: Terraform is used for infrastructure management, but the implementation is incomplete, with some configurations still managed manually, resulting in inconsistent environments across staging and production.

5.3 Monitoring and Logging

- **Monitoring Tools**: Prometheus and Grafana provide real-time insights, but alert thresholds are too broad, leading to **false alarms** and alert fatigue among the operations team.
- **Logging Setup**: The ELK stack is well integrated, but there is **no consistent log rotation policy**, resulting in large log files and storage issues.

5.4 Security (DevSecOps)

- **Security in Pipeline**: Static code analysis (SAST) and dynamic application security testing (DAST) are part of the CI pipeline using Snyk. However, **vulnerabilities** are often discovered late in the pipeline, requiring urgent fixes before release.
- **Access Management**: Role-based access control (RBAC) is implemented, but **audit trails** are incomplete, making it difficult to track privileged access changes and activities.

Step 6. Analysis

Based on the data collected, several inefficiencies and areas for improvement have been identified:

1. **Pipeline Delays**: The lead time is affected by manual interventions and limited automated testing. Implementing **test automation** could reduce this delay and improve deployment frequency.
2. **Infrastructure Costs**: AWS infrastructure is under-optimized, leading to **wasted resources** and higher-than-necessary operational costs.
3. **Monitoring Fatigue**: The current alert configuration produces too many alerts, most of which are false positives. Fine-tuning these alerts can improve response times and reduce operational burden.

4. **Security Integration**: Vulnerabilities are being caught late in the CI/CD pipeline, indicating a need to shift security checks earlier in the process to **catch issues earlier** and prevent last-minute delays.

Step 7. Recommendations

7.1 Continuous Integration and Continuous Delivery

- **Automated Testing**: Implement comprehensive test automation using tools such as **Selenium** for UI tests and **JUnit** for unit tests to improve testing coverage and reduce manual testing.
- **Pipeline Optimization**: Use **Blue-Green Deployment** or **Canary Releases** to reduce the impact of failed deployments on production.
- **Approval Process**: Consider automating parts of the manual approval process to streamline the lead time from code commit to production.

7.2 Infrastructure and Automation

- **Optimize AWS Resources**: Use **AWS Trusted Advisor** and **Cost Explorer** to identify underutilized resources and right-size EC2 instances for cost savings.
- **Kubernetes Autoscaling**: Implement **Horizontal Pod Autoscaling** to ensure applications have the required resources during peak usage without over-allocating resources during low load periods.
- **Complete IaC Adoption**: Expand the use of Terraform to manage all infrastructure configurations, ensuring consistency across environments.

7.3 Monitoring and Logging

- **Fine-tune Alerts**: Adjust the alert thresholds in Prometheus to ensure that only actionable alerts are triggered. Use anomaly detection tools to improve alert accuracy.
- **Implement Log Rotation**: Set up a **log rotation policy** for the ELK stack to manage log storage more effectively and prevent system overloads.

7.4 Security

- **Shift-Left Security**: Integrate security scanning earlier in the CI/CD pipeline, ideally during the development phase, using tools like **OWASP ZAP** for continuous scanning.
- **Audit Trail Enhancements**: Improve the audit trail system for RBAC to track and log all changes in access control for better visibility and compliance.

Step 8. Conclusion

XYZ Enterprise has made significant progress in adopting DevOps practices, but several areas require optimization to improve performance, reduce costs, and enhance security. By focusing on automation, infrastructure optimization, and security integration, the team can achieve faster deployments, lower failure rates, and better overall system performance.

A follow-up review is recommended in six months to measure the impact of the proposed improvements and track further progress.

Step 9. Appendix

- **Glossary of Terms**
  - **CI**: Continuous Integration
  - **CD**: Continuous Delivery
  - **IaC**: Infrastructure as Code
  - **MTTR**: Mean Time to Recovery
  - **RBAC**: Role-Based Access Control
- **Tools Used for Data Collection**:
  - GitLab, Jenkins, Docker, Kubernetes, AWS, Prometheus, Grafana, Snyk, ELK Stack

Points to Remember

- Before performing any monitoring activity, you must understand the different types of monitoring tools such as networking monitoring tools, application monitoring tool and infrastructure monitoring tools.
- When installing a monitoring tool in a DevOps environment, first download tool then after follow the installation steps until tool will completed.
- Describe the type of data such as log data, infrastructure data, application performance data and build and deployment data, version control data, security data user experience data, automation data, collaboration and workflow data and business and financial data.
- While writing there are many Key Points respect for Comprehensive Work Report

- Apart from using commands during installation, there are other monitoring tools which can be downloaded from the official manufacture website and then be used during installation: these include washark, prometheus, grafana, datadog, nagios, zabbix and ELK stack.

Application of learning 3.3.

YA Ltd company has a hosted web application on online server for managing its activities. It needs a competent DevOps to analyse the data in DevOps. As a DevOp analyst, you are requested to write the report of performance metrics and feedback data on the system.

Learning outcome 3 end assessment

Theoretical assessment

Q1. Read carefully and answer the following questions:

i. What factors should be considered when selecting and installing monitoring tools?
ii. What are the key benefits of effective DevOps monitoring?
iii. What are the different types of monitoring tools used in DevOps environments?
iv. What factors should be considered when selecting and installing monitoring tools?
v. What is the importance of integrating feedback loops into the DevOps process?

Q2. Circle the letter corresponding with correct answer:

i. What is the primary purpose of performance metrics in DevOps?

A. To measure the efficiency of development teams

B. To assess the quality of software products

C. To monitor the performance of systems and applications

D. To track user satisfaction

ii. Feedback data is essential in DevOps for:

A. Identifying technical issues

B. Understanding user experiences

C. Measuring team productivity

D. Assessing financial performance

iii. **Why is data analysis crucial in DevOps?**

A. To identify areas for improvement

B. To ensure compliance with regulations

C. To track project timelines

D. To measure team morale

iv. Data analysis can help DevOps teams:

A. Optimize resource utilization

B. Reduce development costs

C. Improve team communication

D. Increase employee turnover

v. **Which of the following is NOT a type of data commonly used in DevOps?**

A. System metrics

B. Financial data

C. User experience metrics

D. Feedback data

vi. Monitoring tools are used to collect:

A. Financial data

B. Employee performance data

C. System metrics

D. Customer satisfaction data

vii. Regular review of performance data helps DevOps teams:

A. Identify trends and anomalies

B. Measure team productivity

C. Reduce development costs

D. Improve employee morale

viii. Root cause analysis is used to:

A. Improve employee morale

B. Measure team productivity

C. Reduce development costs

D. Identify the underlying causes of performance issues

ix. Feedback loop integration involves:

A. Collecting and analyzing feedback from users

B. Measuring team productivity

C. Reducing development costs

D. Improving employee morale

Q3. Answer by using letter T if the statement is True and use letter F is the statement is false.

a. Report findings should be presented in a clear and concise manner, using graphs, charts, and tables where appropriate_______

b. Trends analysis can help identify patterns and anomalies in performance data _________

c. Alerts and incidents should be documented in detail, including the time, date, and severity of the event _________

d. It is not necessary to investigate the root cause of alerts and incidents ____

e. Action items should be specific, measurable, achievable, relevant, and time-bound ______

Practical assessment

Suppose that you are tasked with setting up a monitoring environment for a web application running on a cloud-based infrastructure. You need to install and configure a monitoring tool to track key performance indicators (KPIs) like CPU usage, memory usage, response time, and error rates and generate the report of the work done.

As a DevOps Engineer, you are requested to do the given task.

END

References

DevOps glossary: 78 basic DevOps. (2023, July 12). Retrieved April 27, 2024, from Its Vit: https://itsvit.com/blog/devops-glossary-78-basic-devops-terms-in-simple-words/

Fortinet. (2023, April). *DevOps Security*. Retrieved April 27, 2024, from Fortinet: https://www.fortinet.com/resources/cyberglossary/devops-security

GitLab. (2024). *What is DevOps*. Retrieved April 27, 2024, from About Git Lab: https://about.gitlab.com/topics/devops/

N, V., Batra, A., & Bhandari, V. (2023, April). *What are the best practices for data migration in your DevOps pipeline?* Retrieved April 27, 2024, from Linkedin: https://www.linkedin.com/advice/3/what-best-practices-data-migration-your-devops-ade8e

Poulton, N. (2017). *Docker Deep Dive.* Nigel Poulton.

REHKOPF, M. (2024, April). *Continuous integration tools*. Retrieved April 27, 2024, from Atlassian: https://www.atlassian.com/continuous-delivery/continuous-integration/tools

Simplilearn. (2023, April). *Docker And Containers Explained | Containerization Explained | Docker Tutorial | Simplilearn*. Retrieved April 27, 2024, from Simpli Learn - Yooutube video: https://www.youtube.com/watch?app=desktop&v=A0g7I4A6GN4

RTB | RWANDA TVET BOARD

October 2024