7)Mantissa tool:

https://bugracking.mantishub.io/my_view_page.php login here use one unique domain name.

https://vimeo.com/149359701 video if needed.

8)bugzilla tool:

https://bugzilla-dev.allizom.org/home bugzilla official page

Go there arise issue ,create bug,report to bugs.explore all fields.

9)open source project ghya tyt bug find kra changes kra n push kra.
10)To install and set up **Joomla!** on your own server using the terminal, follow the steps below. This guide will help you set up Joomla! on a Linux-based server (e.g., Ubuntu) and configure it from the terminal.
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@22
https://youtu.be/UTh3fNkhS2s?si=0rrzqfhkugtfTQ6k video for steps.

While using same code make sure that no database is there in mysql.
Use correct credentials:
Database Name: `joomladb`
Database Username: `joomla`
Database Password: `Password`
Database Host: `localhost`

To get ipaddress of localhost: ifconfig
If not work then sudo apt install net-tools
sudo apt update && sudo apt upgrade -y

sudo apt install unzip -y

sudo apt install apache2 -y

sudo systemctl status apache2

sudo systemctl start apache2 && sudo systemctl enable apache2

```
sudo apt install php libapache2-mod-php php-mysql php-xml php-mbstring
php-curl php-zip php-intl -y

php --version

sudo nano /etc/php/8.3/apache2/php.ini

memory_limit = 512M
upload_max_filesize = 60M
max_execution_time = 300
date.timezone = Europe/Amsterdam


sudo apt install mysql-server -y

sudo systemctl status mysql

sudo systemctl start mysql.service && sudo systemctl enable mysql.service

sudo mysql

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY
'root';

quit;

sudo mysql_secure_installation

sudo mysql -u root -p

CREATE DATABASE joomladb;

CREATE USER 'joomla'@'localhost' IDENTIFIED BY 'Password';

GRANT ALL PRIVILEGES ON joomladb.* TO 'joomla'@'localhost';
```

```
FLUSH PRIVILEGES;

SHOW DATABASES;

EXIT;
```

```
wget
https://downloads.joomla.org/cms/joomla5/5-1-2/Joomla_5-1-2-Stable-Full_Pack
age.zip

sudo unzip Joomla_5-1-2-Stable-Full_Package.zip -d /var/www/html/joomla

ls /var/www/html/

sudo chown -R www-data:www-data /var/www/html/joomla

sudo chmod -R 755 /var/www/html/joomla

sudo nano /etc/apache2/sites-available/joomla.conf
```

```
<VirtualHost *:80>
ServerAdmin admin@localhost
DocumentRoot /var/www/html/joomla
ServerName localhost

<Directory /var/www/html/joomla>
Options FollowSymLinks
AllowOverride All
Require all granted
</Directory>

ErrorLog ${APACHE_LOG_DIR}/joomla_error.log
CustomLog ${APACHE_LOG_DIR}/joomla_access.log combined
</VirtualHost>
```

sudo a2ensite joomla.conf

sudo a2enmod rewrite

sudo systemctl restart apache2

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

### **Part 1: Setting Up the Server Environment**

Before installing Joomla!, you need a server with the required software installed. These include:

1. **Apache Web Server**
2. **MySQL Database Server**
3. **PHP**

#### **Step 1: Update the Server**

First, update your server’s package list to ensure all software is up-to-date.

```bash
sudo apt update && sudo apt upgrade -y
```

#### **Step 2: Install Apache Web Server**

To install Apache:

```bash
sudo apt install apache2 -y
```

After the installation, enable and start Apache:

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

Verify Apache is running:

```bash
sudo systemctl status apache2
```

#### **Step 3: Install MySQL Database Server**

Install MySQL by running:

```bash
sudo apt install mysql-server -y
```

Once installed, secure the MySQL installation:

```bash
sudo mysql_secure_installation
```

Follow the prompts to configure MySQL (set the root password, remove insecure default settings, etc.).

Create a database for Joomla:

```bash
```

sudo mysql -u root -p

Once logged in to MySQL, create a database:

```sql
CREATE DATABASE joomla_db;
CREATE USER 'joomla_user'@'localhost' IDENTIFIED BY 'your_password';
GRANT ALL PRIVILEGES ON joomla_db.* TO 'joomla_user'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

This creates a database `joomla_db` and a user `joomla_user` with access to that database.

#### **Step 4: Install PHP and Required Extensions**

Joomla! requires PHP and some PHP extensions. Install them using:

```bash
sudo apt install php php-cli php-fpm php-mysql php-xml php-mbstring php-curl
php-zip php-gd php-json php-ldap libapache2-mod-php -y
```

Restart Apache to load PHP:

```bash
sudo systemctl restart apache2
```

Check PHP version to verify the installation:

```bash
php -v
```

### **Part 2: Download and Install Joomla!**

#### **Step 5: Download Joomla!**

Download the latest Joomla! version directly from the official site using `wget`:

```bash
wget
https://downloads.joomla.org/cms/joomla4/latest/joomla_4.x.x-stable-full_packag
e.tar.gz
```

Extract the downloaded archive:

```bash
tar -xvzf joomla_4.x.x-stable-full_package.tar.gz
```

Move the Joomla! files to your web server's root directory:

```bash
sudo mv Joomla_4.x.x-Stable-Full_Package/* /var/www/html/
```

#### **Step 6: Set Permissions**

Set the correct permissions for Joomla! files:

```bash
sudo chown -R www-data:www-data /var/www/html/
sudo chmod -R 755 /var/www/html/
```

### **Part 3: Configure Apache for Joomla!**

#### **Step 7: Create a Virtual Host for Joomla!**

Create a new Apache configuration file for your site:

```bash
sudo nano /etc/apache2/sites-available/joomla.conf
```

Add the following configuration:

```apache
<VirtualHost *:80>
        ServerAdmin webmaster@yourdomain.com
        DocumentRoot /var/www/html
        ServerName yourdomain.com
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        <Directory /var/www/html>
        AllowOverride All
        Require all granted
        </Directory>
</VirtualHost>
```

Enable the site and rewrite module:

```bash
sudo a2ensite joomla.conf
sudo a2enmod rewrite
```

Restart Apache to apply changes:

```bash
sudo systemctl restart apache2
```

### **Part 4: Complete Joomla! Installation via Web Browser**

#### **Step 8: Open Joomla! Installation in Your Browser**

1. Open your web browser and go to your server's IP address or domain (e.g., `http://yourdomain.com` or `http://your_server_ip`).

2. You’ll be greeted by the Joomla! installation wizard.

3. **Language Selection**: Choose your preferred language and click **Next**.

4. **Pre-Installation Check**: Joomla! will perform a system check. If everything is okay, click **Next**.

5. **Database Configuration**:
   - Database Type: **MySQLi**
   - Host Name: **localhost**
   - Username: `joomla_user`
   - Password: Your password (`your_password`)
   - Database Name: `joomla_db`

   Click **Next**.

6. **FTP Configuration** (Optional): You can skip or configure FTP access for file management (it’s optional).

7. **Site Configuration**:
   - Enter a **Site Name** and **Admin Email**.
   - Set the **Admin Username** and **Admin Password**.

8. **Install Joomla!**: Click **Install** to complete the installation process.

9. **Remove the Installation Folder**: After Joomla! is installed, the wizard will prompt you to remove the installation folder. Click **Remove**.

### **Part 5: Access Your Joomla! Site**

Your Joomla! site is now set up and accessible via your server's IP address or domain.

1. To access the Joomla! administrator area, go to:
   ```bash
   http://yourdomain.com/administrator
   ```

2. Log in with the admin credentials you created during the installation process.

### **Conclusion**

You've now successfully installed Joomla! on your server using the terminal. From here, you can start managing your website by creating content, installing extensions, and customizing the design through the Joomla! admin interface.

If you need help with specific configurations or further steps, feel free to ask!
###############
11)from IT25 Docs.

Assignment 12

Create RPM packages. (multiple modules/code packaging of c/cpp). Pull or take any FOSS project and one feature or remove bug and then

create the package on suitable OS Compare RPM packaging with Debian packaging (on answer sheet)
@@@@@@@@solution:
Now, you can continue with the following steps inside this Fedora container:

Update and install dependencies:
Yes, you're on the right track! You've successfully started a Fedora container as the root user, which will allow you to perform the necessary steps to build the RPM for `htop` within a controlled environment.

Now, you can continue with the following steps inside this Fedora container:

1. **Update and install dependencies**:

```bash

dnf update -y

dnf install -y rpm-build gcc make git autoconf automake ncurses-devel

```

2. **Set up the RPM build environment**:

```bash

mkdir -p /root/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}

echo '%_topdir /root/rpmbuild' > ~/.rpmmacros

```

3. **Clone and modify the `htop` source**:

```bash
cd /root/rpmbuild/SOURCES
git clone https://github.com/htop-dev/htop.git
cd htop
sed -i '1s|^|#include <stdio.h>\n|' htop.c
sed -i '/int main(/a \       printf("Running custom command\\n");' htop.c
```

4. **Create a tarball of the modified source code**:

```bash
cd ..
tar -czvf htop-custom.tar.gz htop
```

5. **Create the RPM spec file**:

```bash
cat << 'EOF' > /root/rpmbuild/SPECS/htop.spec
Name:       htop
Version:    3.2.2
Release:    1%{?dist}
Summary:           Interactive process viewer with custom message
```

```
License:    GPL
URL:        https://github.com/htop-dev/htop
Source0:    %{name}-custom.tar.gz

BuildRequires: ncurses-devel, gcc, make, autoconf, automake
Requires:   ncurses

%description
A modified version of htop with a custom message.

%prep
%setup -q -n htop

%build
./autogen.sh
./configure
make

%install
mkdir -p %{buildroot}/usr/local/bin
install -m 0755 htop %{buildroot}/usr/local/bin/htop
```

```
   %files

   /usr/local/bin/htop


   %changelog

   * Wed Nov 8 2024 User <user@example.com> - 3.2.2-1

  - Added custom message in htop

  EOF
```

6. **Build the RPM package**:

```bash
   cd /root/rpmbuild

   rpmbuild -ba SPECS/htop.spec
```

7. **Install and test the package**:

```bash
   dnf install -y /root/rpmbuild/RPMS/x86_64/htop-3.2.2-1.fc*.rpm

   /usr/local/bin/htop
```

This should install and launch `htop` with the custom message. You can exit `htop` by pressing `Ctrl + C`.

@@@@@@@@@@@@@@@@stop

To create Debian packages for a Python project and compare it with RPM packaging, let's follow these detailed steps:

### Steps to Create Debian Packages for a Python Project

1. **Set up the environment**:
   - We’ll use a controlled environment like a Docker container running Debian.
   - Start by installing Docker and run the following command to set up the Debian environment.

   ```bash
   sudo docker run -it debian:latest /bin/bash
   ```

2. **Install necessary dependencies**:
   - Update the system and install tools required for Debian packaging.

   ```bash
   apt update -y
   apt install -y python3 python3-pip python3-venv python3-setuptools
   python3-wheel dh-make devscripts build-essential
   ```

3. **Choose and Set Up a Python Project**:
   - Select an open-source Python project (e.g., Flask). For simplicity, let’s assume the chosen project is Flask.
   - Create a project directory and initialize the project.

   ```bash
   mkdir my_python_project && cd my_python_project
   python3 -m venv venv
   source venv/bin/activate
   pip install flask
   ```

4. **Prepare the Project for Debian Packaging**:

   - Debian packages require a specific directory structure. Use `dh_make` to set this up.

   ```bash
   dh_make --createorig -p my_python_project_1.0
   ```

5. **Edit Debian Control Files**:
   - Customize the `debian/control` file to add project metadata, dependencies, and maintainers.

   ```bash
   nano debian/control
   ```

   Add or modify sections such as:
   ```plaintext
   Source: my_python_project
   Section: python
   Priority: optional
   Maintainer: Your Name <your.email@example.com>
   Build-Depends: debhelper (>= 9), python3
   Standards-Version: 3.9.8
   Homepage: https://github.com/your-repo/my_python_project

   Package: my_python_project
   Architecture: all
   Depends: ${misc:Depends}, ${python3:Depends}
   Description: A sample Python project for Debian packaging
   ```

6. **Create the `debian/rules` File**:
   - This file contains instructions for building the package.

   ```bash
   nano debian/rules
   ```

   Basic contents of `debian/rules`:
   ```makefile
   #!/usr/bin/make -f
   %:
   	dh $@
   ```

7. **Build the Debian Package**:
   - Now, build the package using `debuild`.

   ```bash
   debuild -us -uc
   ```

   After this step, you should see `.deb` files generated in the parent directory.

8. **Install and Test the Debian Package**:
   - Install the generated `.deb` file to test it.

   ```bash
   sudo dpkg -i ../my_python_project_1.0_all.deb
   ```

9. **Compare RPM and Debian Packaging**:
   - While Debian uses `.deb` files, RPM-based distributions (like Fedora) use `.rpm` files. RPMs are built with tools like `rpmbuild` and generally require `SPEC` files instead of `debian/` directories.
   - For RPM packaging, the structure is different, and the commands involve creating a `.spec` file and using `rpmbuild` to generate the package.

### Key Differences:

- **Packaging System**: Debian uses `debhelper`, while RPM uses `rpmbuild`.
- **Files Used**: Debian requires a `debian/` directory with multiple control files. RPM packages rely on a single `SPEC` file.

- **Package Managers**: Debian packages are managed by `dpkg` and `apt`, whereas RPM packages use `rpm` and `yum/dnf`.
- **Distribution Compatibility**: `.deb` files are for Debian-based systems, and `.rpm` files are for Red Hat-based systems.

This process should give you a working Debian package, with steps highlighting the structure differences between Debian and RPM packaging. Let me know if you need more specific examples on any step!
##################

ASSignment12:

Create RPM packages. (multiple modules/code packaging of c/cpp). Pull or take any FOSS project and one feature or remove bug and then create the package on suitable OS Compare RPM packaging with Debian packaging (on answer sheet)

1. Objective

The goal of this assignment is to modify the `htop` program by adding a custom message, package it as an RPM, and install it on Fedora. We will be using Docker to create a controlled environment for the build.

2. Execution Plan

We will follow these steps:
1. Install Docker and set up a container environment.
2. Prepare the system by installing necessary dependencies.
3. Clone the `htop` repository, modify its code, and package it as an RPM.
4. Verify the RPM installation.

3. Detailed Steps and Commands

Step 1 : Install Docker via link or commands
Link : https://docs.docker.com/engine/install/ubuntu/
Commands:

1. for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done
2. # Add Docker's official GPG key:
   sudo apt-get update
   sudo apt-get install ca-certificates curl
   sudo install -m 0755 -d /etc/apt/keyrings

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

1. sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

Step 2 : Set Up the Build Environment

1. Start Docker and run a Fedora container:

```
sudo docker run -it fedora:latest /bin/bash
```

Now **either** copy this script inside container by following commands

1.dnf install nano
2.nano script.sh
3.paste the script content
4.save and exit by ctrl+o and ctrl+x
5.bash script.sh

script.sh :

```
#!/bin/bash

echo "Updating system and installing build dependencies..."
dnf update -y
dnf install -y rpm-build gcc make git autoconf automake ncurses-devel gdb


echo "Setting up RPM build directories..."
mkdir -p /root/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
echo '%_topdir /root/rpmbuild' > ~/.rpmmacros


echo "Cloning the htop repository from GitHub..."
cd /root/rpmbuild/SOURCES
```

```
git clone https://github.com/htop-dev/htop.git
cd htop


echo "Modifying source code to include a custom message..."
sed -i '1s|^|#include <stdio.h>\n|' htop.c
sed -i '/int main(/a \     printf("Running custom command\\n");' htop.c

echo "Creating a tarball of the modified htop source..."
cd ..
tar -czvf htop-custom.tar.gz htop


echo "Creating the RPM spec file..."
cat << 'EOF' > /root/rpmbuild/SPECS/htop.spec
Name:           htop
Version:        3.2.2
Release:        1%{?dist}
Summary:        Interactive process viewer with custom message

License:        GPL
URL:            https://github.com/htop-dev/htop
Source0:        %{name}-custom.tar.gz

BuildRequires:  ncurses-devel, gcc, make, autoconf, automake
Requires:       ncurses

%description
A modified version of htop with a custom message.

%prep
%setup -q -n htop

%build
./autogen.sh
./configure
make

%install
mkdir -p %{buildroot}/usr/local/bin
install -m 0755 htop %{buildroot}/usr/local/bin/htop

%files
/usr/local/bin/htop
```

```
%changelog
* Wed Nov 8 2024 User <user@example.com> - 3.2.2-1
- Added custom message in htop
EOF

# Step 7: Build the RPM package
echo "Building the RPM package..."
cd /root/rpmbuild
rpmbuild -ba SPECS/htop.spec

# Step 8: Check the generated RPM package
echo "Listing the generated RPM package..."
ls /root/rpmbuild/RPMS/x86_64/

echo "Installing the RPM package after building...."
dnf install -y /root/rpmbuild/RPMS/x86_64/htop-3.2.2-1.fc*.rpm

echo "Verifying ......"
/usr/local/bin/htop
```

or follow below steps :

1. Install necessary packages in the container:

```
  dnf update -y
  dnf install -y rpm-build gcc make git autoconf automake ncurses-devel gdb
```

2. Create the RPM build directories:

```
  mkdir -p /root/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
  echo '%_topdir /root/rpmbuild' > ~/.rpmmacros
```

3. Modify the `htop` Source Code ,Clone the `htop` GitHub repository and make the required code modifications:

```
  cd /root/rpmbuild/SOURCES
 git clone https://github.com/htop-dev/htop.git
  cd htop
```

4. Add a custom message to the source code:

```
sed -i '1s|^|#include <stdio.h>\n|' htop.c
sed -i '/int main(/a \  printf("Running custom command\\n");' htop.c
```

5. Create a compressed tarball of the modified source code:

```
cd ..
tar -czvf htop-custom.tar.gz htop
```

6. Create the RPM Spec File

```
cat << 'EOF' > /root/rpmbuild/SPECS/htop.spec
Name:          htop
Version:       3.2.2
Release:       1%{?dist}
Summary:       Interactive process viewer with custom message

License:       GPL
URL:           https://github.com/htop-dev/htop
Source0:       %{name}-custom.tar.gz
BuildRequires:  ncurses-devel, gcc, make, autoconf, automake
Requires:      ncurses

%description
A modified version of htop with a custom message.

%prep
%setup -q -n htop

%build
./autogen.sh
./configure
make

%install
mkdir -p %{buildroot}/usr/local/bin
install -m 0755 htop %{buildroot}/usr/local/bin/htop

%files
/usr/local/bin/htop

%changelog
* Wed Nov 8 2024 User <user@example.com> - 3.2.2-1
- Added custom message in htop
```

```
   EOF
```

7. Build the RPM Package

```
   cd /root/rpmbuild
   rpmbuild -ba SPECS/htop.spec


   ls /root/rpmbuild/RPMS/x86_64/
```

8. Install and Verify the Custom RPM

```
   dnf install -y /root/rpmbuild/RPMS/x86_64/htop-3.2.2-1.fc*.rpm

   /usr/local/bin/htop
```

PRESS CTRL+C TO EXIT HTOP COMMAND AND YOU CAN SEE A CUSTOM MESSAGE PRINTED ON SCREEN

Same as IT25 file do it by first step.

13. Create Debian packages.

(multiple modules/code packaging of java/c/cpp).
Pull or take any FOSS project and one feature and then create the package on suitable OS
Compare RPM packaging with Debian packaging (on answer sheet)

Step 1) Choose any simple github repository of cpp code i.e. having one code file (so that it'll be easy to implement and understand) and clone it on desktop by running command on in terminal
Cmd: git clone https://github.com/Yashashwi0708/Balanced_Hashmap

Step 2) CD into that folder
Cmd: cd Balanced_Hashmap

Step 3) Create a folder named Debian for keeping control file in it and make a directory structure as usr/local/bin for keeping binary/executable file of code in it

Cmd: mkdir Debian
```
mkdir -p usr/local/bin
```

Step 4) Go inside Debian and make a file named control

Cmd: cd Debian
```
nano control
```

Write following things in it

```
Package: balanced-hashmap
Version: 0.2
Maintainer: User_Name
Architecture: all
Description: A balanced hashmap in cpp
```

Make changes according to you
Step 5) Compile the CPP code in main folder and give a suitable name for the executable file (because it will be the name of your package/command)
Cmd: cd ..
```
g++ Balanced_HMap.cpp -o BalancedMap
```

Step 6) Move the compiled binary to usr/local/bin
Cmd: mv BalancedMap usr/local/bin

Step 7) Build the package and install it
Cmd: dpkg-deb --build Balanced_Hashmap
```
sudo dpkg -i Balanced_Hashmap.deb
```

Step 8) Check if the installed package is working
Cmd: BalancedMap

If the steps are correctly followed then you should see following on the output screen:

```
Value for banana: 20
Bucket 0:
Bucket 1: date: 40 | elderberry: 50 | grape: 70 |
Bucket 2: fig: 60 | lemon: 100 | mango: 50 |
Bucket 3: banana: 20 | dragonfruit: 80 |
Bucket 4: kiwi: 90 |
Bucket 5: cherry: 30 | apricot: 60 |
Bucket 6: jackfruit: 70 |
Bucket 7:
Bucket 8: apple: 10 | honeydew: 80 |
Bucket 9:
```

End!

Follow correctly install cpp compiler all setup on system.
14)same as 13.
15)SonarQube:
Bydefault password: admin username: admin
On webpage

```
sudo docker run -d --name sonar -p 9000:9000 sonarqube
```

This command will start sonarqube on localhost.
After running this, you should be able to access SonarQube at http://localhost:9000 in your browser.
New created password: Siddhi@12345
Create local project
Analyze your project:
Token generated: sqp_1714470043ca8f8b4bdd215e9b9efe44afab7321
To run a SonarQube analysis using `sonar-scanner`, you need to follow a few steps to set up the scanner, configure your environment, and run the analysis on your project.

### Step 1: Install SonarQube Scanner
1. **Download SonarQube Scanner:**
Visit the official SonarQube Scanner documentation to download the latest version of the Scanner. You can download the latest version from the [SonarQube Downloads page](https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/).

2. **Extract the Scanner files:**

   Once downloaded, extract the files to a directory of your choice, for example, `/opt/sonar-scanner`.

   ```bash
   tar -xvzf sonar-scanner-*.tar.gz
   ```

### Step 2: Add the `bin` Directory to the PATH Environment Variable
1. **Edit your shell configuration file**:
   Open the shell profile file (e.g., `.bashrc`, `.zshrc` depending on your shell).

   ```bash
   nano ~/.bashrc   # For Bash users
   ```

   Or:

   ```bash
   nano ~/.zshrc         # For Zsh users
   ```

2. **Add the SonarScanner `bin` directory to the PATH**:
   Add the following line to the end of the file:

   ```bash
   export PATH=$PATH:/path/to/sonar-scanner/bin
   ```

   Replace `/path/to/sonar-scanner/bin` with the actual path to the `bin` directory inside your extracted SonarScanner folder.

3. **Apply the changes**:
   After editing the profile file, apply the changes by running:

   ```bash
   source ~/.bashrc   # For Bash users
   ```

   Or:

   ```bash
   source ~/.zshrc      # For Zsh users
   ```

### Step 3: Configure SonarQube

Make sure that you have SonarQube running on your machine. If it's not installed yet, you can download and set it up from the [SonarQube Downloads page](https://www.sonarqube.org/downloads/).

1. **Start SonarQube**:
   If SonarQube is installed locally, you can start it by navigating to the SonarQube directory and running:

   ```bash
   ./bin/linux-x86-64/sonar.sh start
   ```

   This will start the SonarQube server on `http://localhost:9000`.

2. **Generate a SonarQube Token**:
   - Go to [SonarQube UI](http://localhost:9000).
   - Log in as an administrator.
   - Navigate to `My Account` > `Security` and generate a new token.
   - Copy the token, as you'll need it in the analysis step.

### Step 4: Run SonarQube Analysis

Now, you can run the SonarQube analysis on your project.

1. **Navigate to your project folder** where your source code is located.

   ```bash
   cd /path/to/your/project
   ```

2. **Run the SonarQube Scanner**:
   Execute the following command to start the analysis:

   ```bash
   sonar-scanner \
         -Dsonar.projectKey=samarth \
         -Dsonar.sources=. \
         -Dsonar.host.url=http://localhost:9000 \
         -Dsonar.token=sqp_1714470043ca8f8b4bdd215e9b9efe44afab7321
   ```

Replace `samarth` with your project key, and make sure the token you used (`sqp_1714470043ca8f8b4bdd215e9b9efe44afab7321`) is correct. This command will:
- Use the local SonarQube server at `http://localhost:9000`.
- Analyze the source code in the current directory (`-Dsonar.sources=.`, which means the entire directory).
- Use the SonarQube token for authentication.

3. **Check the Results:**
After running the analysis, visit the SonarQube dashboard at `http://localhost:9000` to see the results of the analysis. You can check for issues, code quality, and other metrics based on the analysis.

### Additional Notes:

- **SonarQube Token: Always make sure you use the correct token that has sufficient permissions to analyze the project.**
- **SonarQube Server URL: If your SonarQube server is hosted remotely, replace `http://localhost:9000` with the actual server URL.**
- **Additional Parameters: The SonarQube scanner supports other parameters like `-Dsonar.language=`, `-Dsonar.exclusions=`, etc., for more advanced configuration, which can be found in the official documentation.**

By following these steps, you should be able to successfully configure and run SonarQube analysis on your project.

20) go on youtrack demo project or create new project.
https://www.jetbrains.com/youtrack/download/get_youtrack.html sign in online if not.

Assignment 22: CMS Software: Drupal

Demonstrate the use/features of CMS software: "Drupal".
Create users and show how Drupal manages contents of web sites for a client. Also implement the working of core features of Drupal.
Compare it with other CMS like schoology/(on answer sheet)

A **CMS**, or **Content Management System**, is a software application or platform that allows users to create, manage, and modify content on a website without needing specialized technical knowledge. In simple terms, it's a tool that helps people easily update and maintain a website by providing a user-friendly interface. You can add text, images, videos, and other content to a site without needing to know how to code.Some popular CMS examples include **WordPress**, **Joomla**, and **Drupal**. Drupal installation - ubuntu:

Updated Step-by-Step Guide to Install Drupal on Ubuntu (Home Directory)

Step 1: Update Your System

```
sudo apt update

sudo apt upgrade -y
```

Step 2: Install Apache Web Server

```
sudo apt install apache2 -y

sudo systemctl enable apache2

sudo systemctl start apache2
```

Step 3: Install PHP

```
sudo apt update
sudo apt install software-properties-common -y
sudo add-apt-repository ppa:ondrej/php
sudo apt update

sudo apt install php8.3 php8.3-cli php8.3-fpm php8.3-mysql php8.3-gd
php8.3-xml php8.3-mbstring php8.3-zip php8.3-curl php8.3-xmlrpc
php8.3-imagick libapache2-mod-php8.3 -y

php -v
```

Step 4: Install MySQL Database

```
sudo apt install mysql-server -y


sudo mysql_secure_installation

sudo mysql -u root -p
```

Create the database and user:

sql

```
CREATE DATABASE drupaldb;

CREATE USER 'drupaluser'@'localhost' IDENTIFIED BY 'your_password';

GRANT ALL PRIVILEGES ON drupaldb.* TO 'drupaluser'@'localhost';

FLUSH PRIVILEGES;

EXIT;
```

Step 5: Install Drupal in Your Home Directory

```
cd ~

wget https://www.drupal.org/download-latest/tar.gz

tar -xvzf tar.gz

mv drupal-* ~/drupal

sudo chown -R www-data:www-data ~/drupal

sudo chmod -R 755 ~/drupal
```

Step 6: Configure Apache for Your Drupal Installation in Home Directory

Edit Apache configuration to point to your home directory:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

For /home/username/drupal if it set bydefault as /var/www/html then keep it as it is.

Keep same documentroot and same directory.

Modify the `DocumentRoot` and `<Directory>` sections:

```
DocumentRoot /home/your_username/drupal


<Directory /home/your_username/drupal>
```

```
    AllowOverride All

    Require all granted

</Directory>
```

Enable mod_rewrite:

```
sudo a2enmod rewrite
```

Restart Apache:

```
sudo systemctl restart apache2
```

Modify permissions:

sudo chown -R www-data:www-data /home/your_username/drupal

sudo chmod -R 755 /home/your_username/drupal

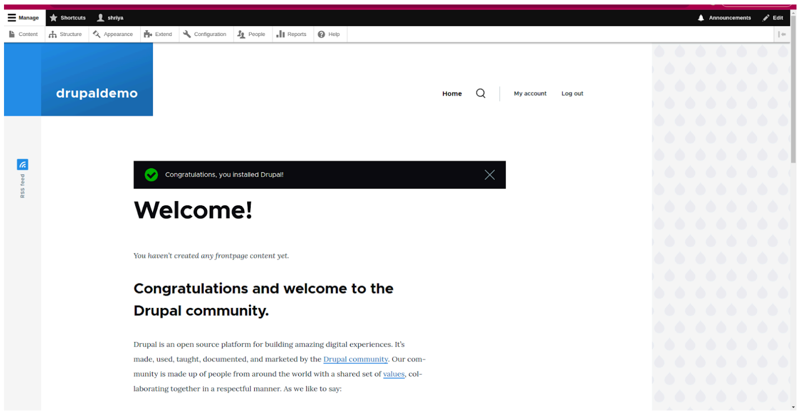
Step 7: Complete Drupal Installation in the Browser

Open your web browser and go to:

```
http://localhost
```

Proceed with the Drupal setup:

1. Choose the "Standard" installation profile.
2. Enter the database details you configured earlier:
   - **Database type**: MySQL
   - **Database name**: `drupaldb`
   - **Database username**: `drupaluser`
   - **Database password**: `your_password`
3. Configure your site details (site name, admin username, etc.)



Congratulations, you installed Drupal!

Welcome!

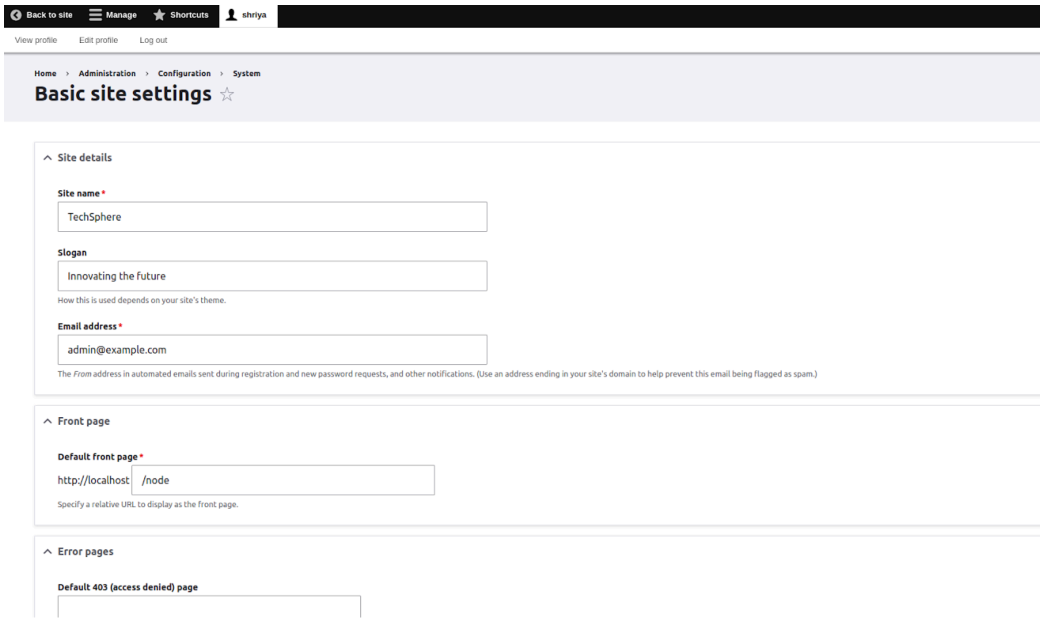
You haven't created any frontpage content yet.

Congratulations and welcome to the Drupal community.

Drupal is an open source platform for building amazing digital experiences. It's made, used, taught, documented, and marketed by the Drupal community. Our community is made up of people from around the world with a shared set of values, collaborating together in a respectful manner. As we like to say:

1: Set Up the Website Name and Basic Settings

1. **Log in to Drupal**:
   - Go to `http://localhost/user/login`
   - Go to **Configuration** → **System** → **Basic site settings**.
   - Give any names you want. I have choosed technical website where they post about projects/blogs



Basic site settings

Site details

Site name *
TechSphere

Slogan
Innovating the future
How this is used depends on your site's theme.

Email address *
admin@example.com
The *From* address in automated emails sent during registration and new password requests, and other notifications. (Use an address ending in your site's domain to help prevent this email being flagged as spam.)

Front page

Default front page *
http://localhost /node
Specify a relative URL to display as the front page.

Error pages

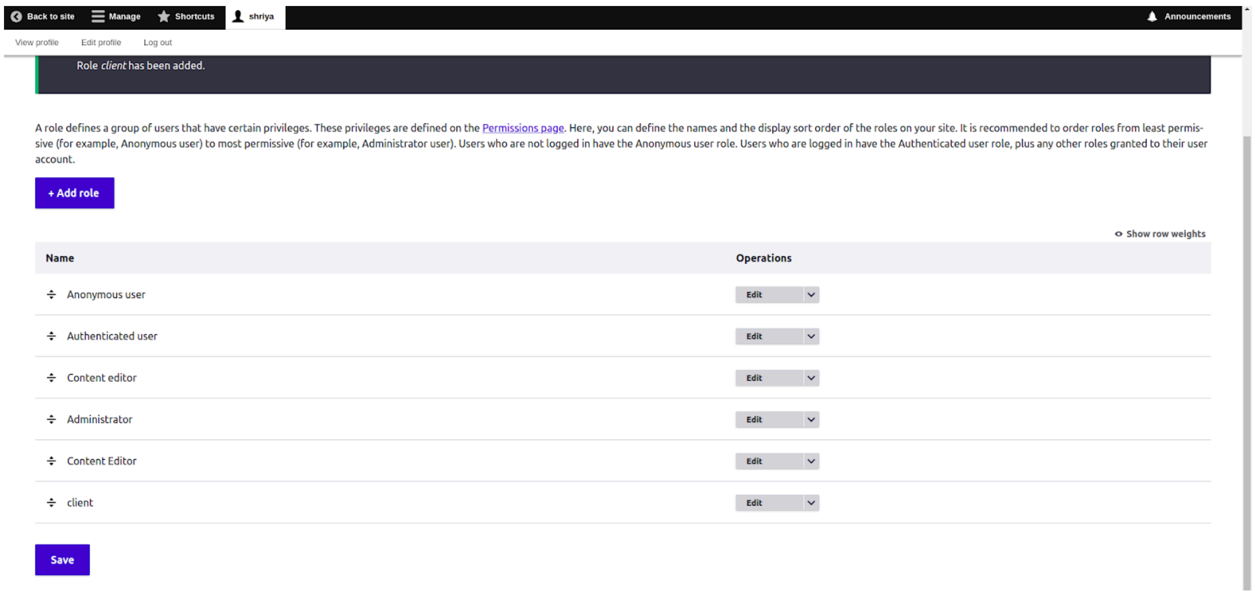
Default 403 (access denied) page

2: Create User Roles and Permissions

Create New User Roles

1. Go to **People** → **Roles**.
2. Click **Add role** and create the following roles: (add any roles acc to your website)

- **Content Editor**: Can create and edit content.
- **Client**: Can view certain content not available to the public.



Role *client* has been added.

A role defines a group of users that have certain privileges. These privileges are defined on the Permissions page. Here, you can define the names and the display sort order of the roles on your site. It is recommended to order roles from least permissive (for example, Anonymous user) to most permissive (for example, Administrator user). Users who are not logged in have the Anonymous user role. Users who are logged in have the Authenticated user role, plus any other roles granted to their user account.

+ Add role

Show row weights

Name	Operations
Anonymous user	Edit
Authenticated user	Edit
Content editor	Edit
Administrator	Edit
Content Editor	Edit
client	Edit

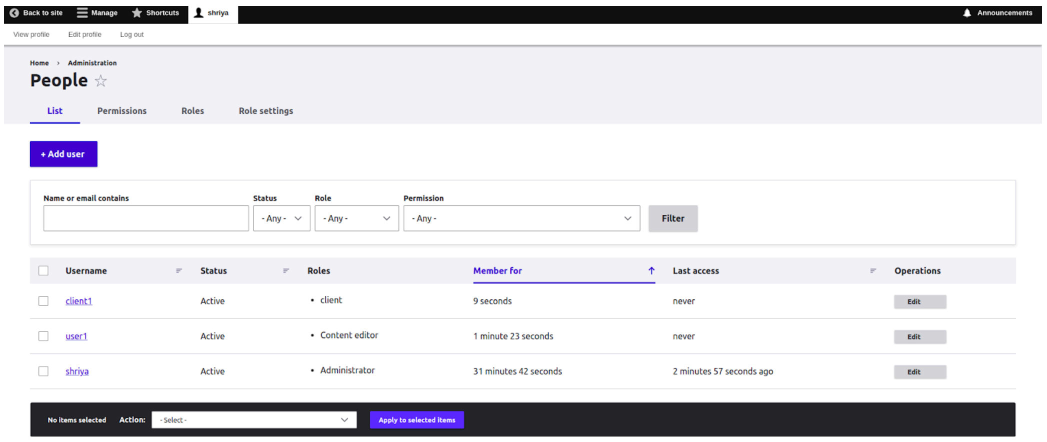
Save

b. Assign Permissions to Each Role

1. Go to **People** → **Permissions**.
2. Assign permissions as follows:

c. Create Users and Assign Roles

1. Go to **People** → **Add user**.
2. Create sample users:
   - **Editor abc**(Role: Content Editor)
3. **Client xyz** (Role: Client)



People

List | Permissions | Roles | Role settings

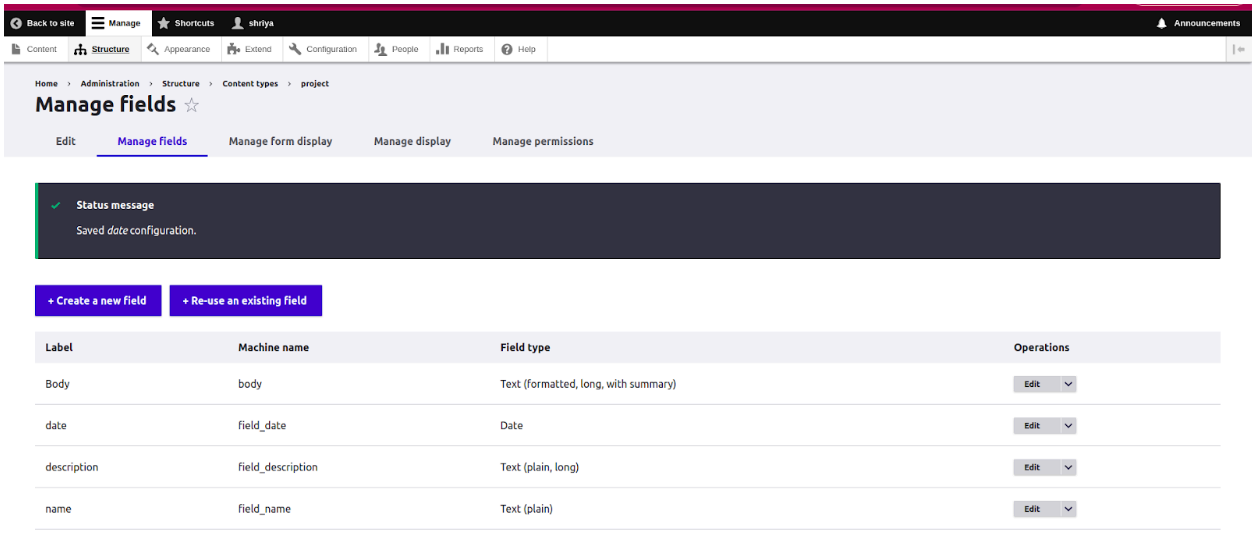
+ Add user

Name or email contains:
Status: - Any -
Role: - Any -
Permission: - Any -
Filter

	Username	Status	Roles	Member for	Last access	Operations
☐	client1	Active	• client	9 seconds	never	Edit
☐	user1	Active	• Content editor	1 minute 23 seconds	never	Edit
☐	shriya	Active	• Administrator	31 minutes 42 seconds	2 minutes 57 seconds ago	Edit

No items selected Action: - Select - Apply to selected items

Add fields.



Manage fields

Status message

Saved *date* configuration.

Label	Machine name	Field type	Operations
Body	body	Text (formatted, long, with summary)	Edit
date	field_date	Date	Edit
description	field_description	Text (plain, long)	Edit
name	field_name	Text (plain)	Edit

3: Create Content Types

a. Create a Custom Content Type: "Project"

1. Go to **Structure** → **Content types** → **Add content type**.
2. **Name**: Project
   - **Description**: Used to showcase TechSphere projects.
3. Click **Save and manage fields**.

b. Add Fields to the "Project" Content Type

c. Configure Content Type Settings

- Go to the **Manage display** tab and adjust the field order as desired.

4: Add Content

a. Create Articles

1. Go to **Content** → **Add content** → create what you want

5: you can go on exploring other options like themes, etc

6: Demonstrate User Login and Content Management

a. Test Different User Roles

1. Log out as admin and log in as **Editor** .
    - Demonstrate adding and editing content.
2. Log in as **Client** .
    - Show access to specific client-only content.

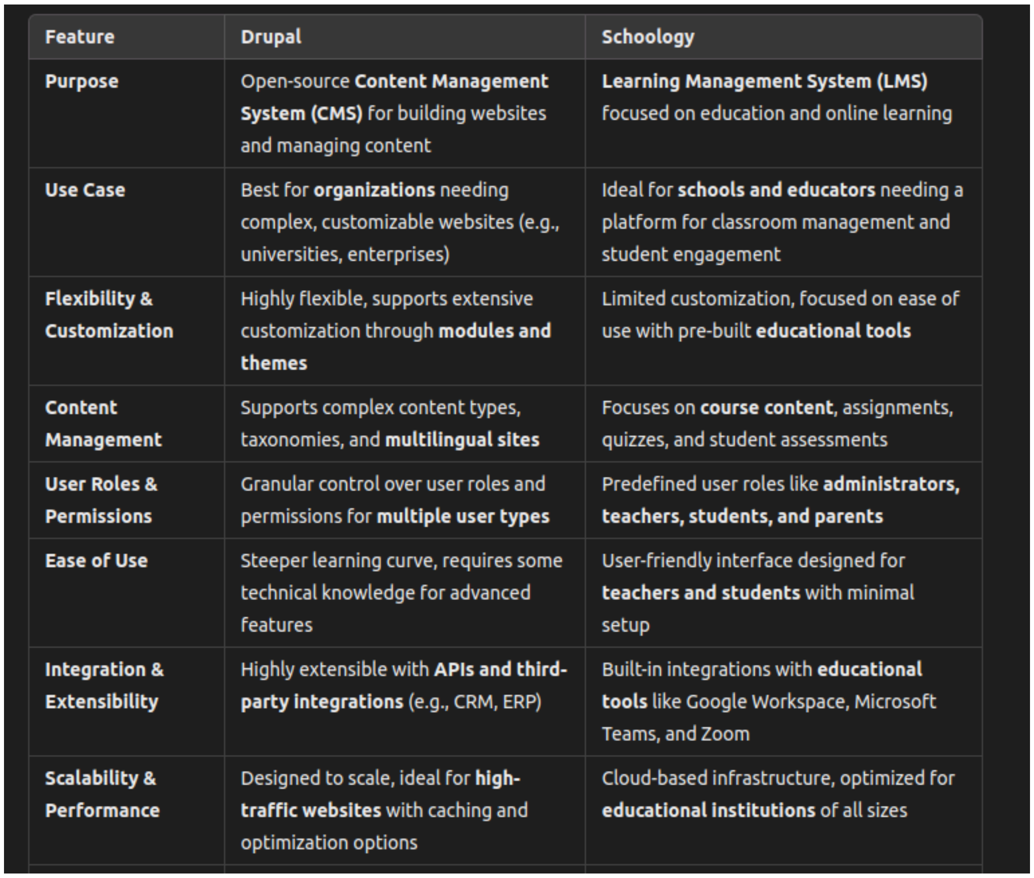
b. Use Drupal's Core Features

- Demonstrate creating a **Content View**:
    1. Go to **Structure** → **Views** → **Add view**.
    2. Create a view for displaying a list of projects.
    3. Enable **Page** and set path to `/projects`.

7: Final Touches and Demonstration

1. **Showcase the Homepage**:
    - Highlight menus, blocks, and recent content.
2. **Demonstrate Content Search**:
    - Use the search bar to find articles or projects.
3. **Show User Access Control**:
    - Log in/out to demonstrate different role permissions.
4. **Discuss the Extensibility**:
    - Mention how to extend Drupal using additional modules (e.g., SEO, Social Media).

Comparison: schoology vs drupal →



Feature	Drupal	Schoology
Purpose	Open-source **Content Management System (CMS)** for building websites and managing content	**Learning Management System (LMS)** focused on education and online learning
Use Case	Best for **organizations** needing complex, customizable websites (e.g., universities, enterprises)	Ideal for **schools and educators** needing a platform for classroom management and student engagement
Flexibility & Customization	Highly flexible, supports extensive customization through **modules and themes**	Limited customization, focused on ease of use with pre-built **educational tools**
Content Management	Supports complex content types, taxonomies, and **multilingual sites**	Focuses on **course content**, assignments, quizzes, and student assessments
User Roles & Permissions	Granular control over user roles and permissions for **multiple user types**	Predefined user roles like **administrators, teachers, students, and parents**
Ease of Use	Steeper learning curve, requires some technical knowledge for advanced features	User-friendly interface designed for **teachers and students** with minimal setup
Integration & Extensibility	Highly extensible with **APIs and third-party integrations** (e.g., CRM, ERP)	Built-in integrations with **educational tools** like Google Workspace, Microsoft Teams, and Zoom
Scalability & Performance	Designed to scale, ideal for **high-traffic websites** with caching and optimization options	Cloud-based infrastructure, optimized for **educational institutions** of all sizes

23)TTo set up and demonstrate FTP and Telnet on Ubuntu, let's go through the configuration process step-by-step. This guide will include creating a public folder for access to all users as requested.

### Step 1: **Install FTP Server (vsftpd)**

1. **Install the `vsftpd` FTP server**:

   ```bash
   sudo apt update
   sudo apt install vsftpd -y
   ```

2. **Configure vsftpd**:

   Open the configuration file for editing:

```bash
   sudo nano /etc/vsftpd.conf
   ```

   Modify the following settings in the `vsftpd.conf` file:

   - **Enable anonymous access** (optional if required for public folder):

        ```bash
        anonymous_enable=YES
        ```

   - **Enable local user access**:

        ```bash
        local_enable=YES
        ```

   - **Allow FTP write access** (only if users will upload files):

        ```bash
        write_enable=YES
        ```

   - **Set up FTP root directory for public folder** (optional if you want everyone to access `/var/ftp/pub`):

        ```bash
        anon_root=/var/ftp
        ```

3. **Create the public FTP folder**:

   ```bash
   sudo mkdir -p /var/ftp/pub
   sudo chmod 755 /var/ftp/pub
   sudo chown nobody:nogroup /var/ftp/pub
   ```

4. **Restart the FTP service**:

   ```bash
   sudo systemctl restart vsftpd
   ```

5. **Allow FTP through the firewall** (if enabled):

   ```bash
   sudo ufw allow 21/tcp
   ```

### Step 2: **Install and Configure Telnet**

1. **Install Telnet Server (`inetutils-inetd`)**:

   ```bash
   sudo apt install inetutils-inetd -y
   ```

2. **Configure Telnet**:

   Edit the configuration file to enable Telnet:

   ```bash
   sudo nano /etc/inetd.conf
   ```

   - Ensure there is a line for Telnet that is uncommented. It should look like this:

     ```
     telnet  stream  tcp  nowait  telnetd  /usr/sbin/telnetd  telnetd
     ```

3. **Restart the inetd service** to apply changes:

```bash
   sudo systemctl restart inetutils-inetd
   ```

4. **Allow Telnet through the firewall** (if enabled):

   ```bash
   sudo ufw allow 23/tcp
   ```

### Step 3: **Create a Test User and Set Permissions**

1. **Create a new user** for FTP and Telnet testing (optional):

   ```bash
   sudo adduser ftpuser
   ```

2. **Set permissions** for the public folder so that this user has access:

   ```bash
   sudo chmod -R 755 /var/ftp/pub
   sudo chown -R ftpuser:ftpuser /var/ftp/pub
   ```

### Step 4: **Testing FTP and Telnet**

- **FTP**: Open an FTP client and connect to the server’s IP on port 21. Log in with the credentials of the `ftpuser` created.

- **Telnet**: Open a Telnet client and connect to the server’s IP on port 23. You should be able to log in with the same `ftpuser`.

This completes the FTP and Telnet configuration on Ubuntu, along with creating and setting permissions for a public folder.

@@@@@
ASSIGNMENT26
It appears that the `nis` service is still not available after installing the `nis` package. Based on the output, it seems like the `nis` service may not have been installed correctly or the required `nis` service unit file is missing. Let's try a more thorough approach to resolve the issue.

### Step 1: Install NIS and Verify Dependencies
First, ensure that you have the correct NIS package installed. Run the following commands:

1. **Install NIS**:

   ```bash
   sudo apt update
   sudo apt install nis -y
   ```

2. **Verify if the NIS service is installed**:

   After installing, run the following to ensure the NIS-related services and configurations are installed:

   ```bash
   dpkg -l | grep nis
   ```

   You should see `nis` and `rpcbind` listed as installed.

### Step 2: Install and Enable the `rpcbind` Service
The NIS service requires `rpcbind` to be active. You've previously enabled `rpcbind`, but let's double-check if it's correctly started.

1. **Ensure `rpcbind` is installed and running**:

```bash
   sudo apt install rpcbind -y
   sudo systemctl enable rpcbind
   sudo systemctl start rpcbind
   ```

2. **Verify `rpcbind` status**:

   ```bash
   sudo systemctl status rpcbind
   ```

   Ensure that the `rpcbind` service is running.

### Step 3: Check the Service Files
If the `nis` service is still missing, it's possible the service file was not created properly during installation. Check if the service file exists:

1. **Verify the service unit file for `nis`**:

   Check if the NIS service unit file exists at `/lib/systemd/system/nis.service` or `/etc/systemd/system/nis.service`.

   ```bash
   sudo ls /lib/systemd/system/ | grep nis
   sudo ls /etc/systemd/system/ | grep nis
   ```

2. **Manually start NIS service**:

   If no service file exists, and you're trying to set up NIS on the server, it might require manual setup. You can try the following:

   ```bash
   sudo /usr/sbin/ypserv
   ```

   This will manually start the NIS server.

### Step 4: Check Logs
If the issue persists, check the system logs for any errors related to NIS:

```bash
sudo journalctl -xe
```

Look for errors regarding `nis`, `rpcbind`, or other related services.

### Step 5: Reinstall NIS
If nothing works, it may be useful to completely remove and reinstall the `nis` package:

1. **Remove `nis` package**:

   ```bash
   sudo apt-get purge nis -y
   ```

2. **Reinstall NIS**:

   ```bash
   sudo apt install nis -y
   ```

After reinstalling, attempt to start and enable the `nis` service again.

Let me know the output of these steps, and we can proceed further if necessary!
If you're working with a **Linux-based system** (such as Ubuntu, Debian, or other distributions), you can follow the steps below to **configure and demonstrate the use of NIS (Network Information Service)** and **NFS (Network File System)** on **Ubuntu** or any **Debian-based distribution**.

These instructions will help you set up both NIS and NFS servers and clients, allowing you to demonstrate the concepts of centralized user management (NIS) and network file sharing (NFS).

## **1. Install and Configure NIS Server**

### **Step 1: Install NIS Server**
On the **NIS server** (the machine that will manage users and groups), install the required NIS packages:
```bash
sudo apt update
sudo apt install -y nis
```

### **Step 2: Configure NIS Domain**
Set the NIS domain name on the NIS server:
```bash
sudo nano /etc/default/nis
```
Change the `NISDOMAIN` to your desired domain name:
```bash
NISDOMAIN=example.com
```

### **Step 3: Configure NIS Server**
Next, configure the server to allow NIS clients to connect. Edit `/etc/yp.conf`:
```bash
sudo nano /etc/yp.conf
```
Add the line:
```bash
ypserver <ip_of_nis_server>
```

### **Step 4: Configure NIS Maps**
Ensure that NIS can manage the user and group maps. Update the maps by running:
```bash
sudo /usr/libexec/ypxfr -v passwd
sudo /usr/libexec/ypxfr -v group
```

### **Step 5: Start NIS Services**
Enable and start the necessary services:
```bash
sudo systemctl enable rpcbind
sudo systemctl start rpcbind
sudo systemctl enable ypserv
sudo systemctl start ypserv
```

### **Step 6: Create Users and Groups**
Create users and assign them to groups. For example, create 5 users and 2 groups:
```bash
sudo groupadd group1
sudo groupadd group2
sudo useradd -m user1 -G group1
sudo useradd -m user2 -G group1
sudo useradd -m user3 -G group2
sudo useradd -m user4 -G group2
sudo useradd -m user5 -G group1
```

### **Step 7: Verify NIS Server**
Check if the NIS maps are being served correctly:
```bash
ypcat passwd
ypcat group
```

## **2. Install and Configure NIS Client**

### **Step 1: Install NIS Client**
On the **NIS client** machine (the machine that will authenticate via the NIS server), install the NIS client package:
```bash
sudo apt update
sudo apt install -y nis
```

### **Step 2: Configure NIS Client**
Set the NIS domain name on the client machine:
```bash
sudo nano /etc/default/nis
```
Ensure that the `NISDOMAIN` matches the domain you set on the server:
```bash
NISDOMAIN=example.com
```

Edit the `/etc/yp.conf` file to add the NIS server IP address:
```bash
sudo nano /etc/yp.conf
```
Add the line:
```bash
ypserver <ip_of_nis_server>
```

### **Step 3: Start NIS Client Services**
Enable and start the necessary services for the NIS client:
```bash
sudo systemctl enable rpcbind
sudo systemctl start rpcbind
sudo systemctl enable ypbind
```

sudo systemctl start ypbind

### **Step 4: Verify NIS Client**
Test if the client can access the NIS server:
```bash
ypcat passwd
ypcat group
```

## **3. Install and Configure NFS Server**

### **Step 1: Install NFS Server**
On the **NFS server** (the machine that will share files), install the `nfs-kernel-server` package:
```bash
sudo apt update
sudo apt install -y nfs-kernel-server
```

### **Step 2: Configure NFS Server**
Create the directory you want to share via NFS. For example:
```bash
sudo mkdir -p /mnt/nfs_share
```

Edit the `/etc/exports` file to specify which directories should be shared:
```bash
sudo nano /etc/exports
```
Add the following line to share the `/mnt/nfs_share` directory:
```bash
/mnt/nfs_share  *(rw,sync,no_root_squash)
```

### **Step 3: Start NFS Services**
Enable and start the NFS services:
```bash
sudo systemctl enable nfs-kernel-server
sudo systemctl start nfs-kernel-server
```

### **Step 4: Export NFS Shares**
To export the shared directory, run:
```bash
sudo exportfs -a
```

### **Step 5: Verify NFS Server**
Check the NFS server's shared directories:
```bash
showmount -e
```

## **4. Install and Configure NFS Client**

### **Step 1: Install NFS Client**
On the **NFS client** machine (the machine that will access the NFS server's shared directories), install the `nfs-common` package:
```bash
sudo apt update
sudo apt install -y nfs-common
```

### **Step 2: Mount NFS Share**
Create a directory to mount the NFS share:
```bash
sudo mkdir -p /mnt/nfs_client
```

Mount the NFS share from the server:
```bash
sudo mount <nfs_server_ip>:/mnt/nfs_share /mnt/nfs_client
```

### **Step 3: Verify NFS Client**
Check that the NFS share is mounted correctly:
```bash
df -h
```

## **5. Demonstrate NIS and NFS Concepts**

### **NIS (Network Information Service)**
- **Centralized Authentication**: NIS allows centralized user and group management. Once you create a user or group on the NIS server, it becomes accessible on any NIS client in the network.
- **User Management**: On the NIS client, you can use the `ypcat` command to access the centralized `/etc/passwd` and `/etc/group` information from the NIS server.
- **Authentication**: Users created on the NIS server can log in to any NIS client with the same credentials.

### **NFS (Network File System)**
- **File Sharing**: NFS allows directories on one machine (NFS server) to be shared with other machines (NFS clients) over a network.
- **Access to Files**: NFS clients can access shared directories on the NFS server as if they were local files, enabling easy file sharing between multiple machines.

## **6. Important Configuration Files**

- **NIS Server Configuration Files**:

  - `/etc/default/nis` – Defines the NIS domain.
  - `/etc/yp.conf` – Configures NIS server address.
  - `/etc/passwd` – Contains user information (managed by NIS).
  - `/etc/group` – Contains group information (managed by NIS).

- **NFS Server Configuration Files**:
  - `/etc/exports` – Defines directories to share via NFS.
  - `/etc/exports.d/` – Optional directory for additional export files.

- **NIS Client Configuration Files**:
  - `/etc/default/nis` – Defines the NIS domain.
  - `/etc/yp.conf` – Configures the NIS server address.

- **NFS Client Configuration Files**:
  - `/etc/fstab` (optional) – Can be used to configure NFS mounts automatically.

## **Conclusion**

By following the above steps, you have successfully configured NIS for centralized user management and NFS for file sharing on your Linux machines. These services are valuable in environments where you need centralized user authentication and efficient file sharing across multiple machines on a network.
@@@@@@@@@@@@@

Assignment 32

32. Demonstrate go Applications: (any one)
Some notable open source applications written in Go include:
• Caddy, an open source HTTP/2 web server with automatic HTTPS capability.
• CockroachDB, an open source, survivable, strongly consistent, scale-out SQL database.
• Docker, a set of tools for deploying Linux containers
• Ethereum, The go-ethereum implementation of the Ethereum Virtual Machine blockchain for the Ether cryptocurrency
• Hugo, a static site generator

• InfluxDB, an open source database specifically to handle time series data with high availability and high performance requirements.
• InterPlanetary File System, a content-addressable, peer-to-peer hypermedia protocol.
• Juju, a service orchestration tool by Canonical, packagers of Ubuntu Linux
• Kubernetes container management system
• Lightning Network, a bitcoin network that allows for fast Bitcoin transactions and scalability.
• Mattermost, a teamchat system
• OpenShift, a cloud computing platform as a service by Red Hat
• Snappy, a package manager for Ubuntu Touch developed by Canonical.
• Syncthing, an open-source file synchronization client/server application
• Terraform, an open-source, multiple cloud infrastructure provisioning tool from HashiCorp.
@@@@@@@
I created simple go application:
To demonstrate a simple Go application, let's create a basic **HTTP server** that serves a "Hello, World!" message. This example is simple and can be easily understood by anyone new to Go.

### Steps to Create the Simple Go Application

1. **Install Go:**
   If you haven't installed Go yet, you can do so by following these steps:

   - On Ubuntu:
     ```bash
     sudo apt update
     sudo apt install golang
     ```

   - Verify installation:
     ```bash
     go version
     ```

2. **Create a New Go Project:**
   Let's create a simple Go project to serve an HTTP server.

   - Create a new directory for your project:
     ```bash
     ```

        mkdir go-hello-world
        cd go-hello-world

3. **Write the Go Code:**

   Create a new Go file, `main.go`, with the following content:

   ```go
   package main

   import (
        "fmt"
        "log"
        "net/http"
   )

   func handler(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "Hello, World!")
   }

   func main() {
        http.HandleFunc("/", handler)
        fmt.Println("Server started at http://localhost:8080")
        log.Fatal(http.ListenAndServe(":8080", nil))
   }
   ```

   Explanation:
   - **http.HandleFunc("/", handler)**: This tells Go to call the `handler` function when the root URL (`/`) is accessed.
   - **fmt.Fprintf(w, "Hello, World!")**: This sends the "Hello, World!" message as the HTTP response.
   - **http.ListenAndServe(":8080", nil)**: Starts the web server on port `8080`.

4. **Run the Go Application:**
   In the project directory, run the Go application using the following command:

   ```bash
   go run main.go
   ```

```

5. **Test the Application:**
   Open your web browser and navigate to:

   ```
   http://localhost:8080
   ```

   You should see the message: **"Hello, World!"** displayed.

### Explanation:
This Go program creates a simple HTTP server that listens on port 8080 and responds with the text "Hello, World!" when accessed via a web browser. It showcases how easy it is to create web applications in Go using its built-in `net/http` package.

### Output:
```
Server started at http://localhost:8080
```

Once you visit `http://localhost:8080`, you should see the output:

```
Hello, World!
```

This is the simplest possible Go application, demonstrating a basic HTTP server. Let me know if you'd like to see more advanced examples!
@@@@@@@CADDY GO TOOL
If you'd like to demonstrate an existing Go application, let's take the example of **"Caddy"**, which is a popular open-source web server written in Go. It’s simple to run and demonstrates the power of Go in handling web traffic.

### Demonstrating Caddy Web Server

Here are the steps to demonstrate **Caddy**, an HTTP/2 web server that is written in Go.

### 1. **Install Caddy**
```

#### On Ubuntu, you can install Caddy using these commands:

```bash
sudo apt update
sudo apt install -y debian-keyring debian-archive-keyring
curl -fsSL https://dl.cloudsmith.io/public/caddy/stable/gpg.key | gpg --dearmor -o
/usr/share/keyrings/caddy-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/caddy-archive-keyring.gpg]
https://dl.cloudsmith.io/public/caddy/stable/deb/ubuntu focal main" | sudo tee -a
/etc/apt/sources.list.d/caddy-stable.list
sudo apt update
sudo apt install caddy
```

### 2. **Verify Installation**

After installation, verify that Caddy is installed and accessible:

```bash
caddy version
```

You should see the version information for Caddy if it's installed correctly.

### 3. **Run Caddy**

To run Caddy with a default configuration, simply use:

```bash
sudo caddy start
```

This will start the web server and automatically serve files from the default web directory `/var/www/html`. Caddy will also enable HTTPS by default.

### 4. **Access the Web Server**

Open your browser and visit `http://localhost`. You should see the default Caddy landing page, confirming that the server is running.

If you want to customize the web server, you can edit the Caddyfile:

```bash
sudo nano /etc/caddy/Caddyfile
```

You can change it to serve a custom directory, like this:

```
localhost

root * /home/your-username/website
file_server
```

This tells Caddy to serve files from `/home/your-username/website` instead of the default location.

### 5. **Stop Caddy**

To stop the Caddy server, use:

```bash
sudo caddy stop
```

### 6. **View Logs and Status**

You can view the Caddy status or logs with:

```bash
sudo systemctl status caddy
sudo journalctl -u caddy
```

### Additional Demonstrations
1. **Using Caddy with HTTPS (Automatic TLS)**:

By default, Caddy automatically provides HTTPS certificates for any domain you configure in your Caddyfile, which is a great feature for production applications.

2. **Serving Static Files**:
   You can use Caddy to serve static websites, like a simple HTML site, by configuring the `root` directive in the Caddyfile.

### Summary:
- **Caddy** is a simple yet powerful Go-based web server with built-in support for HTTPS, easy-to-use configuration, and modern web protocols.
- You can install, configure, and run Caddy in a few simple steps, which makes it a great example to demonstrate the potential of Go in production-grade applications.

Let me know if you'd like help with any specific configuration or other existing Go applications!
@@@@@@@@@
Hugo is a popular static site generator built with **Go (Golang)**, used to create fast, flexible, and customizable websites. It simplifies web development by allowing users to generate a static website based on Markdown content, templates, and configuration files.

Here is a step-by-step guide on how to install and demonstrate Hugo on a Linux system.

### Step 1: **Install Hugo**

1. **Update the system**:

   ```bash
   sudo apt update
   sudo apt upgrade -y
   ```

2. **Install Hugo**:

   You can install Hugo using the Snap package manager, which is the easiest way:

   ```bash
   ```

   sudo snap install hugo --channel=extended

   Alternatively, you can install it using the following steps (if Snap is not available):

   ```bash
   sudo apt install wget
   wget
https://github.com/gohugoio/hugo/releases/download/v0.118.2/hugo_0.118.2_Linux-64bi
t.deb
   sudo dpkg -i hugo_0.118.2_Linux-64bit.deb
   ```

3. **Verify the Installation**:

   After installation, check if Hugo is correctly installed:

   ```bash
   hugo version
   ```

   This will display the installed version of Hugo.

### Step 2: **Create a New Hugo Site**

1. **Create a new Hugo site**:

   Use the `hugo new site` command to create a new site:

   ```bash
   hugo new site my-website
   cd my-website
   ```

2. **Add a theme**:

   Hugo uses themes for its templates and design. You can add a theme by cloning it
from Hugo's themes repository.

   ```bash
   ```

   git init
   git submodule add https://github.com/gohugoio/hugo-theme-ananke.git
themes/ananke

3. **Configure the site**:

   Hugo has a `config.toml` file where you configure site-specific settings. Open this file:

   ```bash
   nano config.toml
   ```

   Modify the `config.toml` to specify the theme and other settings:

   ```toml
   baseURL = "http://example.org/"
   languageCode = "en-us"
   title = "My Hugo Site"
   theme = "ananke"
   ```

4. **Create a new content page**:

   Now, create a new Markdown file (content page) using the following command:

   ```bash
   hugo new posts/my-first-post.md
   ```

   This will create a new `.md` file in the `content/posts` directory. Open it and add some
content:

   ```bash
   nano content/posts/my-first-post.md
   ```

   Add content to the post, such as:

   ```markdown
   ```

   ---
   title: "My First Post"
   date: 2024-11-10T10:00:00+05:30
   draft: true

   Welcome to my first Hugo post. Hugo is an amazing tool for building static websites quickly.

   Make sure to save and exit the editor.

### Step 3: **Build and Preview the Site**

1. **Build the site**:

   Run the following command to generate the static files for your website:

   ```bash
   hugo
   ```

   Hugo will generate the content in the `public/` directory.

2. **Preview the site locally**:

   Start a local development server to preview your site:

   ```bash
   hugo server
   ```

   This will start a local server at `http://localhost:1313/`. You can open your browser and see your Hugo website in action.

### Step 4: **Deploy the Site**

Once you're satisfied with the site, you can deploy it. Since Hugo generates static content, you can upload the files in the `public/` directory to any web hosting platform (like GitHub Pages, Netlify, or a traditional web server).

For example, to deploy using GitHub Pages:

1. **Push the site to a GitHub repository**:

   ```bash
   git init
   git add .
   git commit -m "Initial commit"
   git remote add origin https://github.com/yourusername/my-website.git
   git push -u origin master
   ```

2. **Deploy the `public` directory to GitHub Pages**:

   Hugo provides a simple way to deploy using a GitHub Pages branch:

   ```bash
   hugo --destination=public
   ```

   Push the `public` directory to the `gh-pages` branch:

   ```bash
   git subtree push --prefix public origin gh-pages
   ```

### Step 5: **Additional Features**

Hugo offers several powerful features such as:

- **Content Taxonomies**: You can create tags and categories for organizing your content.
- **Custom Shortcodes**: Embed reusable content components in your posts.
- **Data Files**: Store structured data in JSON, YAML, or TOML files and use them in your templates.
- **Multilingual Support**: Build multilingual sites with ease.

### Conclusion

Hugo is an excellent choice for building static websites, and it integrates seamlessly with other tools like GitHub Pages, Netlify, and many more. Its speed and flexibility make it a preferred option for many developers.
@@@@@@@@@@@

—---

Note: I will underline exact command to be put in the terminal, just copy and paste

There are many specified tools written in Go language, but what we are most familiar with is docker.

1. Docker
   a. First We install docker
   b. We create a simple project to demonstrate docker

a. Installation of docker

—---

Uninstall old versions

Before you can install Docker Engine, you need to uninstall any conflicting packages.

Distro maintainers provide unofficial distributions of Docker packages in APT. You must uninstall these packages before you can install the official version of Docker Engine.

The unofficial packages to uninstall are:

- `docker.io`
- `docker-compose`
- `docker-compose-v2`
- `docker-doc`
- `podman-docker`

Run the following command in terminal:

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

—---

Install using the `apt` repository

Set up Docker's `apt` repository.

# Add Docker's official GPG key:

```
sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc
```

# Add the repository to Apt sources:

```
echo \

  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \

  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \

  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```

Then Install the Docker packages. To install the latest version, run:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Verify that the Docker Engine installation is successful by running the `hello-world` image.

sudo docker run hello-world

Note: from now onwards if you want to use docker command, use it with sudo

—---

a. Create a simple project.
   We will create a simple index.html file, we will install apache or ngnix webserver docker image and run our project in that docker image container.

Create a directory for project, and go to that directory.

Create a index.html file and write below contents and save it

—---

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Hello World</title>
</head>
<body>
   <h1>Hello, World!</h1>
</body>
</html>
```

—---

Or only below is enough

```
<h1>Hello, World!</h1>
```

—---

Now create a file named Dockerfile and paste the below contents

—---

```
# Use the official Nginx image from Docker Hub
```

```
FROM nginx:alpine

# Copy the index.html to the default Nginx HTML location
COPY index.html /usr/share/nginx/html/index.html

# Expose port 80 to the outside world
EXPOSE 80

# Start Nginx server
CMD ["nginx", "-g", "daemon off;"]
```

—--

Now we have to run commands
To build the docker image

```
sudo docker build -t hello-world-app .
```

To run the docker container

```
sudo docker run -p 80:80 hello-world-app
```

Now go to your browser and type localhost:8000
Your docker container should be running and hello world should be displayed.

34)Demonstrate use of bug tracking tool/any foss tool and create the docker image of that tool. Push that image. Run the docker container from recently created image and run that docker container. Push that image.

Let's walk through the steps to achieve this in Docker, including using a bug-tracking tool like **Bugzilla** (a popular FOSS bug-tracking tool). We'll go through:

1. **Installing Docker** (if needed).
2. **Setting up the Dockerfile** for Bugzilla.

3. **Building the Docker image**.
4. **Pushing the Docker image** to Docker Hub.
5. **Running a container** from your Docker image.

### Step 1: Install Docker
If Docker isn’t installed, here’s how to get it:
   - For **Linux**: Use the command `sudo apt-get install docker.io`.
   - For **MacOS/Windows**: Download Docker Desktop from
[docker.com](https://www.docker.com/products/docker-desktop).

### Step 2: Write a Dockerfile for Bugzilla
1. **Create a new directory** for your Docker project, and then **create a file** named `Dockerfile` in this directory.

1. **Create a new directory** for your Docker project:

   ```bash

   mkdir bugzilla-docker

   ```

2. **Navigate into the directory**:

   ```bash

   cd bugzilla-docker

2. Add the following content to the `Dockerfile`:
     Use correct and accurate docker file code .if error in code it will not create image.
3. This Dockerfile does the following:
   - Uses an Apache server base image.
   - Installs the necessary Perl modules for Bugzilla.
   - Downloads Bugzilla and configures it.

### Step 3: Build the Docker Image
In the directory containing your Dockerfile, build the image using:
   ```

```bash
   docker build -t your_dockerhub_username/bugzilla .
   ```

### Step 4: Push the Docker Image to Docker Hub
1. Log in to Docker Hub:

   ```bash
   docker login
   ```

2. Push your image to Docker Hub:

   ```bash
   docker push your_dockerhub_username/bugzilla
   ```

### Step 5: Run a Docker Container from Your Image
Once pushed, you can run a container from your image:

   ```bash
   docker run -d -p 8080:80 your_dockerhub_username/bugzilla
   ```

This command:
- Runs the Bugzilla container in the background (`-d`).
- Maps port 8080 on your local machine to port 80 in the container.

You can access Bugzilla in your browser by navigating to `http://localhost:8080`.

### Summary of Commands:
Here's a quick summary of the main Docker commands we used:
   - **`docker build -t your_dockerhub_username/bugzilla .`** – Builds the Docker image.
   - **`docker push your_dockerhub_username/bugzilla`** – Pushes the image to Docker Hub.

   - **`docker run -d -p 8080:80 your_dockerhub_username/bugzilla`** – Runs a container from the image.

35. Write a Docker File to pull the Ubuntu with open jdk and write any java application.

Create one folder

Create java application in folder and Dockerfile

Write script in Dockerfile

Script:

```
# Use the latest Ubuntu image

FROM ubuntu:latest

# Update the package list and install OpenJDK

RUN apt update && apt install -y default-jdk

# Set the working directory in the container

WORKDIR /usr/src/app

# Copy the Java application code to the working directory

COPY . .

# Compile the Java application

RUN javac javaapp.java

# Set the command to run the Java application

CMD ["java", "javaapp"]
```

Javaapp.java contain some application code.

Build that image ,create container ,run that container.

Bugtracking tool demo.