Linux Server with Package Installation

NFS

NFS on a Linux system (using Ubuntu as an example):

1. Update Package Lists:

sudo apt-get update

2. Install NFS Server:

sudo apt-get install nfs-kernel-server

3. Configure NFS Exports:

Edit the /etc/exports file to specify directories you want to share. For example, if you want to share the /home/user/documents directory:

sudo nano /etc/exports

3. Add a line like:

/home/user/documents
*(rw,sync,no_subtree_check)

3. Save the file and exit.
4. Restart NFS Server:

sudo service nfs-kernel-server restart

5. Check NFS Status:

Ensure NFS server is running without errors:

sudo service nfs-kernel-server status

6. Configure Firewall (if applicable):

If you're using a firewall, you might need to allow NFS traffic. For example, on Ubuntu with UFW:

sudo ufw allow from <client_IP> to any port nfs

7. Mount NFS Share on Client:

On the client machine, create a directory and mount the NFS share:

sudo mkdir /mnt/nfs_share
sudo mount <server_IP>:/home/user/documents /mnt/nfs_share

7. Replace <server_IP> with the IP address of your NFS server.
8. Verify Mount:

Check if the share is mounted successfully:

df -h

8. You should see your NFS share listed.

Samba

Samba on a Linux system (using Ubuntu as an example):

1. Update Package Lists:

sudo apt-get update

2. Install Samba:

```
sudo apt-get install samba
```

3. Create a Directory to Share:

Choose a directory you want to share, for example, /home/user/share.

```
sudo mkdir /home/user/share
```

4. Configure Samba:

Edit the Samba configuration file:

```
sudo nano /etc/samba/smb.conf
```

4. At the end of the file, add the following lines:

```
[share]
  path = /home/user/share
  read only = no
```

browsable = yes
guest ok = yes

4. Save the file and exit.
5. Set Samba Password for User:

sudo smbpasswd -a user

5. Replace user with your Linux username. You'll be prompted to set a Samba password.
6. Restart Samba:

sudo service smbd restart

7. Configure Firewall (if applicable):
If you're using a firewall, you might need to allow Samba traffic. For example, on Ubuntu with UFW:

sudo ufw allow from <client_IP> to any port 139,445

8. Accessing the Share from a Windows Machine:

Open File Explorer on your Windows machine and enter the following in the address bar:

\\<server_IP>\share

8. Replace <server_IP> with the IP address of your Linux server.
You may be prompted to enter the Samba username and password.

FTP

Install an FTP server on Linux, you can use vsftpd (Very Secure FTP Daemon), which is a lightweight and secure FTP server. Here's a step-by-step guide for installing and configuring vsftpd on a Linux system (using Ubuntu as an example):

1. Update Package Lists:

sudo apt-get update

2. Install vsftpd:

sudo apt-get install vsftpd

3. Configure vsftpd:

Edit the vsftpd configuration file:

sudo nano /etc/vsftpd.conf

3. Make sure the following settings are adjusted or uncommented:

anonymous_enable=NO
local_enable=YES
write_enable=YES
chroot_local_user=YES

3. Save the file and exit.
4. Restart vsftpd:

sudo service vsftpd restart

5. Configure Firewall (if applicable):

If you're using a firewall, you might need to allow FTP traffic. For example, on Ubuntu with UFW:

sudo ufw allow 20/tcp
sudo ufw allow 21/tcp

6. Access FTP Server:

You can use an FTP client to connect to your server. Use the server's IP address or domain, and connect with a valid local user account.
If connecting locally, you can use:

ftp localhost

6. If connecting remotely, replace "localhost" with your server's IP address or domain.

Apache

Apache on a Linux system (using Ubuntu as an example):

1. Update Package Lists:

sudo apt-get update

2. Install Apache:

sudo apt-get install apache2

3. Start Apache Service:

Apache should start automatically after installation. If not, you can start it manually:

sudo service apache2 start

4. Configure Firewall (if applicable):

If you're using a firewall, you might need to allow HTTP traffic. For example, on Ubuntu with UFW:

sudo ufw allow 80/tcp

5. Check Apache Status:

Ensure Apache is running without errors:

sudo service apache2 status

6. Access Apache Default Page:

Open a web browser and enter your server's IP address or domain. You should see the default Apache page indicating a successful installation.

7. Configure Virtual Hosts (Optional):

If you want to host multiple websites on the same server, you can set up virtual hosts. Edit the default configuration or create new ones in the /etc/apache2/sites-available/ directory.

8. Secure Apache with Let's Encrypt (Optional):

To enable HTTPS and secure your site, you can use Let's Encrypt. Install Certbot:

```
sudo apt-get install certbot python3-certbot-apache
```

8. Obtain and install SSL certificate:

```
sudo certbot --apache
```

9. Enable Apache Modules (Optional):

Depending on your needs, you may want to enable additional Apache modules. For example, to enable the rewrite module:

sudo a2enmod rewrite

10. Restart Apache:

After making any configuration changes, restart Apache for the changes to take effect:

sudo service apache2 restart

DNS

configuring a DNS server on Linux typically involves using BIND (Berkeley Internet Name Domain). Below is a step-by-step guide using Ubuntu as an example:

1. Update Package Lists:

sudo apt-get update

2. Install BIND9 (DNS Server):

```
sudo apt-get install bind9
```

3. Configure BIND:

The primary configuration file for BIND is /etc/bind/named.conf.options. Open it with a text editor:

```
sudo nano /etc/bind/named.conf.options
```

3. Configure your DNS server settings. Here's an example:

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

    forwarders {
        8.8.8.8;
        8.8.4.4;
    };

    listen-on { any; };
    allow-query { any; };
```

};

3. Save the file and exit.
4. Create Zone Files:

BIND uses zone files to map domain names to IP addresses. Create a forward and reverse zone file for your domain. For example, create /etc/bind/db.example.com for a domain named “example.com”:

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

4. Add content like this (adjust IP addresses and domain as needed):

```
$TTL    604800
@       IN      SOA     ns1.example.com.
admin.example.com. (
                2022021401      ; Serial
                604800          ; Refresh
                86400           ; Retry
                2419200         ; Expire
                604800 )        ; Negative Cache TTL
;
@       IN      NS      ns1.example.com.
```

```
@       IN      A       192.168.1.1
ns1     IN      A       192.168.1.1
```

4. Save and exit. Repeat the process for reverse zone file (/etc/bind/db.192 in this example).

5. Update BIND Configuration:

Modify the named.conf.local file to include your zone files:

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

5. Add lines like this (replace “example.com” with your domain):

```
zone "example.com" {
    type master;
    file "/etc/bind/db.example.com";
};
zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192";
};
```

5. Save and exit.

6. Restart BIND:

sudo service bind9 restart

7. Test DNS Resolution:

Use tools like nslookup or dig to check DNS resolution. For example:

nslookup example.com

7. Ensure the resolved IP matches your DNS configuration.

NGINX

configure Nginx on a Linux system (using Ubuntu as an example):

1. Update Package Lists:

sudo apt-get update

2. Install Nginx:

sudo apt-get install nginx

3. Start Nginx Service:

Nginx should start automatically after installation. If not, you can start it manually:

sudo service nginx start

4. Configure Firewall (if applicable):

If you're using a firewall, you might need to allow HTTP traffic. For example, on Ubuntu with UFW:

sudo ufw allow 80/tcp

5. Check Nginx Status:

Ensure Nginx is running without errors:

sudo service nginx status

6. Access Nginx Default Page:

Open a web browser and enter your server's IP address or domain. You should see the default Nginx welcome page, indicating a successful installation.

7. Create Virtual Hosts (Optional):

If you want to host multiple websites on the same server, you can set up virtual hosts. Edit the default configuration or create new ones in the /etc/nginx/sites-available/ directory.

8. Secure Nginx with Let's Encrypt (Optional):

To enable HTTPS and secure your site, you can use Let's Encrypt. Install Certbot:

```
sudo apt-get install certbot python3-certbot-nginx
```

8. Obtain and install SSL certificate:

```
sudo certbot --nginx
```

9. Enable Nginx Modules (Optional):

Depending on your needs, you may want to enable additional Nginx modules. For example, to enable the rewrite module:

sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/

10. Restart Nginx:

After making any configuration changes, restart Nginx for the changes to take effect:

sudo service nginx restart

OPENSSH-SERVER

configure OpenSSH on a Linux system, using Ubuntu as an example:

1. Update Package Lists:

sudo apt-get update

2. Install OpenSSH:

sudo apt-get install openssh-server

3. Start OpenSSH Service:

OpenSSH server should start automatically after installation. If not, you can start it manually:

```
sudo service ssh start
```

4. Check OpenSSH Status:

Ensure OpenSSH is running without errors:

```
sudo service ssh status
```

5. Configure Firewall (if applicable):

If you're using a firewall, you might need to allow SSH traffic. For example, on Ubuntu with UFW:

```
sudo ufw allow 22/tcp
```

6. Test SSH Connection:

Open a terminal on another machine and try connecting to your server using SSH. Replace

<your_server_ip> with the actual IP address of your server:

ssh username@<your_server_ip>

6. You'll be prompted to enter the user's password.
7. Configure SSH (Optional):

Edit the SSH server configuration file /etc/ssh/sshd_config if you need to customize settings. For example:

sudo nano /etc/ssh/sshd_config

7. Make changes, save the file, and restart the SSH service:

sudo service ssh restart

8. Generate SSH Key Pair (Optional):

You can generate an SSH key pair for secure and passwordless logins:

ssh-keygen -t rsa -b 2048

8. Follow the prompts and keep the default options if unsure. This will create a private key (~/.ssh/id_rsa) and a public key (~/.ssh/id_rsa.pub).

9. Copy SSH Public Key to Server (Optional): If you generated an SSH key pair, you can copy the public key to your server for passwordless authentication:

ssh-copy-id username@<your_server_ip>

9. Enter the user's password when prompted.

10. Secure SSH Configuration (Optional): You can further secure your SSH server by disabling password authentication and using only key-based authentication. Edit the /etc/ssh/sshd_config file:

sudo nano /etc/ssh/sshd_config

10. Update the following settings:

PasswordAuthentication no

ChallengeResponseAuthentication no

10. Save the file, and restart the SSH service.
11. Restart SSH:

After making any configuration changes, restart the SSH service:

sudo service ssh restart

DOCKER

install Docker on a Linux system, using Ubuntu as an example:

1. Update Package Lists:

sudo apt-get update

2. Install Dependencies:

Ensure necessary packages are installed:

sudo apt-get install apt-transport-https ca-certificates curl software-properties-common

3. Add Docker GPG Key:

Add Docker's official GPG key to verify the integrity of the packages:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. Set Up Stable Docker Repository:

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Update Package Lists (Again):

```
sudo apt-get update
```

6. Install Docker Engine:

sudo apt-get install docker-ce docker-ce-cli containerd.io

7. Add User to Docker Group (Optional):

To run Docker commands without sudo, add your user to the docker group:

sudo usermod -aG docker $USER

7. Log out and log back in or restart your system for changes to take effect.

8. Verify Docker Installation:

Check the Docker version to verify the installation:

docker --version

8. Also, test Docker by running a simple container:

docker run hello-world

8. If everything is set up correctly, you should see a “Hello from Docker!” message.

GRAFANA

Here’s a step-by-step guide on how to install Grafana on a Linux system, using Ubuntu as an example:

1. Update Package Lists:

```
sudo apt-get update
```

2. Install Grafana Repository Configuration: Add the Grafana APT repository to your system:

```
sudo apt-get install -y software-properties-common
sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
```

3. Add GPG Key for Repository:

```
wget -q -O -
https://packages.grafana.com/gpg.key | sudo apt-
key add -
```

4. Install Grafana:

```
sudo apt-get update
sudo apt-get install grafana
```

5. Start Grafana Service:

```
sudo service grafana-server start
```

6. Enable Grafana Service on Boot:

```
sudo systemctl enable grafana-server
```

7. Access Grafana Web Interface:

Open a web browser and go to http://localhost:3000. The default login is:

- Username: admin
- Password: admin

You'll be prompted to change the password upon the first login.

8. Configure Data Sources (Optional):

- Once logged in, you can configure data sources to connect Grafana with various databases or services.

9. Create Dashboards (Optional):

- After configuring data sources, you can create dashboards to visualize and monitor your data.

10. Explore Plugins (Optional):

- Grafana supports various plugins that extend its functionality. Explore and install plugins based on your requirements.

11. Secure Grafana (Optional):

- For production environments, consider securing Grafana with SSL/TLS and configuring authentication.

12. Restart Grafana (if needed):

```
sudo service grafana-server restart
```