If I want to develop some APPs on ubuntu 24.04 and it need a dedicated environment such as python3.7, project.toml, pypdf2, libjpeg and etc..., and I want to install docker engine, container, compose and nginx proxy manager etc... should I install the required environment and libraries like mysql or mariadb, postgresql etc... first and then install docker or what is the best practice for this model & note: I use one VPS hosting , Multi domains and multi subdomains to run my APPs & Services?

Setting up your environment for developing and deploying applications on Ubuntu 24.04, especially when using Docker and managing multiple domains and subdomains, involves several steps to ensure a smooth and efficient workflow. Here’s a structured approach you can follow:

### 1. Installing Necessary Dependencies and Libraries

First, ensure your base Ubuntu 24.04 system is up to date:

```bash

sudo apt update

sudo apt upgrade

```

#### Installing Python 3.7 and Required Libraries

Since you need Python 3.7 and specific libraries like `pypdf2` and `libjpeg`, you can install Python 3.7 from a third-party repository or build it from source if necessary. Here’s a general outline:

```bash

# Install Python 3.7 (example using deadsnakes PPA)

sudo apt install software-properties-common

sudo add-apt-repository ppa:deadsnakes/ppa

sudo apt update

sudo apt install python3.7 python3.7-dev

# Set Python 3.7 as default if needed

sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.7 1

# Install pip for Python 3.7

sudo apt install python3.7-distutils python3.7-venv

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

sudo python3.7 get-pip.py

# Install specific libraries (example with pypdf2 and libjpeg)

pip install pypdf2

sudo apt install libjpeg-dev # For libjpeg dependency

# Optionally, create a virtual environment for your project

python3.7 -m venv myenv

source myenv/bin/activate

```

#### Installing Database Servers (MySQL/MariaDB, PostgreSQL)

If your applications require databases like MySQL/MariaDB or PostgreSQL:

```bash

# Install MySQL/MariaDB

sudo apt install mysql-server # for MySQL

# OR

sudo apt install mariadb-server # for MariaDB

# Install PostgreSQL

sudo apt install postgresql postgresql-contrib

```

Configure these databases according to your application needs (setting up users, databases, permissions).

### 2. Installing Docker Engine and Docker Compose

Docker allows you to containerize your applications, which is especially useful for managing multiple services on a single VPS.

#### Installing Docker Engine

Follow the official Docker documentation for installing Docker Engine on Ubuntu: [Install Docker Engine on Ubuntu](https://docs.docker.com/engine/install/ubuntu/)

```bash

# Install Docker dependencies

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

# Add Docker GPG key

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

# Add 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

# Install Docker Engine

sudo apt update

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

# Add your user to the docker group to run Docker commands without sudo

sudo usermod -aG docker $USER

```

#### Installing Docker Compose

Docker Compose simplifies the process of defining and running multi-container Docker applications.

```bash

# Install Docker Compose (adjust version as needed)

sudo curl -L "https://github.com/docker/compose/releases/download/{VERSION}/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# Apply executable permissions to the binary

sudo chmod +x /usr/local/bin/docker-compose

# Verify installation

docker-compose --version

```

### 3. Setting Up Nginx Proxy Manager

Nginx Proxy Manager helps manage Nginx configurations and reverse proxy settings for your multiple domains and subdomains.

```bash

# Install Nginx Proxy Manager (using Docker Compose)

# Create a docker-compose.yml file and configure Nginx Proxy Manager service

nano docker-compose.yml

# Example docker-compose.yml for Nginx Proxy Manager

version: '3'

services:

app:

image: jc21/nginx-proxy-manager:latest

ports:

- "80:80"

- "443:443"

- "81:81"

volumes:

- ./data:/data

- ./letsencrypt:/etc/letsencrypt

# Start Nginx Proxy Manager

docker-compose up -d

```

### Summary

- \*\*Order of Installation\*\*: Install necessary libraries (like Python and its dependencies) and database servers first before Docker.

- \*\*Using Docker\*\*: Dockerize your applications for easier management and scalability.

- \*\*Nginx Proxy Manager\*\*: Use it to manage domain and subdomain routing effectively.

Following this structured approach ensures you have a robust environment for developing and deploying your applications on Ubuntu 24.04 with Docker and Nginx Proxy Manager. Adjust configurations and versions as per your specific application requirements.

Can I install nginx proxy manager first, so I can use it to run docker through subdomain for example: docker.mydomain.org? and the install docker or it doesn't make sense?

Yes, you can install Nginx Proxy Manager first and configure it to handle subdomains like `docker.mydomain.org`. This approach allows you to manage incoming traffic and route it to different Docker containers based on subdomains or paths.

Here’s how you can approach it:

### 1. Install Nginx Proxy Manager

First, set up Nginx Proxy Manager using Docker Compose or any method you prefer. Docker Compose is straightforward for this purpose:

```yaml

# docker-compose.yml for Nginx Proxy Manager

version: '3'

services:

app:

image: jc21/nginx-proxy-manager:latest

ports:

- "80:80"

- "443:443"

- "81:81"

volumes:

- ./data:/data

- ./letsencrypt:/etc/letsencrypt

# Start Nginx Proxy Manager

docker-compose up -d

```

This configuration will start Nginx Proxy Manager on your VPS, listening on ports 80 (HTTP), 443 (HTTPS), and 81 (Admin panel).

### 2. Configure Nginx Proxy Manager

After Nginx Proxy Manager is running, you can access its web interface (usually on port 81) to configure your domains and subdomains. Set up a subdomain like `docker.mydomain.org` and configure it to proxy to Docker containers.

### 3. Install Docker and Configure Containers

Next, install Docker and Docker Compose on your VPS if you haven’t already (following the steps mentioned earlier). Docker containers can then be configured to run various services (like web servers, databases, etc.).

### 4. Configure Docker Containers

When setting up Docker containers, you can use custom network configurations (like Docker Compose networks) to isolate services and connect them to Nginx Proxy Manager through specified subdomains.

### Benefits of this Approach:

- \*\*Centralized Management\*\*: Nginx Proxy Manager allows you to manage and configure all your subdomains and their corresponding Docker containers from a single interface.

- \*\*Security and Scalability\*\*: By routing traffic through Nginx Proxy Manager, you can apply SSL certificates, rate limiting, and other security features easily.

- \*\*Flexibility\*\*: You can scale your applications by adding more Docker containers and configuring additional subdomains without modifying Nginx configurations manually.

### Summary

Installing Nginx Proxy Manager first is a sensible approach as it provides a centralized point for managing your domain and subdomain routing. It simplifies the process of adding and configuring Docker containers for various services under different subdomains (`docker.mydomain.org` in your case). This setup allows for efficient management and scaling of your applications on a single VPS.