Setting up Ubuntu 22.04.2 LTS for Python Development

1. Installing Ubuntu 22.04.2 LTS

Ensure that you have a clean installation of Ubuntu 22.04.2 LTS on your system. You can download the ISO file and create a bootable USB drive or install it directly from the ISO image.

2. Updating System Packages

Before installing any software, it's essential to update the system packages to their latest versions. Open a terminal and run the following commands:

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

This will fetch the latest package information and update your system.

3. Installing Python 3 and pip

We will install Python 3 and pip, the package manager for Python. In this example, we'll install Python 3.10.12 and pip version 22.0.2. Run the following commands:

```
sudo apt install python3-pip
```

This command installs Python 3 and pip on your system. You can verify the installation by running:

```
python3 --version
pip --version
```

4. Installing curl

Curl is a command-line tool for transferring data with URLs. You can install it with the following command:

```
sudo apt install curl
```

Curl is commonly used for various tasks, such as downloading files from the internet.

5. Installing Node.js and npm

Node.js is a JavaScript runtime that allows you to run JavaScript on the server-side. npm is the Node Package Manager used to manage JavaScript packages. We will install Node.js version 18.17.1 and npm version 9.6.7. Run the following commands to add the Node.js repository and install both Node.js and npm:

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs
```

This adds the Node.js repository, updates the package list, and installs Node.js and npm.

6. Installing Miniconda

Miniconda is a minimal installer for conda, a package manager for Python and other languages. We will install Miniconda from the official website.

1. Download Miniconda from the official website (https://docs.conda.io/en/latest/miniconda.html (https://docs.conda.io/en/latest/miniconda.html)) for your specific system. Choose the 64-bit Linux version (Miniconda3-latest-Linux-x86_64.sh).
2. Open a terminal and navigate to the folder where you downloaded the Miniconda installer.
3. Run the following command to execute the installer (replace "Miniconda3-latest-Linux-x86_64.sh" with the actual filename):

```
bash Miniconda3-latest-Linux-x86_64.sh
```

Follow the on-screen prompts to install Miniconda. You can choose the installation location and other settings according to your preferences.

7. Restarting the Terminal

After the installation of Miniconda, it's a good practice to restart the terminal to ensure that the changes take effect.

You have successfully set up Ubuntu 22.04.2 LTS for Python development with the necessary tools and packages.

Setting up JupyterHub with Conda

This documentation outlines the steps to set up JupyterHub using Conda on a Linux system. JupyterHub allows multiple users to access JupyterLab instances on a single server.

Step 1: Create Symbolic Link to Conda Environment

To ensure that Conda is available in the system environment when users log in, create a symbolic link to the Conda profile script:

```
sudo ln -s /home/vuswh/miniconda3/etc/profile.d/conda.sh /etc/profile.d/conda.sh
```

Step 2: Install JupyterHub with Conda

To install JupyterHub using Conda, you'll need to switch to a root shell:

```
sudo -s
```

Then, source the Conda environment for the root user:

```
. /etc/profile.d/conda.sh
```

Step 3: Create Conda Environment for JupyterHub

Create a Conda environment named "jupyterhub" with JupyterHub, JupyterLab, and ipywidgets installed:

```
conda create --name jupyterhub jupyterhub jupyterlab ipywidgets
```

Step 4: Create JupyterHub Configuration File

Create a JupyterHub configuration file and specify the full path to the JupyterHub executable within the Conda environment:

```
mkdir -p /home/vuswh/miniconda3/envs/jupyterhub/etc/jupyterhub
cd /home/vuswh/miniconda3/envs/jupyterhub/etc/jupyterhub
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyterhub --generate-config
```

Step 5: Use systemd to Start JupyterHub on Boot

Create a systemd "Unit" file to start JupyterHub as a service:

```
sudo mkdir -p /home/vuswh/miniconda3/envs/jupyterhub/etc/systemd
sudo cat << EOF > /home/vuswh/miniconda3/envs/jupyterhub/etc/systemd/jupyterhub.service
[Unit]
Description=JupyterHub
After=syslog.target network.target

[Service]
User=root
Environment="PATH=/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/home/vuswh/miniconda3/envs/jupyterhub/bin"
ExecStart=/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyterhub -f /home/vuswh/miniconda3/envs/jupyterhub/etc/jupyterhub/jupyterhub_

[Install]
WantedBy=multi-user.target
EOF
```

Step 6: Link to systemd Directory

Create a symbolic link to the systemd directory to enable the JupyterHub service:

```
sudo ln -s /home/vuswh/miniconda3/envs/jupyterhub/etc/systemd/jupyterhub.service /etc/systemd/system/jupyterhub.service
```

Step 7: Start and Enable JupyterHub Service

Start JupyterHub and enable it as a service to run on boot:

```
systemctl start jupyterhub.service
systemctl enable jupyterhub.service
```

Step 8: Check JupyterHub Service Status

To check if JupyterHub is running or not, use the following command:

```
systemctl status jupyterhub.service
```

Configuring JupyterHub for nbgrader in a Multi-Class Lab Environment

In a multi-class lab environment, configuring JupyterHub to work seamlessly with nbgrader requires specific steps to enable autograding services, promote users to instructors, and create new courses. This section outlines the necessary configurations and customizations.

1.9 JupyterHub Configuration

1.9.1 Autograding Service

To facilitate multi-instructor access and ensure student code doesn't run in an instructor's user account, you need to set up an autograding service. The following steps provide the necessary configuration:

1. Replace `JUPYTERHUB_API_TOKEN` with `JUPYTERHUB_API_TOKEN_CUSTOM` in `/home/vuswh/miniconda3/envs/jupyterhub/lib/python3.8/site-packages/nbgrader/auth/jupyterhub.py`.

2. Go to JupyterHub's Hub Control Panel, click the admin button, and create an API token. Then add the following line to your JupyterHub config file:

```
c.Spawner.environment = { 'JUPYTERHUB_API_TOKEN_CUSTOM': '0123456789abcdef0123456789abcdef' }
```

3. In JupyterHub's config file, create a token service for this API token:

```
c.JupyterHub.services = [
    {
        'name': 'nbgrader_token_service',
        'api_token': '0123456789abcdef0123456789abcdef'
    }
]
```

4. Add the following role to your JupyterHub config file if it doesn't already exist:

```
c.JupyterHub.load_roles = [
    {
        'name': 'nbgrader_token_role',
        'scopes': ['read:users:groups', 'list:services', 'groups', 'admin:users'],
        'services': ['nbgrader_token_service']
    }
]
```

Ensure that you add these configurations appropriately, taking care not to duplicate `c.JupyterHub.services` and `c.JupyterHub.load_roles` lines in your config file.

1.10 Promote Regular Hub Users to Instructors

Instructors in your JupyterHub are users with admin access and specific nbgrader Jupyter extensions enabled. To promote a user to an instructor, follow these steps:

1. Add the instructor's username to `c.Authenticator.admin_users` in your JupyterHub's config file.

2. Log in to the instructor user's account via SSH or JupyterLab terminal and run the following commands to enable relevant nbgrader Jupyter extensions:

```
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension enable --user course_list/main --section=tree
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension enable --user nbgrader.server_extensions.course_list
```

1.11 Create a New Course

To create a new nbgrader course, you need to set up a dedicated user (the grader) responsible for running the autograding tool. Follow these steps:

1. Create a new user for the grader:

```
sudo adduser grader-test-course
```

2. In your JupyterHub's config file, grant the grader access to the hub by adding their username to `c.Authenticator.allowed_users`.

3. Create two groups in your JupyterHub's config file:

```
c.JupyterHub.load_groups = {
    'formgrade-test-course': ['instructor', 'grader-test-course'],
    'nbgrader-test-course': []
}
```

Note that the group names are prescribed by nbgrader, and the second group remains empty.

4. Configure the autograding service in your hub's config file:

```
c.JupyterHub.services = [
    {
        'name': 'test-course',
        'url': 'http://127.0.0.1:8101',
        'command': ['jupyterhub-singleuser', '--group=formgrade-test-course', '--debug'],
        'user': 'grader-test-course',
        'cwd': '/home/grader-test-course',
        'api_token': '0123456789abcdef0123456789abcdef',
        'environment': { 'JUPYTERHUB_API_TOKEN_CUSTOM': '0123456789abcdef0123456789abcdef' }
    }
]
```

Ensure that you include only one `c.JupyterHub.services` line and adjust `c.JupyterHub.load_roles` if necessary.

5. Log in to the grader's account and activate relevant nbgrader Jupyter extensions:

```
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension enable --user create_assignment/main
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension enable --user formgrader/main --section=tree
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension enable --user nbgrader.server_extensions.formgrader
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter nbextension disable --user assignment_list/main --section=tree
/home/vuswh/miniconda3/envs/jupyterhub/bin/jupyter serverextension disable --user nbgrader.server_extensions.assignment_list
```

6. Create the grader's `~/.jupyter/nbgrader_config.py` file with appropriate configurations. For example:

```
c = get_config()
c.CourseDirectory.root = '/home/grader-test-course/test-course'
c.CourseDirectory.course_id = 'test-course'
c.GenerateFeedback.preprocessors = ['nbgrader.preprocessors.GetGrades', 'nbconvert.preprocessors.CSSHTMLHeaderPreprocessor']
```

   Adjust the configuration as needed for your specific course setup.

7. Create the directory to hold all the course assignments:

```
mkdir ~/test-course
```

1.12 C++ Kernel Installation

To use a C++ kernel within JupyterHub, you must install it first. Follow these steps:

1. Install Mamba, a package manager, using Conda:

```
conda install -n base -c conda-forge mamba
```

2. Use Mamba to install the C++ kernel (xeus-cling):

```
mamba install xeus-cling -c conda-forge
```

This completes the configuration of JupyterHub for nbgrader in a multi-class lab environment, along with the installation of the C++ kernel for use within JupyterHub. Make sure to adapt the configurations to match your environment and specific requirements.