Step 1 — Set Up Python

To begin the process, we'll install the dependencies we need for our Python programming environment from the Ubuntu repositories. Ubuntu 18.04 comes preinstalled with Python 3.6. We will use the Python package manager pip to install additional components a bit later.

We first need to update the local `apt` package index and then download and install the packages:

```
sudo apt update
```

Next, install pip and the Python header files, which are used by some of Jupyter's dependencies:

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

We can now move on to setting up a Python virtual environment into which we'll install Jupyter.

Step 2 — Create a Python Virtual Environment for Jupyter

Now that we have Python 3, its header files, and pip ready to go, we can create a Python virtual environment to manage our projects. We will install Jupyter into this virtual environment.

To do this, we first need access to the `virtualenv` command which we can install with pip.

Upgrade pip and install the package by typing:

```
sudo -H pip3 install --upgrade pip
sudo -H pip3 install virtualenv
```

The `-H` flag ensures that the security policy sets the `home` environment variable to the home directory of the target user.

With `virtualenv` installed, we can start forming our environment. Create and move into a directory where we can keep our project files. We'll call this `my_project_dir`, but you should use a name that is meaningful for you and what you're working on.

```
mkdir ~/my_project_dir
cd ~/my_project_dir
```

Within the project directory, we'll create a Python virtual environment. For the purpose of this tutorial, we'll call it `my_project_env` but you should call it something that is relevant to your project.

```
virtualenv my_project_env
```

This will create a directory called `my_project_env` within your `my_project_dir` directory. Inside, it will install a local version of Python and a local version of pip. We can use this to install and configure an isolated Python environment for Jupyter.
Before we install Jupyter, we need to activate the virtual environment. You can do that by typing:

```
source my_project_env/bin/activate
```

Your prompt should change to indicate that you are now operating within a Python virtual environment. It will look something like this: `(my_project_env)user@host:~/my_project_dir$`.
You're now ready to install Jupyter into this virtual environment.

Step 3 — Install Jupyter

With your virtual environment active, install Jupyter with the local instance of pip.

Note: When the virtual environment is activated (when your prompt has `(my_project_env)` preceding it), use `pip` instead of `pip3`, even if you are using Python 3. The virtual environment's of the tool is always named `pip`, regardless of the Python version.

```
pip install jupyter
```

At this point, you've successfully installed all the software needed to run Jupyter. We can now start the Notebook server.

Step 4 — Run Jupyter Notebook

You now have everything you need to run Jupyter Notebook! To run it, execute the following command:

```
(my_project_env)sammy@your_server:~/my_project_dir$ jupyter notebook
```

A log of the activities of the Jupyter Notebook will be printed to the terminal. When you run Jupyter Notebook, it runs on a specific port number. The first Notebook you run will usually use port `8888`. To check the specific port number Jupyter Notebook is running on, refer to the output of the command used to start it:

```
Output
[I 21:23:21.198 NotebookApp] Writing notebook server cookie secret to
/run/user/1001/jupyter/notebook_cookie_secret
[I 21:23:21.361 NotebookApp] Serving notebooks from local directory:
/home/sammy/my_project_dir
[I 21:23:21.361 NotebookApp] The Jupyter Notebook is running at:
[I 21:23:21.361 NotebookApp]
http://localhost:8888/?token=1fefa6ab49a498a3f37c959404f7baf16b9a2eda3eaa6d72
[I 21:23:21.361 NotebookApp] Use Control-C to stop this server and shut down
all kernels (twice to skip confirmation).
[W 21:23:21.361 NotebookApp] No web browser found: could not locate runnable
browser.
[C 21:23:21.361 NotebookApp]

    /paste this URL into your browser when you connect for the first time,
    to login with a token:

http://localhost:8888/?token=1fefa6ab49a498a3f37c959404f7baf16b9a2eda3eaa6d72
```

If you are running Jupyter Notebook on a local computer (not on a server), you can navigate to the displayed URL to connect to Jupyter Notebook. If you are running Jupyter Notebook on a server, you will need to connect to the server using SSH tunneling as outlined in the next section.

At this point, you can keep the SSH connection open and keep Jupyter Notebook running or you can exit the app and re-run it once you set up SSH tunneling. Let's choose to stop the Jupyter Notebook process. We will run it again once we have SSH tunneling set up. To stop the Jupyter Notebook process, press `CTRL+C`, type `Y`, and then `ENTER` to confirm. The following output will be displayed:

```
Output
[C 21:28:28.512 NotebookApp] Shutdown confirmed
```

```
[I 21:28:28.512 NotebookApp] Shutting down 0 kernels
```
We'll now set up an SSH tunnel so that we can access the Notebook.

Step 5 — Connect to the Server Using SSH Tunneling

SSH Tunneling using macOS or Linux

If your local computer is running Linux or macOS, it's possible to establish an SSH tunnel just by running a single command.

`ssh` is the standard command to open an SSH connection, but when used with the `-L` directive, you can specify that a given port on the local host (that is, your local machine) will be forwarded to a given host and port on the remote host (in this case, your server). This means that whatever is running on the specified port on the remote server (`8888`, Jupyter Notebook's default port) will appear on the specified port on your local computer (`8000` in the example command).
To establish your own SSH tunnel, run the following command. Feel free to change port `8000` to one of your choosing if, for example, `8000` is in use by another process. It is recommended that you use a port greater than or equal to `8000`, as those port numbers are unlikely to be used by another process. Be sure to include your own server's IP address and the name of your server's non-root user:

```
ssh -L 8000:localhost:8888 sammy@your_server_ip
```

If there are no errors from this command, it will log you into your remote server. From there, activate the virtual environment:

```
source ~/environments/my_env/bin/activate
```

Then run the Jupyter Notebook application:

```
jupyter notebook
```

To connect to Jupyter Notebook, use your favorite web browser to navigate to the local port on the local host: `http://localhost:8000`. Now that you're connected to Jupyter Notebook..