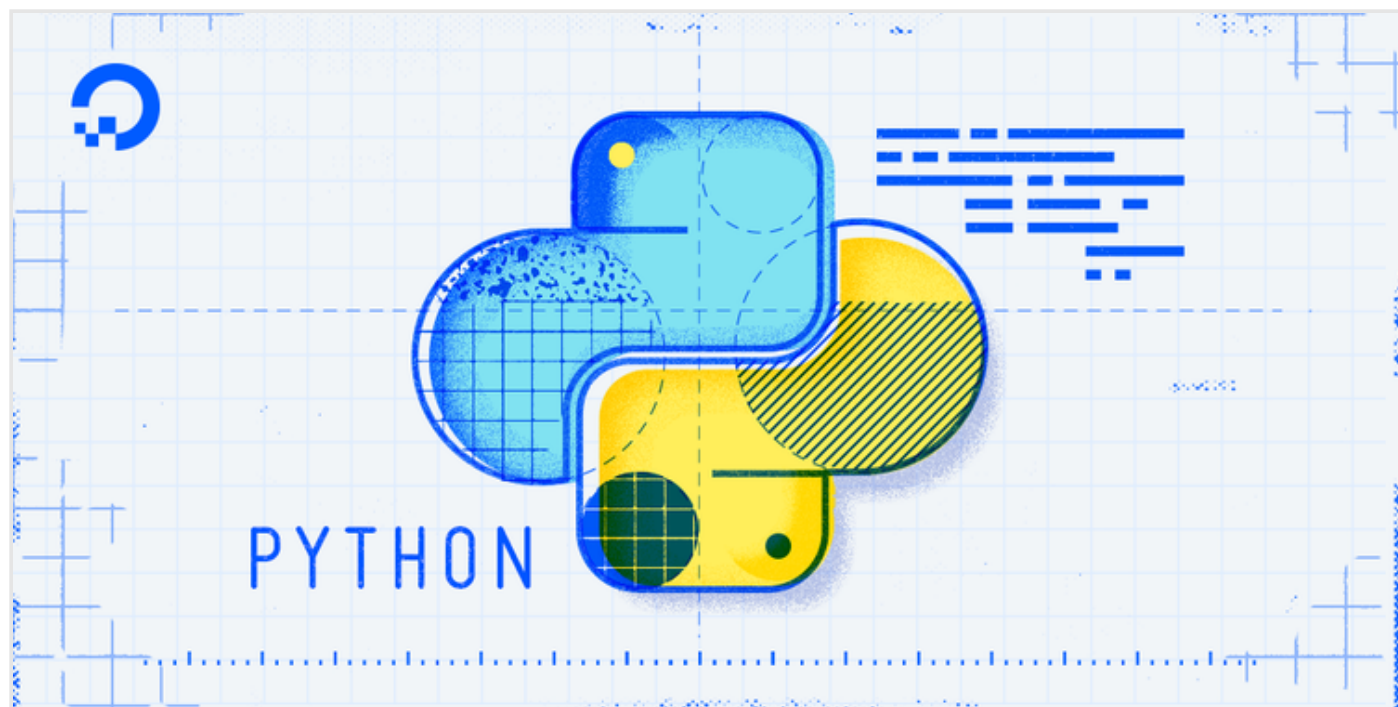


NEW Now available: Memory-Optimized Droplets >

 CommunityLanguage: EN ▾  Contents ▾

How To Set Up Jupyter Notebook with Python 3 on Ubuntu 18.04

Posted November 28, 2018  97.1k PYTHON DATA ANALYSIS DEVELOPMENT UBUNTU 18.04By [Lisa Tagliaferri](#)
[Become an author](#)Not using **Ubuntu 18.04**? Choose a different version:

Introduction

An open-source web application, [Jupyter Notebook](#) lets you create and share interactive code, visualizations, and more. This tool can be used with several programming languages,

including Python, Julia, R, Haskell, and Ruby. It is often used for working with data, statistical modeling, and machine learning.

This tutorial will walk you through setting up Jupyter Notebook to run from an Ubuntu 18.04 server, as well as teach you how to connect to and use the notebook. Jupyter Notebooks (or simply Notebooks) are documents produced by the Jupyter Notebook app which contain both computer code and rich text elements (paragraph, equations, figures, links, etc.) which aid in presenting and sharing reproducible research.

By the end of this guide, you will be able to run Python 3 code using Jupyter Notebook running on a remote server.

Prerequisites

In order to complete this guide, you should have a fresh Ubuntu 18.04 server instance with a basic firewall and a non-root user with sudo privileges configured. You can learn how to set this up by running through our [initial server setup tutorial](#).

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 copy 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 21:23:21.158 NotebookApp] Writing notebook server cookie secret to /home/sammy/.jupyter_notebook_cookie.txt
[I 21:23:21.361 NotebookApp] Serving notebooks from local directory: /home/sammy/my_pi
[I 21:23:21.361 NotebookApp] The Jupyter Notebook is running at:
[I 21:23:21.361 NotebookApp] http://localhost:8888/?token=1fefaf6ab49a498a3f37c959404f7baf16b9a2eda3eaa6d72
[I 21:23:21.361 NotebookApp] Use Control-C to stop this server and shut down all kernels
[W 21:23:21.361 NotebookApp] No web browser found: could not locate runnable browser.
[C 21:23:21.361 NotebookApp]
```

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

```
http://localhost:8888/?token=1fefaf6ab49a498a3f37c959404f7baf16b9a2eda3eaa6d72
```

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

In this section we will learn how to connect to the Jupyter Notebook web interface using SSH tunneling. Since Jupyter Notebook will run on a specific port on the server (such as `:8888`, `:8889` etc.), SSH tunneling enables you to connect to the server's port securely.

The next two subsections describe how to create an SSH tunnel from 1) a Mac or Linux, and 2) Windows. Please refer to the subsection for your local computer.

SSH Tunneling with a Mac or Linux

If you are using a Mac or Linux, the steps for creating an SSH tunnel are similar to using SSH to log in to your remote server, except that there are additional parameters in the `ssh` command. This subsection will outline the additional parameters needed in the `ssh` command to tunnel successfully.

SSH tunneling can be done by running the following SSH command in a new local terminal window:

```
$ ssh -L 8888:localhost:8888 your_server_username@your_server_ip
```

The `ssh` command opens an SSH connection, but `-L` specifies that the given port on the local (client) host is to be forwarded to the given host and port on the remote side (server). This means that whatever is running on the second port number (e.g. `8888`) on the server will appear on the first port number (e.g. `8888`) on your local computer.

Optionally change port `8888` to one of your choosing to avoid using a port already in use by another process.

`server_username` is your username (e.g. `sammy`) on the server which you created and `your_server_ip` is the IP address of your server.

For example, for the username `sammy` and the server address `203.0.113.0`, the command would be:

```
$ ssh -L 8888:localhost:8888 sammy@203.0.113.0
```

If no error shows up after running the `ssh -L` command, you can move into your programming environment and run Jupyter Notebook:

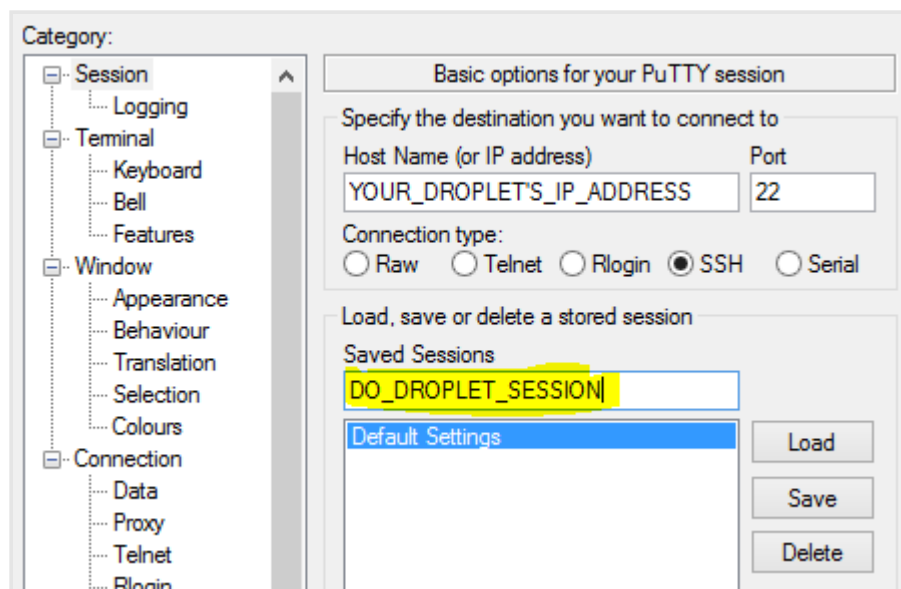
```
(my_project_env)sammy@your_server:~/my_project_dir$ jupyter notebook
```

You'll receive output with a URL. From a web browser on your local machine, open the Jupyter Notebook web interface with the URL that starts with `http://localhost:8888`. Ensure that the token number is included, or enter the token number string when prompted at `http://localhost:8888`.

SSH Tunneling with Windows and Putty

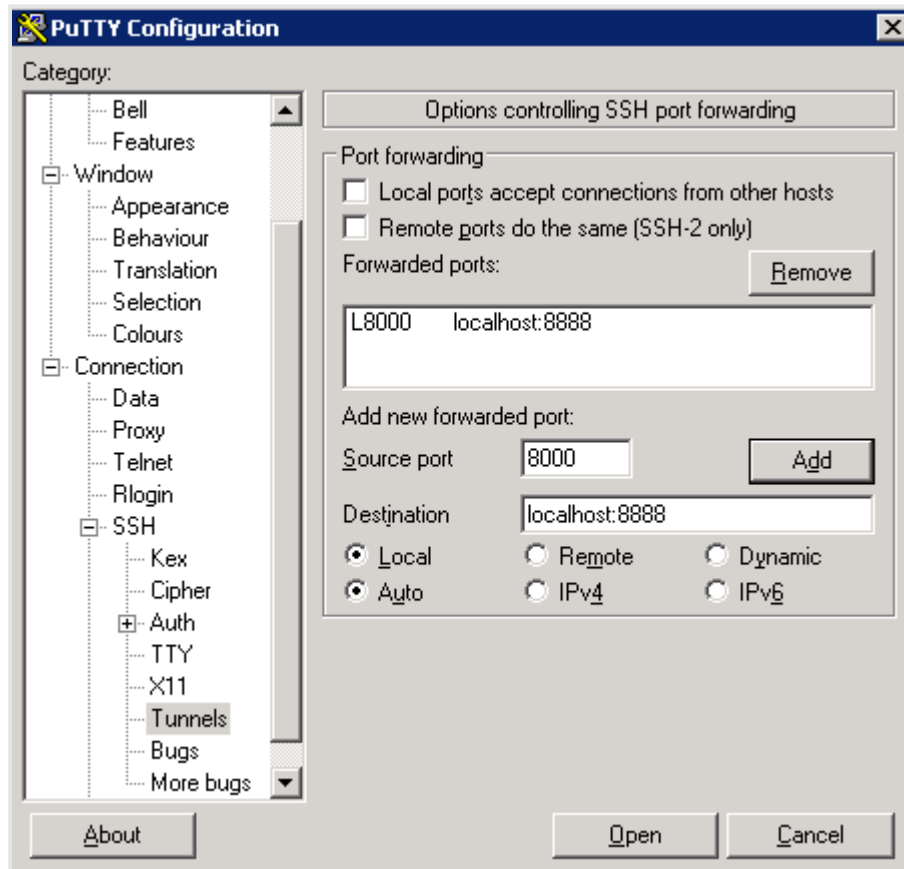
If you are using Windows, you can create an SSH tunnel using Putty.

First, enter the server URL or IP address as the hostname as shown:



Next, click **SSH** on the bottom of the left pane to expand the menu, and then click **Tunnels**. Enter the local port number you want to use to access Jupyter on your local machine. Choose **8000** or greater to avoid ports used by other services, and set the destination as `localhost:8888` where `:8888` is the number of the port that Jupyter Notebook is running on.

Now click the **Add** button, and the ports should appear in the **Forwarded ports** list:



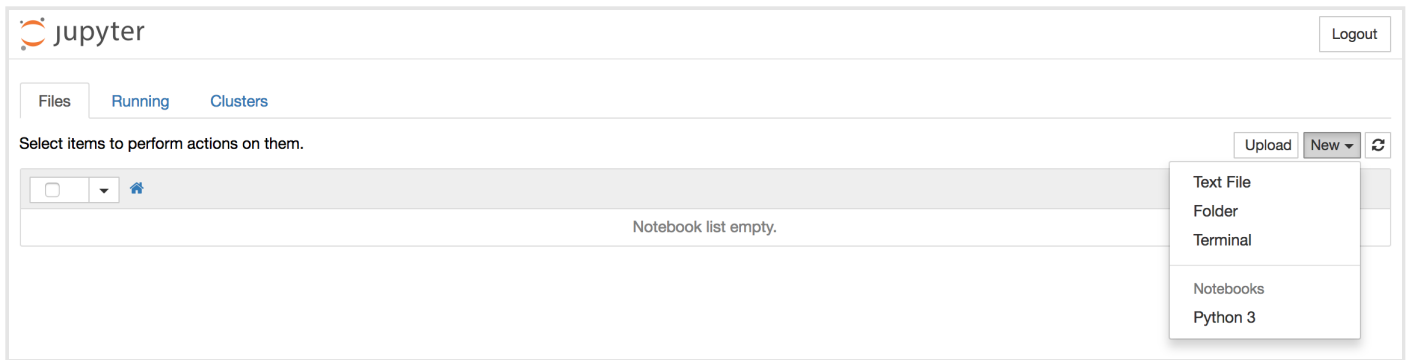
Finally, click the **Open** button to connect to the server via SSH and tunnel the desired ports. Navigate to `http://localhost:8000` (or whatever port you chose) in a web browser to connect to Jupyter Notebook running on the server. Ensure that the token number is included, or enter the token number string when prompted at `http://localhost:8000`.

Step 6 – Using Jupyter Notebook

This section goes over the basics of using Jupyter Notebook. If you don't currently have Jupyter Notebook running, start it with the `jupyter notebook` command.

You should now be connected to it using a web browser. Jupyter Notebook is a very powerful tool with many features. This section will outline a few of the basic features to get you started using the Notebook. Jupyter Notebook will show all of the files and folders in the directory it is run from, so when you're working on a project make sure to start it from the project directory.

To create a new Notebook file, select **New > Python 3** from the top right pull-down menu:



This will open a Notebook. We can now run Python code in the cell or change the cell to markdown. For example, change the first cell to accept Markdown by clicking **Cell > Cell Type > Markdown** from the top navigation bar. We can now write notes using Markdown and even include equations written in LaTeX by putting them between the `$$` symbols. For example, type the following into the cell after changing it to markdown:

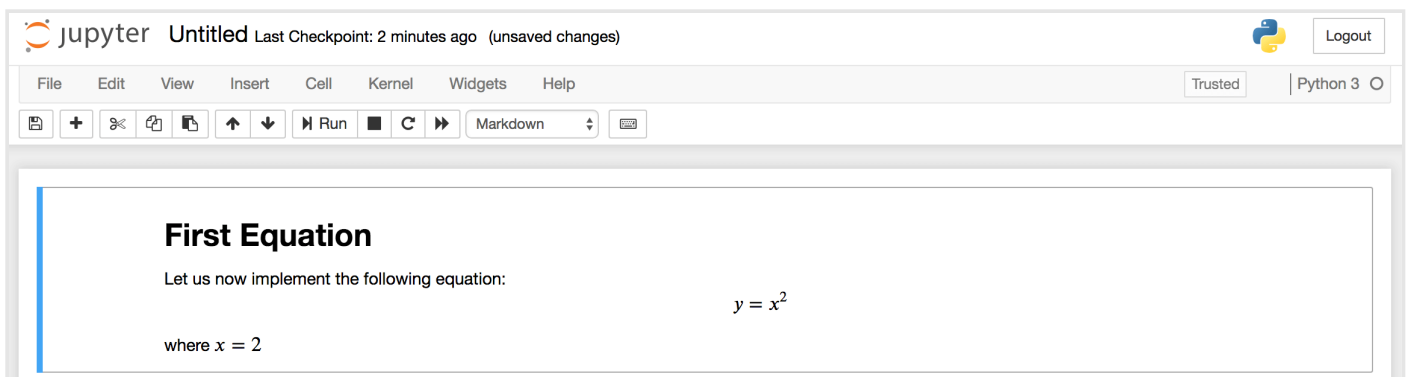
```
# First Equation
```

Let us now implement the following equation:

```
$$ y = x^2 $$
```

where $x = 2$

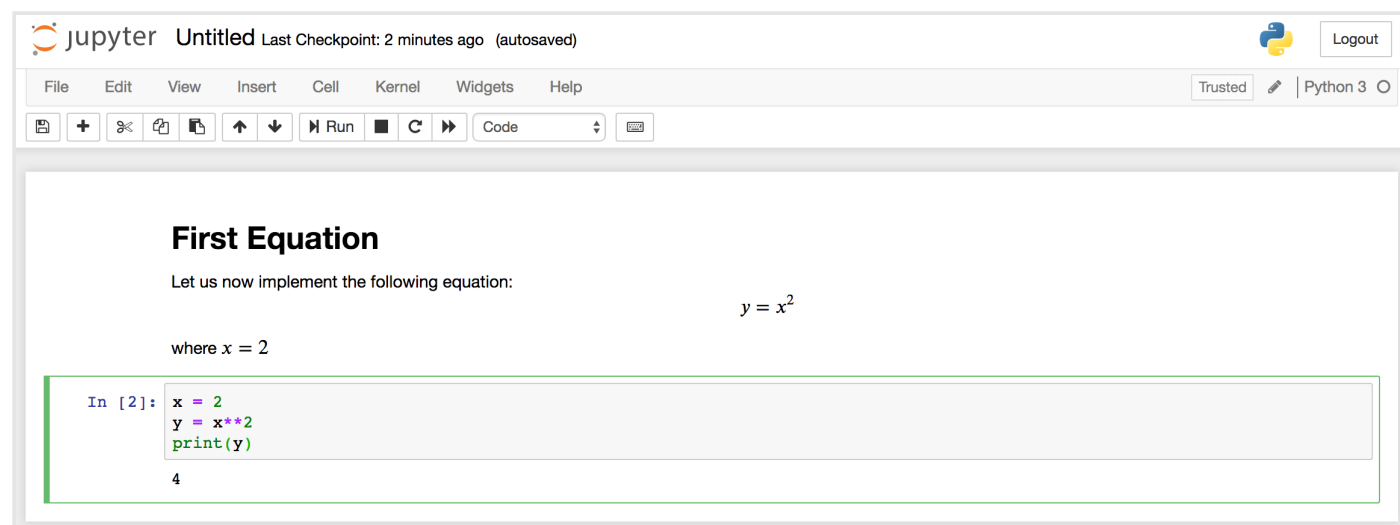
To turn the markdown into rich text, press **CTRL+ENTER**, and the following should be the results:



You can use the markdown cells to make notes and document your code. Let's implement that equation and print the result. Click on the top cell, then press **ALT+ENTER** to add a cell below it. Enter the following code in the new cell.

```
x = 2
y = x**2
print(y)
```

To run the code, press **CTRL+ENTER**. You'll receive the following results:



The screenshot shows a Jupyter Notebook window titled "Untitled" with a last checkpoint of "2 minutes ago (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The main content area is titled "First Equation" and contains the text "Let us now implement the following equation:" followed by the equation $y = x^2$ and "where $x = 2$ ". Below this is a code cell with the following Python code:

```
In [2]: x = 2
        y = x**2
        print(y)
```

The output of the code cell is the number 4.

You now have the ability to [import modules](#) and use the Notebook as you would with any other Python development environment!

Conclusion

Congratulations! You should now be able to write reproducible Python code and notes in Markdown using Jupyter Notebook. To get a quick tour of Jupyter Notebook from within the interface, select **Help > User Interface Tour** from the top navigation menu to learn more.

From here, you can begin a data analysis and visualization project by reading [Data Analysis and Visualization with pandas and Jupyter Notebook in Python 3](#).

If you're interested in digging in more, you can read our series on [Time Series Visualization and Forecasting](#).

By [Lisa Tagliaferri](#)

Was this helpful?

Yes

No

[Report an issue](#)

Related

CONCEPTUAL ARTICLE

Understanding This, Bind, Call, and Apply in JavaScript

The `this` keyword is a very important concept in JavaScript, and also a...

TUTORIAL

Handling Panics in Go

Panics are unforeseeable errors that will spontaneously terminate and exit a running Go program. Common...

TUTORIAL

How To Add Stimulus to a Ruby on Rails Application

In this tutorial, you will install and use Stimulus to build on an existing Rails...

CHEATSHEET

How To Run Transactions in Redis

Redis allows you to plan a sequence of commands and run them one after another, a procedure...

Still looking for an answer?



Ask a question



Search for more help

1 Comment

Leave a comment...

Log In to Comment

 [azbeltk](#) *May 9, 2019*

0 Hi! Thanks for this tutorial, I could finally install jupyter but when I try to import a module (like numpy), jupyter cant find it. Is this cause the modules are not on my virtual enviroment? how can I load then then?

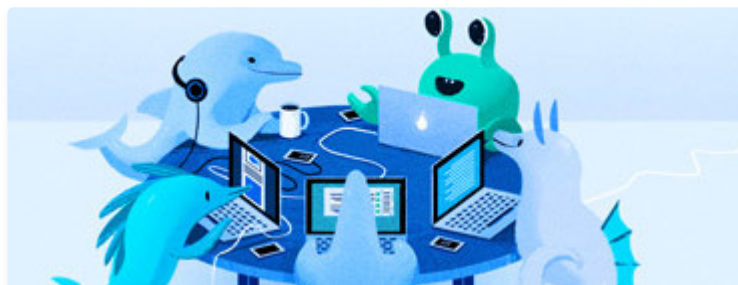


This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



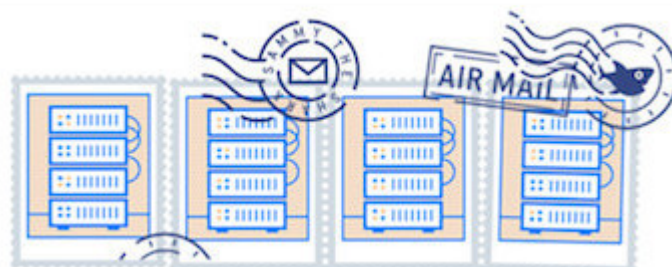
BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.



CONNECT WITH OTHER DEVELOPERS

Find a DigitalOcean Meetup near you.



GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

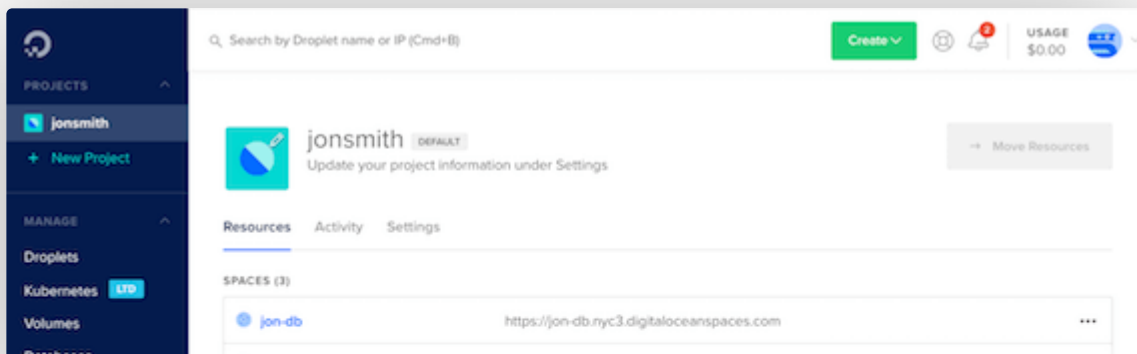
Featured on Community Intro to Kubernetes Learn Python 3 Machine Learning in Python Getting started with Go Migrate Node.js to Kubernetes

DigitalOcean Products Droplets Managed Databases Managed Kubernetes Spaces Object Storage Marketplace

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow – whether you’re running one virtual machine or ten thousand.

Learn More



© 2019 DigitalOcean, LLC. All rights reserved.

Company

- About
- Leadership
- Blog
- Careers
- Partners

Products

- Products Overview
- Pricing
- Droplets
- Kubernetes
- Managed Databases

[Referral Program](#)

[Press](#)

[Legal & Security](#)

[Spaces](#)

[Marketplace](#)

[Load Balancers](#)

[Block Storage](#)

[Tools & Integrations](#)

[API](#)

[Documentation](#)

[Release Notes](#)

[Community](#)

[Contact](#)

[Tutorials](#)

[Support](#)

[Q&A](#)

[Sales](#)

[Tools and Integrations](#)

[Report Abuse](#)

[Tags](#)

[System Status](#)

[Product Ideas](#)

[Meetups](#)

[Write for DOnations](#)

[Droplets for Demos](#)

[Hatch Startup Program](#)

[Shop Swag](#)

[Research Program](#)

[Currents Research](#)

[Open Source](#)