# jupyterhub

jupyterhub 和 jupyter notebook一样是python的一个包，可以通过pip安装，也可以通过 conda安装，在服务器端安装就可以供大家使用。

支持多用户的 Jupyter Notebook 服务器，用于创建、管理、代理多个 Jupyter Notebook 实例。具有扩展性和可定制性。

## 组件

JupyterHub 包含三个组件 [1] ：

1.一个多用户的 Hub

2.一个可配置的 http代理

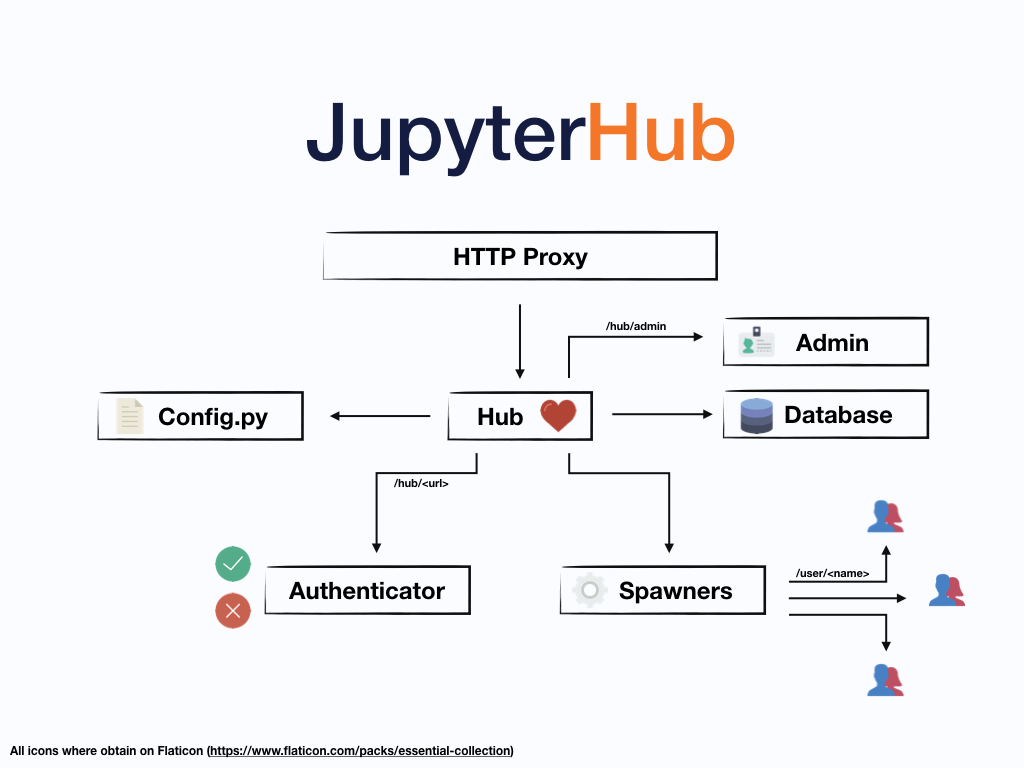
3.多个单用户JupyterNotebook服务器

## 工作原理

JupyterHub 的基本操作流程如下 [1] ：

Hub创建一个代理；

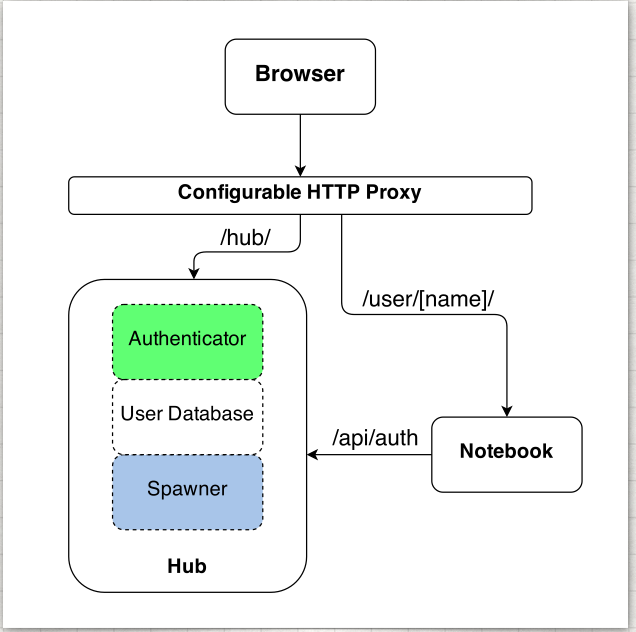
默认情况下，代理将所有请求转发到 Hub；



## The Subsystems: Hub, Proxy, Single-User Notebook Server

JupyterHub is a set of processes that together provide a single user Jupyter Notebook server for each person in a group. Three major subsystems are started by the jupyterhub command line program:

* **Hub** (Python/Tornado): manages user accounts, authentication, and coordinates Single User Notebook Servers using a Spawner.
* **Proxy**: the public facing part of JupyterHub that uses a dynamic proxy to route HTTP requests to the Hub and Single User Notebook Servers. [configurable http proxy](https://github.com/jupyterhub/configurable-http-proxy) (node-http-proxy) is the default proxy.
* **Single-User Notebook Server** (Python/Tornado): a dedicated, single-user, Jupyter Notebook server is started for each user on the system when the user logs in. The object that starts the single-user notebook servers is called a **Spawner**.



## 安装

环境需求：

1.>python3.5

2.nodejs/npm

sudo apt-get install npm nodejs-legacy

安装

方法一：

pip, npm:

python3 -m pip install jupyterhub

npm install -g configurable-http-proxy

python3 -m pip install notebook # needed if running the notebook servers locally

安装完成后进行测试

jupyterhub -h

configurable-http-proxy -h

## 启动

jupyterhub

 如果要进行多用户登陆则要运行：

sudo jupyterhub

在浏览器中访问 <https://localhost:8000>  即可出现登陆页面。

## 配置jupyterhub

如果上述过程都没有问题，我们就可以在命令行中敲入jupyterhub -h看看是否有帮助提示，有帮助提示代表安装成功，在虚拟环境中安装的需要先进入到虚拟环境中source activate \*\*

jupyterhub --generate-config

这个命令会在你的当前目录下生成一个jupyterhub\_config.py文件，接下来我们需要在这个文件中配置我们的网络和用户管理。

jupyterhub -f /path/to/jupyterhub\_config.py

上述命令可以根据你的配置文件启动jupyterhub。

c.JupyterHub.ip = '192.168.24.\*\*\*'

c.JupyterHub.port = 445

c.PAMAuthenticator.encoding = 'utf-8'

c.LocalAuthenticator.create\_system\_users = False

c.Authenticator.whitelist = {'dada',

'haha',

‘name’

}

c.Authenticator.admin\_users = {'famu'}

c.Spawner.env\_keep.append('LD\_LIBRARY\_PATH')

c.JupyterHub.statsd\_prefix = 'jupyterhub'

JupyterHub.ip是你本机局域网的ip，记得是局域网

JupyterHub.port是我们指定的端口，随便指定一个不和其他服务冲突的端口就行

Authenticator.whitelist 比较重要，这里面需要将linux的用户名添加进入，这样该用户就可以通过浏览器利用linux的用户名和密码登录自己的账户，jupyterhub采用和linux系统相同的认证方式，所以我们不需要另外建立用户，只需要登录linux的用户和密码即可。

c.Spawner.env\_keep.append('LD\_LIBRARY\_PATH')这行是我们踩的坑，因为用了GPU版的tensorflow，这个目的是将LD\_LIBRARY\_PATH的路径放到jupyterhub中，这样才能正确使用GPU版的tensorflow。

## 多用户问题

4、多用户：在Linux下使用useradd命令添加新用户并设置密码，以新用户名和密码登录jupyterhub即可。

5、网上常用方法是设置 c.Authenticator.whitelist 添加白名单，但是jupyterhub对centos添加新用户的命令支持不好，出错，并且它的底层依然是添加linux新用户。另一种方法是在配置文件中自定义 c.Authenticator.add\_user\_cmd 配置，例如：c.LocalAuthenticator.add\_user\_cmd = ['adduser', '-m', '-G', 'jupyterhub-users'] ，此时添加白名单方法生效。但是此时还需要先为新用户设置密码，才能登录，所以还是建议在linux终端直接添加新用户。