**Autumn(Qiuying) Li (RIS-ATL)**

Qiuying.li@lecisnexisrisk.com

Abstract

With the power of JupyterHub, Git, and Papermill, data scientists can automate the end to end statistical modeling process. It will eliminate the cost of re-coding, and enhance the efficiency of model training, model production, and model updating. .

Statistical Modeling   
Automation Guide

A manual of End to End Modeling



Documentation Guide

# [**JupyterHub**](#_JupyterHub)

### [What is JupyterHub](#_What_is_JupyterHub)

### [JupyterHub advantages](#_JupyterHub_advantages)

### [JupyterHub and Spark](#_JupyterHub_and_Spark)

### [JupyterHub and Python](#_JupyterHub_and_Python)

### [JupyterHub and SQL Server](#_JupyterHub_and_SQL)

### [Walkthrough Example: how to implement an end to end machine learning project with fido dataset](#_Walkthrough_Example:_how)

## **GitHub**

### [GitHub workflow chart](#_GitHub_workflow_chart)

### [Common GitHub terminologies](#_Common_GitHub_terminologies)

### [Basic Git command lines](#_Basic_Git_command)

# **Setup GitHub on JupyterHub**

### [Why do we need GitHub on Jupyter Hub?](#_Why_do_we)

### [GitHub Authentication steps](#_GitHub_Authentication_steps)

### [Fork the project](#_Fork_the_project)

### [Setup the repos](#_Setup_the_repos)

### [Setup branches](#_Setup_Branches)

### [Walkthrough Example: how to commit/push local changes to GitHub in 2 ways](#_Walkthrough_Example:_how_1)

**SSH**

### [What is SSH, and how it works](#_What_is_SSH,)

### [Generated SSH keys](#_Generated_SSH_keys)

### [How to use SSH keys for authentication](#_How_to_use)

**From local Git Bash to GitHub**

### [Add ssh-key agent](#_Add_ssh-key_agent(back)

### [Clone the repo](#_Clone_the_repo)

### [Walkthrough example: how to run the cloned jobs on your local machine](#_Walkthrough_example:_how_2)

**Papermill**

### [What is Papermill](#_What_is_Papermill(back)

### [How to install Papermill](#_How_to_install)

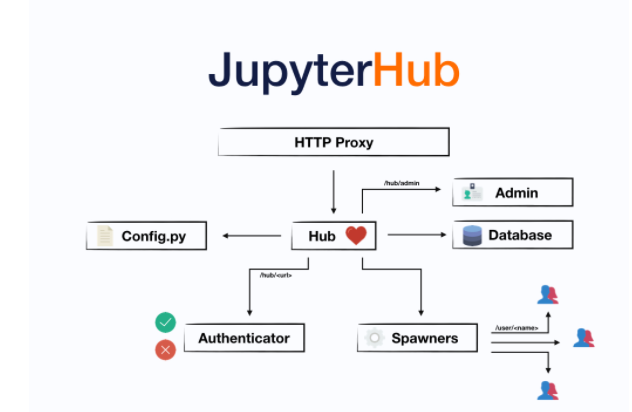
### [How to parametrize notebook by Papermill](#_How_to_parametrize)

### [Walkthrough example: how to git clone the notebook from Jupyter Hub and use the command line to run the notebook locally via ssh key authentication. Finally, use Papermill to parameterize the notebook for automation.](#_Walkthrough_example:_how_3)

# **JupyterHub** [(back to top)](#_top)

### What is JupyterHub

* JupyterHub runs in the cloud or on your own hardware and makes it possible to serve the best pre-configured data science environment to any Jupyter user in the world.



* Four subsystems make up JupyterHub:

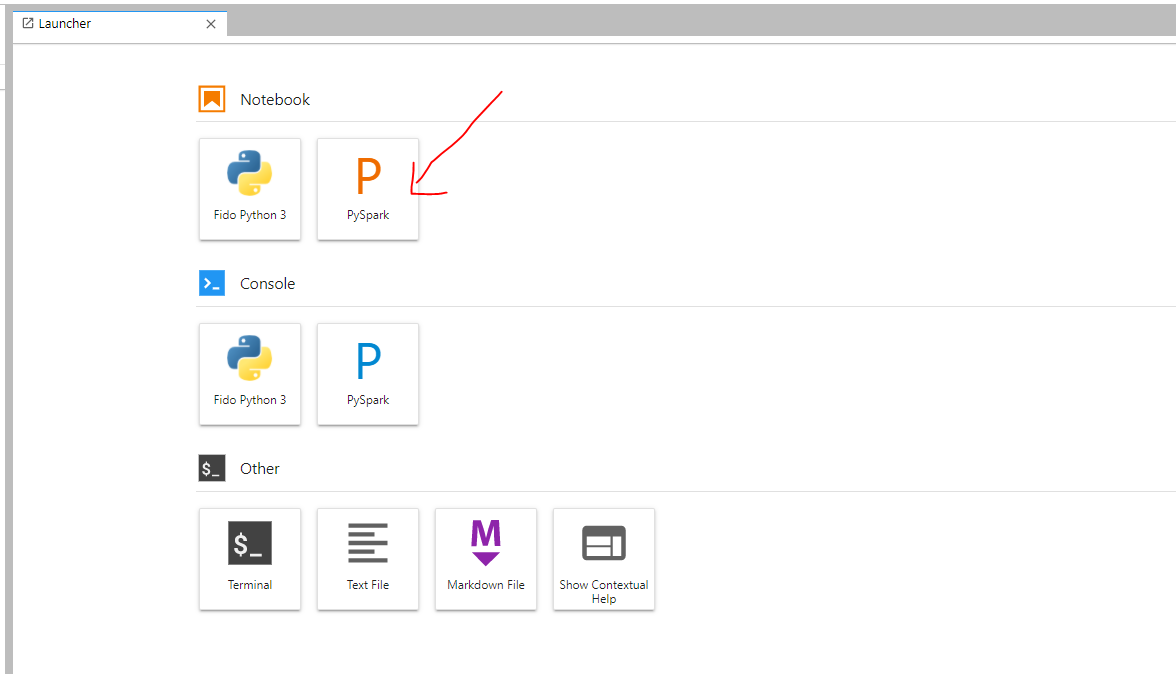
1. a Hub (tornado process) that is the heart of JupyterHub
2. a configurable http proxy (node-http-proxy) that receives the requests from the client’s browser
3. multiple single-user Jupyter notebook servers (Python/IPython/tornado) that are monitored by Spawners
4. an authentication class that manages how users can access the system

### JupyterHub advantages [(back to top)](#_top)

1. **Customizable** - JupyterHub can be used to serve a variety of environments. It supports dozens of kernels with the Jupyter server and can be used to serve a variety of user interfaces, including the Jupyter Notebook, Jupyter Lab, RStudio, nteract, and more.
2. **Flexible** - JupyterHub can be configured with authentication to provide access to a subset of users. Authentication is pluggable, supporting a number of authentication protocols (such as OAuth and GitHub).
3. **Scalable** - JupyterHub is container-friendly, and can be deployed with modern-day container technology. It also runs on Kubernetes, and can run with up to tens of thousands of users.
4. **Portable** - JupyterHub is entirely open-source and designed to be run on a variety of infrastructure. This includes commercial cloud providers, virtual machines, or even your laptop hardware.

### JupyterHub and Spark [(back to top)](#_top)

* + You can create a Pyspark notebook for the spark Job
  + Follow the notes below, and you’ll find the ways to set up the spark environment



* + Usage- Setup environment for Spark

Given configuration like above, users may not need to enter any parameters when creating a SparkContext - the default values may already be sufficiently set:

**import** **pyspark**

*# Create a spark context from the defaults set in the configuration*

sc = pyspark.SparkContext()

Of course, overrides can always be provided at runtime if needed:

**import** **pyspark**

conf = pyspark.SparkConf()

*# Override a few default parameters*

conf.set('spark.executor.memory', '512m')

conf.set('spark.executor.instances', 1)

*# Create a spark context with the overrides*

sc = pyspark.SparkContext(conf=conf)

If all nodes are configured to use the same Python path/archive, then all dependencies should be available on all workers:

**def** some\_function(x):

*# Libraries are imported and available from the same environment as the*

*# notebook*

**import** **sklearn**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

*# Use the libraries to do work*

**return** ...

rdd = sc.parallelize(range(1000)).map(some\_function).take(10)

When you’re done, the Spark clusters can be shutdown manually, or will be automatically shutdown when the notebook exits.

### JupyterHub and Python [(back to top)](#_top)

* + How to Install the python packages, use NumPy as an example

sudo python3 -m pip install numpy

* + Packages currently available to use ( no need to install in your notebook)

conda install -y pandas

conda install -y numpy

conda install -y matplotlib

conda install -y datetime

conda install -y pyodbc

conda install -y sys

conda install -y IPython

conda install -y random

conda install -y sklearn

conda install -y operator

conda install -y statsmodels

conda install -y xgboost

conda install -y scipy

conda install -y Seaborn

### JupyterHub and SQL Server [(back to top)](#_top)

* + JupyterHub uses a database to store information about users, services, and other data needed for operating the Hub.
  + Connect to the SQL server, use Fido-Dev as an example

**import** pyodbc ## this is the library for SQL connection

server = 'ALAWDREDSQL201.risk.regn.net, 50255'

database = 'red' # enter database name

username = your username

password = your password

cnxn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)

cursor = cnxn.cursor()

* Read Data from SQL server

query = '''

## Insert your SQL queries here

Select top 10 \*

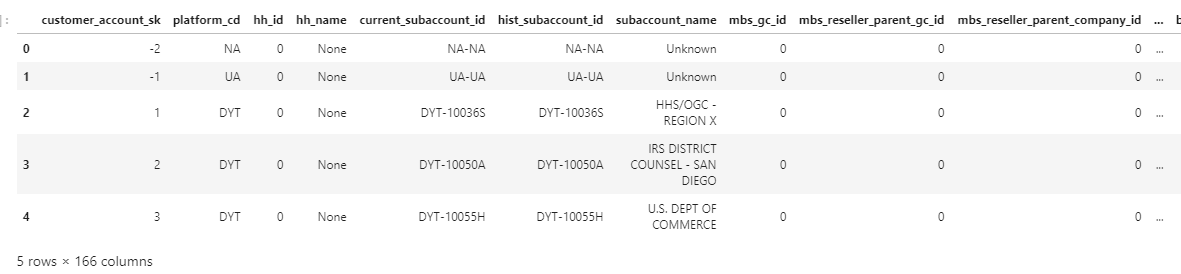
From dim\_customer\_account

''';

## Execute your quries on the SQL server

**data = pd.read\_sql(query, cnxn)**

**data.head()**



* Write your data back to the SQL server

## Connect to SQL Server

server = 'ALAWDREDSQL201.risk.regn.net, 50255'

database = 'red' # enter database name

username = your username

password = your password

cnxn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password)

cursor = cnxn.cursor()

## Excute your tbales on SQL server

cursor.execute(table\_sql)

cnxn.commit()

## Define your data schema

table\_sql = ("CREATE TABLE my\_table ( policy\_state\_sk real, \

coverage\_type\_sk real, \

shopgroup\_name text, \

coverage\_desc text)



* Please close the connection after you read/write the data from the SQL server

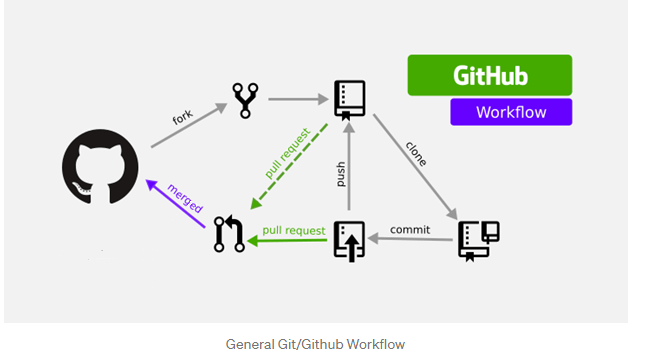
**cnxn.close()**

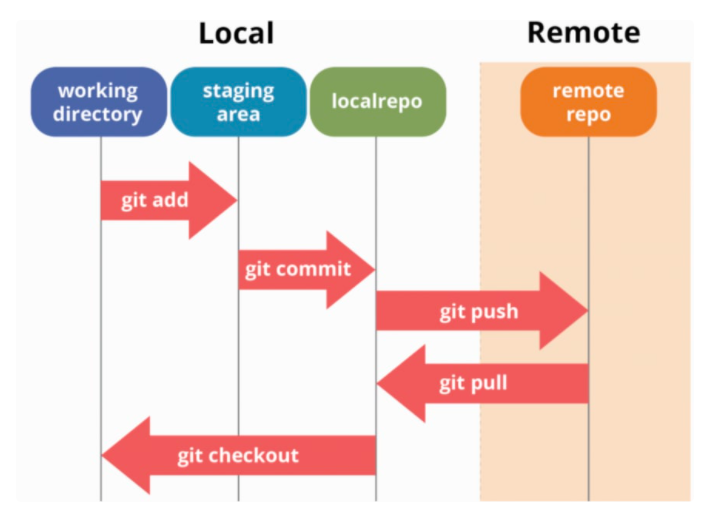
### Walkthrough Example: how to implement an end to end machine learning project with fido dataset

# **GitHub** [(back to top)](#_top)

### GitHub workflow chart

* + **Git** is a version control system that will help you keep track of the changes and history of your project throughout its lifestyle.  Two main components of version control
    - Branching - allows you to duplicate the source code for yourself.
    - Merging - the process of joining your branch with the original source code after complete testing.
  + **GitHub** is the service that will host your Git repositories.





### Common GitHub terminologies [(back to top)](#_top)

* + **A fork is a copy of a repository.** Forking a repository allows you to freely experiment with changes without affecting the original project.
  + **git clone**: To clone the repository to your local machine.
  + **git pull origin master**: pull down from the remote repository (reminder, this lives on GitHub) to ensure you have up-to-date information.
  + **git checkout -b new\_branch\_name:** Create your branch. For example, you probably want to create a new branch for the feature you are working on.
  + **git add**: After your code has been tested and is complete, add your changes to the staging area.
  + **git status:** Run git status to confirm that your files have been added to the staging area.
  + **git commit -m "comments of your changes":** Commit your changes back to the repository. Add a brief, but detailed comment about the changes that were made
  + **git push:** push your changes to the repository!
  + **git pull:** create a **pull request** through GitHub. Submitting a pull request will notify the owner of the repository that you are ready to merge your code with the master branch.

### Basic Git command lines [(back to top)](#_top)

git remote add origin https://github.com/LiQiuy01/fido-models.git

git remote add upstream <https://github.com/LexisNexis-RBA/fido-models.git>

#1. check remote path

git remote -v

#2. check status

git status

#3. add the file with changes, for example, my file with change is test.ipynp

git add test.ipynb

#4. Commit changes with comment

git commit -m "Autumn's commit changes"

# 5. start push

git push

#6. Then type in your GitHub username and password

##7. Git pull

git pull

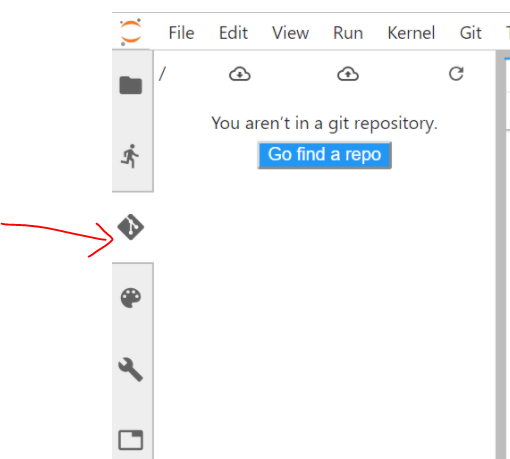
# **Setup GitHub on JupyterHub** [(back to top)](#_top)

### Why do we need GitHub on Jupyter Hub?

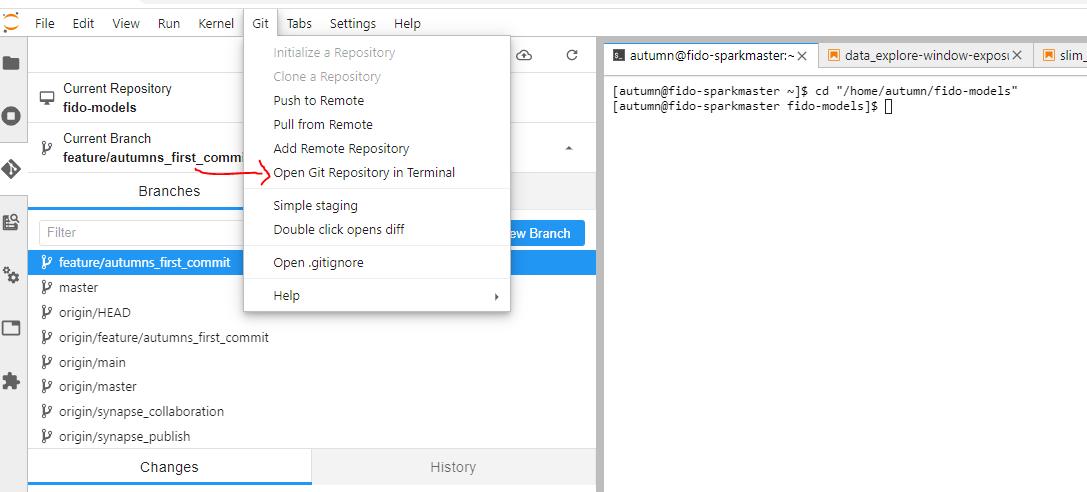
* + To use the version control for Jupyter Notebooks, you can compare files, identify differences, and merge the changes if needed prior to committing any code.

### GitHub Authentication steps [(back to top)](#_top)

* + Step1: Create your GitHub personal token ( if you have two factor authentication)
    - You should create a personal access token to use in place of a password with the command line or with the API. Personal access tokens (PATs) are an alternative to using passwords for authentication to GitHub when using the GitHub API or the command line.
    - Follow the [link](https://reedelsevier-my.sharepoint.com/personal/liqiuy01_risk_regn_net/Documents/Desktop/%09https:/docs.github.com/en/github/authenticating-to-github/creating-a-personal-access-token) for the specific instructions
  + Step2: Configuring GitHub on Jupyter Hub



* + Step3: Git Setup on Jupyter Hub via terminal. Useful command line to set up the username and email address. Please use your GitHub username and email address



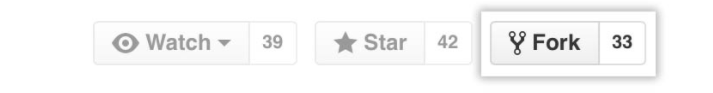
$ git config **--global** user.name " LiQiuy01"

$ git config **--global** user.email " LiQiuy01@github.com"

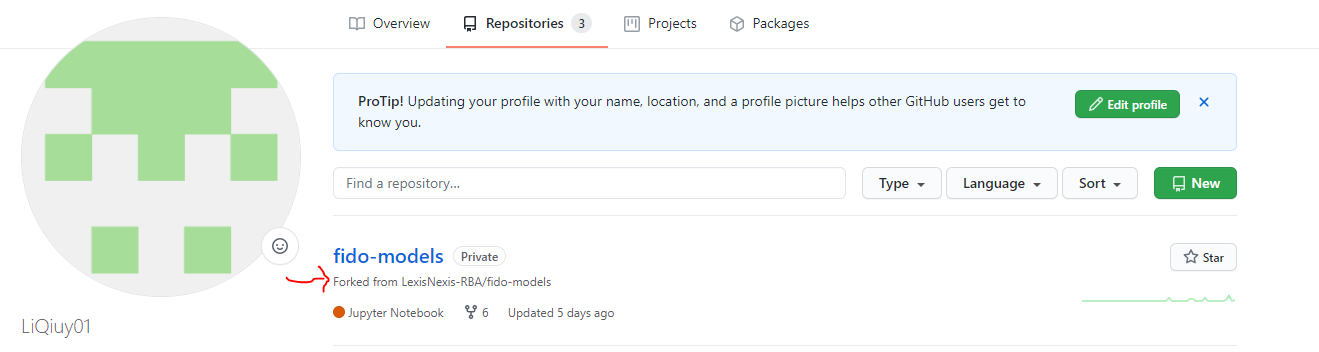
$ git config **–list ## you can verify your settings at any time**

### Fork the project [(back to top)](#_top)

* + If you are working on a group project, you can fork the repo. A fork is a copy of a repository. Forking a repository allows you to experiment with changes without affecting the original project freely. Rather than logging an issue for a bug you've found, you can:
    - Fork the repository.
    - Make the fix.
    - Submit a pull request to the project owner.
  + On GitHub, navigate to the [LexisNexis-RBA /fido-models](https://github.com/LexisNexis-RBA/fido-models) repository.
  + In the top-right corner of the page, click **Fork**.

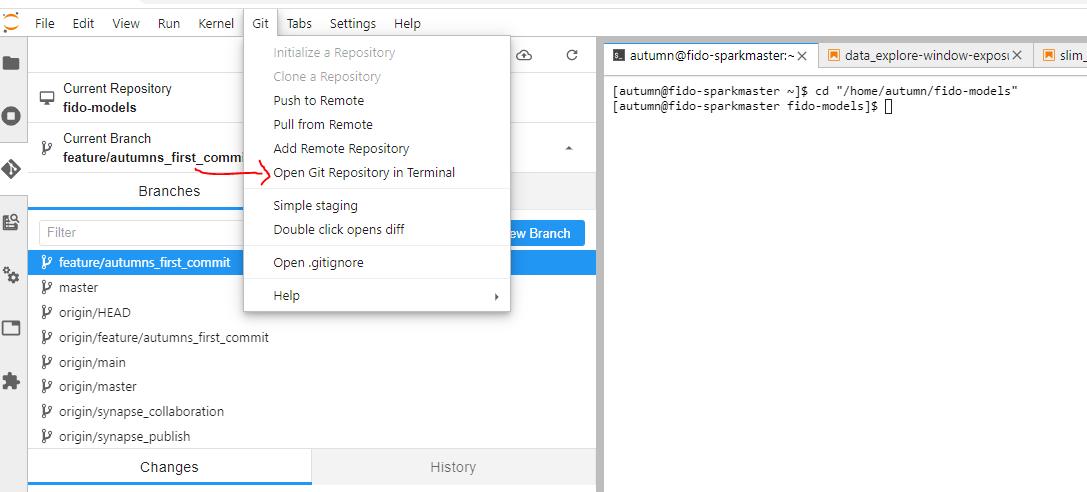


* + Under your personal account, you’ll find a fido-models becomes your repo.



### Setup the repos [(back to top)](#_top)

* + Since we have setup the GitHub authentication on JupyterHub, and we also fork the project into our personal repo. It’s time to configure Git to pull changes from the original, or upstream, repository into the local clone of your fork.
  + Now we need to go back to **terminal** on JupyterHub and setup the origin and upstream:

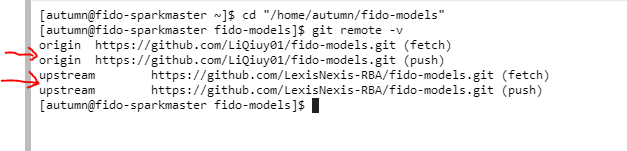


$ git remote add origin <https://github.com/LiQiuy01/fido-models.git> (SEE NOTE)

$ git remote add upstream https://github.com/LexisNexis-RBA/fido-models.git$ git config (SEE NOTE)

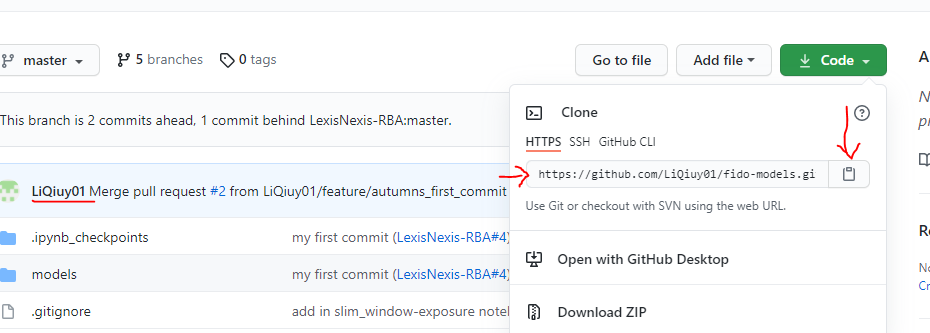
Ideally, you can check your setup, and you should see similar results

$ git remote -v

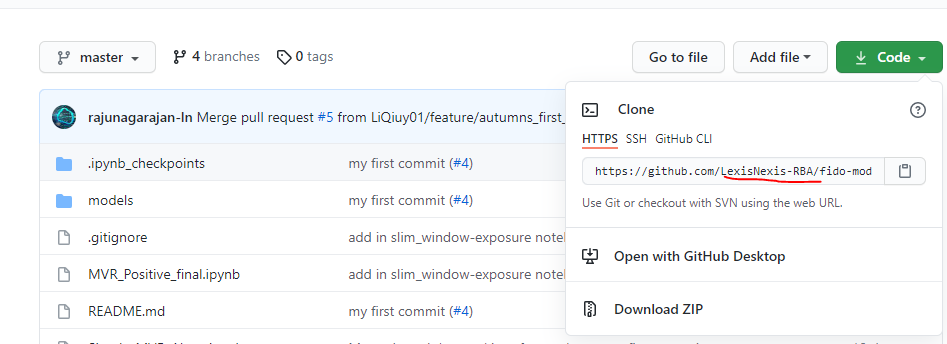


**Note: how to find the links for the Origin and Upstream?**

You can find the Origin repo link for origin under your **persnal** repo, such as **LiQiuy01/fido-models**, click code, and copy HTTPS.



You can find the **upstream repo link** under **the LexisNexis-BRA/fido-models** repo, click code, and copy HTTPS.



### Setup Branches [(back to top)](#_top)

* + After we set up the repo, it’s time to set up the branches. Please note that Master is a permanent branch that always reflects a production-ready state. So yes, it is for ready-product which can be downloaded on the market by a user. For the ongoing features, you can always set up a branch for it.

$ git checkout -b new\_feature

## Switch to a new branch," new\_feature "

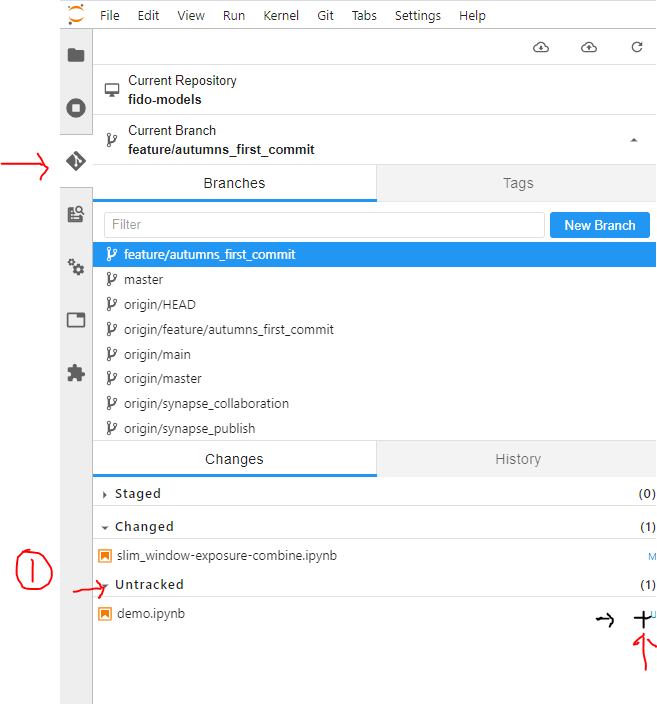
* Assume you’ve committed all your changes, so you can switch back to your master branch:

$ git checkout master

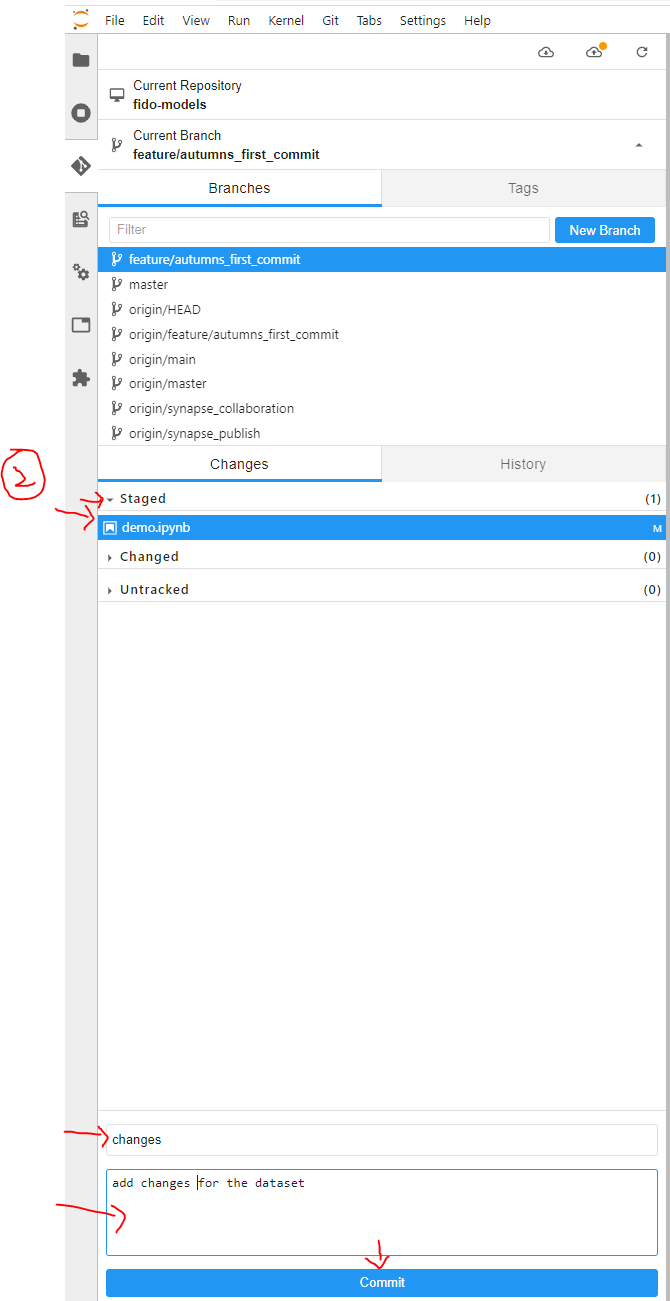
Switch to branch 'master'

### Walkthrough Example: how to apply Git commit, pull, push, merge in 2 ways [(back to top)](#_top)

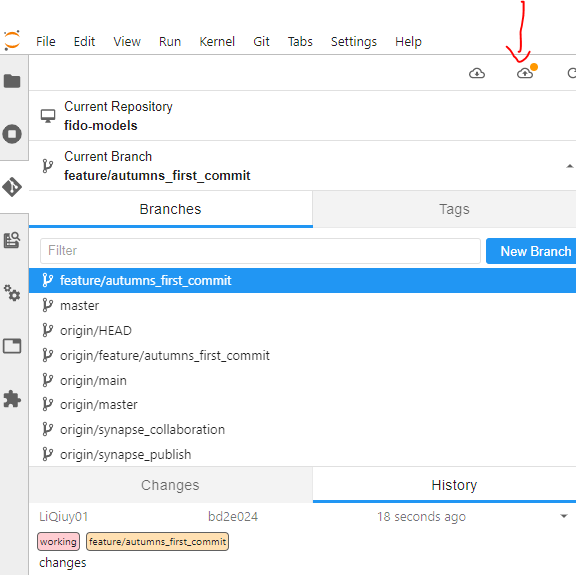
* + Method1: push changes without terminal. If you don’t like terminal and command line, you can also push your changes to GitHub.
  + Overall workflow: track the “untracked file” 🡪 staged a tracked file 🡪 commit changes
    - Step 1: Click the “+: track this file” on the right side of the untracked file



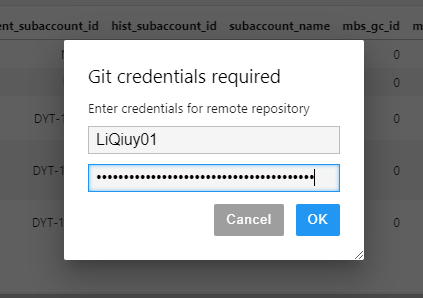
* + - Step2: go to staged file, and add comments and description in the bottom for the change details, then click “commit”



* + - Step3: click the second cloud to push your changes



Step 4: type your GitHub username and your personal token



Please note, the password is your GitHub personal token

Wait and see if you can get the notification for push successfully

* + Method2: use the command line to add staged files, commit changes and push the files to GitHub. [(back to top)](#_top)

$ git remote -v

$ git add demo.ipynb

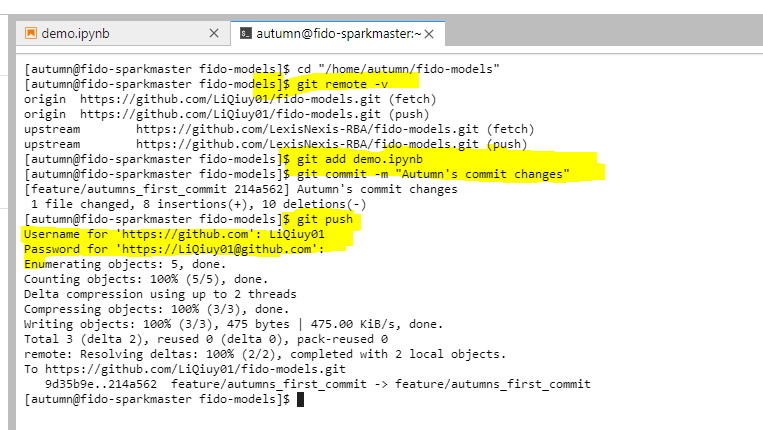
$ git commit -m "Autumn's commit changes"

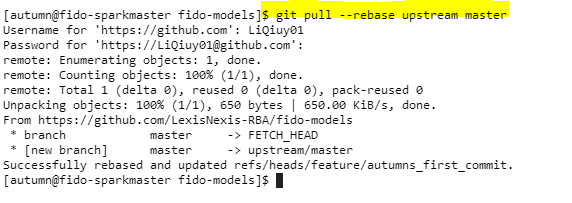
$ git push

$ git pull --rebase upstream master (be careful, only use it when your features are ready to merge back to the upstream)

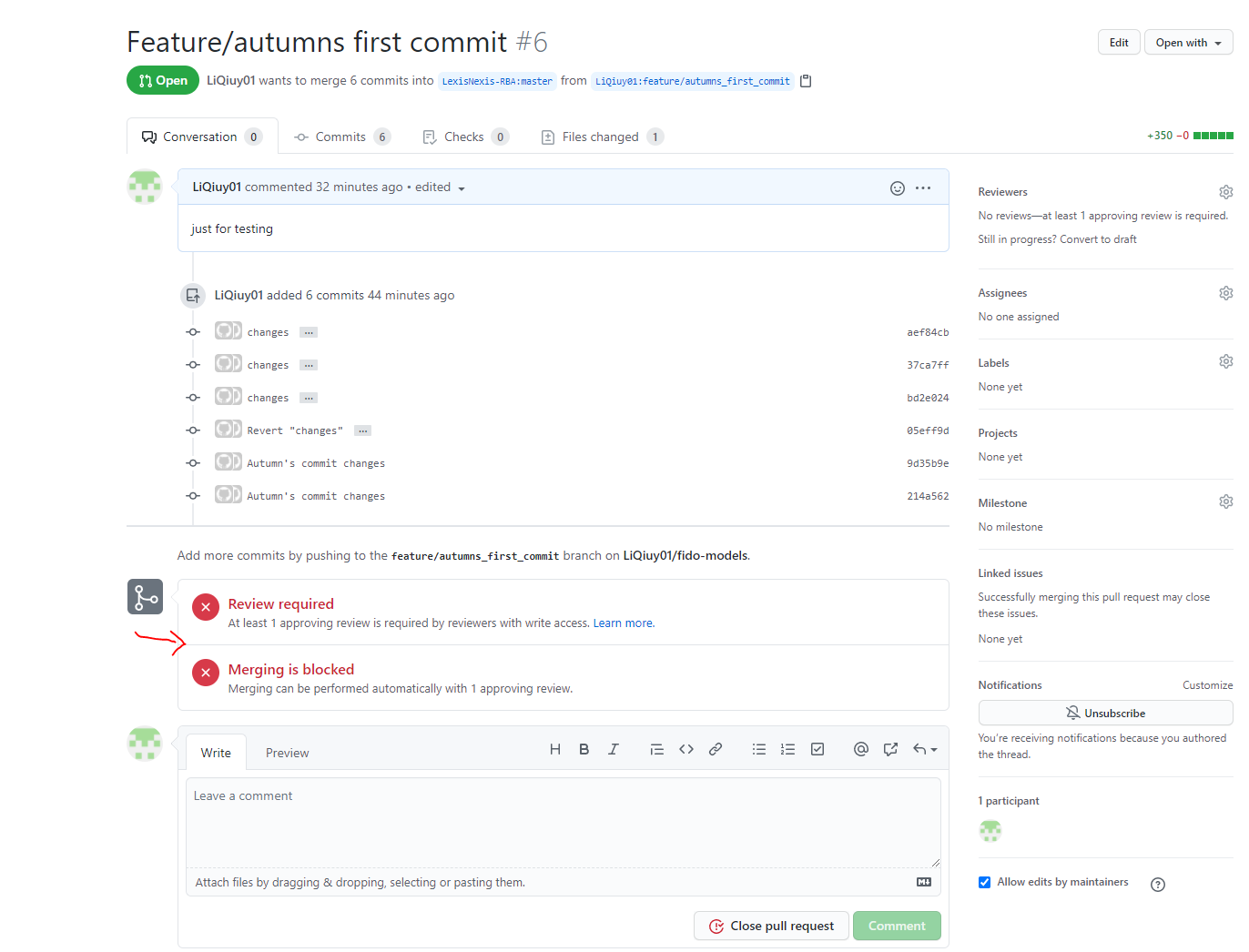
Username for 'https://github.com': LiQiuy01

Password for 'https://LiQiuy01@github.com': \*\*\*\*\*\*





* + After the above steps, wait for your supervisor to review and approve your changes [(back to top)](#_top)



**SSH**

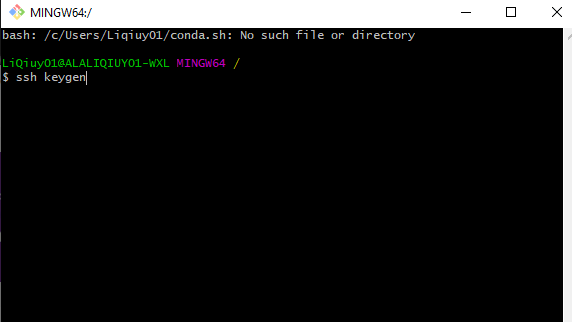
### What is SSH, and how it works [(back to top)](#_top)

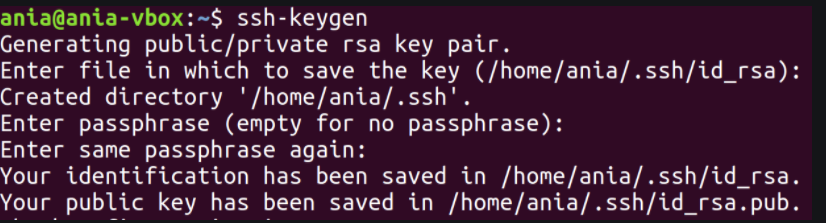
* + SSH keys are an authentication method used to gain access to an encrypted connection between systems and then ultimately use that connection to manage the remote system.

### Generated SSH keys [(back to top)](#_top)

* + Open your Git Bash, and type in ssh-keygen

$ ssh-keygen





### How to use SSH keys for authentication [(back to top)](#_top)

* + Public key: A public key is used to encrypt information, be shared, and be used by the user and the remote server.
  + Private key: Private keys should never be shared with anyone and should be secured on a system – i.e., the system is secured (full disk encryption, MFA), in the user’s possession, and the private key is secured via passphrase.

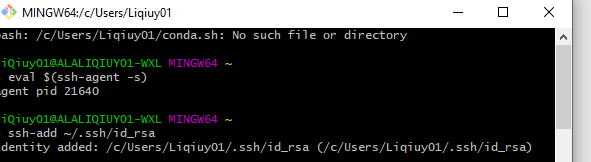
**From local Git Bash to GitHub**

### Add ssh-key agent[(back to top)](#_top)

* + **Step1:** Add key agent locally by below code, and expect to see the results in the screenshot:

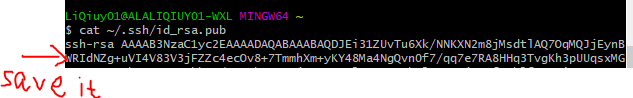
eval $(ssh-agent -s)

ssh-add ~/.ssh/id\_rsa



* + **Step2: find your public key and save it.**

cat ~/.ssh/id\_rsa.pub



* + Step3: add your public keys to your GitHub account
    - Go to <https://github.com/settings/keys>, and click the green button “New SSH Key.” Give a name for the key, and copy-paste your public key from step 2 into the box. Then click “Add SSH Key.” After authentication, you’ll see your SSH public key is added to your GitHub profile pages

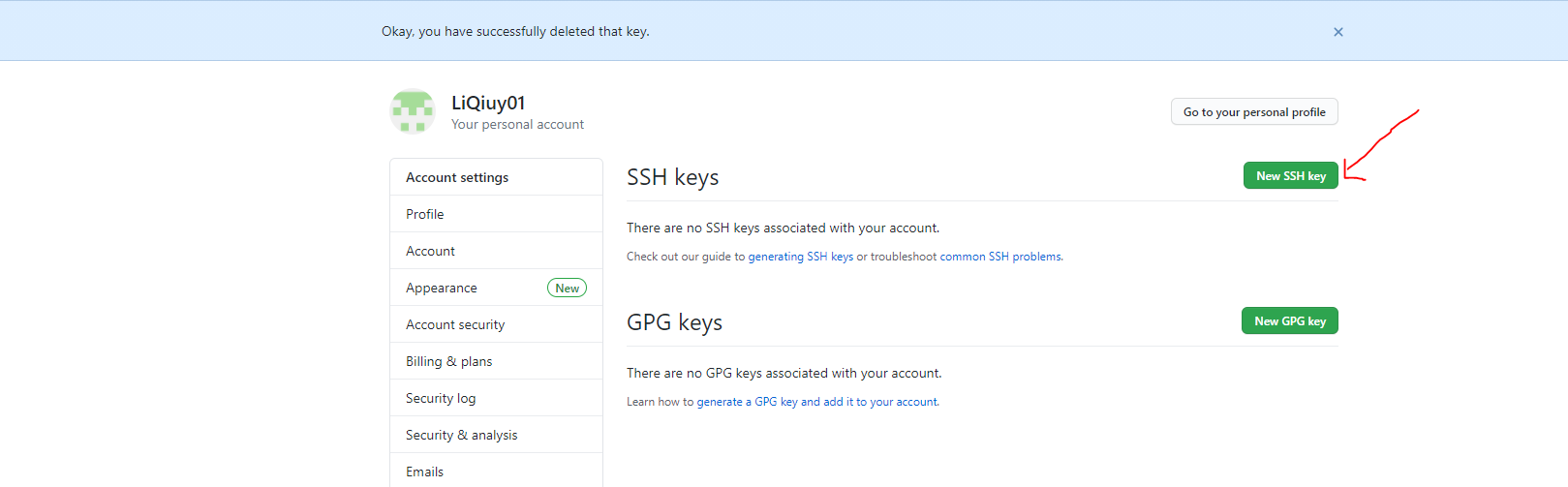


Figure 1: Go to <https://github.com/settings/keys>, and click the green button “New SSH Key.”

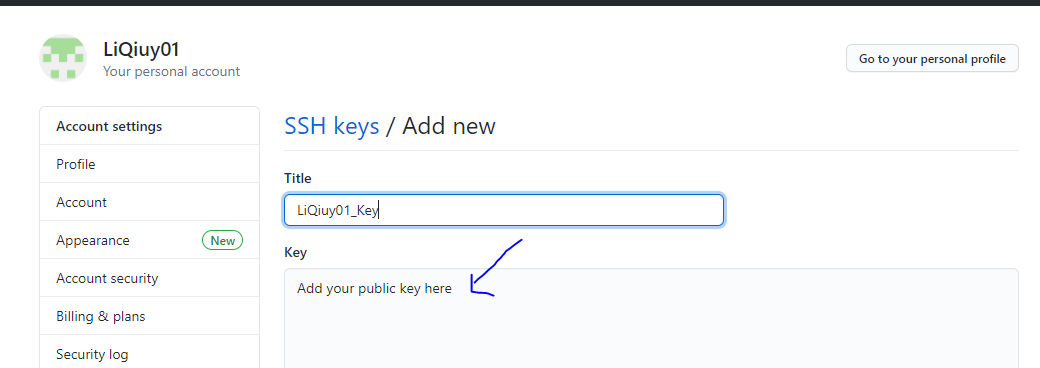


Figure 2: Name your key in the Title box, and copy and paste your SSH key from step 2 into the Key box.

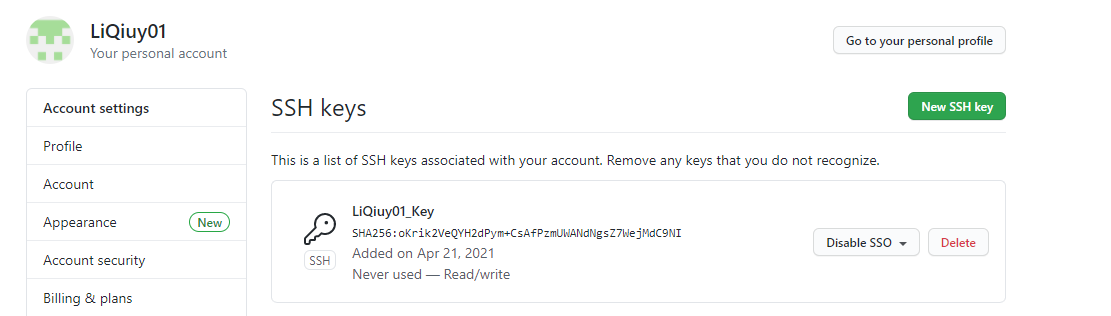
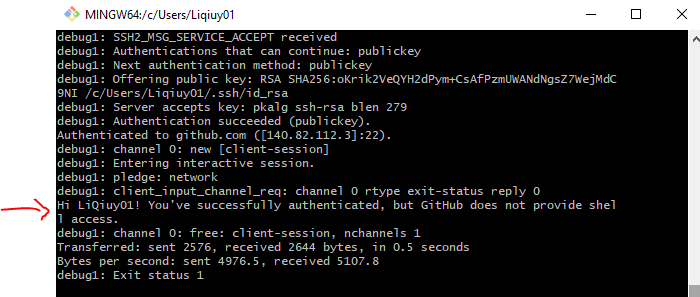


Figure 3: your SSH key is added to your GitHub account

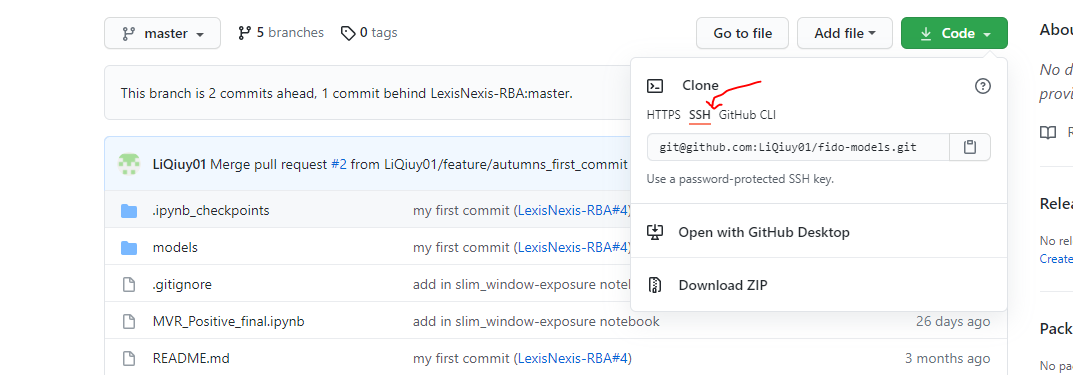
* + Step4: verify that your SSH key is linked to your GitHub account. Type in the below code in your Git Bash, and then you are expecting to see “Hi LiQiuy01! You've successfully authenticated.”

ssh -Tv git@github.com



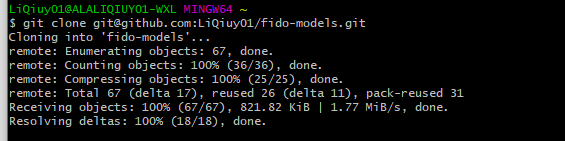
### Clone the repo [(back to top)](#_top)

* + Please Note: If you didn’t set up the previous step- set up the SSH agent. Please finish the last step first.
  + Step1: Go to your personal repo, and click the green button “Code.” Copy the link for the SSH clone option.



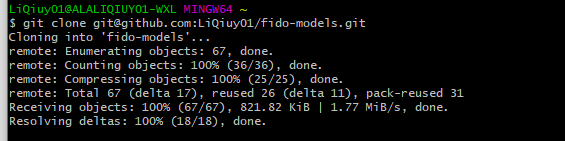
Step2: go to your Git Bash, and start clone: git clone with the link you just copied from step 1

$ git clone git@github.com:LiQiuy01/fido-models.git



### Walkthrough example: how to run the cloned jobs on your local machine

$ git clone git@github.com:LiQiuy01/fido-models.git



If you find permission- deny the issue, then please double-check if your SSH key is linked to your GitHub account successfully

**Papermill**

### What is Papermill[(back to top)](#_top)

* + Papermill is a tool for parameterizing and executing Jupyter Notebooks.

Papermill lets you:

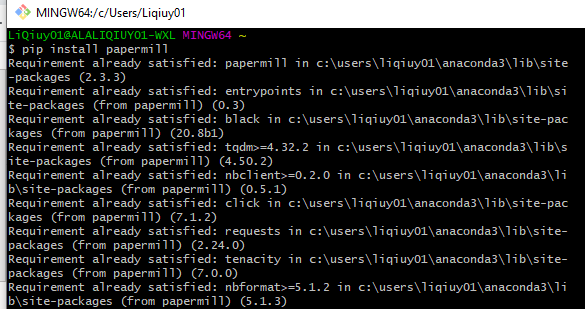
* + - parameterize notebooks
    - execute notebooks

### How to install Papermill[(back to top)](#_top)

python3 -m pip install papermill

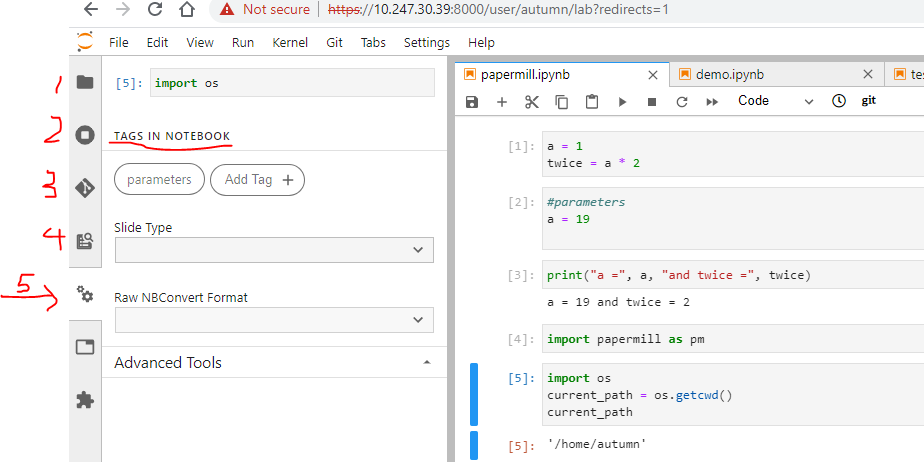
OR

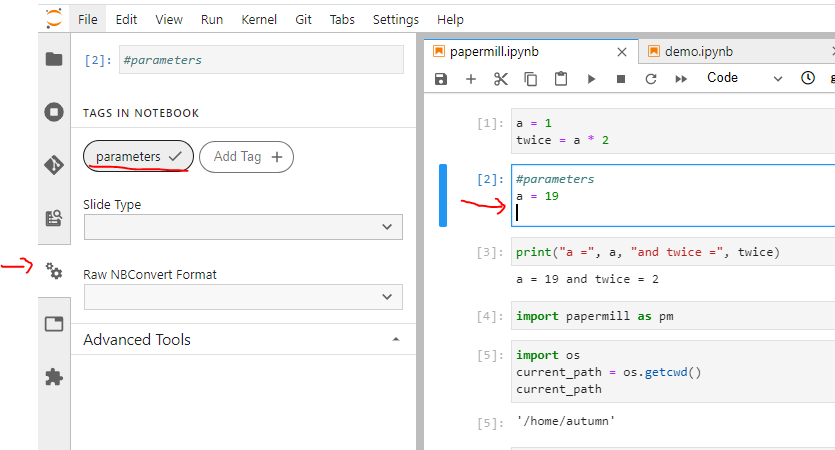
pip install papermill



### How to parametrize notebook by Papermill [(back to top)](#_top)

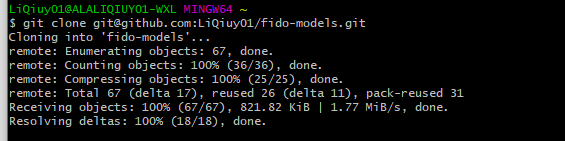
* + Perhaps you have a model that needs to be updated every month. Using parameters makes this task easier.
  + Please check the [link](https://papermill.readthedocs.io/en/latest/usage-parameterize.html) for more information
  + In my example below, I set the variable “a” as my parameter. Go to Jupyter Hub notebook, find the 5th tab on the left toolbar, then add tag ” parameter” to the cell with the variable “a.” In my case, it is the cell [2].



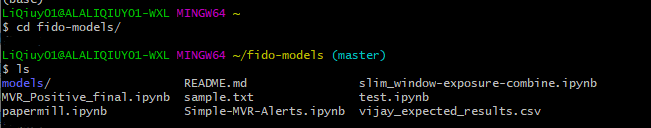


### Walkthrough example: how to git clone the notebook from Jupyter Hub and use the command line to run the notebook locally via ssh key authentication. Finally, use Papermill to parameterize the notebook for automation. [(back to top)](#_top)

* + Step1: [git clone the GitHub](#_Clone_the_repo) repo which has the notebook that you want to automate



* + Step2： If you successfully cloned the repo folder to your local machine, go to the repo folder. For example, my repo folder is ‘fido-models.’



* + Step3: execute your target notebook via papermill
    - Here is the command line format:
* $ papermill input output -p parameter\_name parameter\_value
* for example, my input and output is the same: papermill.ipynb
* $ papermill papermill.ipynb papermill.ipynb -p a 30

