Git and GitHub

, which are popular environments

among developers and data scientists for performing version control of source code files and projects

and collaborating with others.

You can’t talk about Git and GitHub without a basic understanding of what version control is.

Play video starting at ::30 and follow transcript0:30

A version control system allows you to keep track of changes to your documents.

This makes it easy for you to recover older versions of your document if you make a mistake,

and it makes collaboration with others much easier.

Here is an example to illustrate how version control works.

Let’s say you’ve got a shopping list and you want your roommates to confirm the things

you need and add additional items.

Without version control, you’ve got a big mess to clean up before you can go shopping.

With version control, you know EXACTLY what you need after everyone has contributed their ideas.

Play video starting at :1:9 and follow transcript1:09

Git is free and open source software distributed under the GNU General Public License.

Git is a distributed version control system, which means that users anywhere in the world

can have a copy of your project on their own computer; when they’ve made changes, they

can sync their version to a remote server to share it with you.

Git isn’t the only version control system out there, but the distributed aspect is one

of the main reasons it’s become one of the most common version control systems available.

Version control systems are widely used for things involving code, but you can also version

control images, documents, and any number of file types.

You can use Git without a web interface by using your command line interface, but

GitHub is one of the most popular web-hosted services for Git repositories.

Others include GitLab, BitBucket, and Beanstalk.

There are a few basic terms that you will need to know before you can get started.

The SSH protocol is a method for secure remote login from one computer to another.

A repository contains your project folders that are set up for version control.

A fork is a copy of a repository.

A pull request is the way you request that someone reviews and approves your changes

before they become final.

A working directory contains the files and subdirectories on your computer that are associated

with a Git repository.

There are a few basic Git commands that you will always use.

When starting out with a new repository, you only need create it once: either locally,

and then push to GitHub, or by cloning an existing repository by using the command "git init".

Play video starting at :3:3 and follow transcript3:03

"git add" moves changes from the working directory to the staging area.

"git status" allows you to see the state of your working directory and the staged snapshot

of your changes.

"git commit" takes your staged snapshot of changes and commits them to the project.

"git reset" undoes changes that you’ve made to the files in your working directory.

"git log" enables you to browse previous changes to a project.

"git branch" lets you create an isolated environment within your repository to make changes.

"git checkout" lets you see and change existing branches.

"git merge" lets you put everything back together again.

To learn how to use Git effectively and begin collaborating with data scientists around

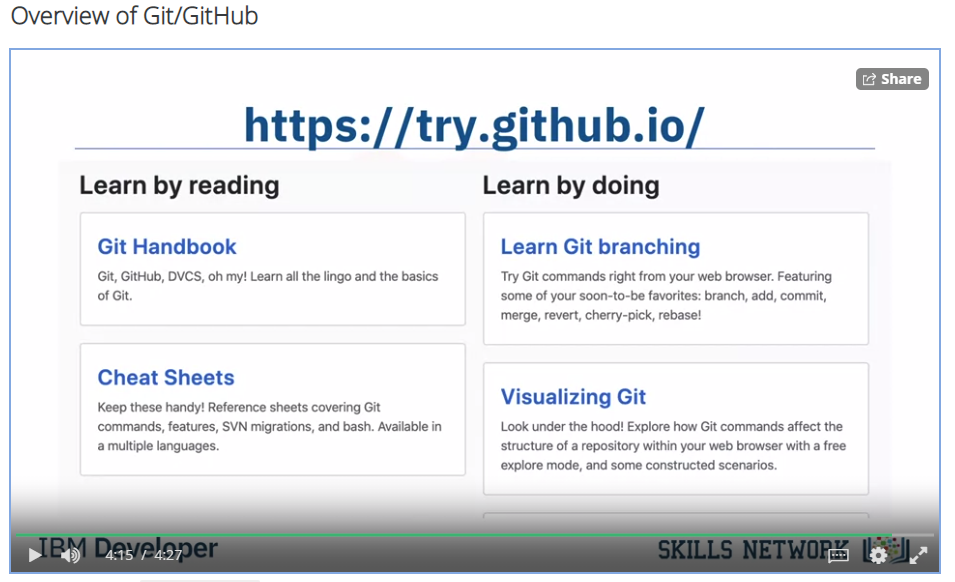
the world, you will need to learn the essential commands.

Luckily for us, GitHub has amazing resources available to help you get started.

Go to try.github.io to download the cheat sheets and run through the tutorials.

In the following modules, we'll give you a crash course on setting up your local environment

and getting started on a project.



In the previous video, you learned about Git  and GitHub. Before you continue with this video,

register for a GitHub account and log in.  Let’s start by creating a new repository.

Click + then click New Repository. To  create a new repository, you need to provide

these details: give your new repository a name;  optionally, add a description of your repository;

choose the repository visibility - whether you  want it to be public or private; and choose the option

to Initialize this repository with readme file. Then click Create Repository.

Play video starting at ::54 and follow transcript0:54

You will now be redirected to the  repository you have created.

The root folder of your repository is listed  by default and it has just one file ReadMe.md.

Play video starting at :1:8 and follow transcript1:08

Now, it’s time to edit the readme. You can do this  in your browser. Just click the pencil to open the

online editor and you can change the text of the  readme. To save your changes to the repository,

you must commit them. After you have made your  changes, scroll down to the Commit changes

section. Add a commit message and optionally add  a description, then click Commit changes. The

"commit changes" is used to save your changes  to the repository. Go back to the home screen by

clicking the repository name link. Note that the  readme file is updated and verify your changes.

Play video starting at :1:49 and follow transcript1:49

Let’s learn how to create a new file using the  built-in web editor provided by GitHub which

runs in the browser. Click Add File, then  click Create New File to create the new file.

Play video starting at :2:4 and follow transcript2:04

To create a python file called firstpython.py.

First, provide the file name. Next, add a comment  that describes your code, then add the code.

Play video starting at :2:17 and follow transcript2:17

Once finished, commit the change to the  repository. You can see that your file is

now added to the repository and the repository  listing shows when the file was added or changed.

When you need to change the  file, you can edit it again.

Click the file name, and then click the pencil  icon, make your edits and commit the changes.

Play video starting at :2:40 and follow transcript2:40

You can also upload a file from your  local system into the repository.

From the home screen of the repository, click  Add File and choose the Upload files option.

Play video starting at :2:53 and follow transcript2:53

Click Choose Your Files and select the files  you want to upload from your local system.

Play video starting at :2:59 and follow transcript2:59

The file upload process may take a short time,

depending on what you are uploading.  Once the files finish uploading,

click Commit Changes. The repository now reflects  the files that were uploaded. In this video,

you learned how to create a repository, edit  files, and commit changes using the web interface.

**Effort:** 20 min

In this lab, you will get started with GitHub by creating a GitHub account and project and adding a file to it using its Web interface.

Objectives

After completing this lab, you will be able to:

1. Describe GitHub
2. Create a GitHub account
3. Add a Project / Repo
4. Edit / Create a file
5. Upload a file & Commit

GitHub Overview

First, let us introduce to GitHub. GitHub in simple words is a collection of folders and files. It is a Git repository hosting service, but it adds many of its own features. While Git is a command-line tool and a server needs to be hosted and maintained via command line as well, GitHub provides this Git server for you and a Web-based graphical interface. It also provides access control and several collaboration features, such as wikis and basic task management tools for every project. GitHub provides cloud storage for source code, supports all popular programming languages, and streamlines the iteration process. GitHub includes a free plan for individual developers and for hosting open source projects.

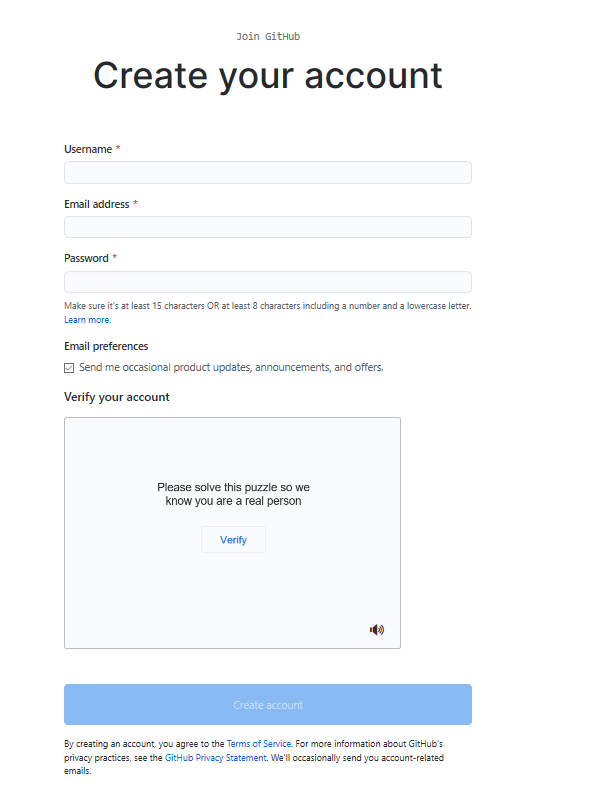
Exercise 1: Creating a GitHub Account

Please follow the steps given below to create an account in GitHub:

Step 1: Create an account: <https://github.com/join>

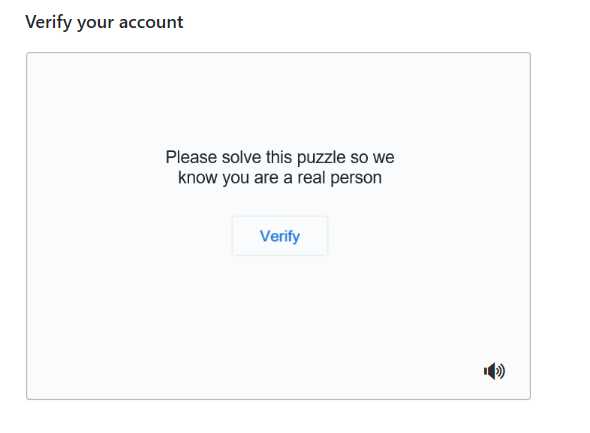
**NOTE:** If you already have a GitHub account, you can skip this step and simply login to your account.

Step 2: Provide the necessary details to create an account as shown below:

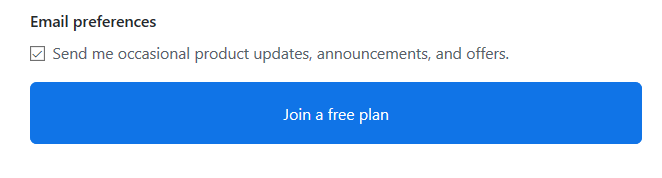


and click Create account.

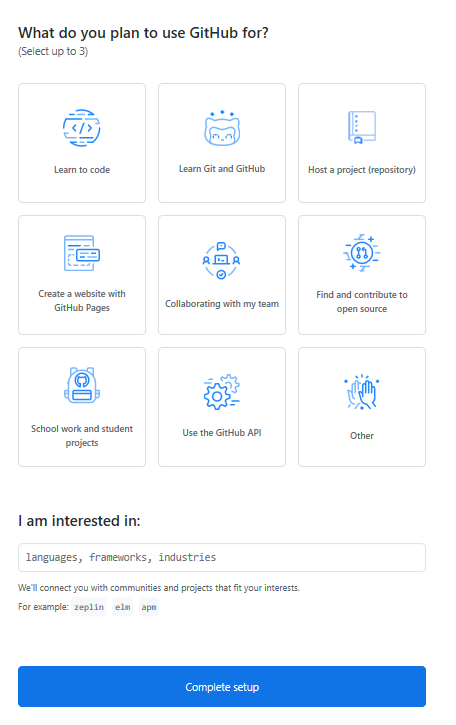
Step 3: Click Verify to verify the account and click Done



Step 4: After verification, click Join a Free Plan

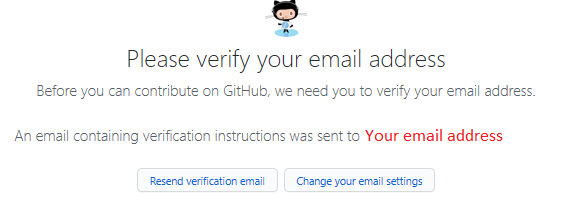


Step 5: Select the details as shown below and click Complete Setup

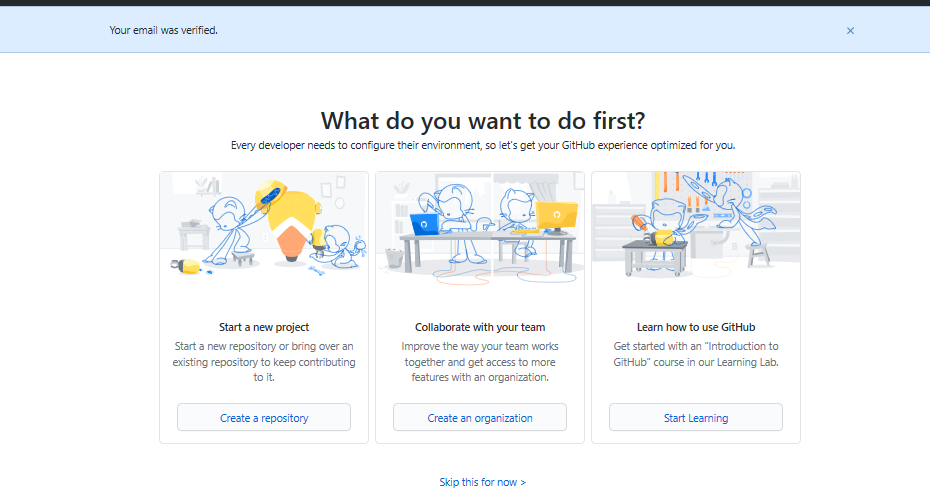


Step 6: Go to your email, find the verification email from GitHub, and click on the link/button in that email to verify your email.

**NOTE:** If you do not receive verification email, click Resend verification email.

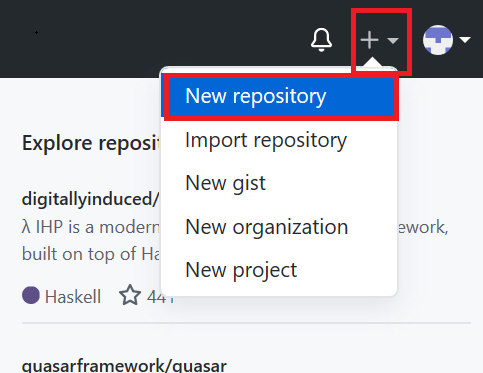


Email is verified

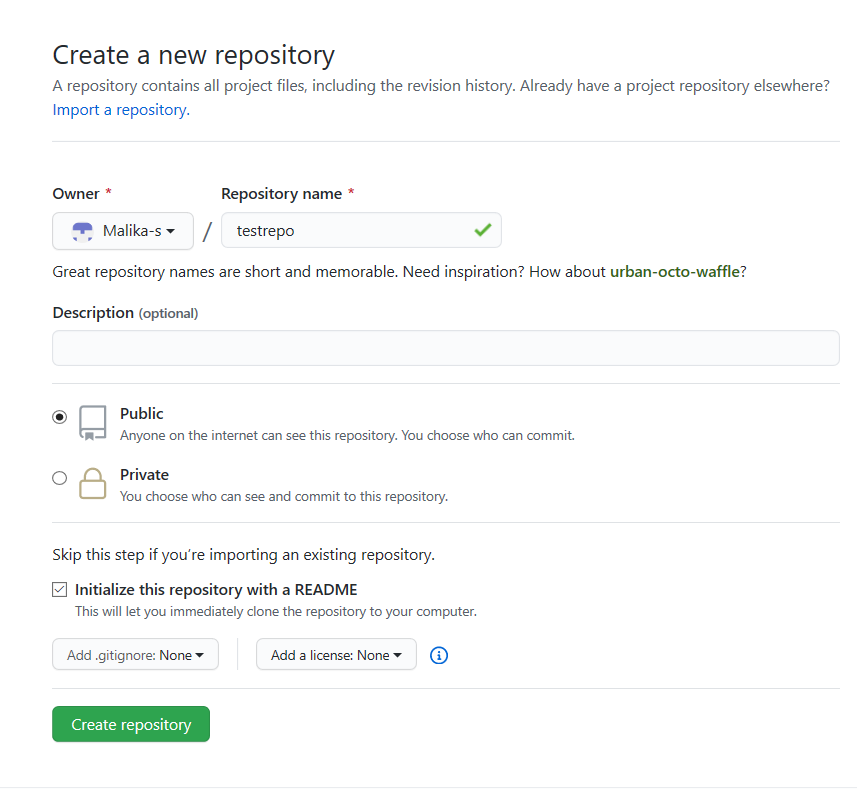


Exercise 2: Adding a Project / Repo

Step 1: Click on the + symbol and click New repository.



Step 2: Provide a repository a name and initialize with the empty README.md file.



and click Create repository.

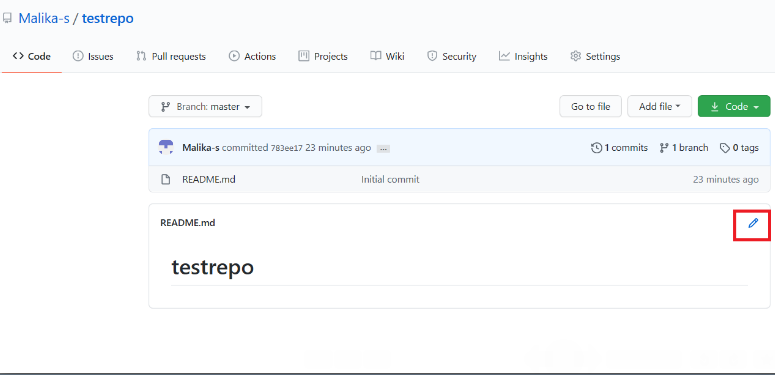
Now, you will be redirected to the repository you have created.

Let's start editing the repository.

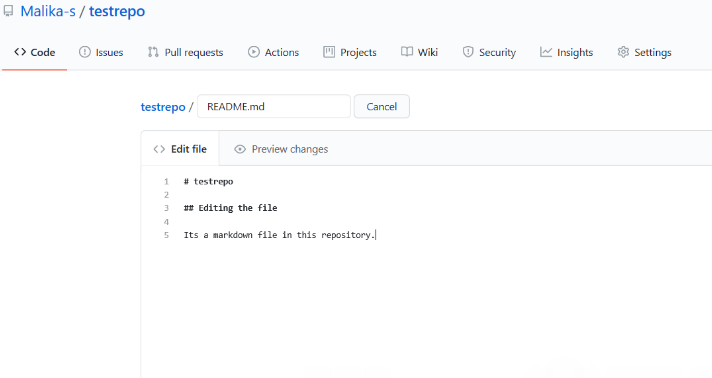
Exercise 3: Create / edit a file

Exercise 3a: Edit a file

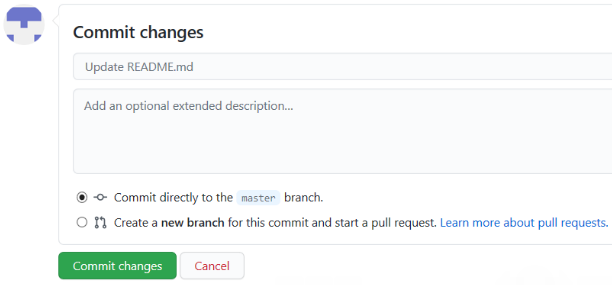
Step 1: Once the repository is created, the root folder of your repository is listed by default and it has just one file ReadMe.md. Click on the pencil icon to edit the file.



Step 2: Add text to file.



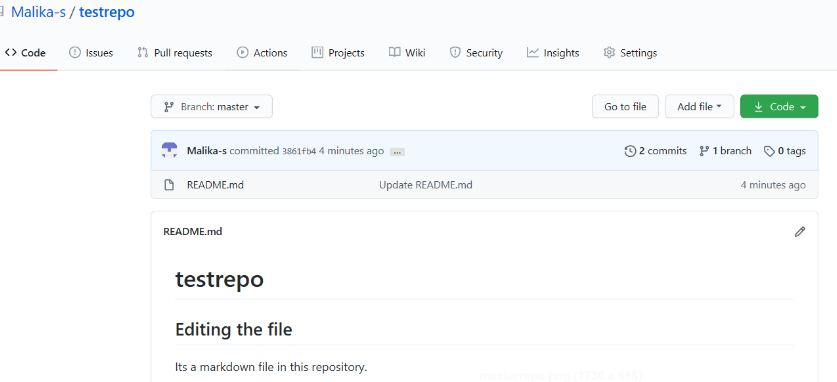
Step 3: Scroll down the page after adding the text and click Commit Changes.



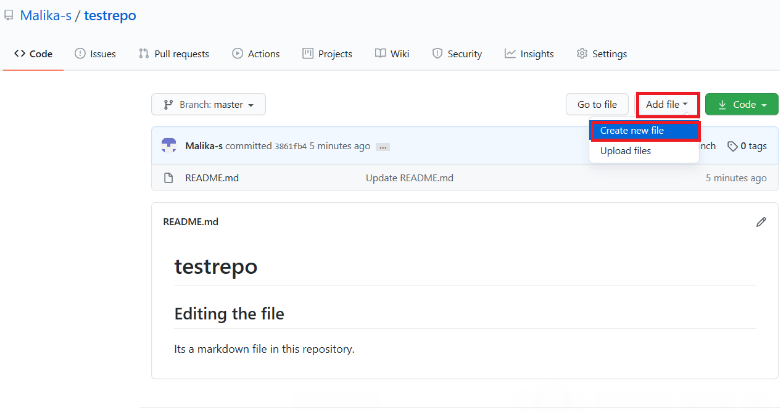
Now, check your file is edited with the new text.

Exercise 3b: Create a new file

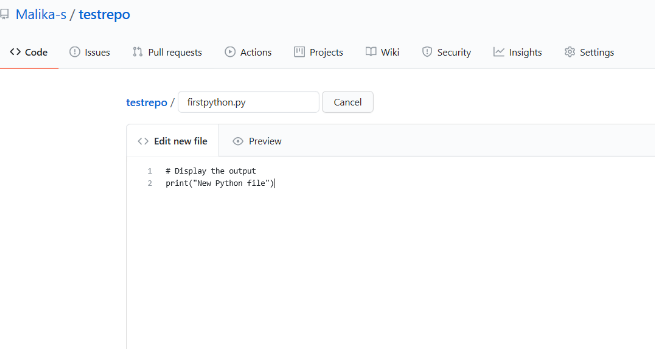
Step 1: Click on the repository name to go back to the master branch like in this testrepo.



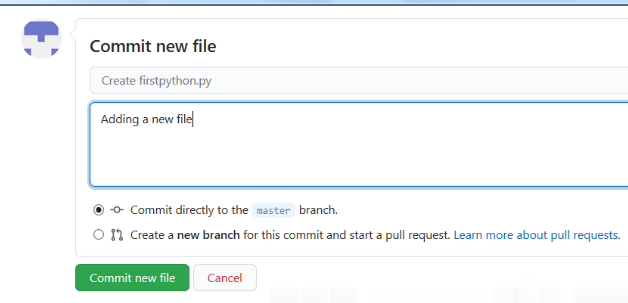
Step 2: Click Add file and select Create New file to create a file in the repository.



Step 3: Provide the file name and the extension of the file. For example, firstpython.py and add the lines.



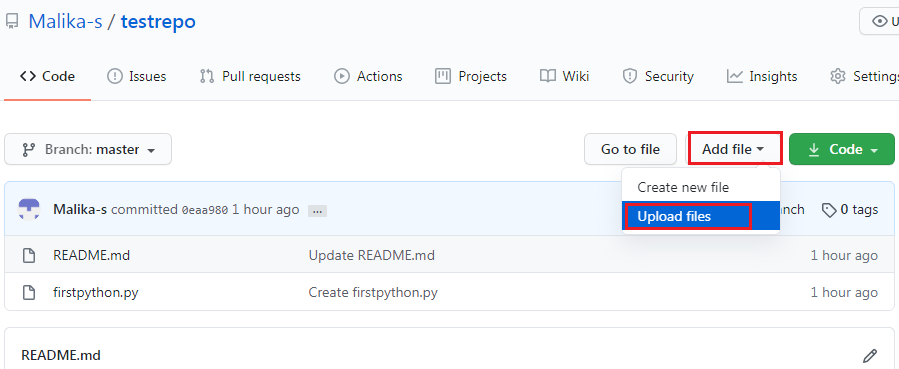
Step 4: Scroll down the page after adding the text. Add description of the file (optional) and click Commit new file.



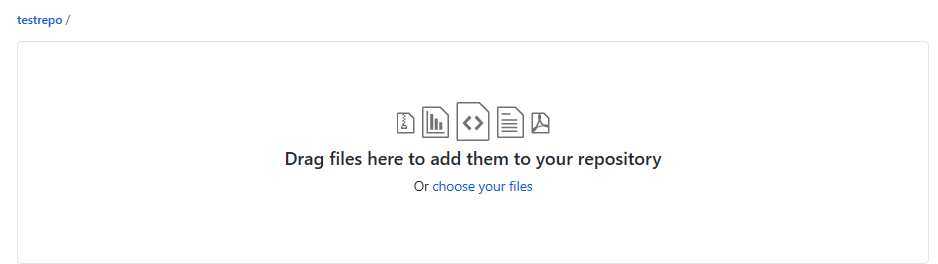
Step 5: Your file is now added to your repository and the repository listing shows when the file was added/changed.

Exercise 4: Upload a file & Commit

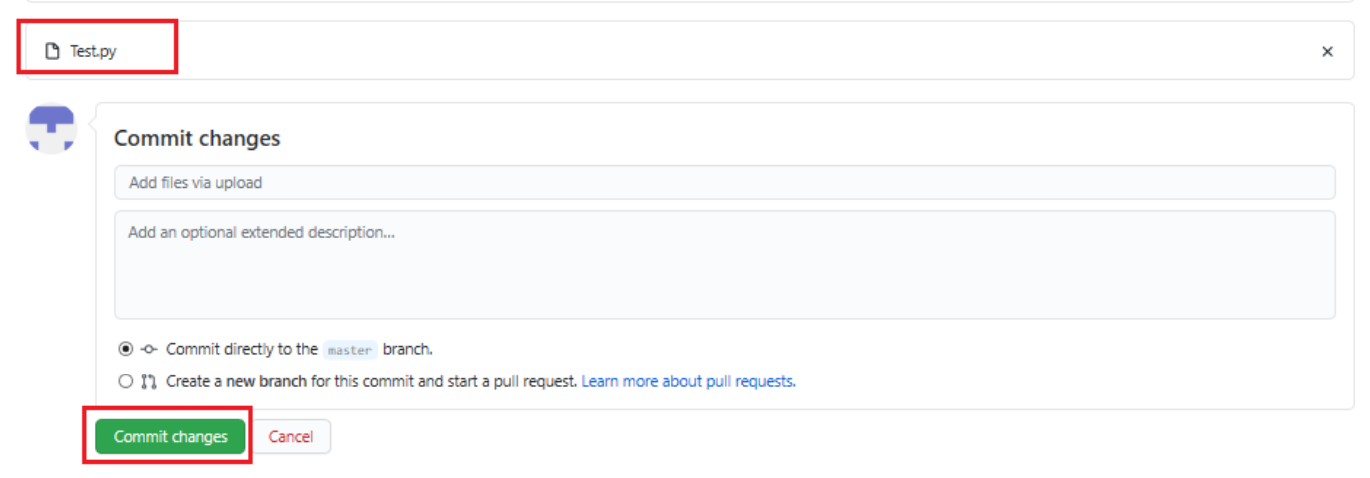
Step 1: Click Add file and select Upload files to upload a file (Upload any .txt,.ipynb, .png file) in the repository from the local computer.



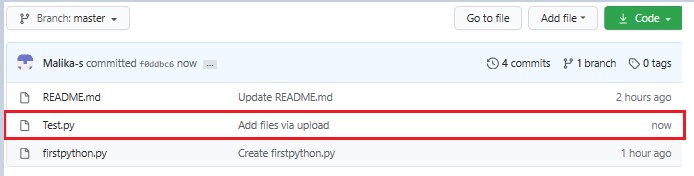
Step 2: Click on choose your files and choose any files from your computer.



Step 3: Once the file finishes uploading, click on Commit Changes



Step 4: Now, your file is uploaded in the repository.



Summary

In this document, you have learned how to create a new repository, adding a new file, editing a file, and uploading a file in a repository and commit the changes.

In this video, you will learn how to create and

merge a branch using the GitHub web interface.

A branch is a snapshot of your repository to which you can make changes.

It is a copy of the master branch and can be used to develop and test changes to the

workflow before merging it back to the master branch. In Git and GitHub, there is a main

branch.

The main branch which is called Master, is the one with deployable code

and the official working version of your project.

It is meant to be stable and it is always advisable never to push any code that is not

tested to master.

Many times, we want to make changes to the code and workflow in the master branch.

That is when we create a copy of the Master branch.

Let’s call it Child Branch.

We will then create a copy of the workflow to the child branch in the child branch, changes

and experiments are done.

We will build and make edits, test the changes and when we are satisfied with the changes

we will merge it back to the master branch where we prepare the model for deployment.

We can see that all of this is done outside of the main branch and until we merge, changes

will not be made to the workflow before we branched.

To ensure that changes done by one member, does not impede or affect the flow of work

of other members, multiple branches can be created and merged appropriately to master

after the workflow is properly tested and approved.

To create branches in GitHub, let’s look at this repository.

There is currently one branch in the repository.

I want to make some changes, but I don’t want to alter the master in case something

goes wrong.

We will create a branch.

To do that, we will click the drop-down arrow and create a new branch.

Let's name it - child branch and then we will click enter.

The repository now has two branches, the Master and the Child branch.

You can check this by selecting Child branch in the Branch selector drop-down list.

Whatever was in the Master branch was copied to the child branch.

But we can add files in the child branch without adding any files to the master branch.

To add a file, make sure Child branch is selected in the branch selector drop-down list.

Click on create new file. In the space provided, name the file - we will name it

testchild.py and then we will add a few lines of code.

We will print the statement – Inside child branch.

At the bottom of the screen, we will see a section called Commit new file.

Commit messages are very important as it helps to keep track of the changes that were made.

It is important to add a descriptive commit message so that other team

members can understand it.

Here we will add a commit message,

Create testchild.py, then we will commit the new file.

The file gets added to only the child branch.

We can check this by going to the master branch by clicking ‘master’ from the Branch selector

menu and here we can see that the new file is not added to the master branch.

After we have created the new file, tested and made sure that is up to standards.

We then want to merge the changes in the child branch to reflect in the master branch.

To merge the changes, we will first have to create a pull request, also known as a PR.

A pull request in simple terms is a way to notify other team members of your changes

and edits and ask them for review so they can be pulled or merged into the master branch.

Pull requests are the heart of collaboration on GitHub.

When you open a pull request, you’re proposing your changes and requesting that someone review

and pull in your contribution and merge them into the target branch.

Pull requests show the differences of the content from both branches.

To open a pull request and see the differences between the branches, click on the Compare

and pull request button.

If you scroll down to the bottom of the screen, you will see something like this that shows

you the difference between both branches.

As you can see on the screen it shows that one file has changed and the file has two

additions, which are the two lines we added to the file and 0 deletions.

We will now create the pull request.

Add the title and an optional comment for the pull request.

Click Create Pull request to create the pull request.

You can assign team members to review and approve pull requests.

On the next page you will see this image. If you are okay with the changes, click on Merge

pull request and click confirm. You will get a confirmation that the pull request has been

successfully merged.

You can now delete the branch if you no longer need to make any edits or add new information.

Now, the child branch has completely merged with the Master branch.

You can check the Master branch and we can now see it contains the

testchild.py file.

You should now be familiar with how to create and merge branches using the web interface.

Hands-on Lab: Creating and merging branches in your GitHub repo

**Effort:** 20 min

Objectives

After completing this lab you will be able to:

1. Create a branch
2. Commit changes in the child branch
3. Open a pull request (PR)
4. Merge the PR into the master branch

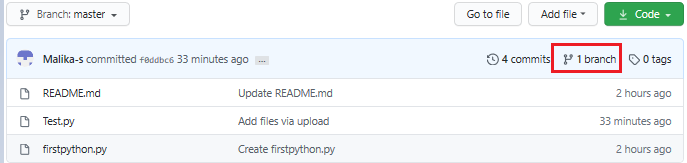
Pre-requisites

This hands-on lab requires you to have already created a GitHub account, and added a project to it, as illustrated in the [previous lab](https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DS0105EN-SkillsNetwork/labs/Module2/GitHub1_Getting_Started.md.html)

Exercise 1: Creating a branch

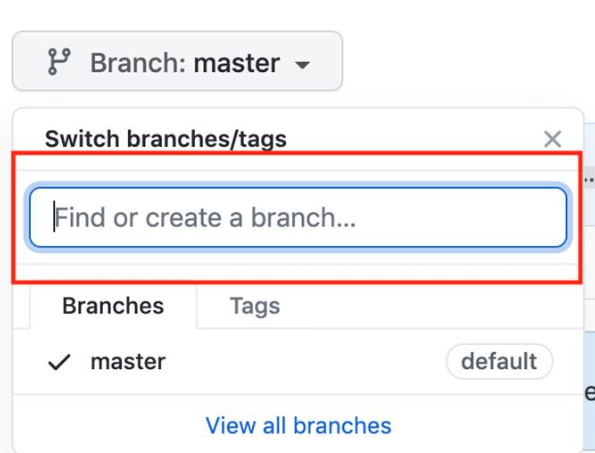
You can create or delete branches directly on GitHub.

Step 1: Currently, there is one branch as shown below:

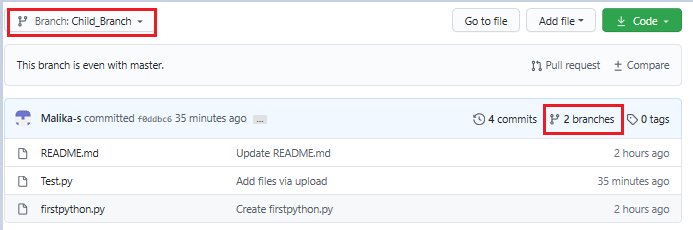


Step 2: On GitHub, navigate to the main page of the repository.

Step 3: Click the Branch selector menu. Enter the name of the branch you want to create and press Enter.



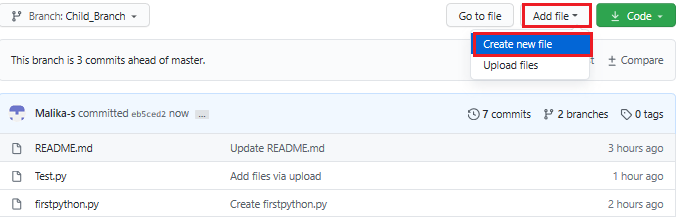
Step 4: Observe that your repository has two branches Master and Child\_Branch(check using arrow).



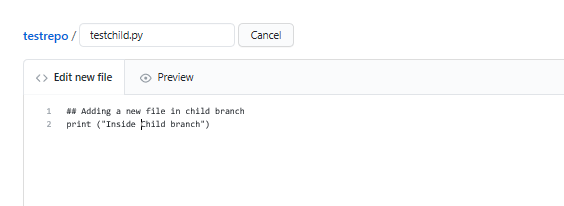
Whatever, is in the master file is copied to the child branch. But when we add a file or edit any file in child branch that will not reflect in the 'Master' branch.

Exercise 2: Adding a file in the Child Branch

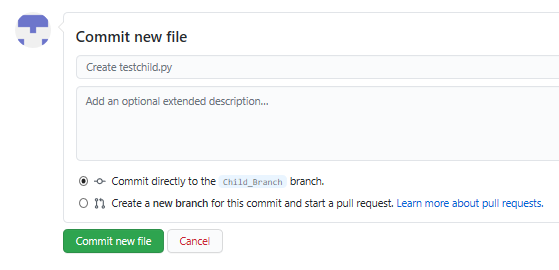
Step 1: Click Add file and select Create New file to create a file in the repository.



Step 2: Provide the file name and the extension of the file. For example, testchild.py and add the lines.



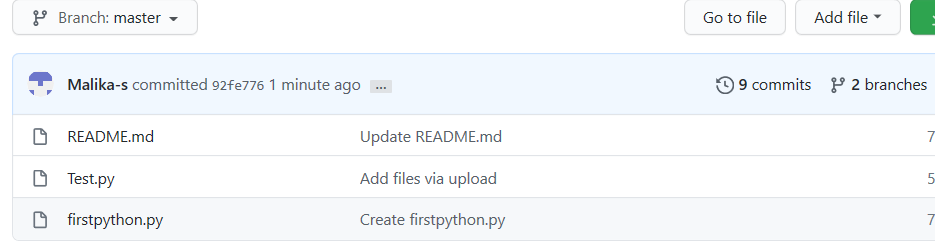
Step 3: Scroll down the page after adding the text. Add a description of the file (optional) and click Commit new file.



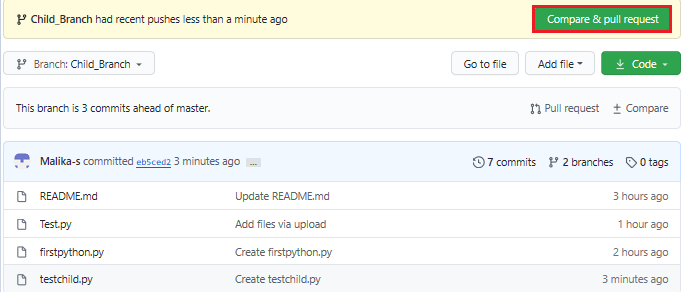
The file has added to the child branch.

Exercise 3: Open a Pull Request

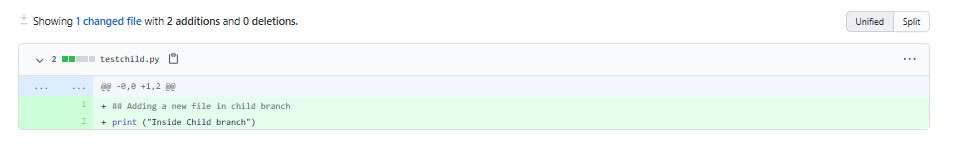
You can check the master branch now there is no testchild.py file by selecting the Branch selector menu.



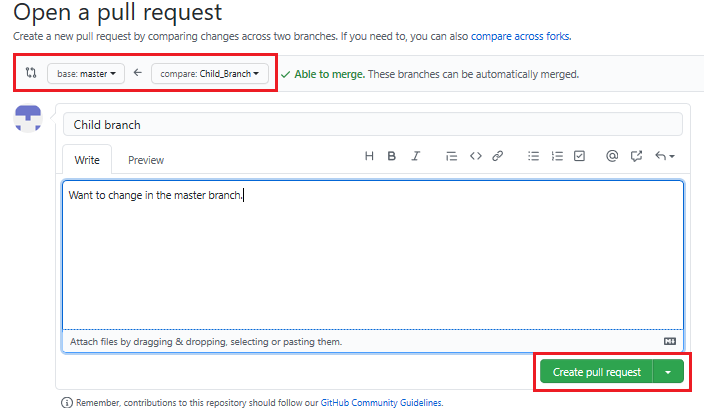
Or you can also compare the file as shown below with the option given Compare and pull request.



Step 1: Scroll down the page, you will get 1 file changed

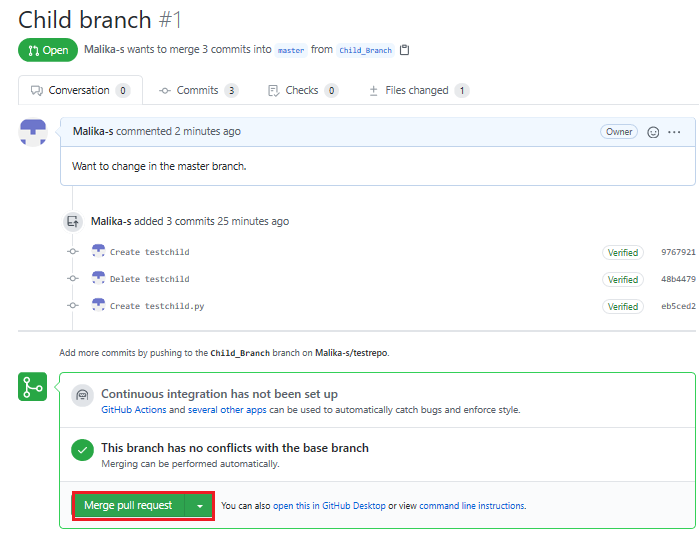


Step 2: Scroll up and create a pull request using the option Create Pull request. In the highlight, you can see the arrow which means that you are comparing and creating a pull request. Add the comments (optional) to create a request.

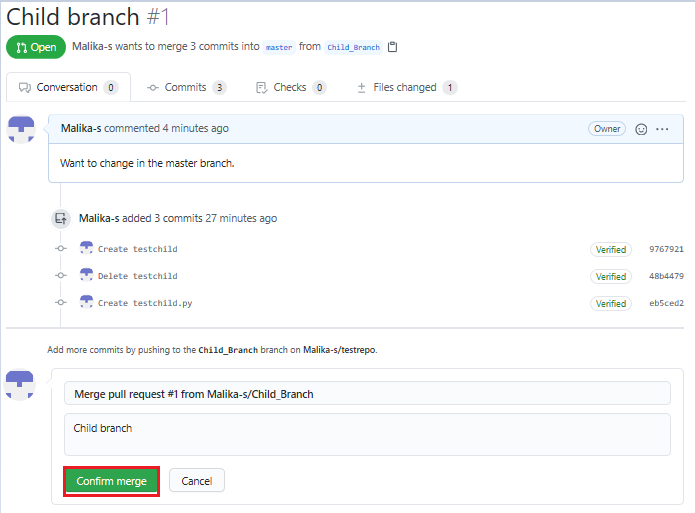


Exercise 4: Merge the Pull Request

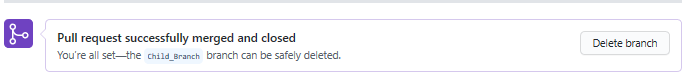
To accept the pull request, click the Pull Requests tab to see a summary of pending pull requests. If you are happy with the changes, click Merge Pull request to accept the pull request and perform the merge. You can add a comment if you want.



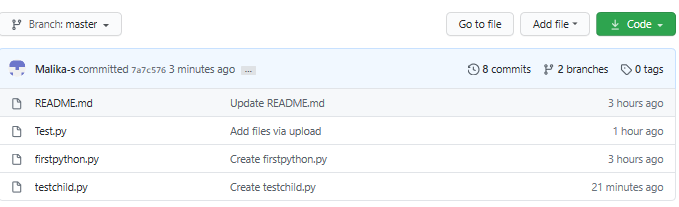
* Once you click Merge Pull request, you will see a button Confirm merge.



Your request has now merged successfully.



Now, the child branch has completely merged with the Master branch. You can check the Master branch is having the testchild.py file.



Summary

In this document, you have learned how to create a branch, edit and commit the changes, open a pull request and merge the request.

Welcome to again, introduction to Git and GitHub.

It's very basic.

If you have absolutely no clue,

then this is for you.

Otherwise it's maybe boring.

I'm just telling you how I'm using Git,

and I am able to

survive in most of the cases with that knowledge.

Sometimes I have to

read the documentation for a more advanced stuff,

but in let's say, 80 percent of the cases

that I will tell you now is sufficient.

Let's actually start using GitHub.

It makes things a bit more easy.

There's also competition to GitHub. This is GitLab.

GitLab is particular interesting because

the whole system behind GitLab is open-source.

This is a hosted version,

but you can have the whole system on-prem,

and also purchase support from GitLab.

Those guys are partially in

Berlin, so they are pretty cool.

Anyway, let's start with GitHub here.

What I will do now is create a repository.

A repository is the location where your stuff goes into.

Let's call it, I don't know,

demo, and we can choose

whether we want to create

a public or a private repository.

That repository is remote,

that's hosted at GitHub,

and let's initialize with a README.

Play video starting at :1:45 and follow transcript1:45

Let's skip gitignore for now,

I will show you that later.

We create now repository.

Once this is created,

it's very easy to have a copy of this repository locally.

There are two options: one is

cloning it via HTTPS or via SSH.

I'm always using SSH,

it just to make things a bit more easy.

The only thing you need to do is you have to add

your public identity to GitHub.

I can show you that later.

Now, the only thing we need to do

is we go to empty folder,

and then we say git clone, and this link.

What now happens is the contents

of this is copied to my machine.

Here, I have to specify my password for my SSH identity,

so that's something I need to explain you later.

If you know, go to demo.

You will see here the README file,

Play video starting at :3:8 and follow transcript3:08

and you see here there is content of the README file.

Now, let's add a file.

Play video starting at :3:26 and follow transcript3:26

You see here there's only the README, and the.git folder.

Please never ever touch it.

Play video starting at :3:35 and follow transcript3:35

Let's create another file.

Let's say, vi test.txt,

and hello, this is a test.

Now, we add this to the repo.

First of all, we say git add test.txt.

Now, if we say now git status,

then we see that this file

is now in the so-called staging area.

That's something in-between your local folder

and your local repository.

You have a local folder with the contents,

you have a local repository,

and you have a remote repository.

Personally, you work with your local repository,

and all the other guys working together with you

are working also with their local repositories,

and do sync all your contents with the remote repository.

Now, we want to commit or let's do something else.

Let's say, we create a file test2,

and we say git add test2.txt.

Now interestingly, if we now say git status,

those are both in the staging area.

Now, if we say git commit,

and we say message initial commit,

and that is now in our local repository on this machine.

The idea is why you have the staging area,

you can control which files

go into a commit because let's say,

the atomic entity here in git is a commit,

so everything is around beat on the commit.

You can yet definitely control

which files are going into a single commit.

Now, what you want to do is you want to push

the contents of

your local repository to remote repository.

Because if you now go to this here,

to this GitHub project,

you see that there is no additional file.

We say now, git push.

What now happens is,

I again have to provide a password.

I don't have the key agent that running.

Now, you see here we have committed

this local change in

the local repository to remote repository.

Now, you can see that you have these two files here.

If you click on that "File",

you'll see first of all the content,

and also you'll see here which commit

was responsible for adding this file.

Another interesting thing is

if you are starting from scratch,

you can also do the following: Now,

let's delete this project,

and this is a bit hard.

Just to make sure you don't do that accidentally,

so delete this repository,

and you have to say demo here,

although I think you have to type the whole thing here.

Now, this project is deleted.

Play video starting at :6:58 and follow transcript6:58

Let's do something else.

Let's create a new repository,

but now let's init a repository on my machine.

Let's call it demo again.

We don't initialize it with a README.

We say create repository.

If you're lucky, GitHub will show

us how to do it. Yeah, here.

Let's actually do it ourselves.

Let's get rid of the demo folder,

Play video starting at :7:30 and follow transcript7:30

and let's create it again.

Play video starting at :7:35 and follow transcript7:35

Now, do what they tell us.

We create a README,

and let me say, git init.

Now, we have it initialized because now

we have the.git folder which you never ever should touch.

We say git add README,

but we can also say git add.,

so it's adding everything which was in our folder.

It's only the README.md,

and then we say,

git commit as usual, test or whatever.

Now, we can't say git push now.

If we say git push,

it doesn't know where to push it

because actually we haven't

told this local folder that it should

connect to remote repository.

That we do with git remote add origin,

and that's the URL of our project.

Then we can say git push u origin master,

and that's basically it.

We have to re-enter the password for my RSA key,

and if you're lucky,

and everything worked fine,

we can now go to the project,

and we should see the README file,

which is the case.

That's it for now. That's the first part.

In the next video, I will show you how to do branches,

and actually also how to create a pull request.

Thanks for watching, and see you later.