GitHub Desktop Tutorial

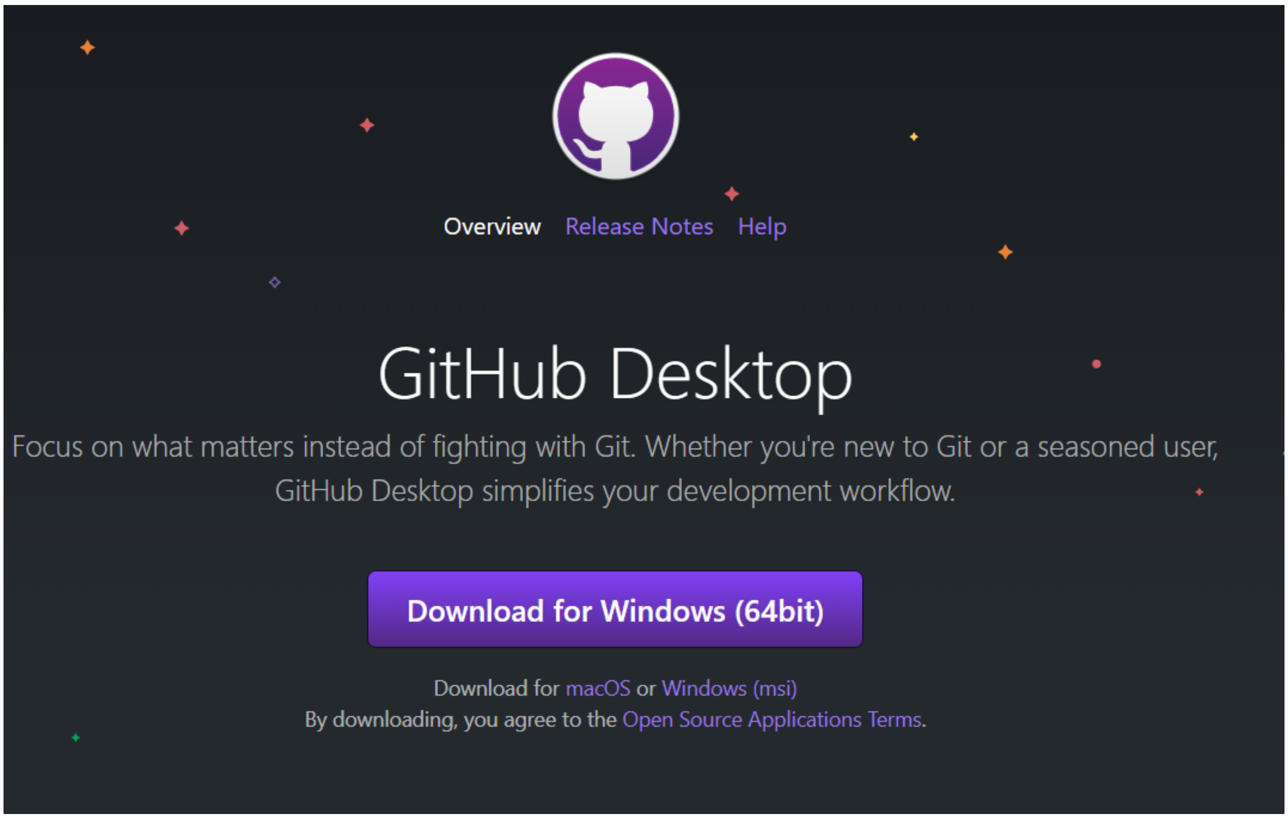
> created by hejing 2020/1/17

Introduction

This guide will walk you through the process of using GitHub Desktop to work on a Git repository. GitHub Desktop extends and simplifies your GitHub.com workflow, using a visual interface instead of text commands on the command line. By the end of this guide, you'll have used GitHub Desktop to create a repository, make changes to the repository, and publish the changes to GitHub.com.

Step 1. Install and sign into GitHub Desktop

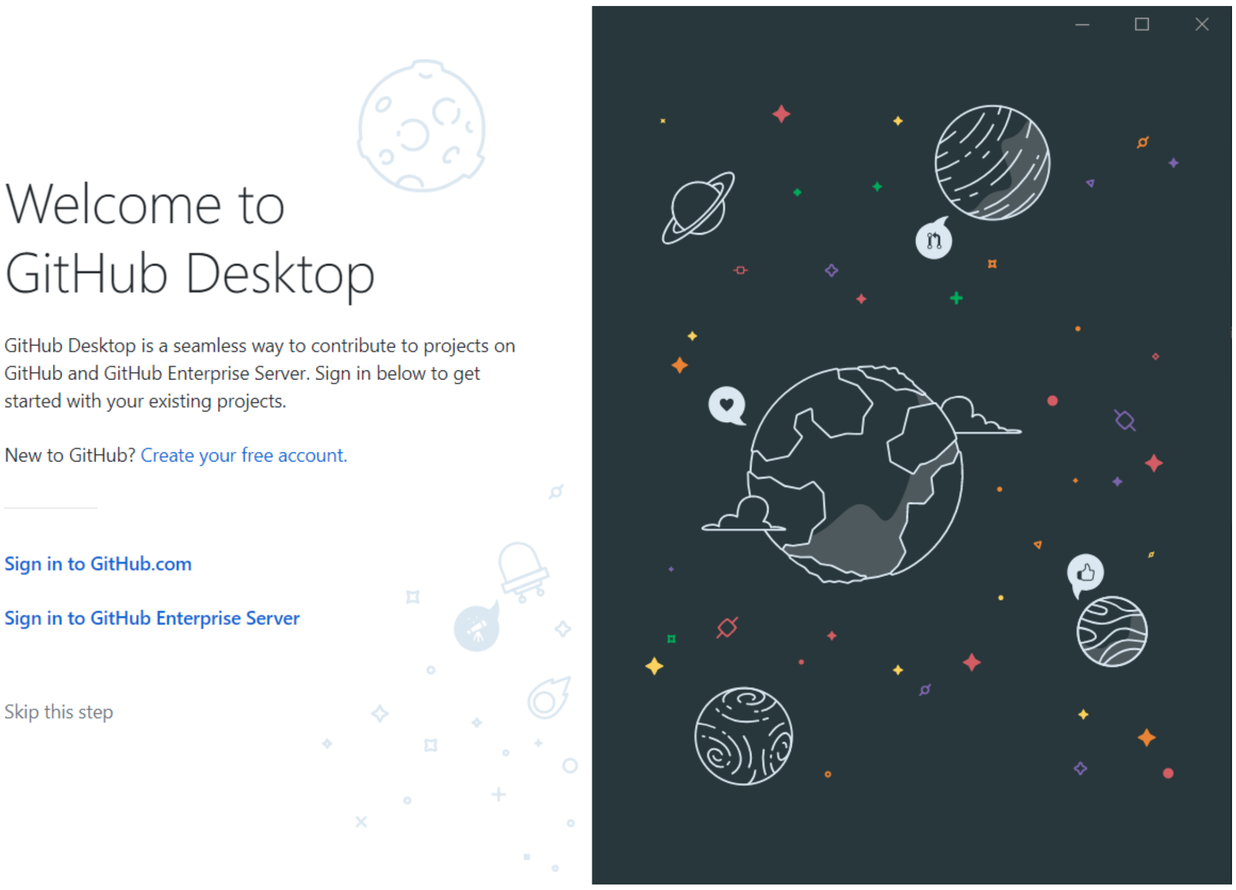
1. Download GitHub Desktop from https://desktop.github.com/. GitHub Desktop supports recent versions of Windows and macOS. For specific installation instructions for your operating system, see "Installing GitHub Desktop."



Focus on what matters instead of fighting with Git. Whether you're new to Git or a seasoned user, GitHub Desktop simplifies your development workflow.

Download for macOS or Windows (msi)
By downloading, you agree to the Open Source Applications Terms.

2. Launch GitHub Desktop and follow the initial welcome flow to sign into your GitHub account. You'll see a "Configure Git" step, where you can set your name and email address. To ensure your commits are correctly attributed to your GitHub account, use the email address associated with your GitHub account. For more information about commit attribution, see "Setting your commit email address."



Welcome to GitHub Desktop

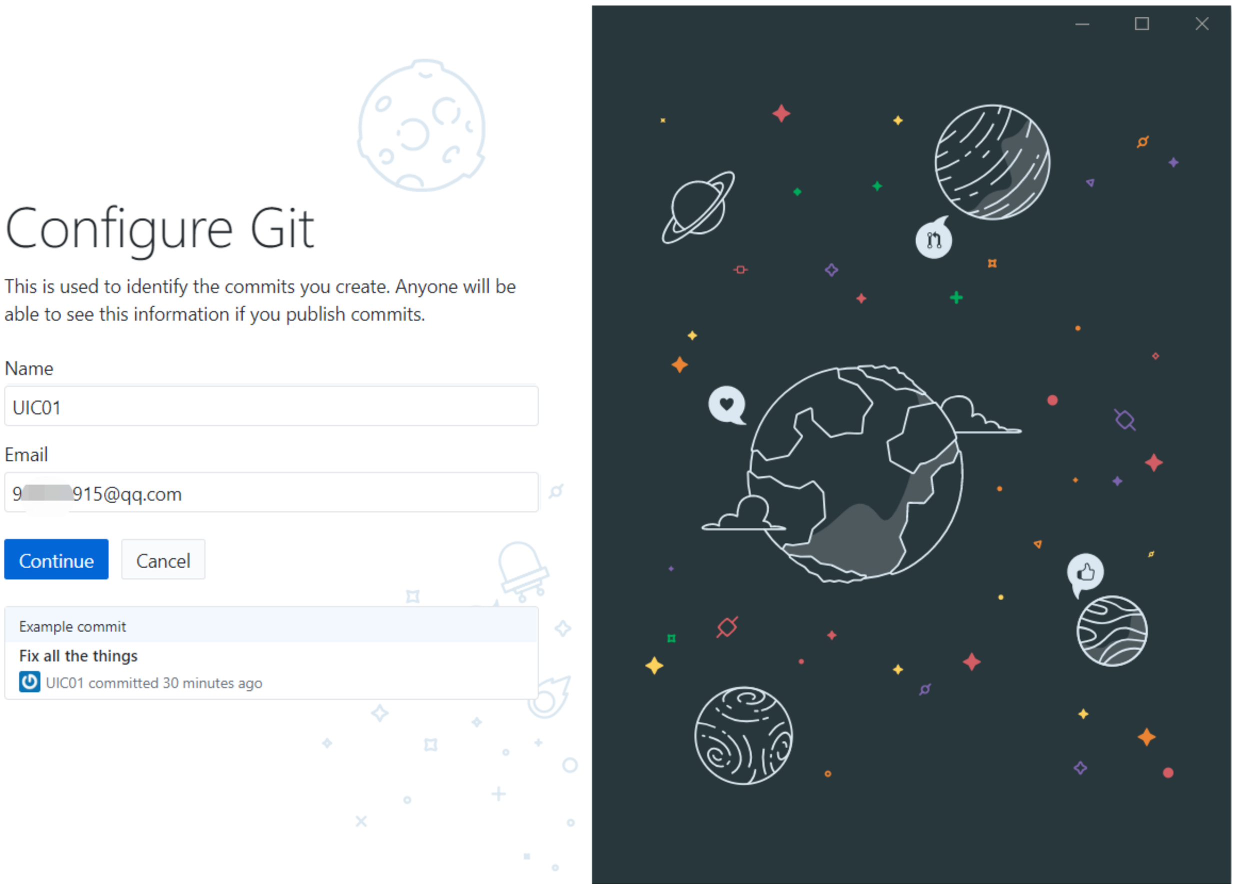
GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise Server. Sign in below to get started with your existing projects.

New to GitHub? Create your free account.

Sign in to GitHub.com

Sign in to GitHub Enterprise Server

Skip this step



Configure Git

This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.

Name

UIC01

Email

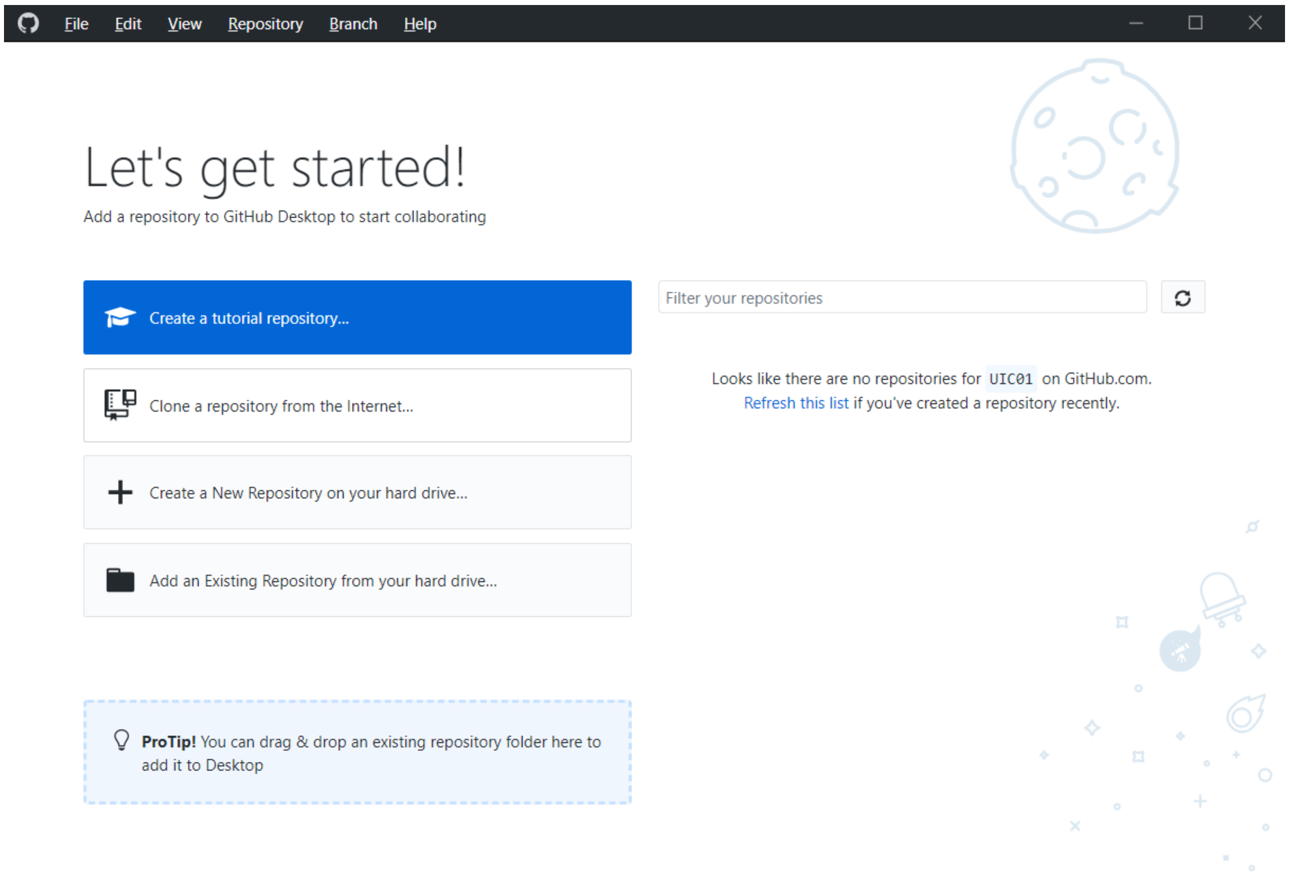
9[redacted]915@qq.com

Continue Cancel

Example commit

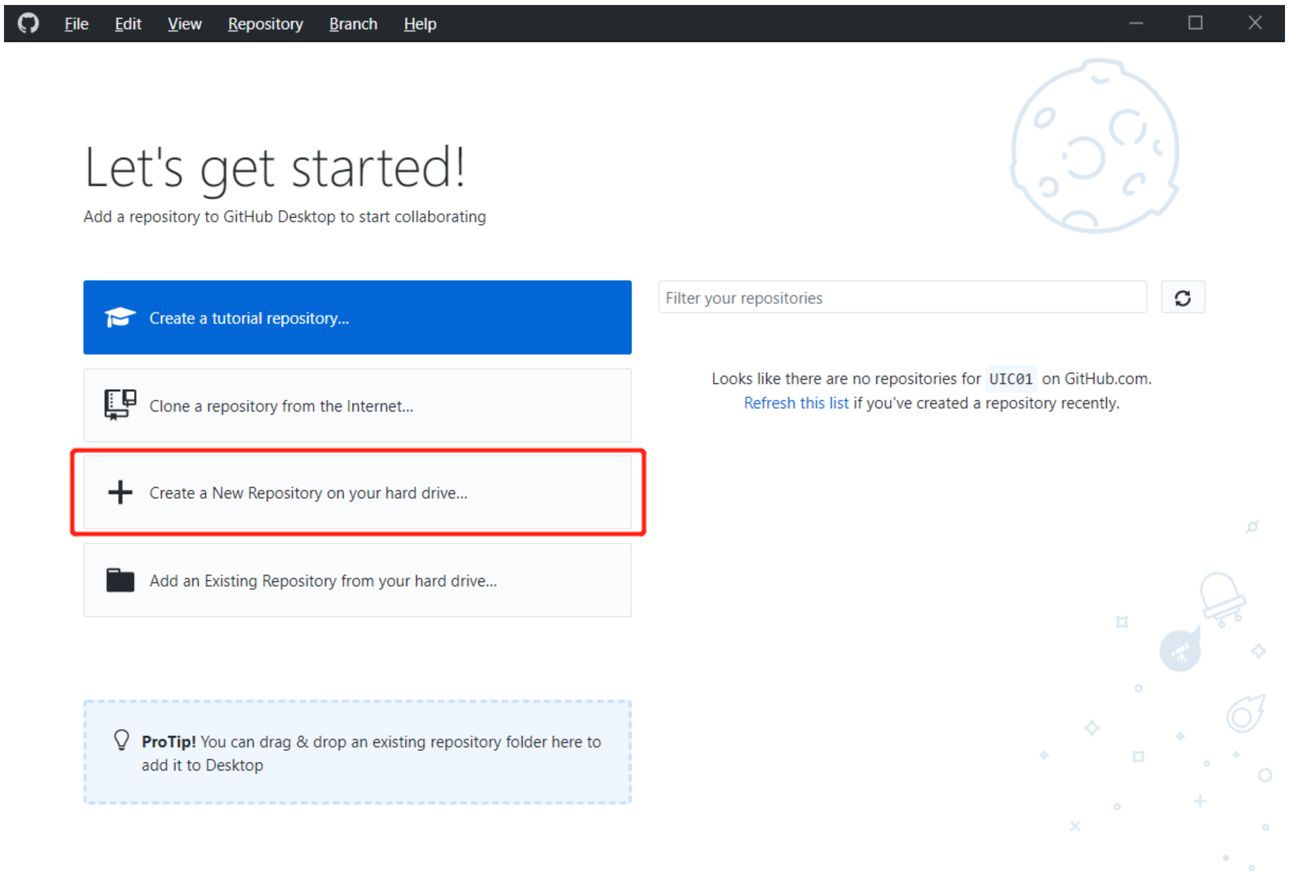
Fix all the things

UIC01 committed 30 minutes ago

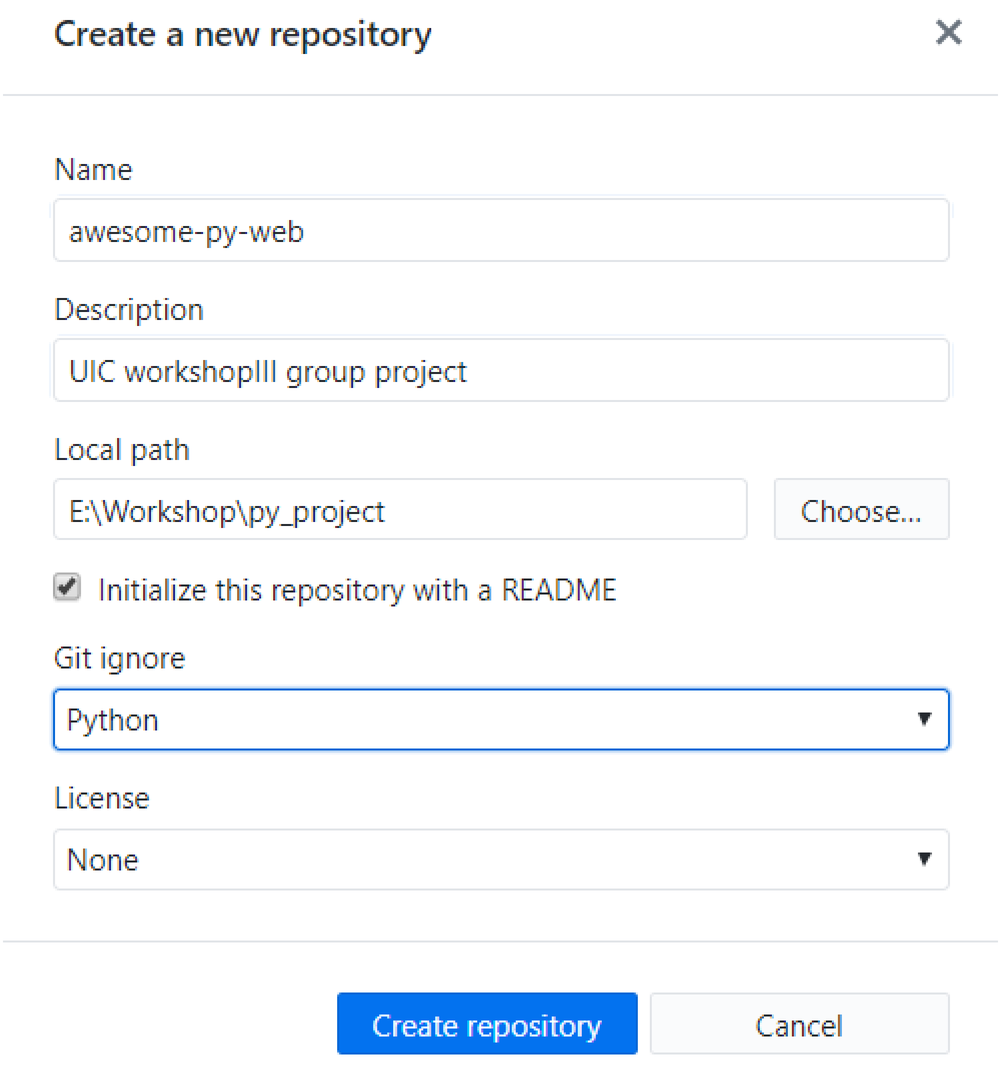

File
Edit
View
Repository
Branch
Help
Let's get started!
Add a repository to GitHub Desktop to start collaborating
Create a tutorial repository...
Clone a repository from the Internet...
Create a New Repository on your hard drive...
Add an Existing Repository from your hard drive...
ProTip! You can drag & drop an existing repository folder here to add it to Desktop
Filter your repositories
Looks like there are no repositories for UIC01 on GitHub.com.
Refresh this list if you've created a repository recently.

Step 2. Create a new repository

1. Click Create a New Repository on your Hard Drive....


File
Edit
View
Repository
Branch
Help
Let's get started!
Add a repository to GitHub Desktop to start collaborating
Create a tutorial repository...
Clone a repository from the Internet...
Create a New Repository on your hard drive...
Add an Existing Repository from your hard drive...
ProTip! You can drag & drop an existing repository folder here to add it to Desktop
Filter your repositories
Looks like there are no repositories for UIC01 on GitHub.com.
Refresh this list if you've created a repository recently.

2. To create a new repository, fill out the fields:


Create a new repository

Name: awesome-py-web

Description: UIC workshopIII group project

Local path: E:\Workshop\py_project

[✓] Initialize this repository with a README

Git ignore: Python

License: None

- "Name" defines the name of your repository both locally and on GitHub.
- "Description" is an optional field that you can use to provide more information about the purpose of your repository.
- "Local path" sets the location of your repository on your computer. By default, GitHub Desktop creates a GitHub folder inside your Documents folder to store your repositories, but you can choose any location on your computer. Your new repository will be a folder inside the chosen location. For example, if you name your repository Tutorial, a folder named Tutorial is created inside the folder you selected for your local path. GitHub Desktop remembers your chosen location the next time you create or clone a new repository.
- Initialize this repository with a README creates an initial commit with a README.md file. READMEs helps people understand the purpose of your project, so we recommend selecting this and filling it out with helpful information. When someone visits your repository on GitHub, the README is the first thing they'll see as they learn about your project. For more information, see "About READMEs."
- The Git ignore drop-down menu lets you add a custom file to ignore specific files in your local repository that you don't want to store in version control. If there's a a specific language or framework

that you'll be using, you can select an option from the available list. If you're just getting started, feel free to skip this selection. For more information, see "Ignoring files."

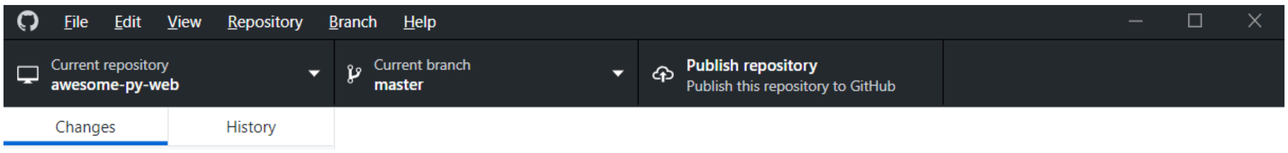
- The License drop-down menu lets you add an open-source license to a LICENSE file in your repository. You don't need to worry about adding a license right away. For more information about available open-source licenses and how to add them to your repository, see "Licensing a repository."

3. Click Create repository.

Step 3. Explore GitHub Desktop

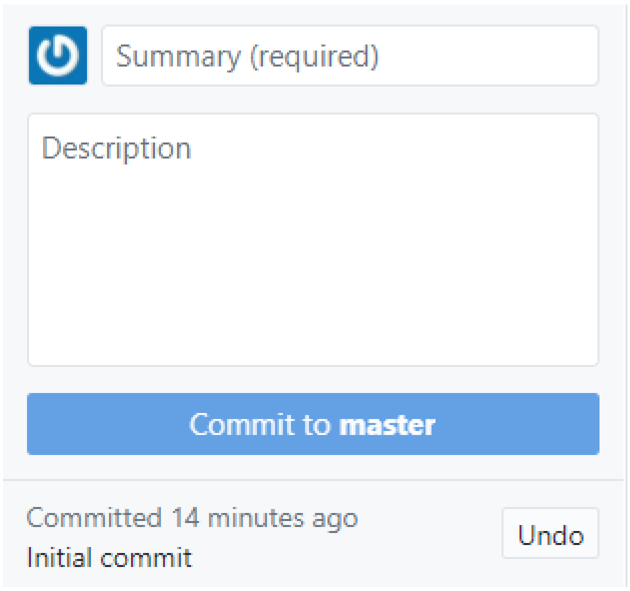
Now that you've created a repository, you'll see the file menu at the top of the screen. This is where you can access settings and actions that you can perform in GitHub Desktop. Most actions also have keyboard shortcuts to help you work more efficiently. For a full list of keyboard shortcuts, see "Keyboard shortcuts in GitHub Desktop."

1. Below the menu is a bar that shows the current state of your repository in GitHub Desktop:

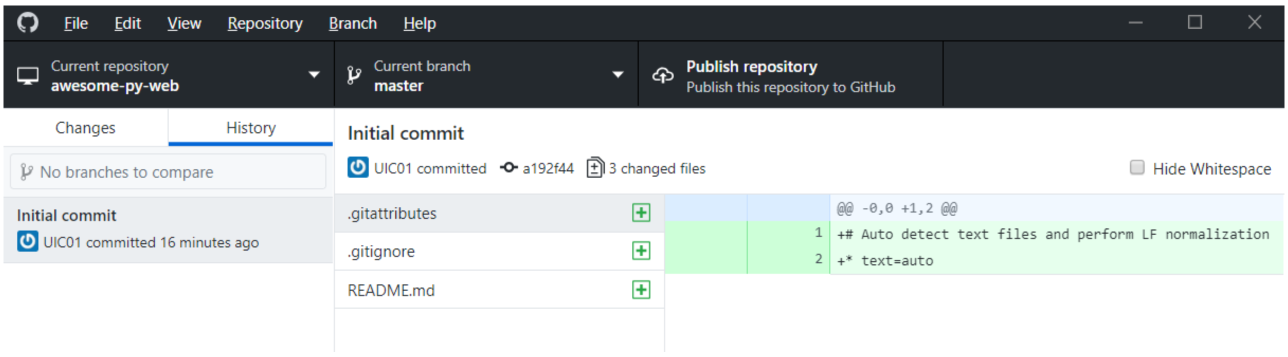


   - Current repository shows the name of the repository you're working on. You can click Current repository to switch to a different repository in GitHub Desktop.
   - Current branch shows the name of the branch you're working on. You can click Current branch to view all the branches in your repository, switch to a different branch, or create a new branch. Once you create pull requests in your repository, you can also view these by clicking on Current branch.
   - Publish repository appears because you haven't published your repository to GitHub yet, which you'll do later in the next step.

2. In the left sidebar, you'll find the Changes and History views.



- The Changes view shows changes you've made to files in your current branch but haven't committed to your local repository. At the bottom, you'll also notice a box with "Summary" and "Description" text boxes and a Commit to master button. This is where you'll commit new changes. The Commit button lets you know which branch you're committing your changes to.



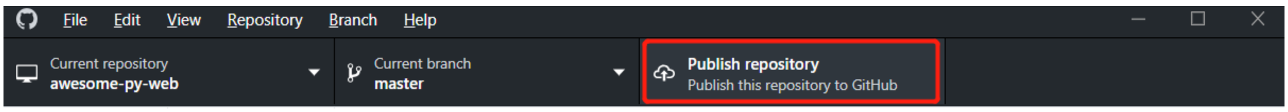
```
@@ -0,0 +1,2 @@
1 +# Auto detect text files and perform LF normalization
2 +* text=auto
```

- The History view shows the previous commits on the current branch of your repository. You should see an "Initial commit" that was created by GitHub Desktop when you created your repository. To the right of the commit, depending on the options you selected while creating your repository, you may see .gitattributes, .gitignore, LICENSE, or README files. You can click each file to see a diff for that file, which is the changes made to the file in that commit. The diff only shows the parts of the file that have changed, not the entire contents of the file.

Step 4. Push your repository to GitHub

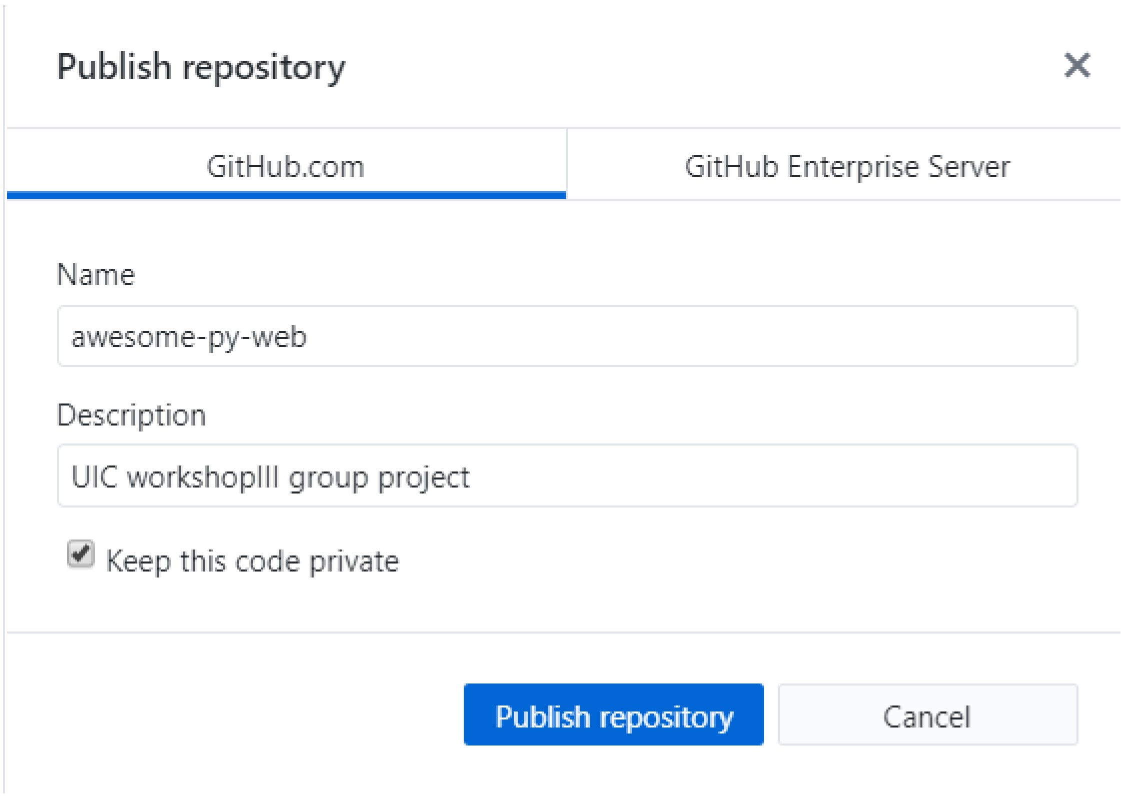
Currently, your repository only exists on your computer, and you're the only one who can access the repository. Publishing your repository to GitHub keeps it synchronized across multiple computers and team members on the same project. To publish the repository, you'll "push" it to GitHub, which makes it available on GitHub.com as well.

1. Click Publish repository.



   - You'll see a few familiar fields. "Name" and "Description" match the fields you completed when you created the repository.
   - You'll see the option to Keep this code private. Select this option if you don't want to share your code publicly with other users on GitHub.
   - The Organization dropdown, if present, lets you publish your repository to a specific organization that you belong to on GitHub. It's okay if you're not a member of an organization yet!

2. Click Publish repository.


Publish repository

GitHub.com | GitHub Enterprise Server

Name: awesome-py-web

Description: UIC workshopIII group project

[✓] Keep this code private

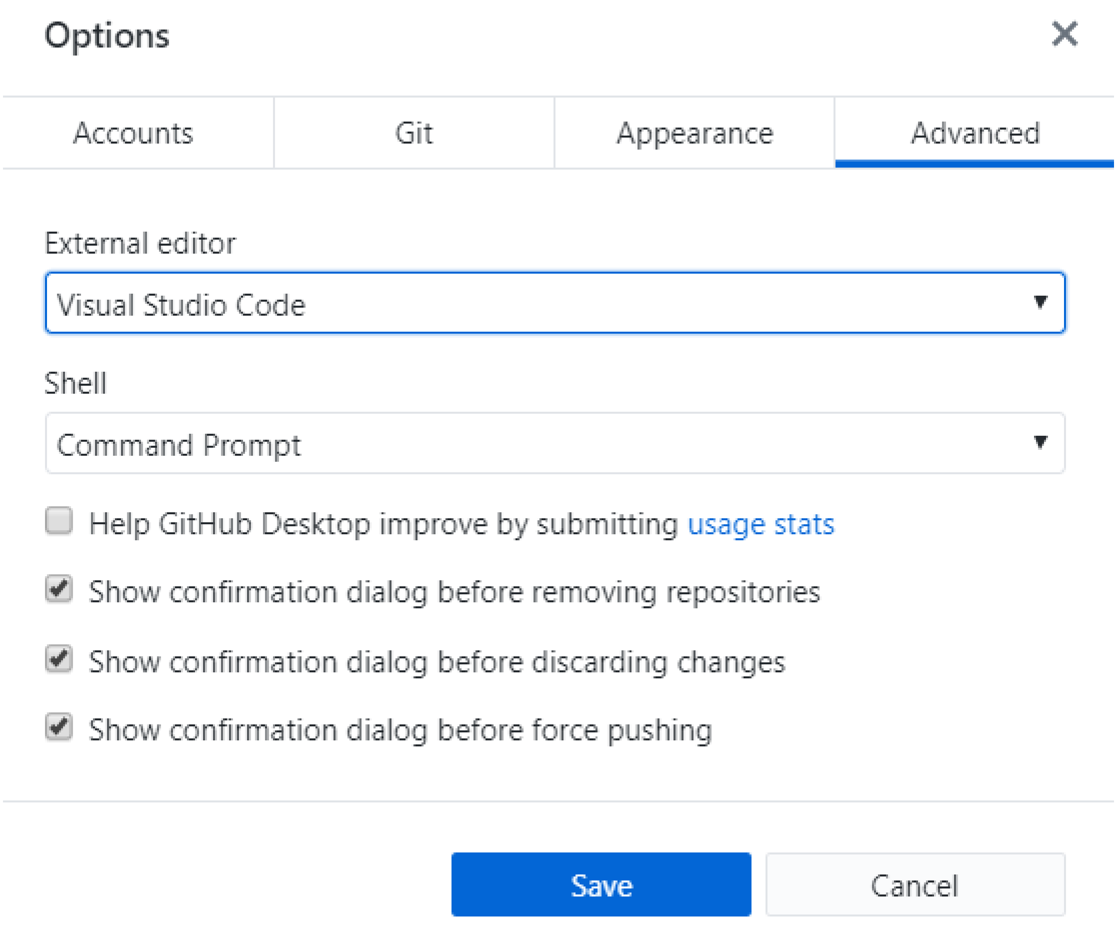
3. You can access the repository on GitHub.com from within GitHub Desktop. In the file menu, click Repository, then click View on GitHub. This will take you directly to the repository in your default browser.

Step 5. Set up a text editor

To reduce time spent setting up your development environment, you can launch a number of text editors and integrated development environments (IDEs) directly from GitHub Desktop. From a repository in GitHub Desktop, you can seamlessly open the project folder in your favorite text editor.

1. Click File, then click Options, and then click Advanced.

2. Use the External editor dropdown menu and select an editor from the list. You should see any editors you have installed in the list. If you don't see any editors, install a supported editor like Visual Studio Code. For a list of supported editors, see "Open External Editor" integration in the GitHub Desktop repository.



Options

Accounts | Git | Appearance | Advanced

External editor: Visual Studio Code

Shell: Command Prompt

- [ ] Help GitHub Desktop improve by submitting usage stats
- [x] Show confirmation dialog before removing repositories
- [x] Show confirmation dialog before discarding changes
- [x] Show confirmation dialog before force pushing

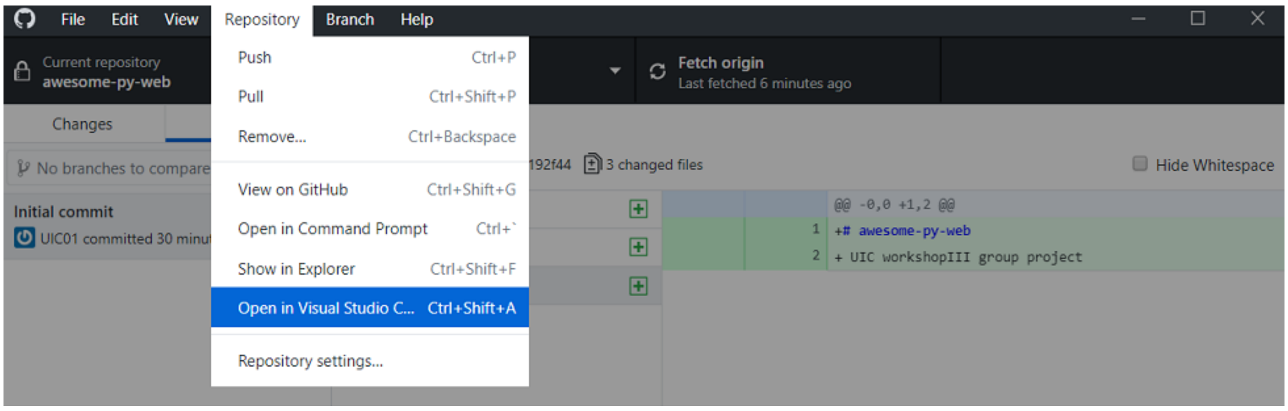
Save | Cancel

3. If you installed a new editor, restart GitHub Desktop to make the editor available in the External editor dropdown menu.

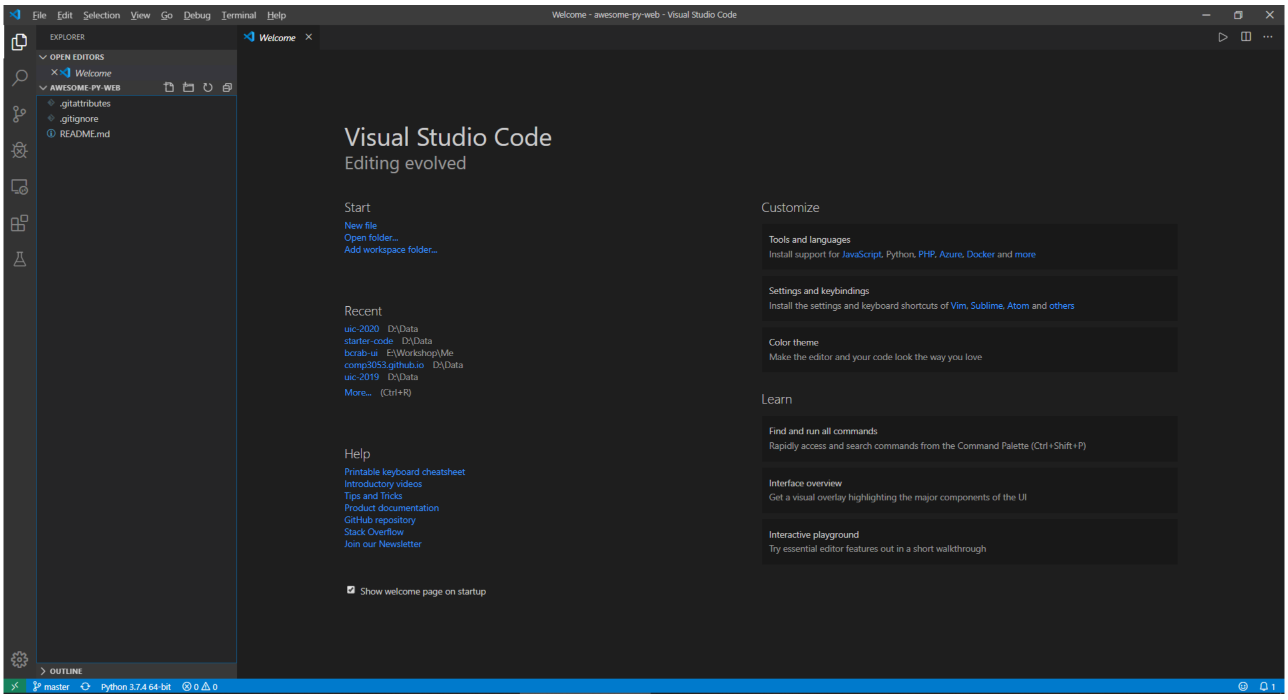
Step 6. Make, commit, and push changes

Now that you've configured a default editor, you're ready to make changes to your project and start crafting the first commit of your own to your repository.

1. To launch your external editor from within GitHub Desktop, click Repository and then click Open in EDITOR.



```
@@ -0,0 +1,2 @@
1 +# awesome-py-web
2 + UIC workshopIII group project
```



Visual Studio Code
Editing evolved

Start
New file
Open folder...
Add workspace folder...

Recent
uic-2020 D:\Data
starter-code D:\Data
bcrab-ui E:\Workshop\Me
comp3053.github.io D:\Data
uic-2019 D:\Data
More... (Ctrl+R)

Help
Printable keyboard cheatsheet
Introductory videos
Tips and Tricks
Product documentation
GitHub repository
Stack Overflow
Join our Newsletter

[✓] Show welcome page on startup

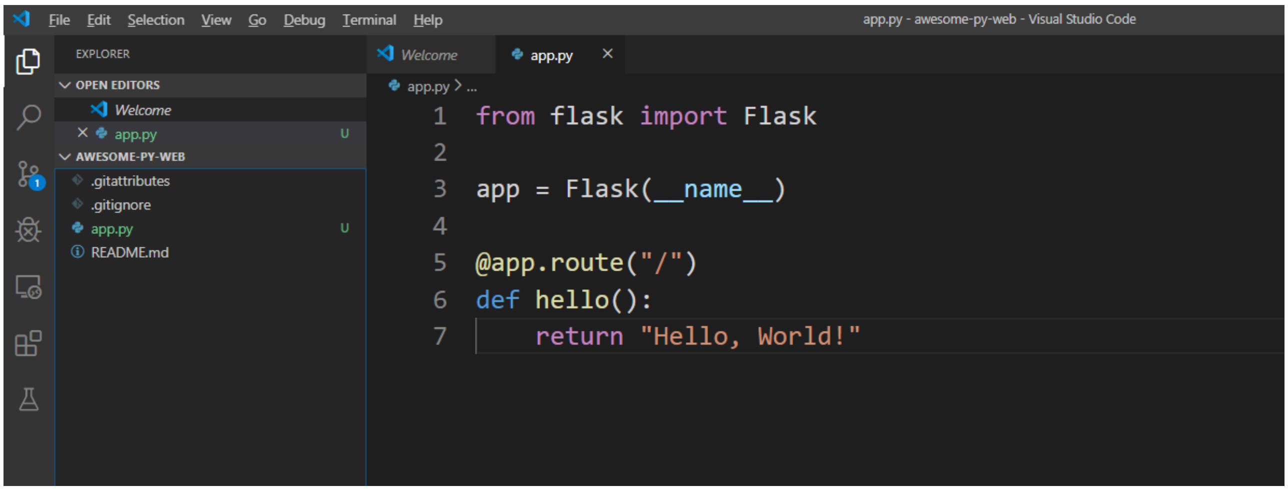
Customize
Tools and languages
Install support for JavaScript, Python, PHP, Azure, Docker and more
Settings and keybindings
Install the settings and keyboard shortcuts of Vim, Sublime, Atom and others
Color theme
Make the editor and your code look the way you love

Learn
Find and run all commands
Rapidly access and search commands from the Command Palette (Ctrl+Shift+P)
Interface overview
Get a visual overlay highlighting the major components of the UI
Interactive playground
Try essential editor features out in a short walkthrough

2. Create a flask simple example

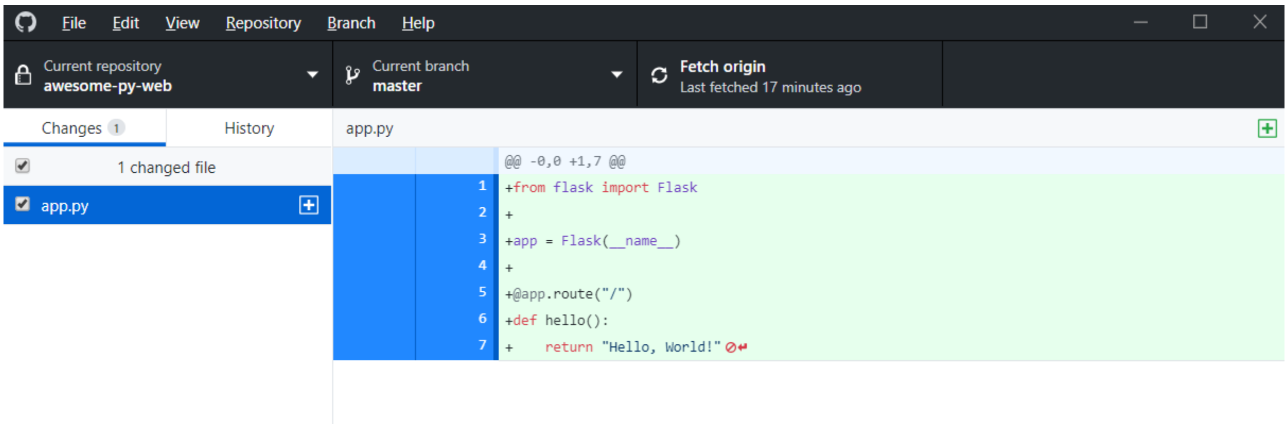
   app.py

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"
```



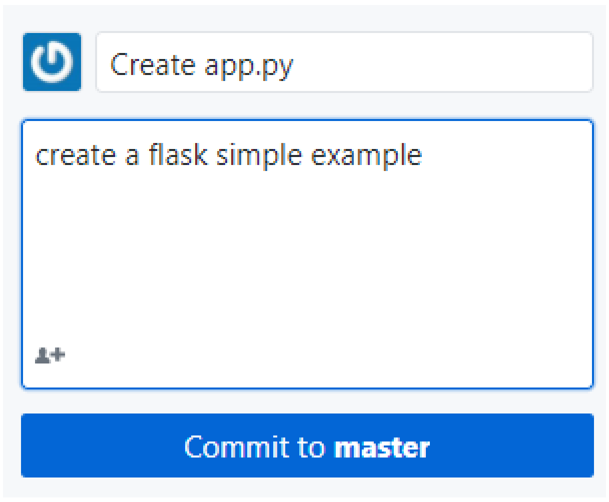
```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"
```

3. Switch from your text editor back to GitHub Desktop and navigate to the Changes tab. In the file list, you should see your app.py file. The checkmark by the app.py file indicates that the changes you've made to the file will be part of the commit you make. In the future, you might make changes to multiple files but only want to commit the changes you've made to some of the files. GitHub Desktop allows you to select specific changes you want to commit.



```
@@ -0,0 +1,7 @@
+from flask import Flask
+
+app = Flask(__name__)
+
+@app.route("/")
+def hello():
+    return "Hello, World!"
```

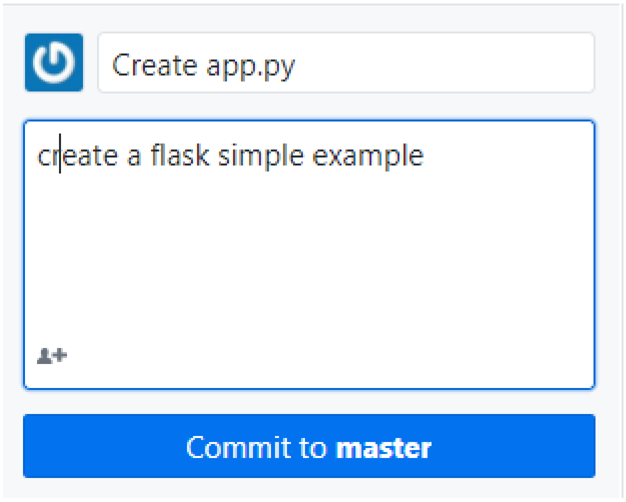
4. At the bottom of the Changes list, enter a commit message. To the right of your profile picture, type a short description of the commit. Since we're changing the app.py file, "Add information about purpose of project" would be a good commit summary. Below the summary, you'll see a "Description" text field, where you can type a longer description of the changes in the commit, which is helpful when looking back at the history of a project and understanding why changes were made. Since you're making a basic update of a app.py file, you can skip the description.



Create app.py

create a flask simple example

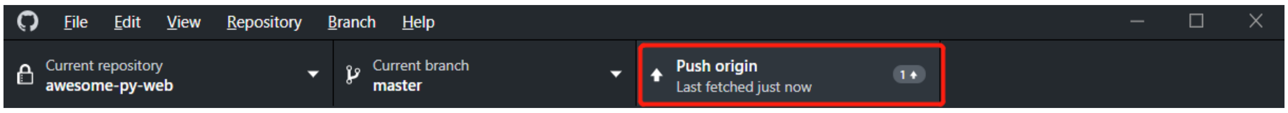
5. Click Commit to master. The commit button shows your current branch, which in this case ismaster, so you can be sure to commit to the branch you want.



Create app.py

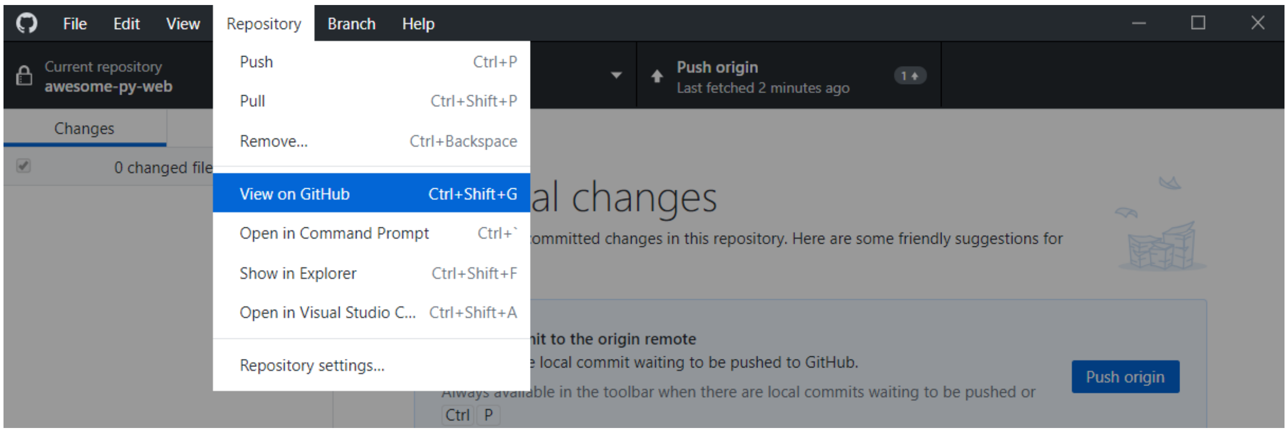
create a flask simple example

6. To push your changes to the remote repository on GitHub, click Push origin.

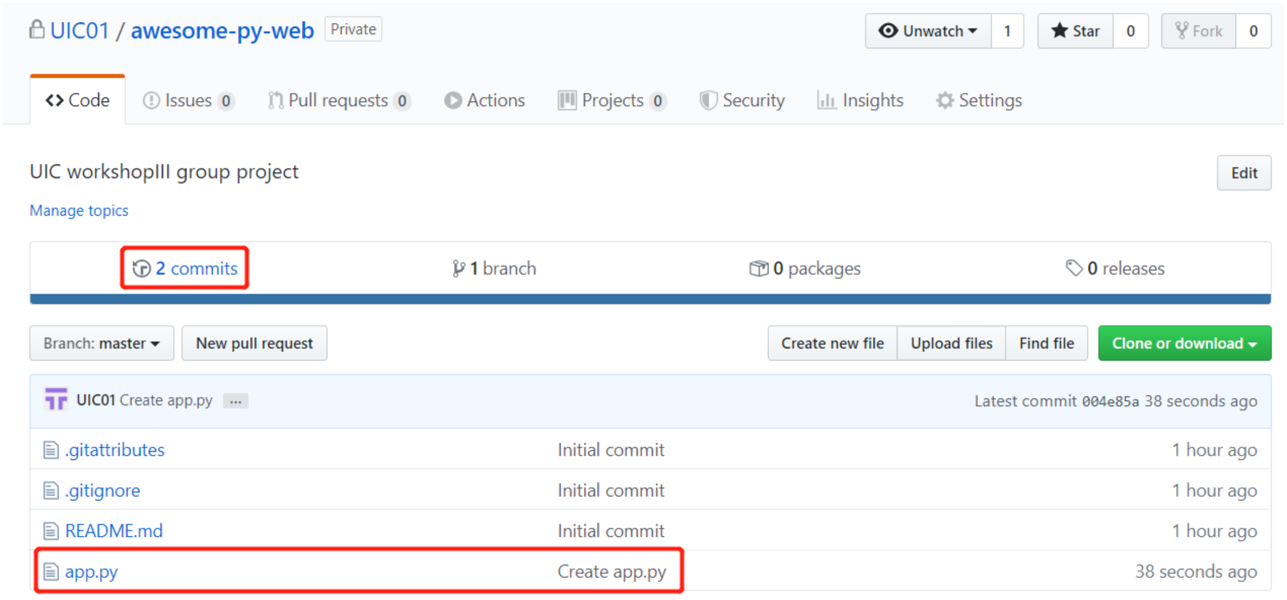


- Remember the Publish button that you used to publish your repository to GitHub? It should now say Push origin instead, with a 1 next to it, indicating that there is one commit that has not been pushed up to GitHub.
- The "origin" in Push origin means that we're pushing changes to the remote called origin, which in this case is your project's repository on GitHub.com. Until you push any new commits to GitHub, there will be differences between your project's repository on your computer and your project's repository on GitHub.com. This allows you to work locally and only push your work to GitHub.com when you're ready.

7. In the open area next to the Changes tab, you'll see suggestions for things you can do next. To open the repository on GitHub in your browser, click View on GitHub.


File
Edit
View
Repository
Branch
Help
Current repository
awesome-py-web
Push origin
Last fetched 2 minutes ago
1
Changes
0 changed file
Push Ctrl+P
Pull Ctrl+Shift+P
Remove... Ctrl+Backspace
View on GitHub Ctrl+Shift+G
Open in Command Prompt Ctrl+`
Show in Explorer Ctrl+Shift+F
Open in Visual Studio C... Ctrl+Shift+A
Repository settings...
al changes
ommitted changes in this repository. Here are some friendly suggestions for
it to the origin remote
local commit waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or
Ctrl P
Push origin

8. In your browser, click 2 commits. You'll see a list of the commits in this repository on GitHub. The first commit should be the commit you just made in GitHub Desktop!



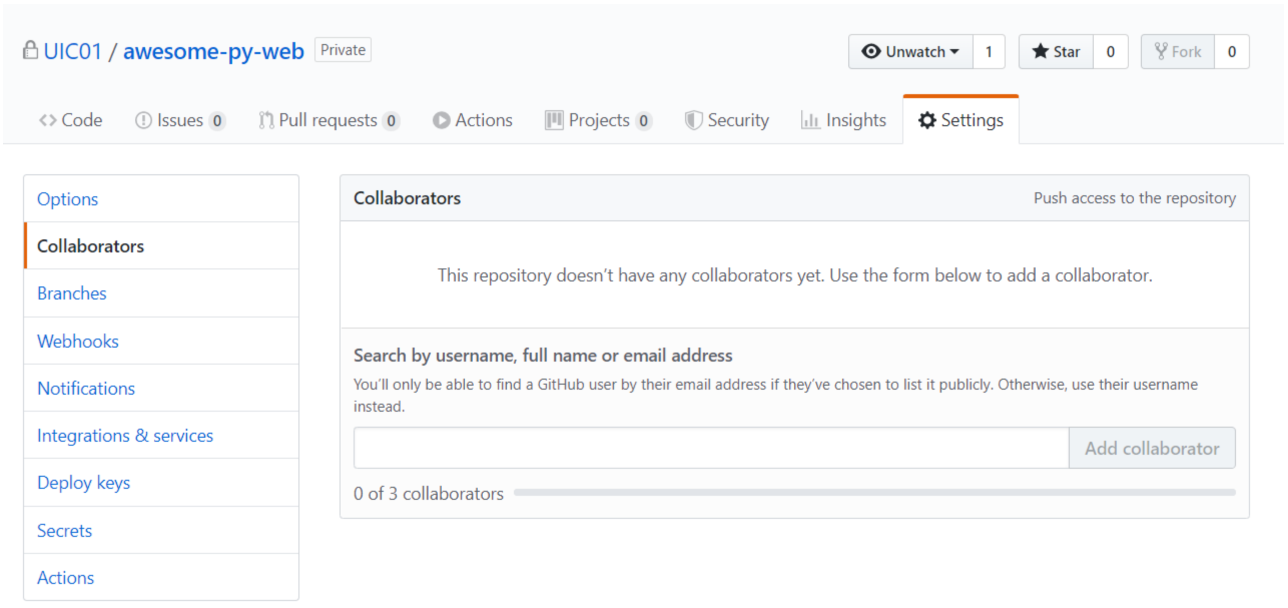
UIC workshopIII group project

UIC01 Create app.py ... Latest commit 004e85a 38 seconds ago

.gitattributes	Initial commit	1 hour ago
.gitignore	Initial commit	1 hour ago
README.md	Initial commit	1 hour ago
app.py	Create app.py	38 seconds ago

Step 7. Invite collaborators

1. In your browser and navigate to Settings and choose the collaborations tab.



Collaborators — Push access to the repository

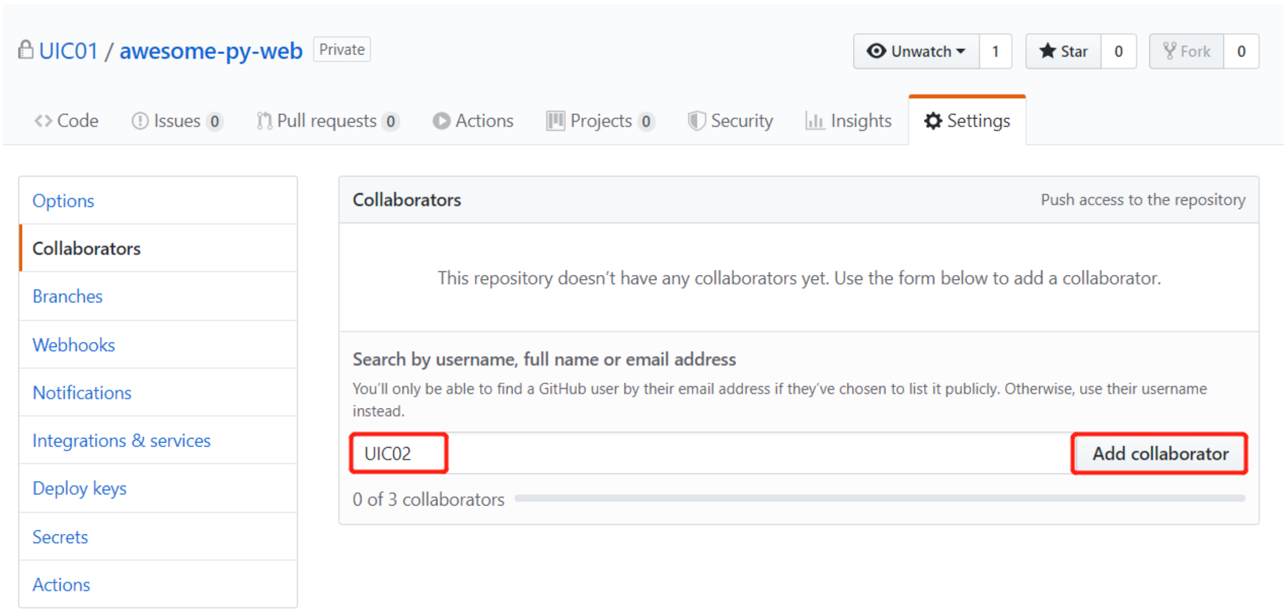
This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

0 of 3 collaborators

2. Add a new collaborator into your repository.



Collaborators — Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

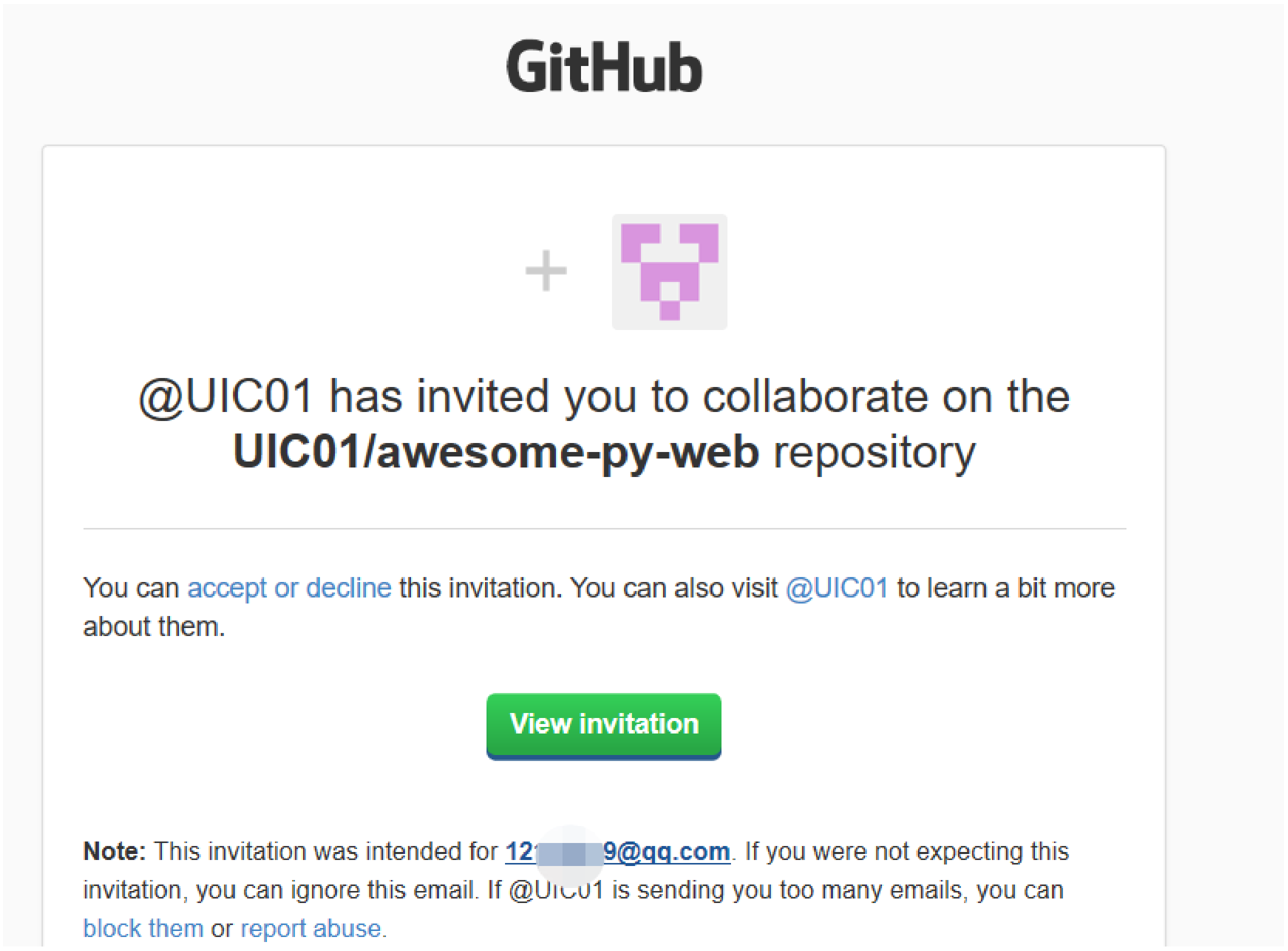
Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

UIC02

0 of 3 collaborators

3. Collaborator (example here said UIC02) will receive an invitation email and accept it.



@UIC01 has invited you to collaborate on the **UIC01/awesome-py-web** repository

You can accept or decline this invitation. You can also visit @UIC01 to learn a bit more about them.

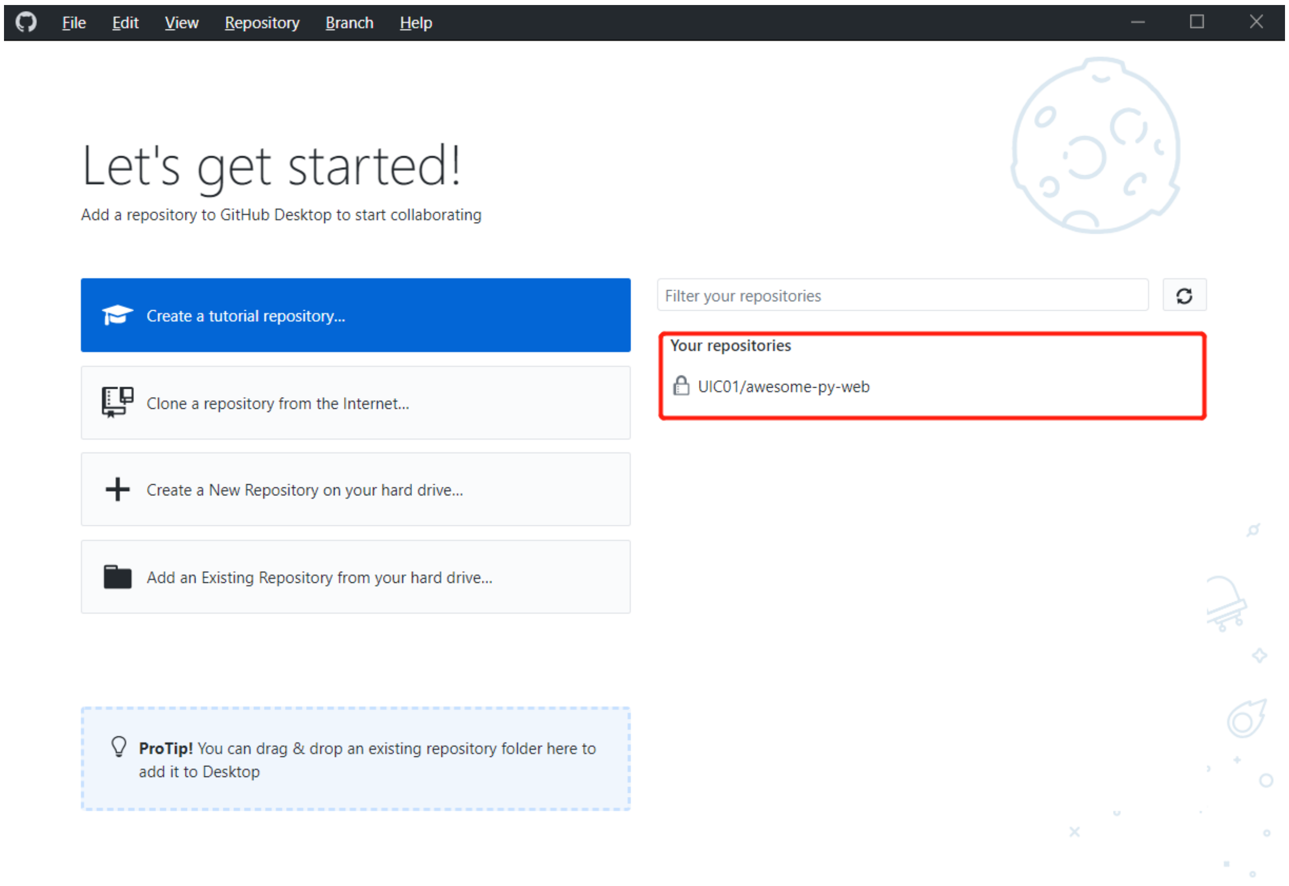
Note: This invitation was intended for 12[redacted]9@qq.com. If you were not expecting this invitation, you can ignore this email. If @UIC01 is sending you too many emails, you can block them or report abuse.

Step 8. Work with collaborators

Be noted, the following procedures are under the collaborator ***UIC02***.

Fetch the awesome-py-web repository from GitHub.

1. GitHub Desktop shows your repositories list.


File
Edit
View
Repository
Branch
Help

Let's get started!

Add a repository to GitHub Desktop to start collaborating

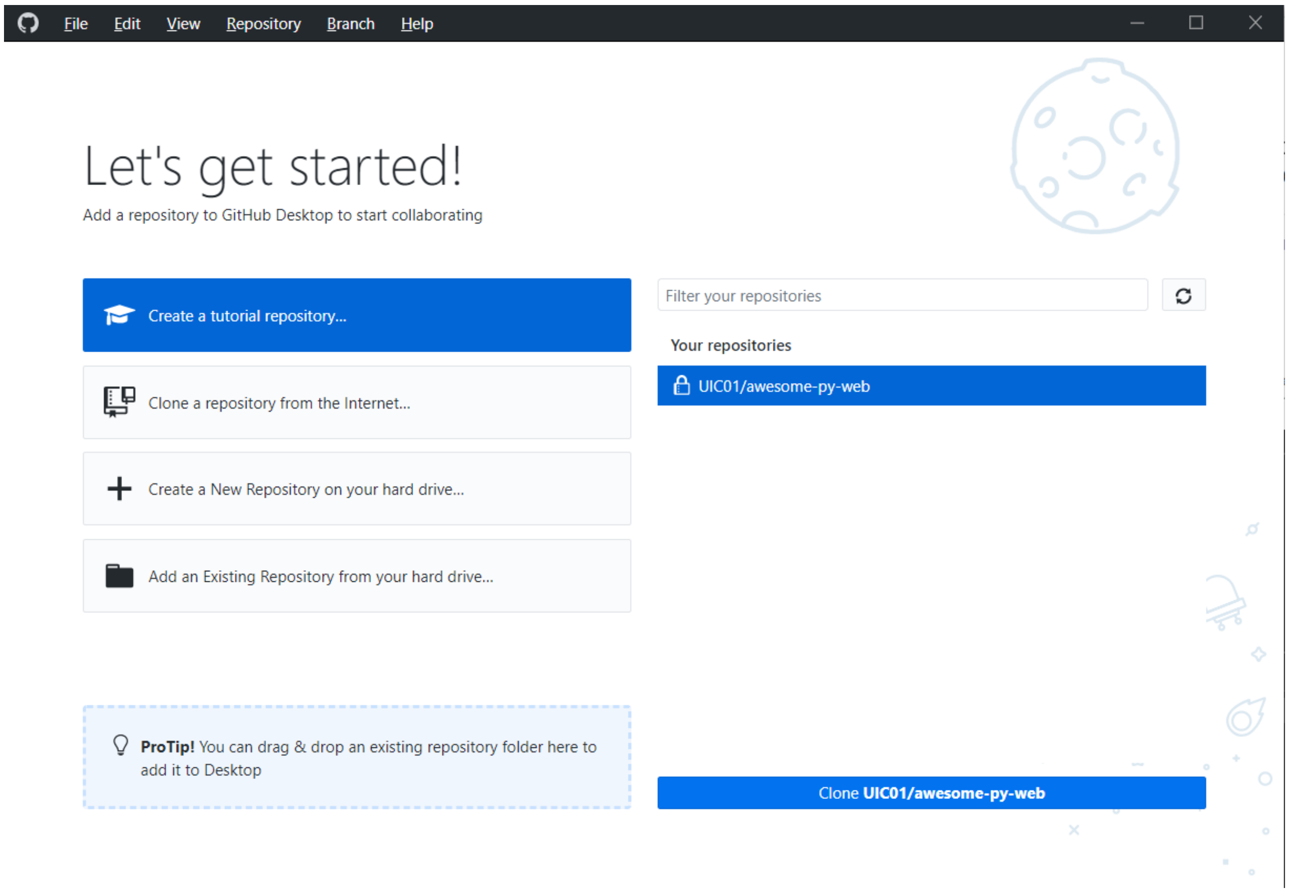
Create a tutorial repository...
Clone a repository from the Internet...
Create a New Repository on your hard drive...
Add an Existing Repository from your hard drive...
Filter your repositories

Your repositories

UIC01/awesome-py-web

ProTip! You can drag & drop an existing repository folder here to add it to Desktop

2. Select the awesome-py-web and clone to your local machine.


File
Edit
View
Repository
Branch
Help

Let's get started!

Add a repository to GitHub Desktop to start collaborating

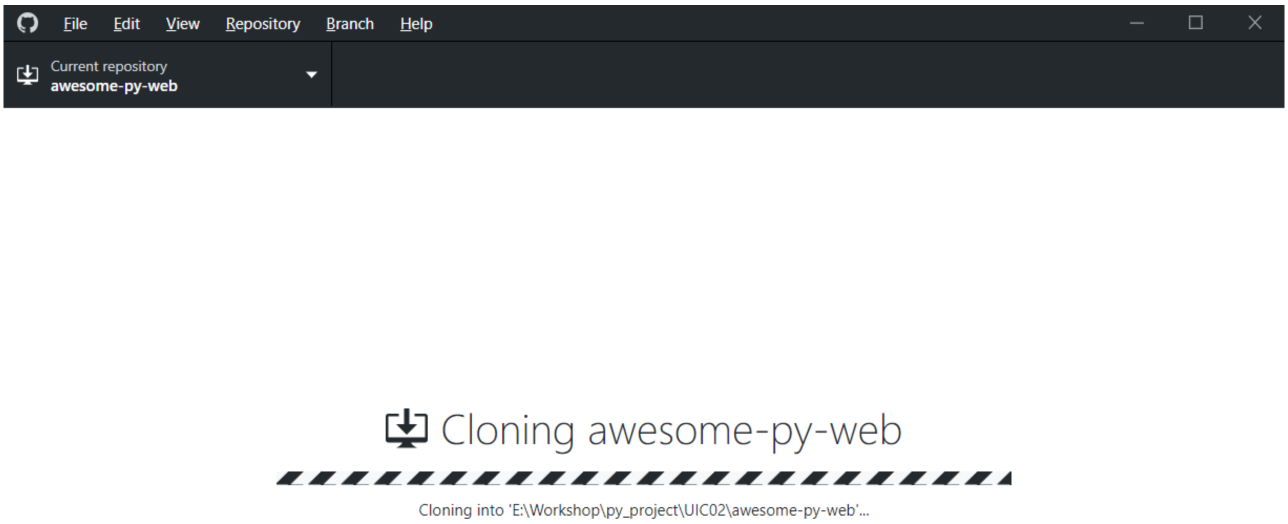
Create a tutorial repository...
Clone a repository from the Internet...
Create a New Repository on your hard drive...
Add an Existing Repository from your hard drive...
Filter your repositories

Your repositories

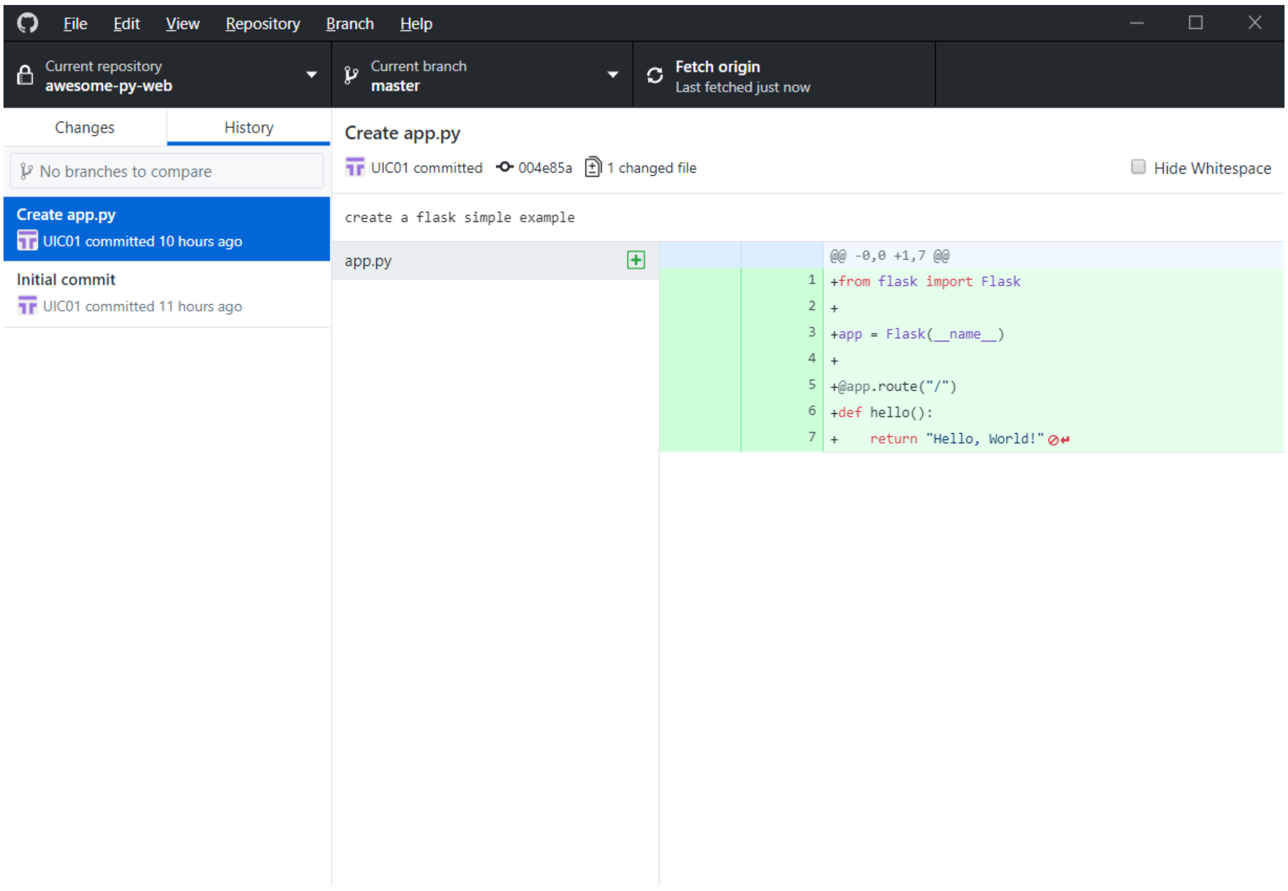
UIC01/awesome-py-web

ProTip! You can drag & drop an existing repository folder here to add it to Desktop

Clone **UIC01/awesome-py-web**



Cloning into 'E:\Workshop\py_project\UIC02\awesome-py-web'...



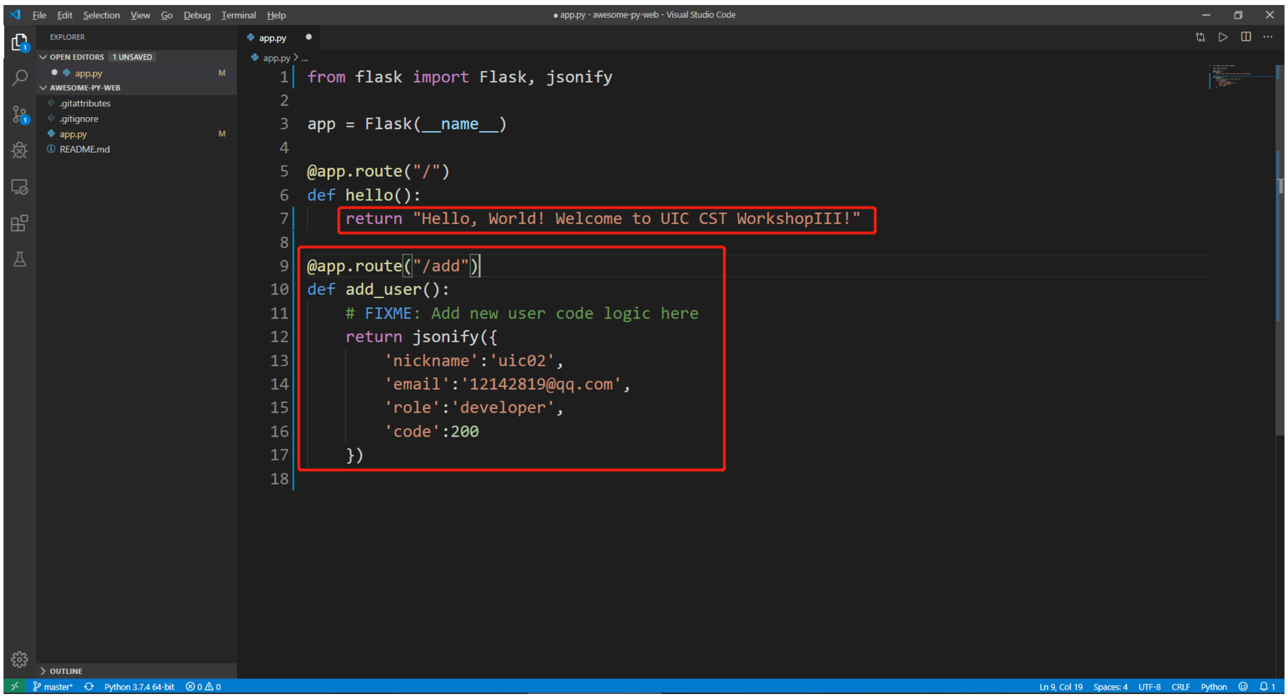
create a flask simple example

```
@@ -0,0 +1,7 @@
+from flask import Flask
+
+app = Flask(__name__)
+
+@app.route("/")
+def hello():
+    return "Hello, World!"
```

3. Edit app.py with IDE

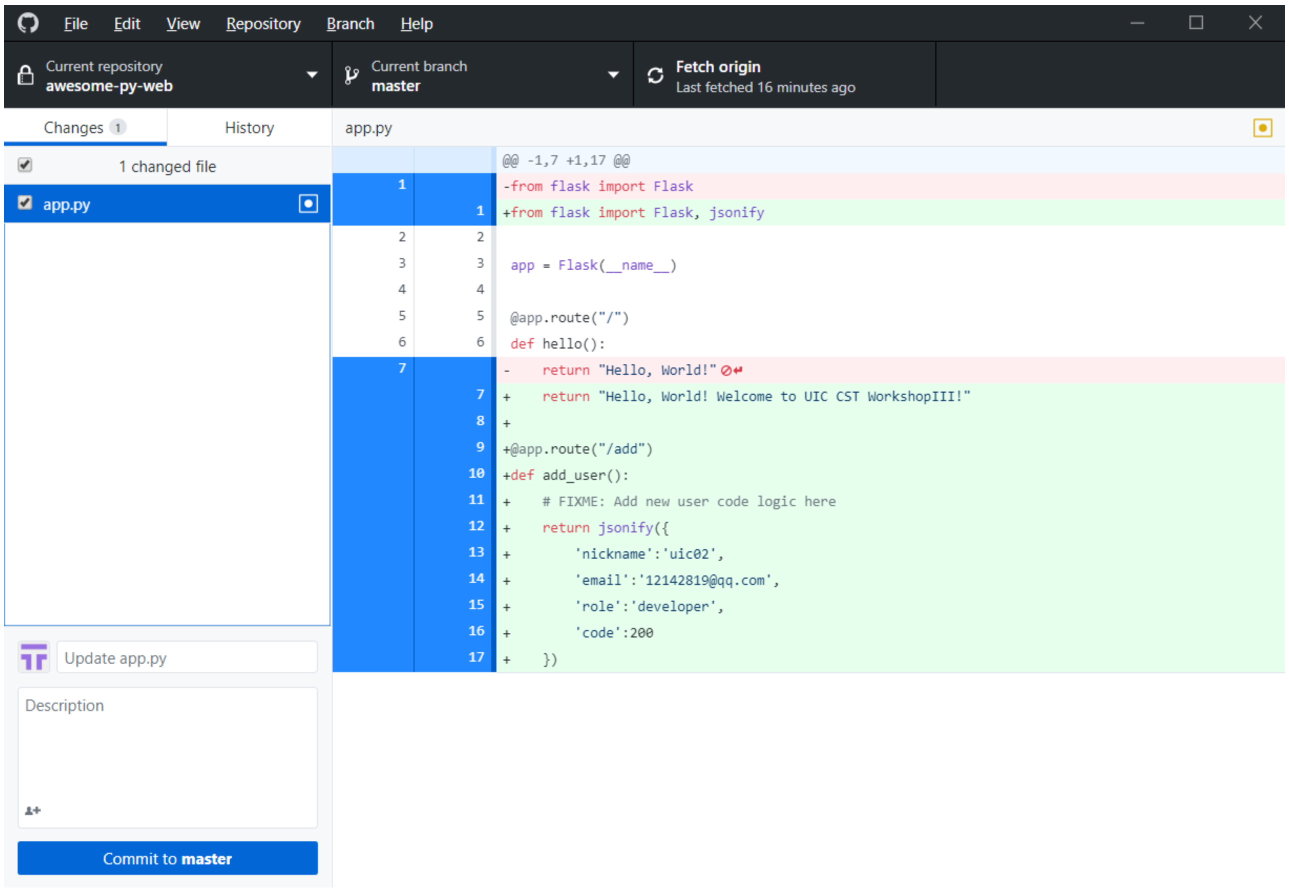
```
# app.py
from flask import Flask, jsonify
```

```
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World! Welcome to UIC CST WorkshopIII!"

@app.route("/add")
def add_user():
    # FIXME: Add new user code logic here
    return jsonify({
        'nickname':'uic02',
        'email':'12142819@qq.com',
        'role':'developer',
        'code':200
    })
```


File Edit Selection View Go Debug Terminal Help
• app.py - awesome-py-web - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
app.py
AWESOME-PY-WEB
.gitattributes
.gitignore
app.py
README.md
OUTLINE
master*
Python 3.7.4 64-bit
Ln 9, Col 19 Spaces: 4 UTF-8 CRLF Python

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World! Welcome to UIC CST WorkshopIII!"

@app.route("/add")
def add_user():
    # FIXME: Add new user code logic here
    return jsonify({
        'nickname':'uic02',
        'email':'12142819@qq.com',
        'role':'developer',
        'code':200
    })

```

4. Check the changes on GitHub Desktop



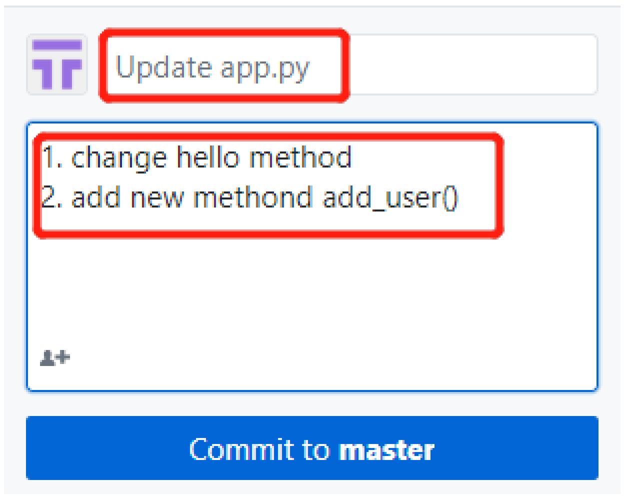
app.py

```
@@ -1,7 +1,17 @@
-from flask import Flask
+from flask import Flask, jsonify

 app = Flask(__name__)

 @app.route("/")
 def hello():
-    return "Hello, World!"
+    return "Hello, World! Welcome to UIC CST WorkshopIII!"
+
+@app.route("/add")
+def add_user():
+    # FIXME: Add new user code logic here
+    return jsonify({
+        'nickname':'uic02',
+        'email':'12142819@qq.com',
+        'role':'developer',
+        'code':200
+    })
```

Update app.py

Description

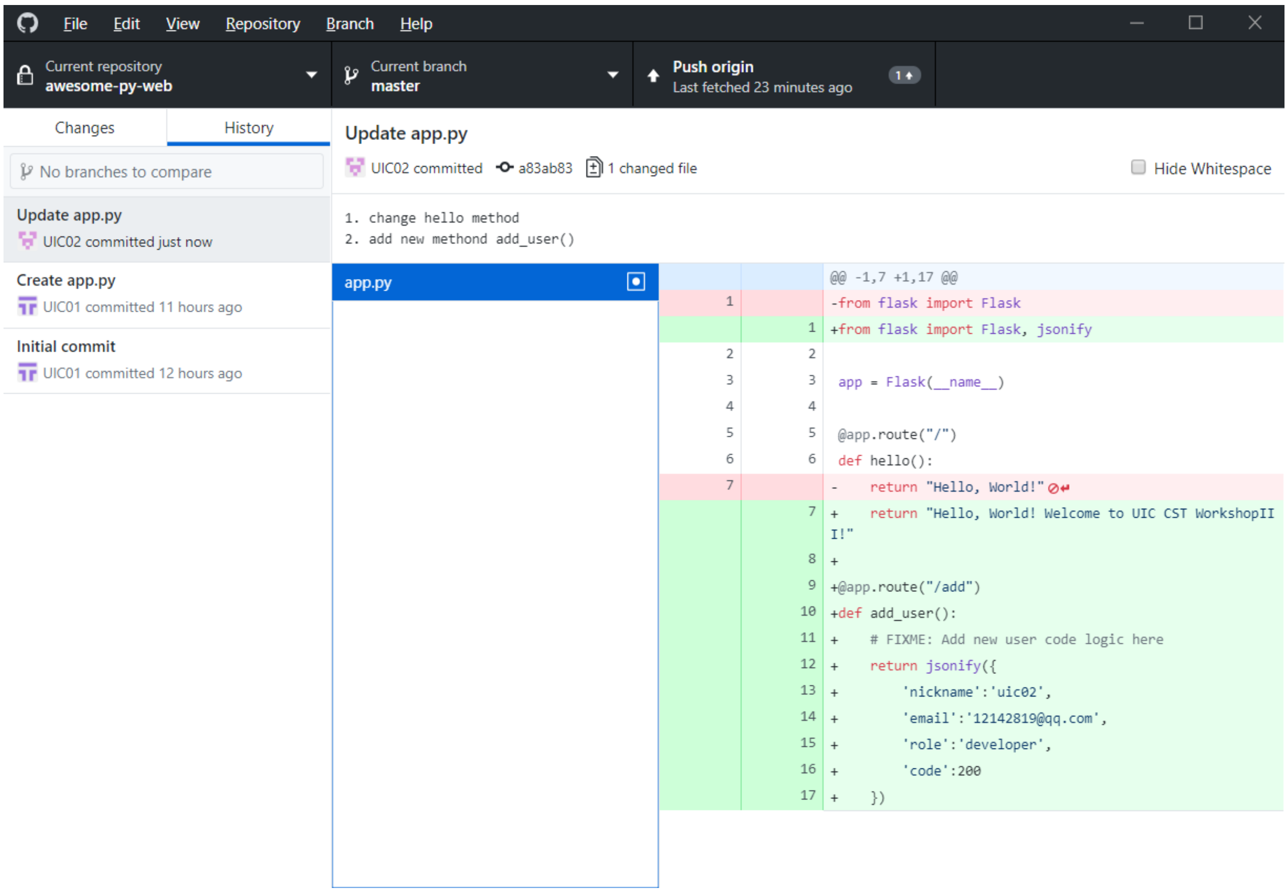
5. Commit to your local repository



Update app.py

1. change hello method
2. add new methond add_user()

6. Navigate to History tab and will see one more commit by UIC02.

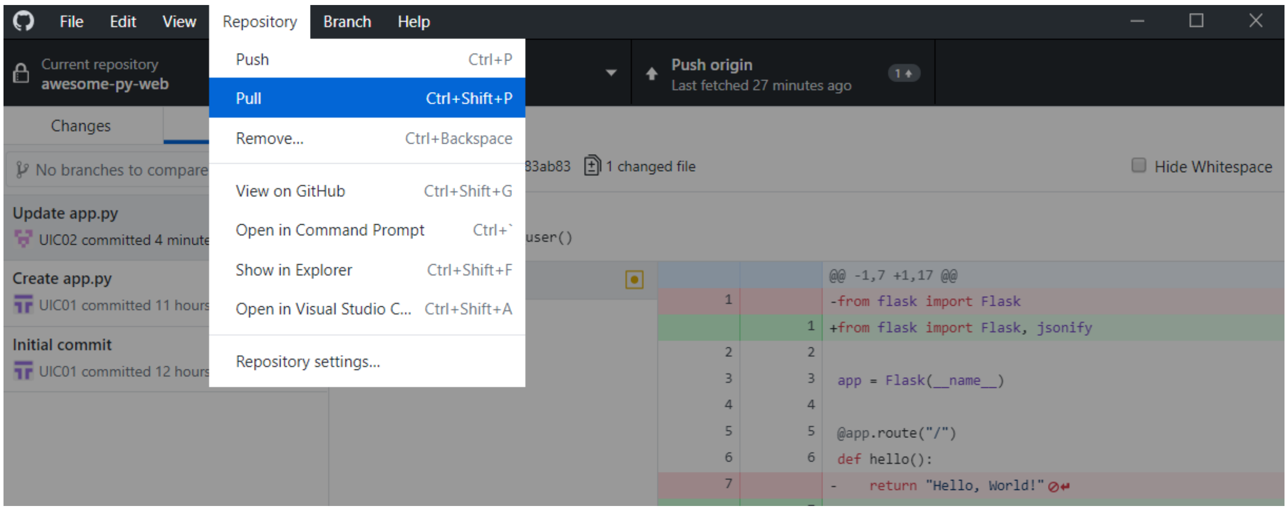


```
1. change hello method
2. add new methond add_user()
```

```
@@ -1,7 +1,17 @@
-from flask import Flask
+from flask import Flask, jsonify

 app = Flask(__name__)

 @app.route("/")
 def hello():
-    return "Hello, World!"
+    return "Hello, World! Welcome to UIC CST WorkshopII I!"
+
+@app.route("/add")
+def add_user():
+    # FIXME: Add new user code logic here
+    return jsonify({
+        'nickname':'uic02',
+        'email':'12142819@qq.com',
+        'role':'developer',
+        'code':200
+    })
```

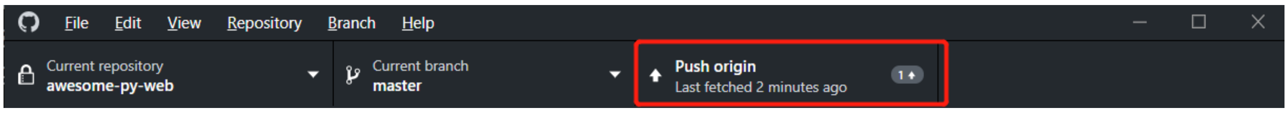
7. Syncing your branch.

   Be noted, before push to GitHub, pull from GitHub first to make it synchronization.

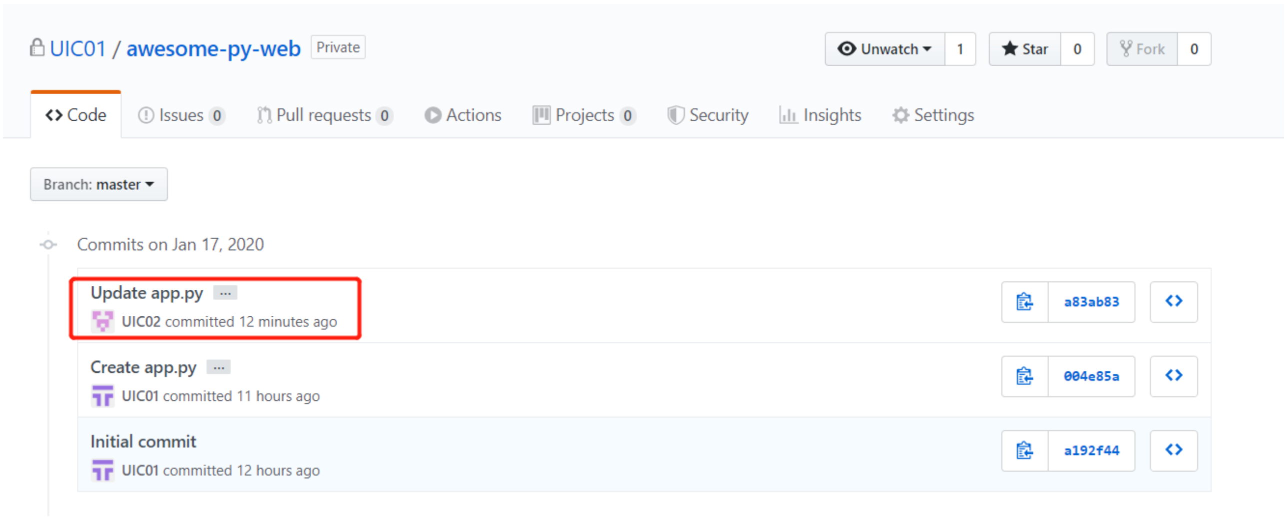


```
@@ -1,7 +1,17 @@
-from flask import Flask
+from flask import Flask, jsonify

 app = Flask(__name__)

 @app.route("/")
 def hello():
-    return "Hello, World!"
```

8. Push to GitHub.



9. In the browser, see the commits.



Commits on Jan 17, 2020

Update app.py
UIC02 committed 12 minutes ago
a83ab83

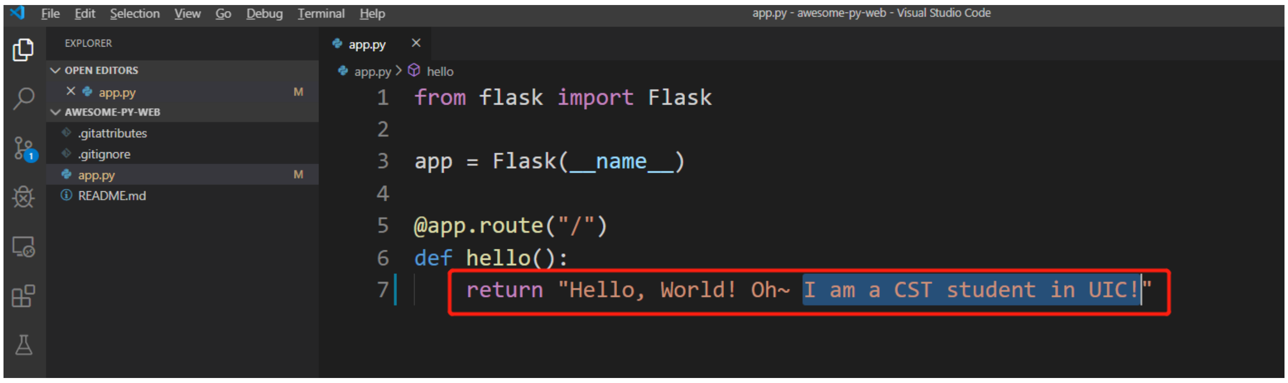
Create app.py
UIC01 committed 11 hours ago
004e85a

Initial commit
UIC01 committed 12 hours ago
a192f44

Step 9. Solve the conflicts

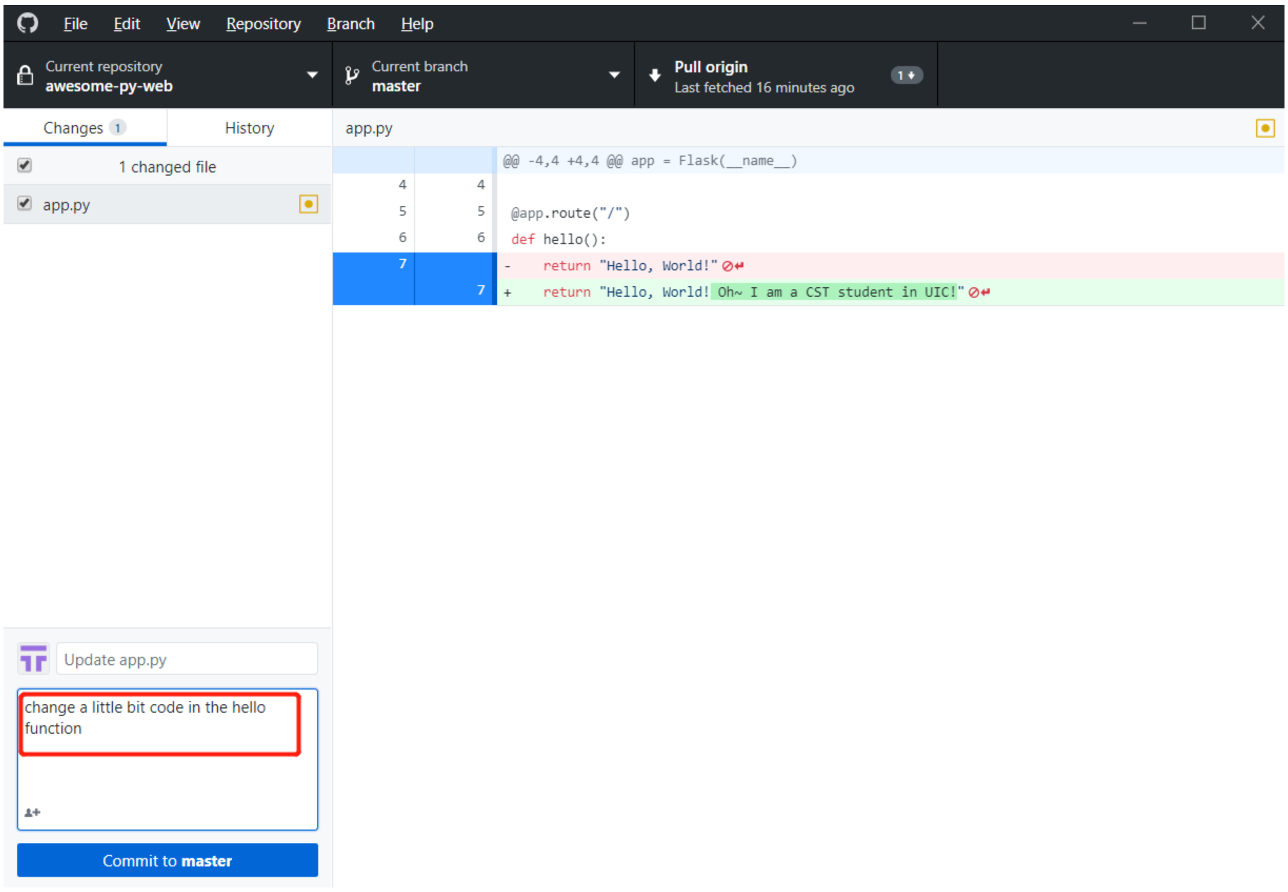
Be noted, now switch back to collaborator **UIC01**.

1. Now the UIC01 is going to change the code in the hello function.



```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World! Oh~ I am a CST student in UIC!"
```

2. Commit to local repository.

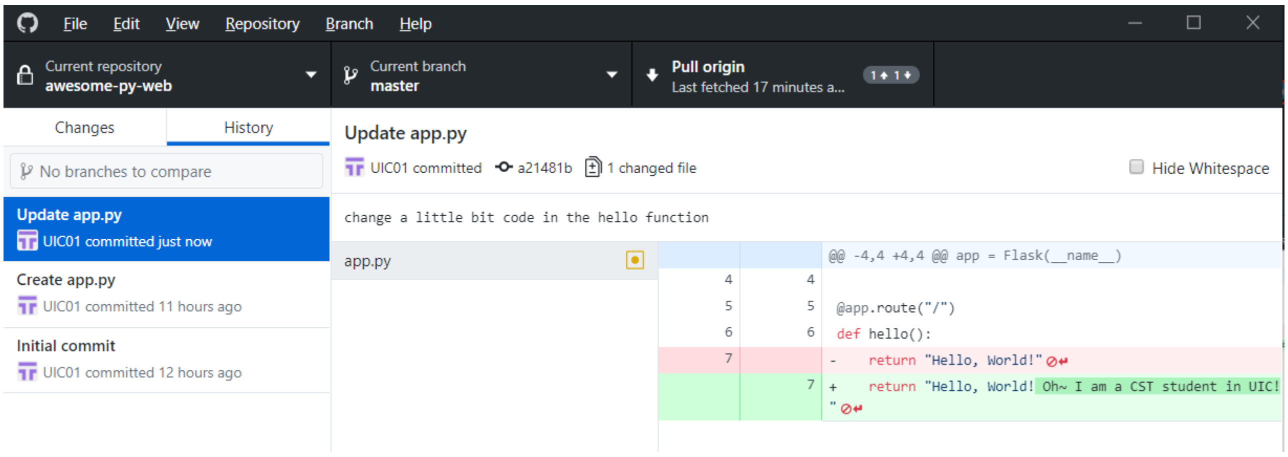


Update app.py

change a little bit code in the hello function

```
@@ -4,4 +4,4 @@ app = Flask(__name__)

 @app.route("/")
 def hello():
-    return "Hello, World!"
+    return "Hello, World! Oh~ I am a CST student in UIC!"
```

3. Check in the History tab.



Update app.py
UIC01 committed just now

Create app.py
UIC01 committed 11 hours ago

Initial commit
UIC01 committed 12 hours ago

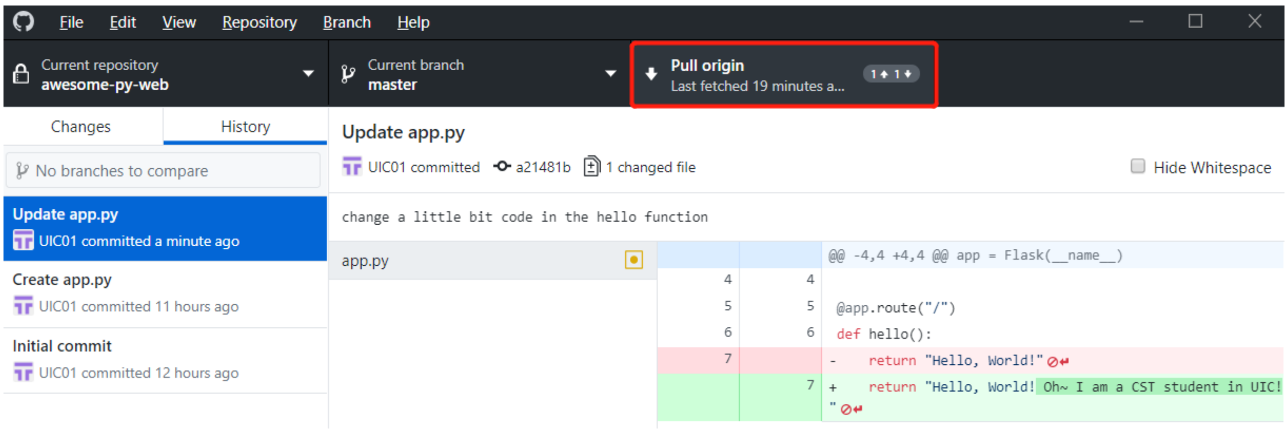
Update app.py
UIC01 committed a21481b 1 changed file

change a little bit code in the hello function

app.py

```
@@ -4,4 +4,4 @@ app = Flask(__name__)

 @app.route("/")
 def hello():
-    return "Hello, World!"
+    return "Hello, World! Oh~ I am a CST student in UIC!
"
```

4. Pull from GitHub.



Update app.py
UIC01 committed a minute ago

Create app.py
UIC01 committed 11 hours ago

Initial commit
UIC01 committed 12 hours ago

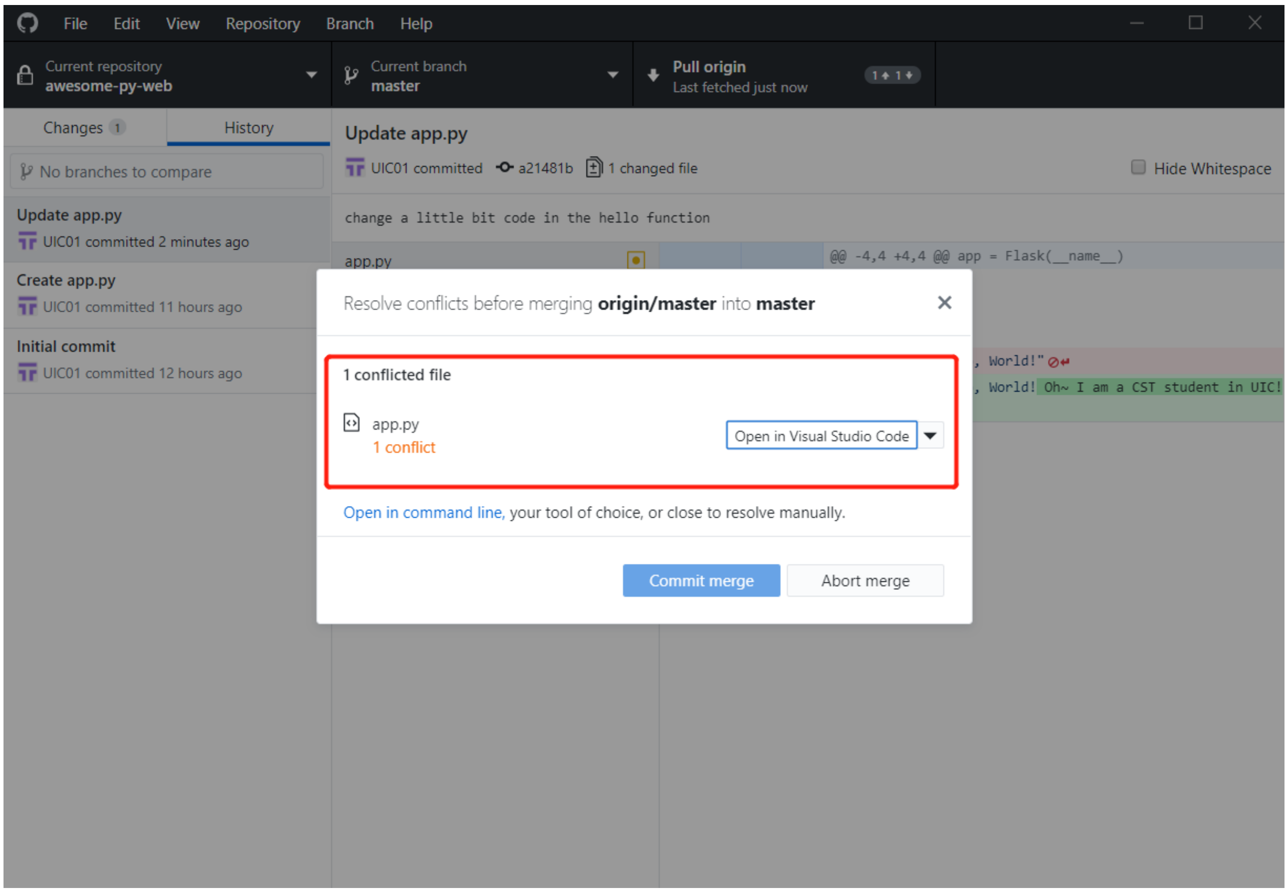
Update app.py
UIC01 committed a21481b 1 changed file

change a little bit code in the hello function

app.py

```
@@ -4,4 +4,4 @@ app = Flask(__name__)

 @app.route("/")
 def hello():
-    return "Hello, World!"
+    return "Hello, World! Oh~ I am a CST student in UIC!
"
```

5. Conflicts appear.


File Edit View Repository Branch Help
Current repository awesome-py-web
Current branch master
Pull origin Last fetched just now
Changes 1
History
No branches to compare
Update app.py
UIC01 committed 2 minutes ago
Create app.py
UIC01 committed 11 hours ago
Initial commit
UIC01 committed 12 hours ago
Update app.py
UIC01 committed a21481b 1 changed file
Hide Whitespace
app.py

change a little bit code in the hello function

Resolve conflicts before merging **origin/master** into **master**

1 conflicted file

app.py
1 conflict

Open in Visual Studio Code

Open in command line, your tool of choice, or close to resolve manually.

Commit merge | Abort merge

6. Fix the conflicts with IDE.

   Be noted, user should choose their favorite editor. Different editor may provide different approaches to fix conflicts. Please refere to your editor user manual. In this example, the author is using VS Code editor.

   If you follow my previous steps, you should see the result like this in editor:

```
<<<<<<< HEAD
    return "Hello, World! Oh~ I am a CST student in UIC!"
=======
    return "Hello, World! Welcome to UIC CST WorkshopIII!"

@app.route("/add")
def add_user():
    # FIXME: Add new user code logic here
    return jsonify({
        'nickname':'uic02',
        'email':'12142819@qq.com',
        'role':'developer',
        'code':200
    })
>>>>>>> a83ab83d40d8c9ffaf4eca57cd0f4bfb83a00fd7
```

   - The code between `<<<<<<< HEAD` and `=======` is your current local code.

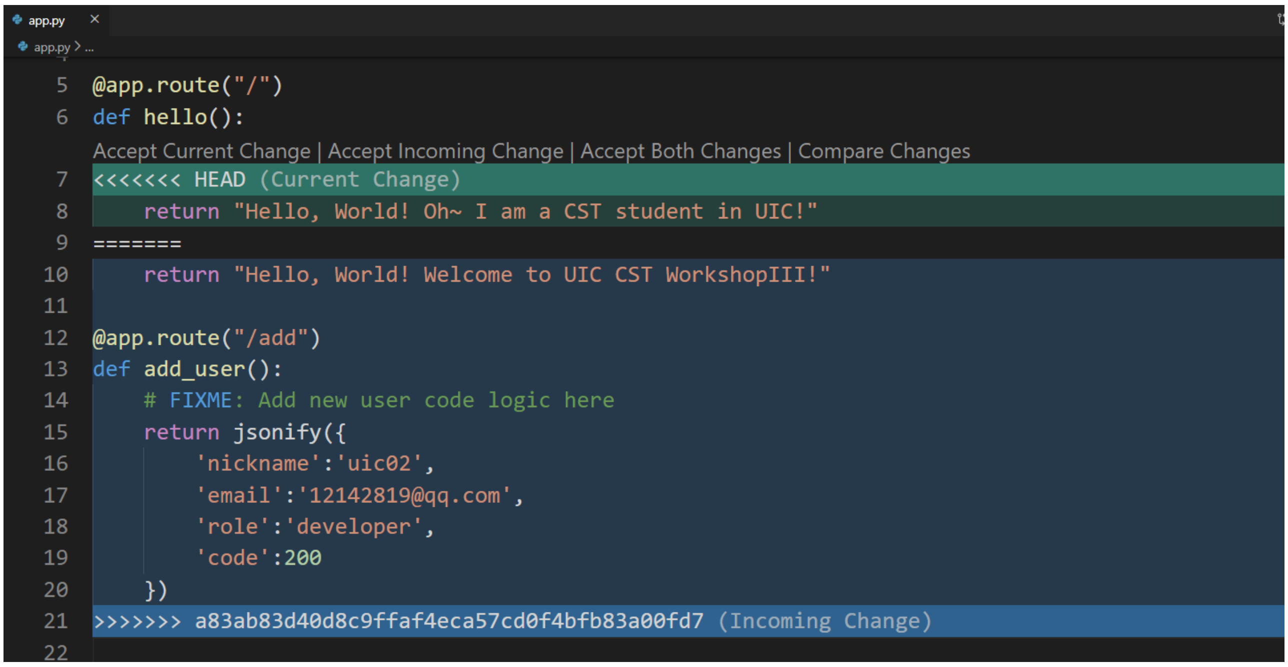
- The code between `=======` and `>>>>>>>` is incoming code from GitHub.

GitHub Desktop will compare you current local code version with the remote GithHub

What you need to do is to decide which part of code need to remove / merge.

Once you have fixed the conflicts, make sure removing all the conflict symbols `<<<<<<< HEAD`, `=======`, `>>>>>>>` etc.

Before fix conflicts.



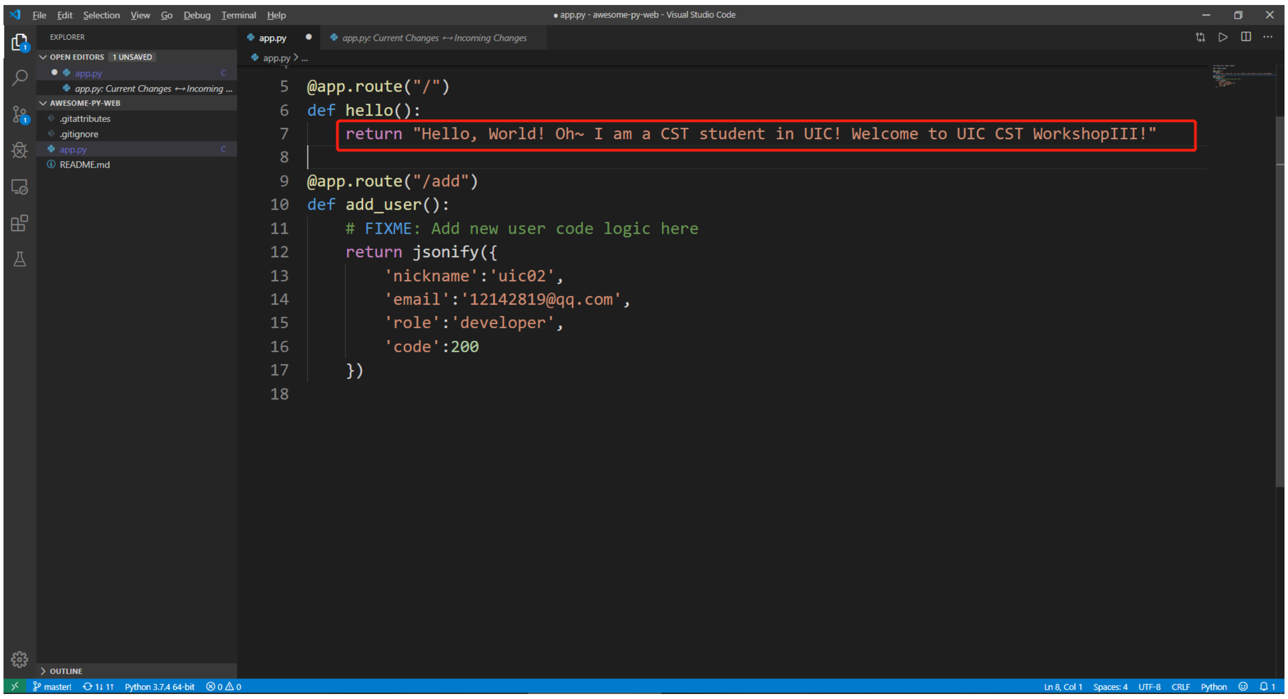
app.py
app.py > ...
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

```
@app.route("/")
def hello():
<<<<<<< HEAD (Current Change)
    return "Hello, World! Oh~ I am a CST student in UIC!"
=======
    return "Hello, World! Welcome to UIC CST WorkshopIII!"

@app.route("/add")
def add_user():
    # FIXME: Add new user code logic here
    return jsonify({
        'nickname':'uic02',
        'email':'12142819@qq.com',
        'role':'developer',
        'code':200
    })
>>>>>>> a83ab83d40d8c9ffaf4eca57cd0f4bfb83a00fd7 (Incoming Change)
```

VS Code provides convenient ways to solve the conflicts by Accep Current change / Accept Incoming Change / Accept Both Changes /Compare Changes.

You can also edit it manually if you want.

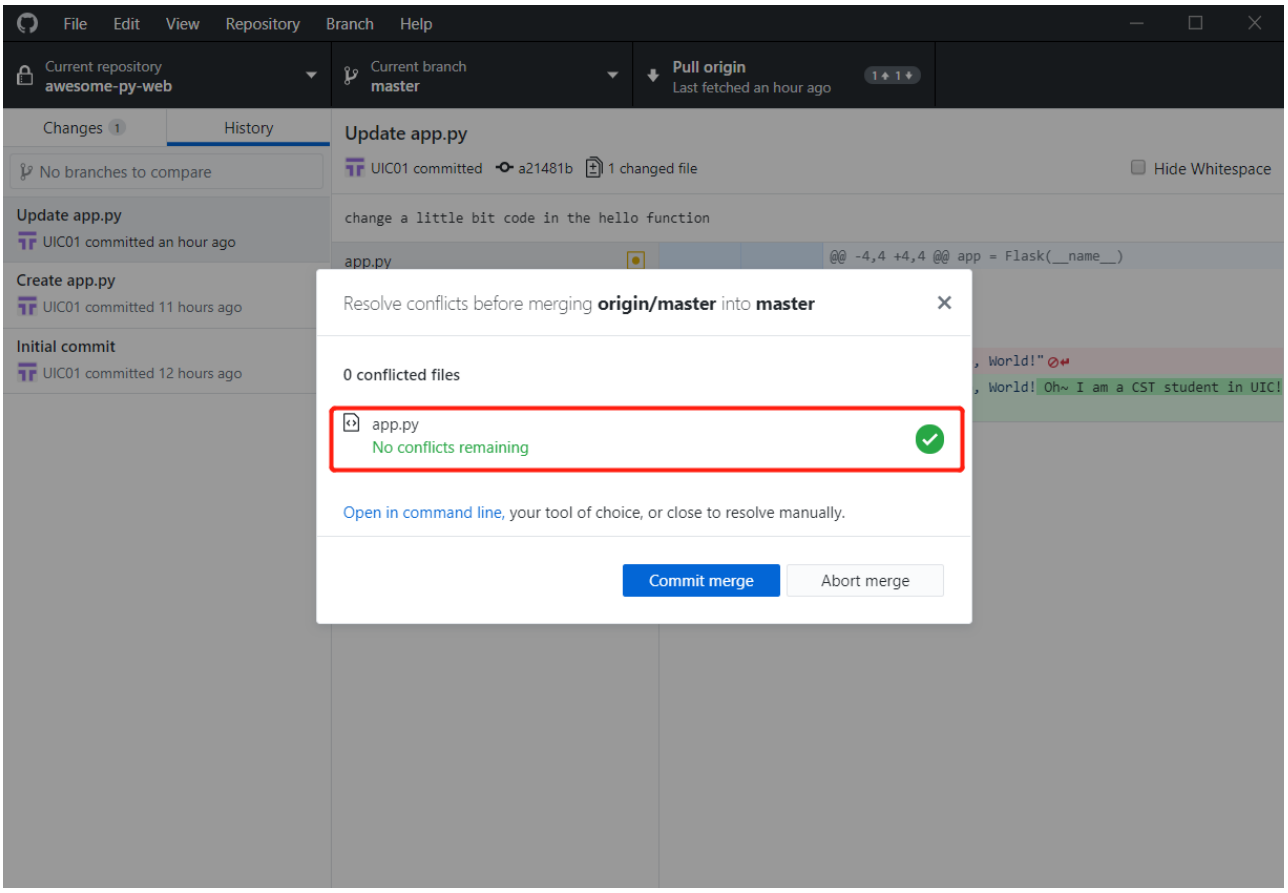
After fix conflicts.



```python
@app.route("/")
def hello():
    return "Hello, World! Oh~ I am a CST student in UIC! Welcome to UIC CST WorkshopIII!"

@app.route("/add")
def add_user():
    # FIXME: Add new user code logic here
    return jsonify({
        'nickname':'uic02',
        'email':'12142819@qq.com',
        'role':'developer',
        'code':200
    })

```

File Edit Selection View Go Debug Terminal Help
● app.py - awesome-py-web - Visual Studio Code
EXPLORER
OPEN EDITORS 1 UNSAVED
app.py
app.py: Current Changes ↔ Incoming ...
AWESOME-PY-WEB
.gitattributes
.gitignore
app.py
README.md
OUTLINE
master! 11 11 Python 3.7.4 64-bit ⊗ 0 ⚠ 0
Ln 8, Col 1 Spaces: 4 UTF-8 CRLF Python

Also the GitHub Desktop will show No conflicts remaining.



Resolve conflicts before merging **origin/master** into **master**

0 conflicted files

app.py
No conflicts remaining

Open in command line, your tool of choice, or close to resolve manually.

Update app.py
UIC01 committed an hour ago

Create app.py
UIC01 committed 11 hours ago

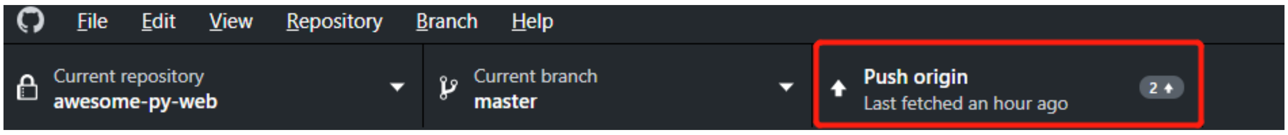
Initial commit
UIC01 committed 12 hours ago

Update app.py
UIC01 committed a21481b 1 changed file

change a little bit code in the hello function

7. Click commit merge.

8. Push to GitHub.



Conclusion

Congratulations! You've now created a repository, published the repository to GitHub, made a commit, and pushed your changes and aslo known how to solve conflicts. We've only scratched the surface of all the things you can do with GitHub and GitHub Desktop. We hope this exercise has you excited to explore further!

Acknowledge

- https://git-scm.com/
- https://help.github.com/en