Git Merge

Lab 1 hour 30 minutes No cost Introductory

Rate Lab

This lab may incorporate AI tools to support your learning.

Introduction

In this lab, you'll use your knowledge of Git and Git commit history to check out an existing repo and make some changes to it. You'll also test what you learned about rolling back commits after bad changes in order to fix a script in the repo and run it to produce the correct output.

What you'll do

- Check the status and history of an existing Git repo
- Create a branch
- Modify content on the branch
- Make rollback changes
- Merge the branch

You'll have 90 minutes to complete this lab.

Disclaimer: For optimal performance and compatibility, it is recommended to use either **Google Chrome** or **Mozilla Firefox** browsers while accessing the labs.

Start the lab

You'll need to start the lab before you can access the materials. To do this, click the green "Start Lab" button at the top of the screen.

Start Lab

After you click the "Start Lab" button, you will see a shell, where you will be performing further steps in the lab. You should have a shell like this:

Explore repository

There is a Git repository named `food-scripts` consisting of a couple of food-related Python scripts.

Navigate to the repository using the following command:

```
cd ~/food-scripts
```

Now, list the files using the `ls` command. There are three files named **favorite_foods.log**, **food_count.py**, and **food_question.py**.

```
favorite_foods.log  food_count.py  food_question.py
```

Let's explore each file. Use the `cat` command to view each file.

1. **favorite_foods.log**: This file consists of a list of food items. You can view it using the following command:

```
cat favorite_foods.log
```

```
cat favorite_foods.log
```

Output:

```
pie
burgers
pizza
pie
tacos
fried chicken
spaghetti
rice
cake
broccoli
cake
cereal
salad
avocados
burgers
```

2. **food_count.py**: This script returns a list of each food and the number of times the food appeared in the `favorite_foods.log` file.

Let's execute the script `food_count.py`:

```
./food_count.py
```

Output:

```
rice, 12
burgers, 10
fried chicken, 9
pie, 8
pizza, 7
salad, 7
tacos, 6
avocados, 6
spaghetti, 5
broccoli, 5
ice cream, 5
bananas, 5
fish, 4
cake, 3
cereal, 3
strawberries, 3
watermelon, 2
```

3. **food_question.py**: This prints a list of foods and prompts the user to enter one of those foods as their favorite. It then returns an answer of how many others in the list like that same food.

Run the following command to see the output of `food_question.py` script:

```
./food_question.py
```

Output:

```
Traceback (most recent call last):
  File "./food_question.py", line 10, in <module>
    if item not in counter:
NameError: name 'item' is not defined
```

Uh oh , this gives us an error. One of your colleagues reports that this script was working fine until the most recent commit. We'll be fixing this error later during the lab.

Understanding the repository

Let's use the following Git operations to understand the workflow of the repository:

- git status
- git log
- git branch

Git status: This displays paths that have differences between the index file and the current HEAD commit; paths that have differences between the working tree and the index file; and paths in the working tree that are not tracked by Git. You can view the status of the working tree using the command: **[git status]**

```
git status
```

You can now view the status of the working tree.

Git log: This lists the commits done in the repository in reverse chronological order; that is, the most recent commits show up first. This command lists each commit with its SHA-1 checksum, the author's name and email, date, and the commit message.

You can see logs by using the following command:

```
git log
```

Output:

```
commit 21cf376832fa6eace35c0bf9e4bae4a3400452e9 (HEAD -> master)
Author: Alex Cooper <alex_cooper@gmail.com>
Date:   Wed Jan 8 14:09:39 2020 +0530

    Rename item variable to food_item.

commit b8d00e33237b24ea1480c363edd972cf4b49eedf
Author: Alex Cooper <alex_cooper@gmail.com>
Date:   Wed Jan 8 14:08:35 2020 +0530

    Added file food_question.py that returns how many others in the list
like that same food.
```

```
commit 8d5a3189b88d273ef08286e5074b5e38d616da21
Author: Alex Cooper <alex_cooper@gmail.com>
Date:   Wed Jan 8 14:07:15 2020 +0530

    Added file food_count.py that returns a list of each food and the
number of times each food appears in favorite_foods.log file.

commit 0a202e03e0356d2b5c323e51905d3d06e5e2d0ed
Author: Alex Cooper <alex_cooper@gmail.com>
Date:   Wed Jan 8 14:06:21 2020 +0530

    Added file favorite_foods.log that contains list of foods.
```

Git branch: Branches are a part of the everyday development process on the master branch. Git branches effectively function as a pointer to a snapshot of your changes. When you want to add a new feature or fix a bug, no matter how big or small, you spawn a new branch to encapsulate your changes. This makes it difficult for unstable code to get merged into the main codebase.

Configure Git

Before we move forward with the lab, let's configure Git. Git uses a username to associate commits with an identity. It does this by using the **git config** command. Set the Git username with the following command:

```
git config user.name "Name"
```

Replace **Name** with your name. Any future commits you push to GitHub from the command line will now be represented by this name. You can even use **git config** to change the name associated with your Git commits. This will only affect future commits and won't change the name used for past commits.

Let's set your email address to associate them with your Git commits.

```
git config user.email "user@example.com"
```

Replace **user@example.com** with your email-id. Any future commits you now push to GitHub will be associated with this email address. You can also use **git config** to change the user email associated with your Git commits.

Add a new feature

In this section, we'll be modifying the repository to add a new feature, without affecting the current iteration. This new feature is designed to improve the food count (from the file food_count.py) output. So, create a branch named **improve-output** using the following command:

```
git branch improve-output
```

Move to the `improve-output` branch from the master branch.

```
git checkout improve-output
```

Here, you can modify the script file without disturbing the existing code. Once modified and tested, you can update the master branch with a working code.

Now, open `food_count.py` in the nano editor using the following command:

```
nano food_count.py
```

Add the line below before **printing for loop** in the `food_count.py` script:

```
print("Favourite foods, from most popular to least popular")
```

Chromebook users: Instructions for saving a file in the nano editor >

Save the file by pressing **Ctrl-o**, the **Enter** key, and **Ctrl-x**. Then run the script `food_count.py` again to see the output:

```
./food_count.py
```

Output:

```
Favourite foods, from most popular to least popular
rice, 12
burgers, 10
fried chicken, 9
pie, 8
pizza, 7
salad, 7
tacos, 6
avocados, 6
spaghetti, 5
broccoli, 5
ice cream, 5
bananas, 5
fish, 4
cake, 3
cereal, 3
strawberries, 3
watermelon, 2
```

After running the `food_count.py` script successfully, commit the changes from the `improve-output` branch by adding this script to the staging area using the following command:

```
git add food_count.py
```

Now, commit the changes you've done in the `improve-output` branch.

```
git commit -m "Adding a line in the output describing the utility
of food_count.py script"
```

Output:

```
[improve-output 09b9ce4] Adding a line in the output describing the
utility of food_count.py script
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Fix the script

In this section, we'll fix the script `food_question.py`, which displayed an error when executing it. You can run the file again to view the error.

```
./food_question.py
```

Output:

```
Traceback (most recent call last):
  File "./food_question.py", line 10, in <module>
    if item not in counter:
NameError: name 'item' is not defined
```

This script gives us the error: "**NameError: name 'item' is not defined**" but your colleague says that the file was running fine before the most recent commit they did.

In this case, we'll revert back the previous commit.

For this, check the git log history so that you can revert back to the commit where it was working fine.

```
git log
```

Output:

```
commit 09b9ce441654a361477405d3eea785b3be5f8e8f (HEAD -> improve-output)
Author: Name <user@example.com>
Date:   Thu Oct 12 10:48:08 2023 +0000

    Adding a line in the output describing the utility of food_count.py
script

commit 21cf376832fa6eace35c0bf9e4bae4a3400452e9 (master)
Author: Alex Cooper <alex_cooper@gmail.com>
Date:   Wed Jan 8 14:09:39 2020 +0530

    Rename item variable to food_item.

commit b8d00e33237b24ea1480c363edd972cf4b49eedf
Author: Alex Cooper <alex_cooper@gmail.com>
Date:   Wed Jan 8 14:08:35 2020 +0530

    Added file food_question.py that returns how many others in the list
like that same food.
```

Here, you'll see the commits in reverse chronological order and find the commit having "**Rename item variable to food_item**" as a commit message. Make sure to note the commit ID for this particular commit.

To revert, use the following command:

```
git revert [commit-ID]
```

Replace `[commit-ID]` with the commit ID you noted earlier.

This creates a new commit again. You can continue with the default commit message on the screen or add your own commit message.

Then continue by clicking **Ctrl-o**, the **Enter** key, and **Ctrl-x**.

Then continue by clicking **Ctrl-o**, the **Enter** key, and **Ctrl-x**.

Now, run `food_question.py` again and verify that it's working as intended.

```
./food_question.py
```

Output:

```
Select your favorite food below:
pie
burgers
pizza
tacos
fried chicken
spaghetti
rice
cake
broccoli
cereal
salad
avocados
ice cream
fish
strawberries
bananas
watermelon
Which of the foods above is your favorite? rice
12 of your friends like rice as well!
```

Merge operation

Before merging the branch `improve-output,` switch to the master branch from the current branch `improve-output` branch using the command below:

```
git checkout master
```

Merge the branch `improve-output` into the master branch.

```
git merge improve-output
```

Output:

```
Updating 21cf376..70bd534
Fast-forward
 food_count.py    | 2 +-
 food_question.py | 2 +-
 2 files changed, 2 insertions(+), 2 deletions(-)
```

Now, all your changes made in the `improve-output` branch are on the master branch.

```
./food_question.py
```

Output:

```
Select your favorite food below:
pie
burgers
pizza
tacos
fried chicken
spaghetti
rice
cake
broccoli
cereal
salad
avocados
ice cream
fish
strawberries
bananas
watermelon
Which of the foods above is your favorite? burgers
10 of your friends like burgers as well!
```

To get the status from the master branch, use the command below:

```
git status
```

Output:

```
On branch master
nothing to commit, working tree clean
```

To track the git commit logs, use the following command:

```
git log
```

Output:

```
commit 57c54e10c808118898bb0f15625bda2c0f55fec3 (HEAD -> master,
improve-output)
Author: Name <user@example.com>
Date:   Tue Oct 31 08:57:01 2023 +0000

    Revert "Rename item variable to food_item."

    This reverts commit 21cf376832fa6eace35c0bf9e4bae4a3400452e9.

commit 80b47ced952bffe110e2f9f00769cd43ed047ac1
Author: Name <user@example.com>
Date:   Tue Oct 31 08:55:24 2023 +0000

    Adding a line in the output describing the utility of food_count.py
script

commit 21cf376832fa6eace35c0bf9e4bae4a3400452e9
Author: Alex Cooper <alex_cooper@gmail.com>
Date:   Wed Jan 8 14:09:39 2020 +0530

    Rename item variable to food_item.
```