# Git 101

# Using Vim

Vim is the default text editor for terminal. Very useful when writing scripts on the HPC servers. Always comes up when using Git

- Make edits: `esc -> i`
- Write edits: `esc -> :w -> enter`
- Quit: `esc -> :q -> enter`
- Write and quit: `esc -> :wq -> enter`
- Quit without saving -> `esc -> :q! -> enter`
- Delete line -> `esc -> dd`

# Useful Vim Settings

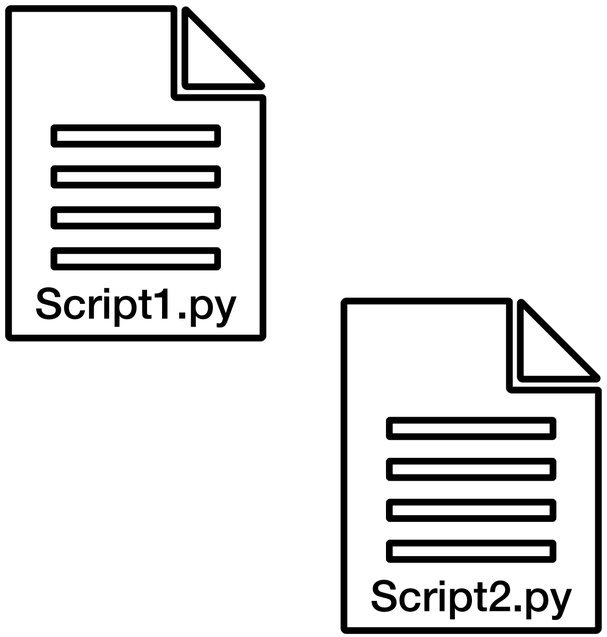
The Vim settings are stored in a file named `~/.vimrc`

We will add the following commands to `~/vimrc`. In order they (1) turn on syntax highlighting, (2) Enable cursor, (3) Tab length of 4 and (4) Auto-indentation when coding

```
syntax on
set mouse=a
set tabstop=4
set autoindent
```

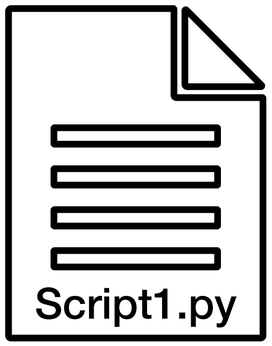
# Initialising Git

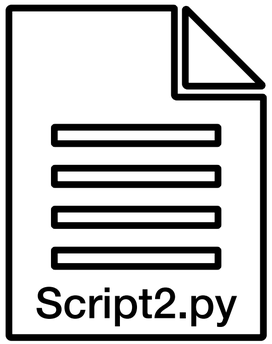
Current Folder



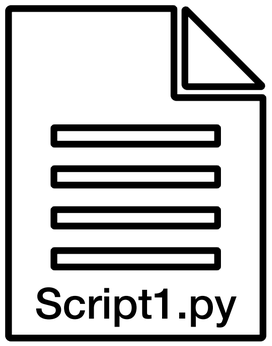
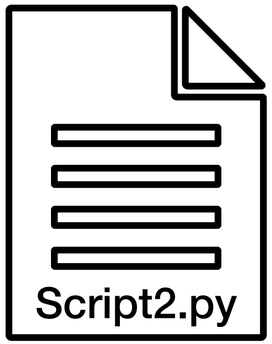
# Initialising Git

| Current Folder | Staged Files | Git |
| --- | --- | --- |
| `git init` | | |

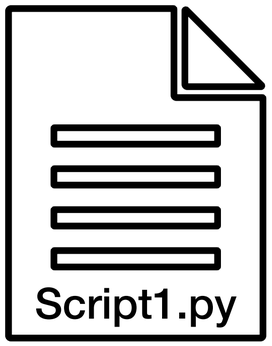
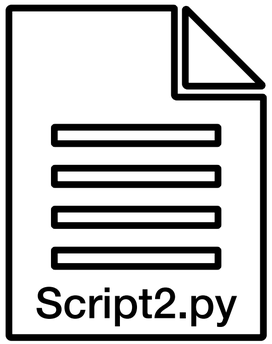
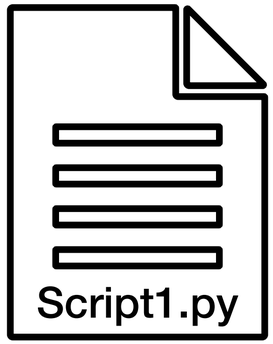


# Adding Single Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | | |

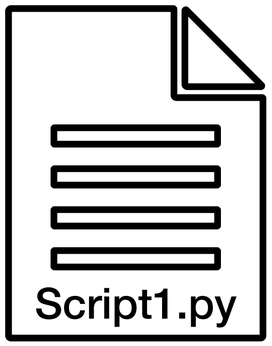
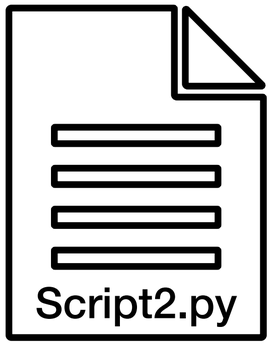
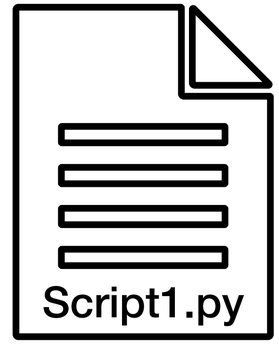
# Adding Single Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | `git add Script1.py`   | |

# Adding Single Files

Optional Command:

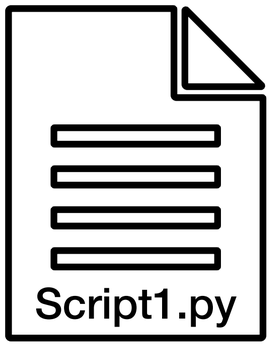
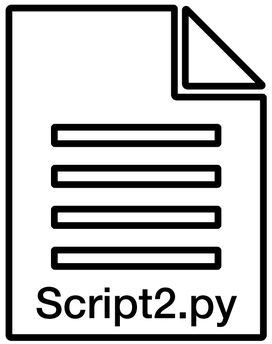
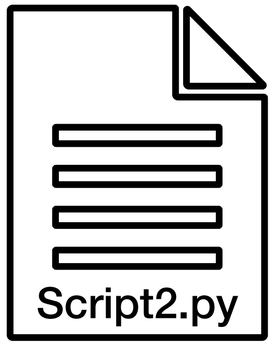
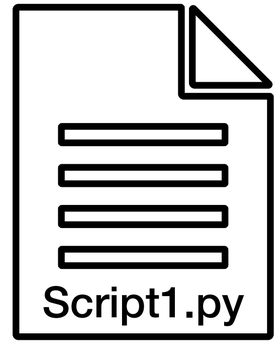
```
git status
```

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|   | `git commit` | `-m ‘adding script1’`  |

# Adding Single Files

Optional Command:

```
git status
```

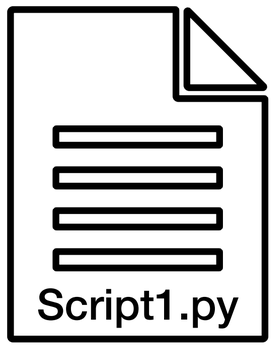
| Current Folder | Staged Files | Git |
| --- | --- | --- |
|   | `git add Script2.py`  |  |

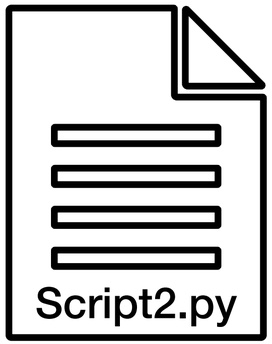
# Adding Single Files

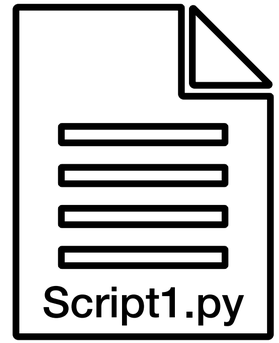
Optional Command:

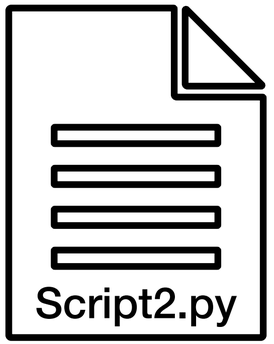
```
git status
```

| Current Folder | Staged Files | Git |
| --- | --- | --- |
| | `git commit -m 'adding script2'` | |

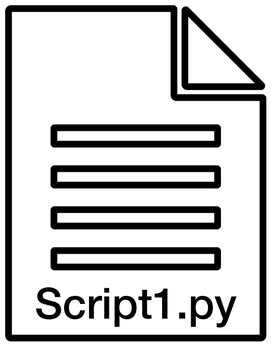
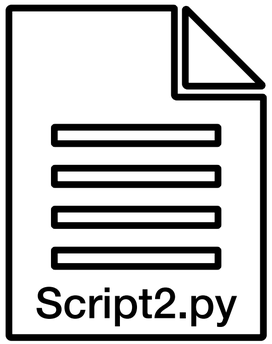
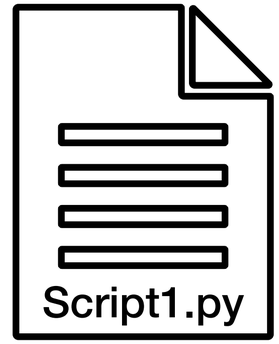
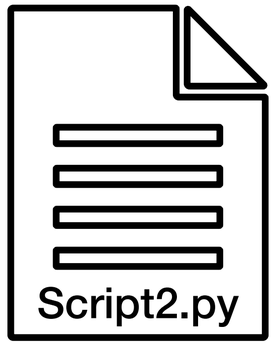


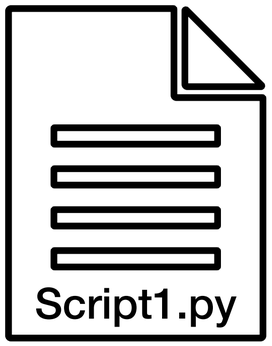
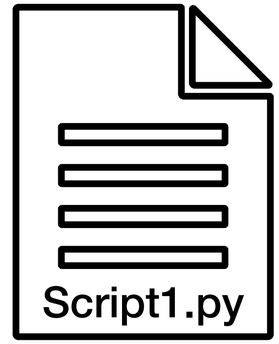
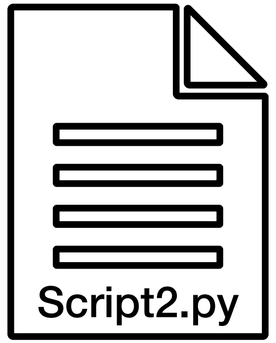



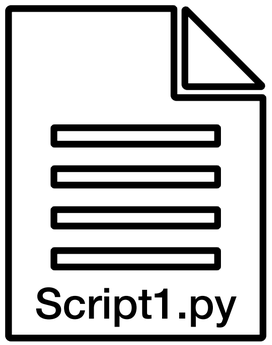
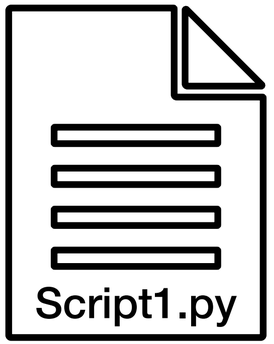
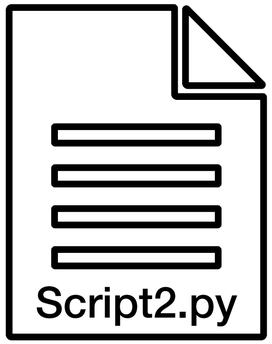
# Removing Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | |     |

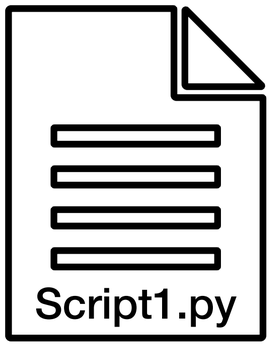
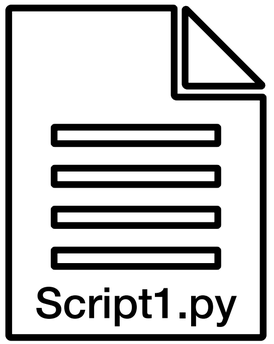
# Removing Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
| `rm Script2.py` | | |
|  | |   |

# Removing Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|   | `git rm Script2.py` |     |

# Removing Files

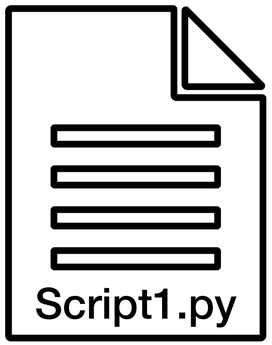
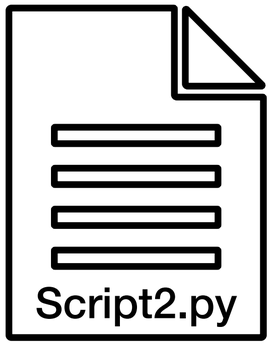
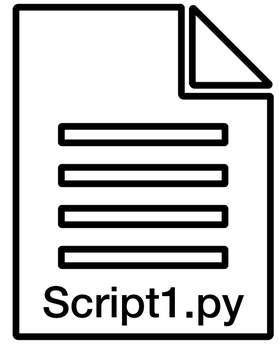
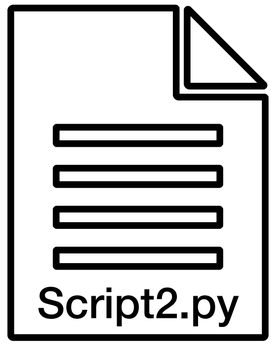
| Current Folder | Staged Files | Git |
| --- | --- | --- |
|   | `git commit -m ‘removing Script2’` |   |

# Removing Folders

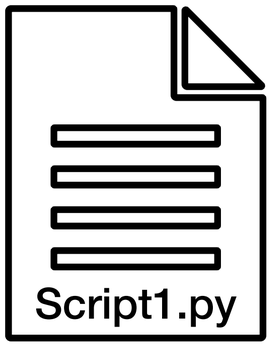
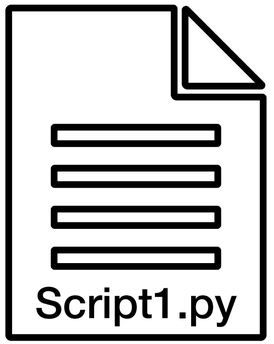
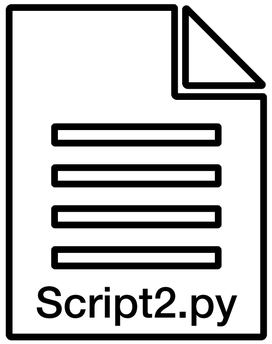
Removing folder requires an extra argument

git rm -r folder_name

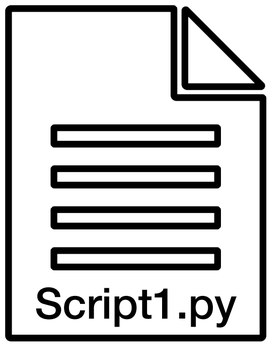
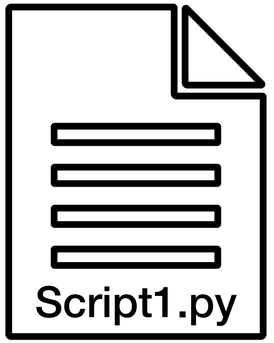
# Removing Files ++

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|   | |   |

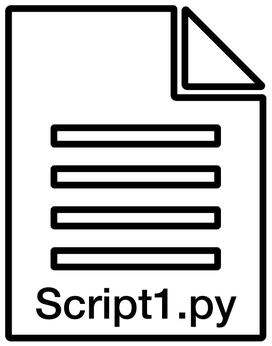
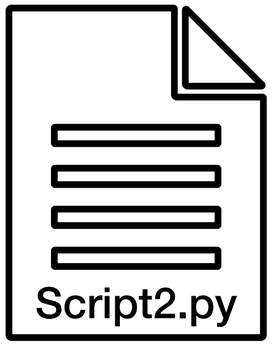
# Removing Files ++

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|  | `git rm Script2.py` |   |

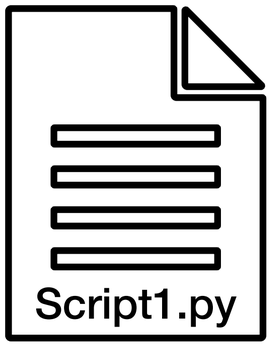
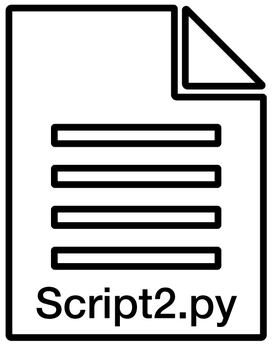
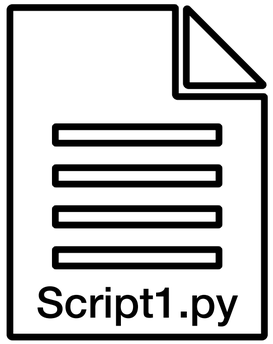
# Removing Files ++

| Current Folder | Staged Files | Git |
| --- | --- | --- |
| | `git commit -m ‘removing Script2’` | |
|   | |   |

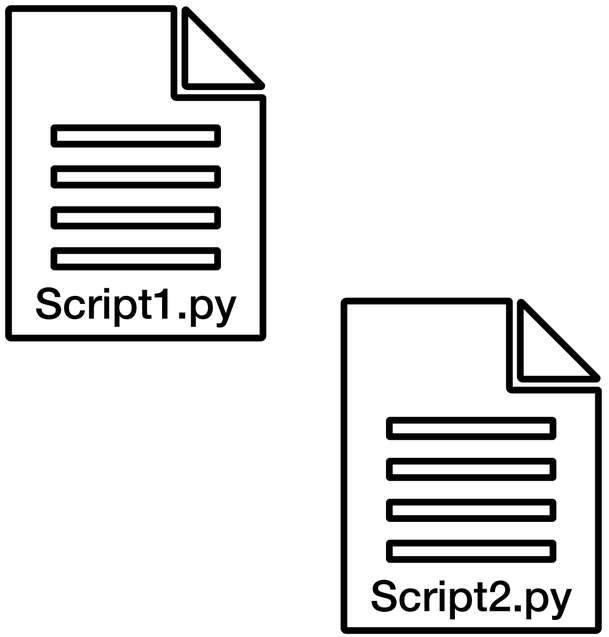
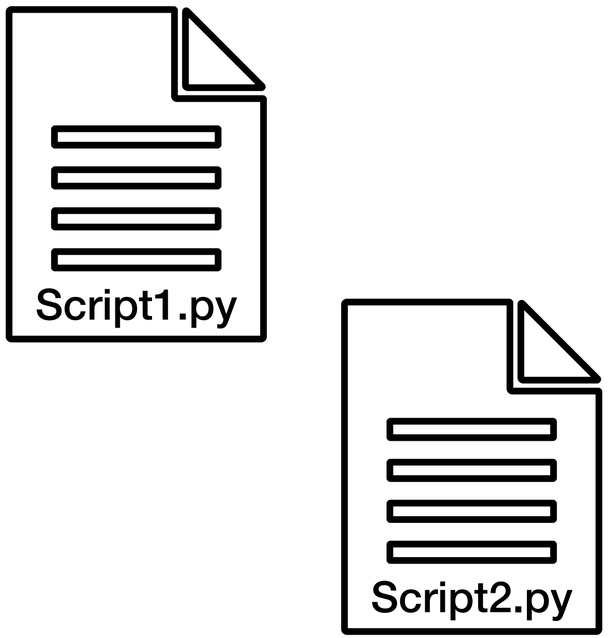
# Adding Multiple Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | | |

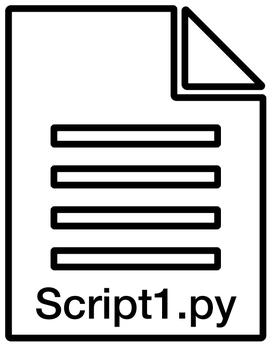
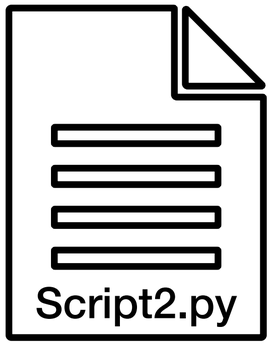
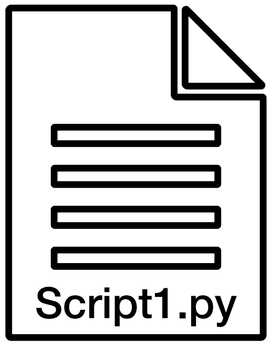
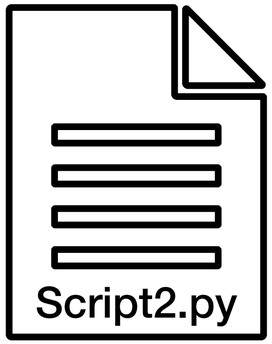
# Adding Multiple Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | `git add Script1.py`   | |

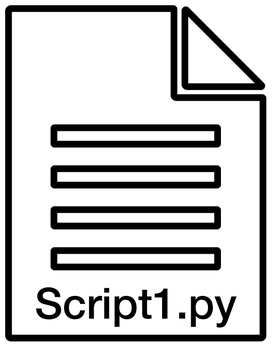
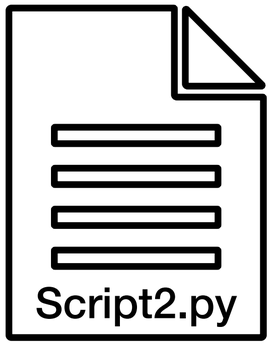
# Adding Multiple Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|  | `git add Script2.py`  | |

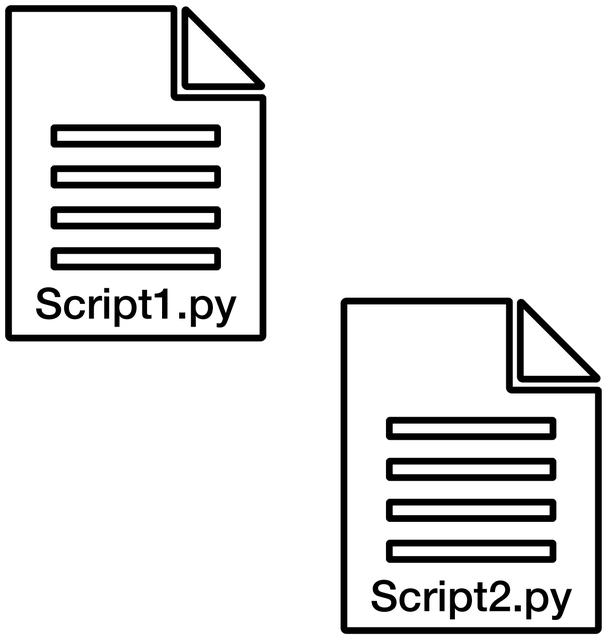
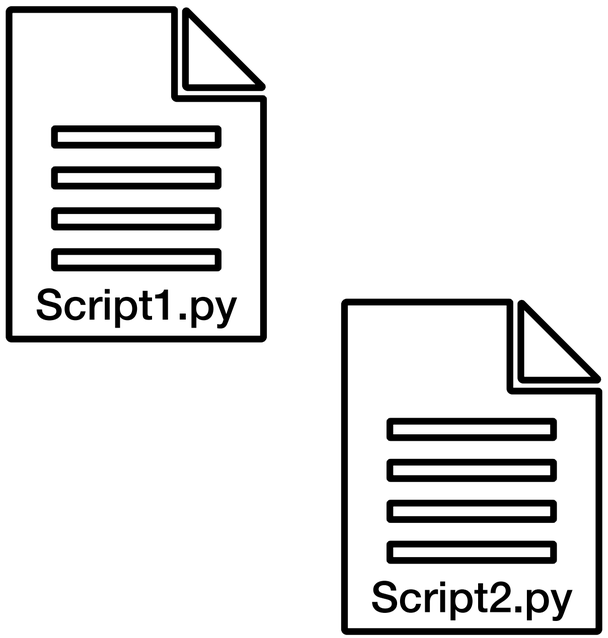
# Adding Multiple Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | `git commit -m 'first commit'` |     |

# Adding Multiple Files But Quicker

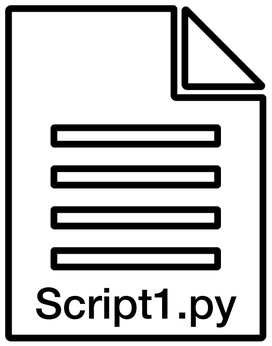
| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | | |

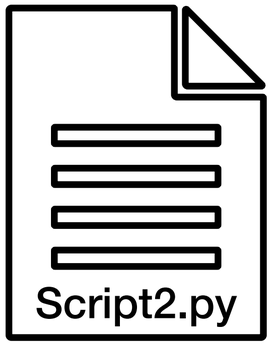
# Adding Multiple Files But Quicker

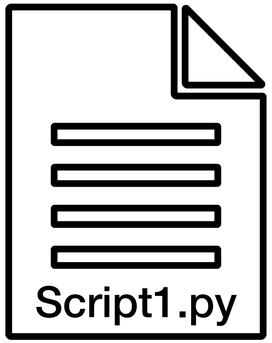
| Current Folder | Staged Files | Git |
| --- | --- | --- |
|   | `git add *`   | |

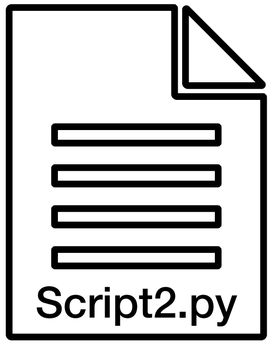
# Adding Multiple Files But Quicker

| Current Folder | Staged Files | Git |
| --- | --- | --- |
| | `git commit -m 'first commit'` | |

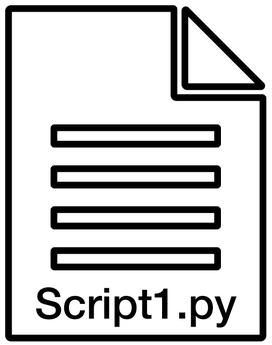
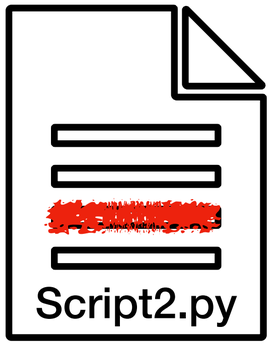
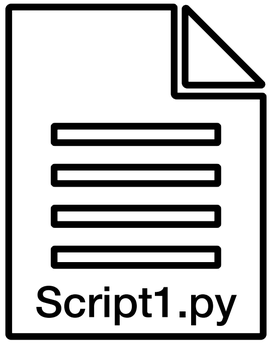
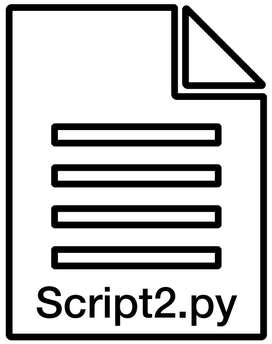


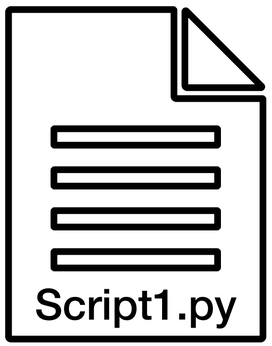
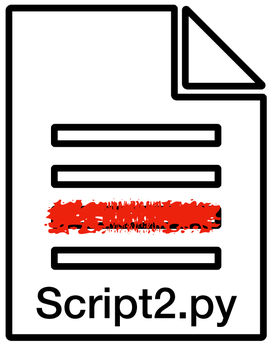
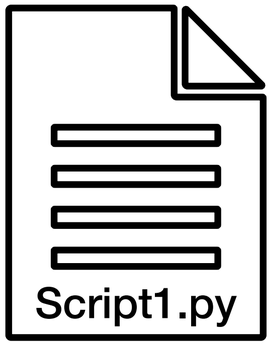
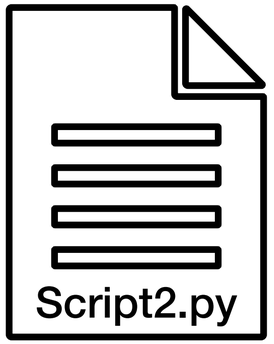



# Adding Edits Files

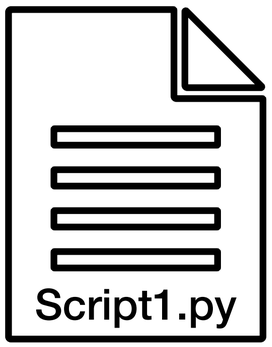
| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | |     |

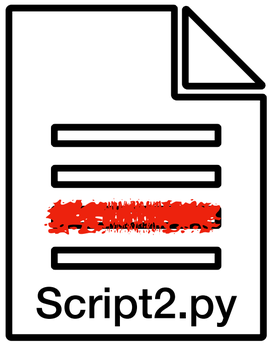
# Adding Edits Files

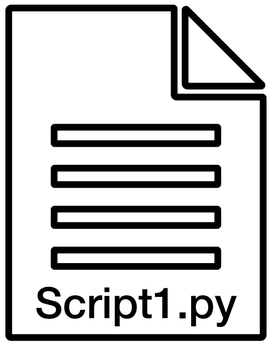
| Current Folder | Staged Files | Git |
| --- | --- | --- |
|     | `git add Script2.py`<br>Or<br>`git add *`<br> |     |

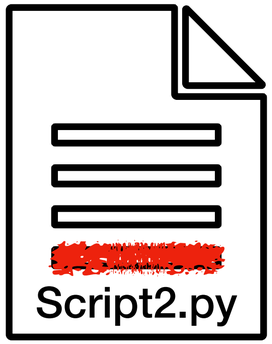
# Adding Edits Files

| Current Folder | Staged Files | Git |
| --- | --- | --- |
| | `git commit -m 'updating script2'` | |



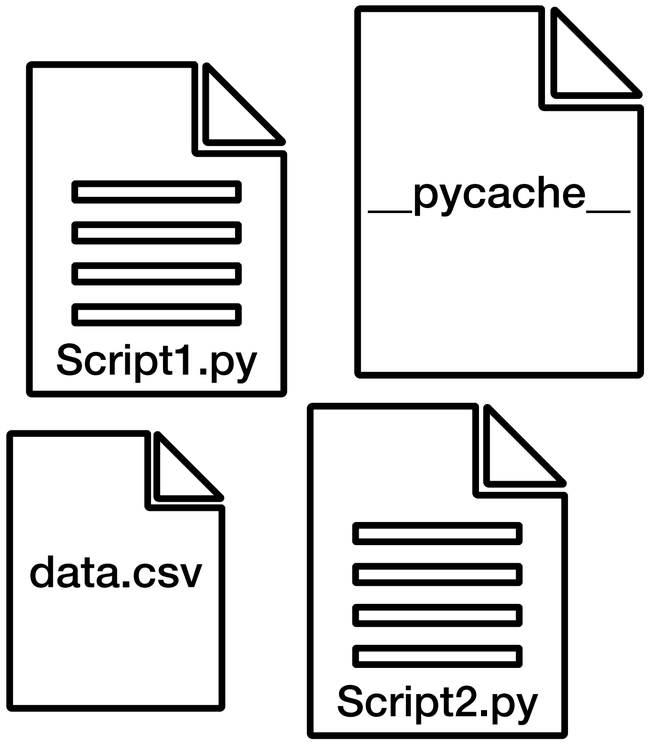



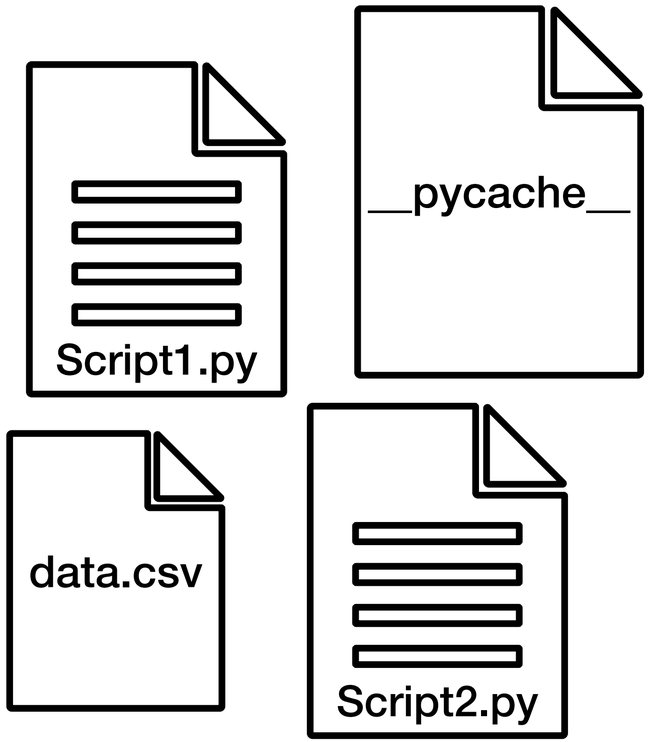
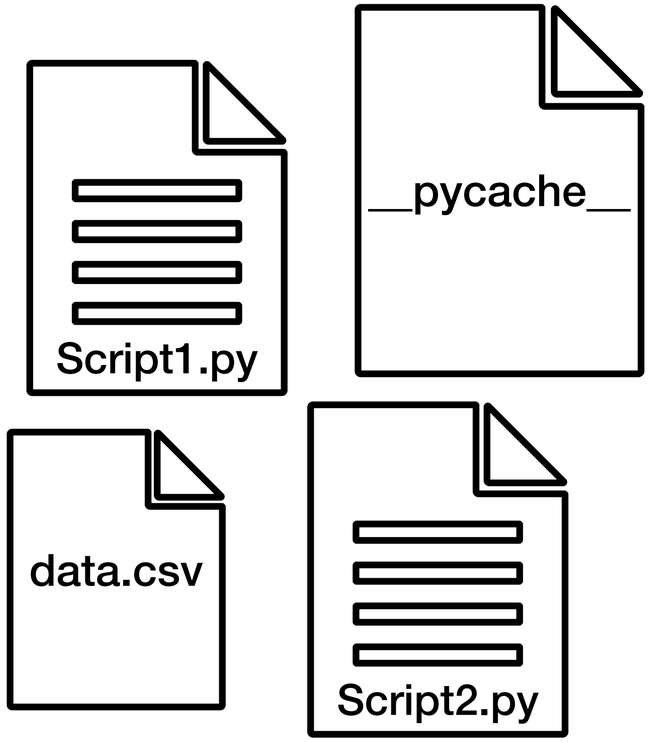



# .gitignore File

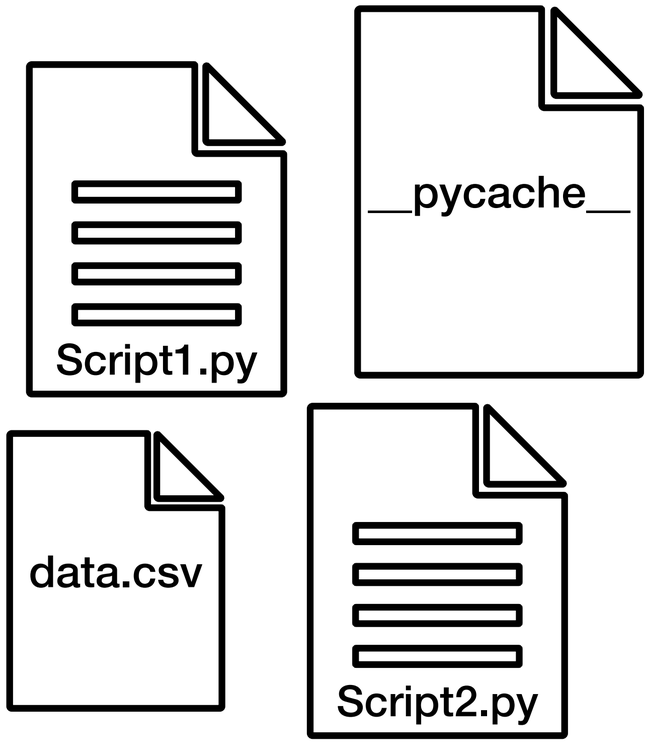
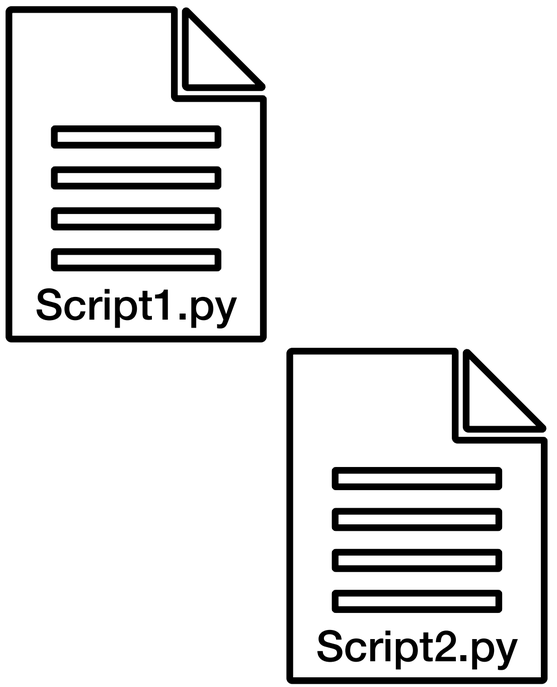
| Current Folder | Staged Files | Git |
| --- | --- | --- |



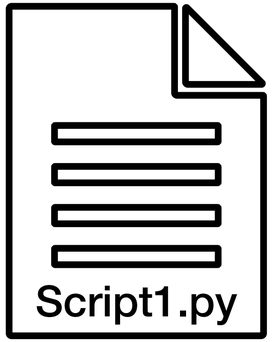
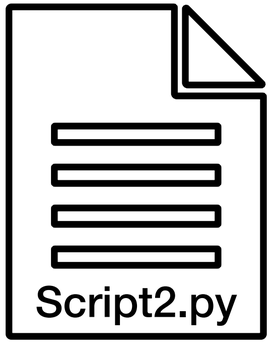
# .gitignore File

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|  | `git add *`  | |

# .gitignore File

| Current Folder | Staged Files | Git |
| --- | --- | --- |
|  | `git add *`  | |

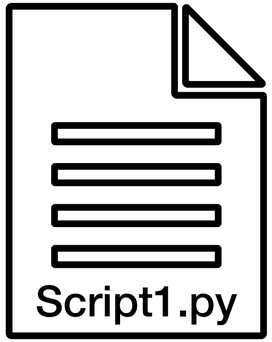
# Restoring Staged Files

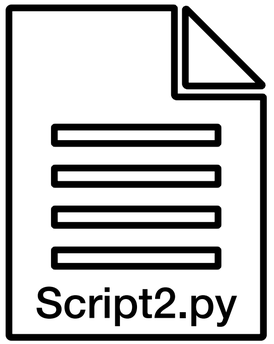
| Current Folder | Staged Files | Git |
| --- | --- | --- |
| | |   |

# Restoring Staged Files

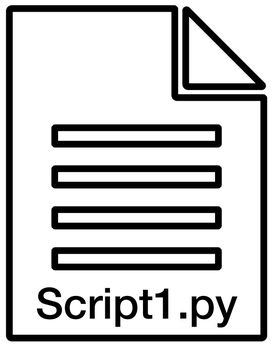
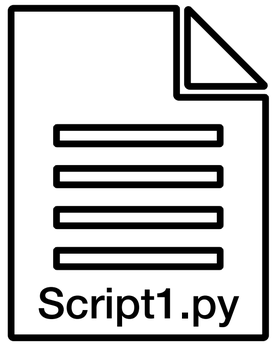
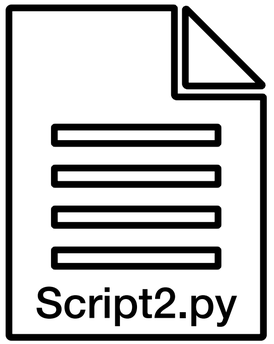
| Current Folder | Staged Files | Git |
| --- | --- | --- |

```
git restore --staged Script1.py
```



# Restoring Staged Files

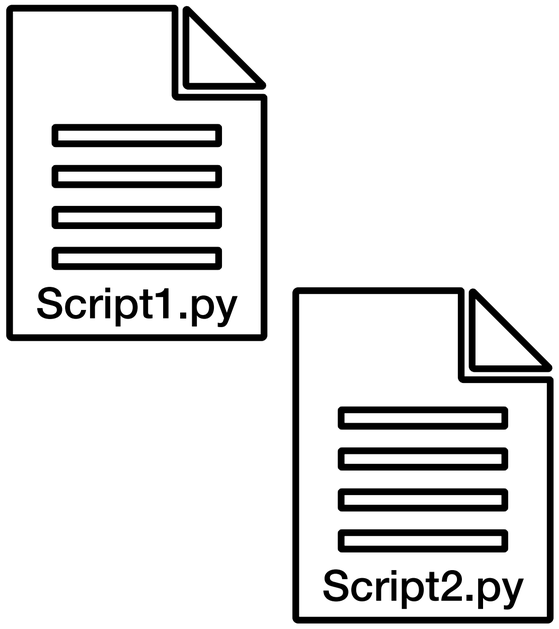
| Current Folder | Staged Files | Git |
| --- | --- | --- |
| `git restore Script1.py` | | |
|   | |     |

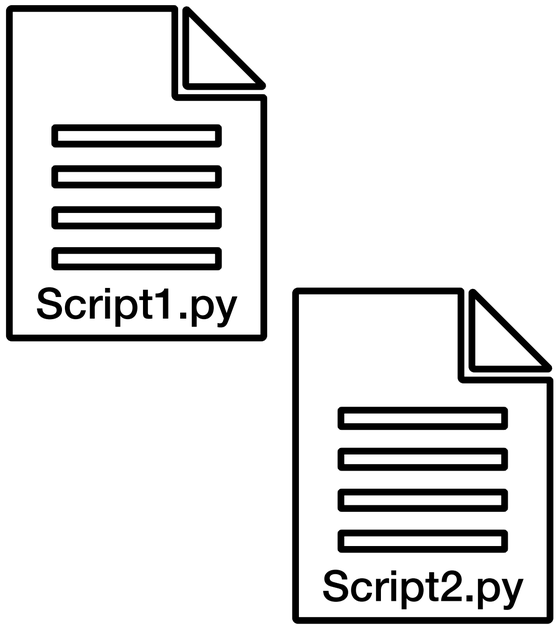
# Using GitHub

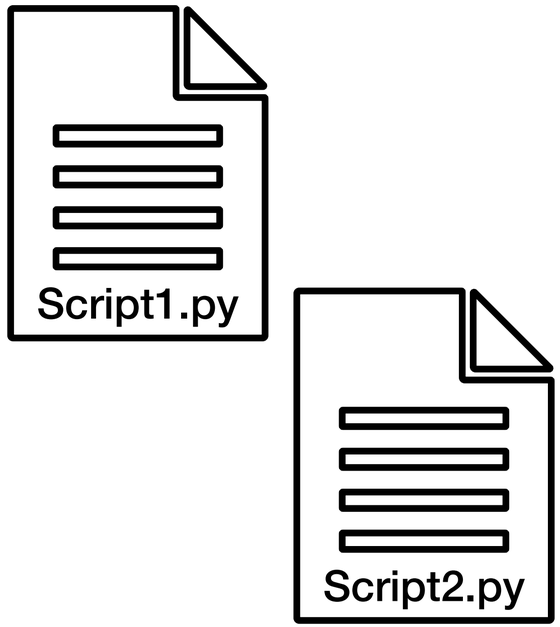
| Current Folder | Staged Files | Git | The Cloud |
| --- | --- | --- | --- |

# Using GitHub

| Current Folder | Staged Files | Git | The Cloud |
| --- | --- | --- | --- |
| | | | `git push` |

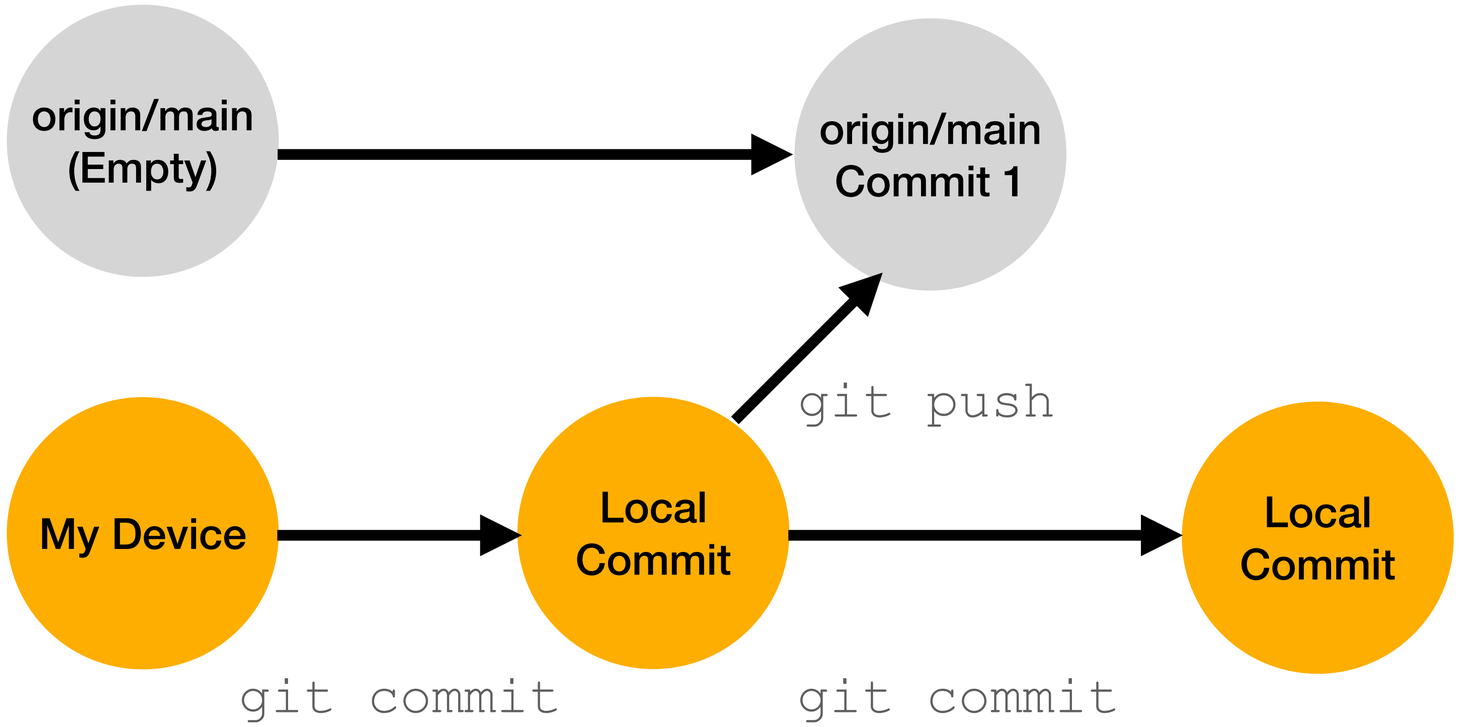


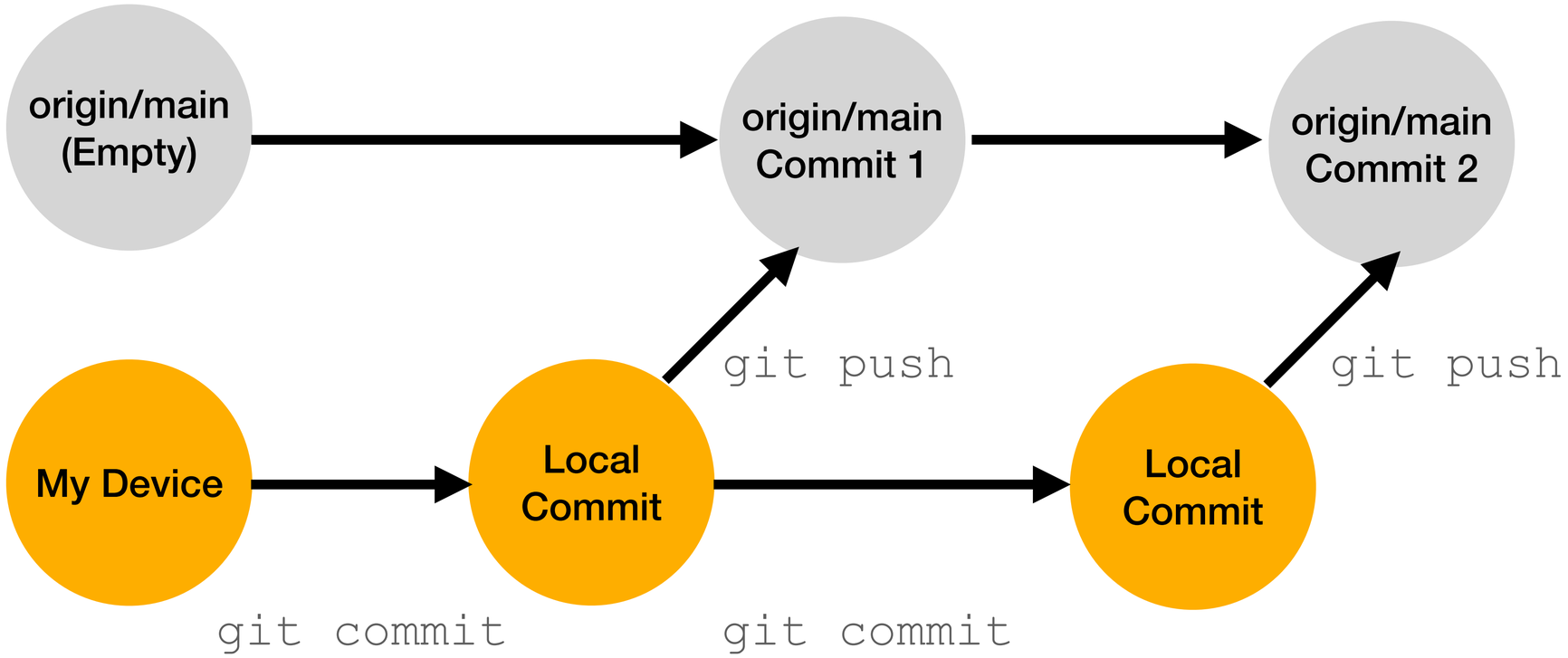
# Using Github

You can make multiple commits before you push to the cloud

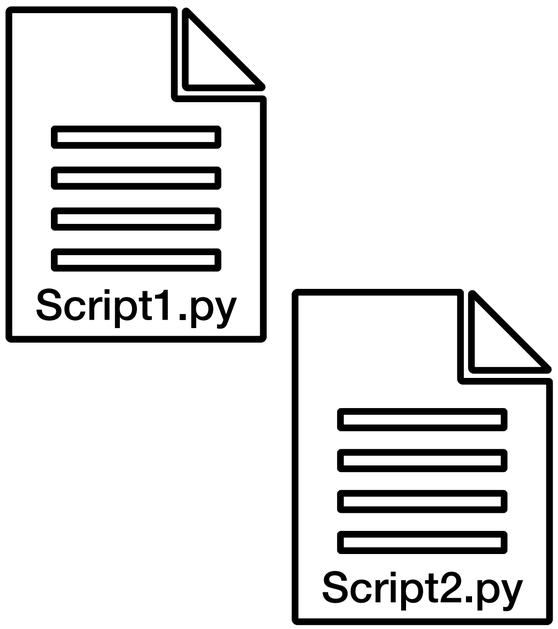
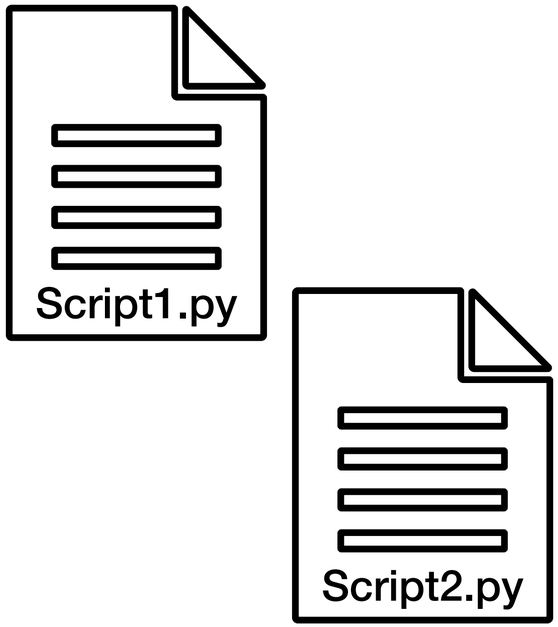


# Using Github

You can make multiple commits before you push to the cloud

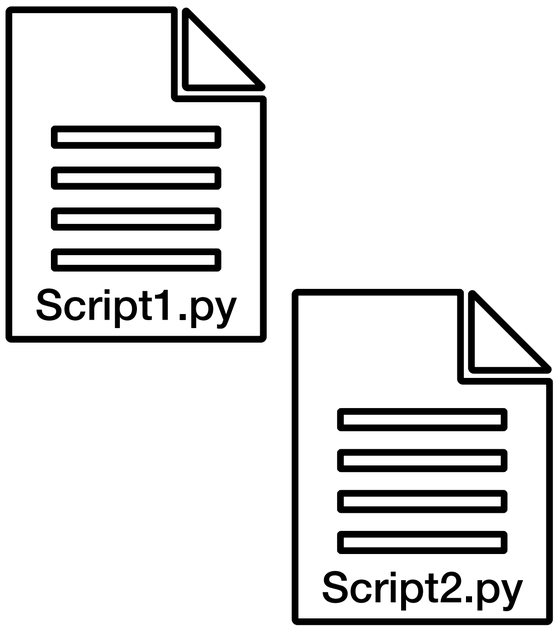


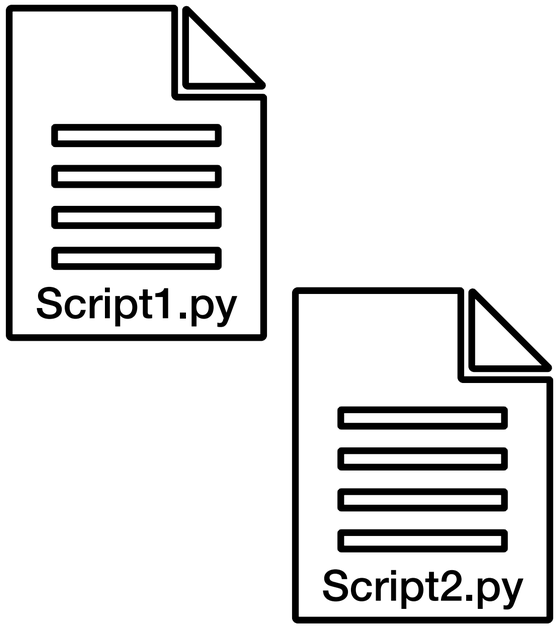
# Collaborating With Github

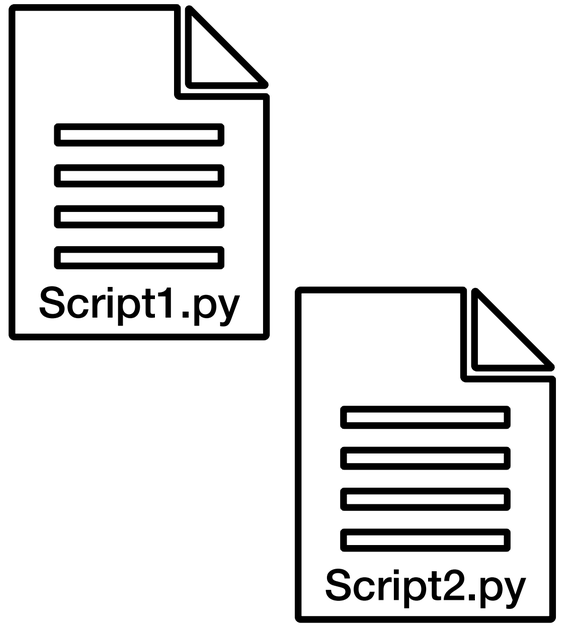
| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
|  |  | |

# Collaborating With Github

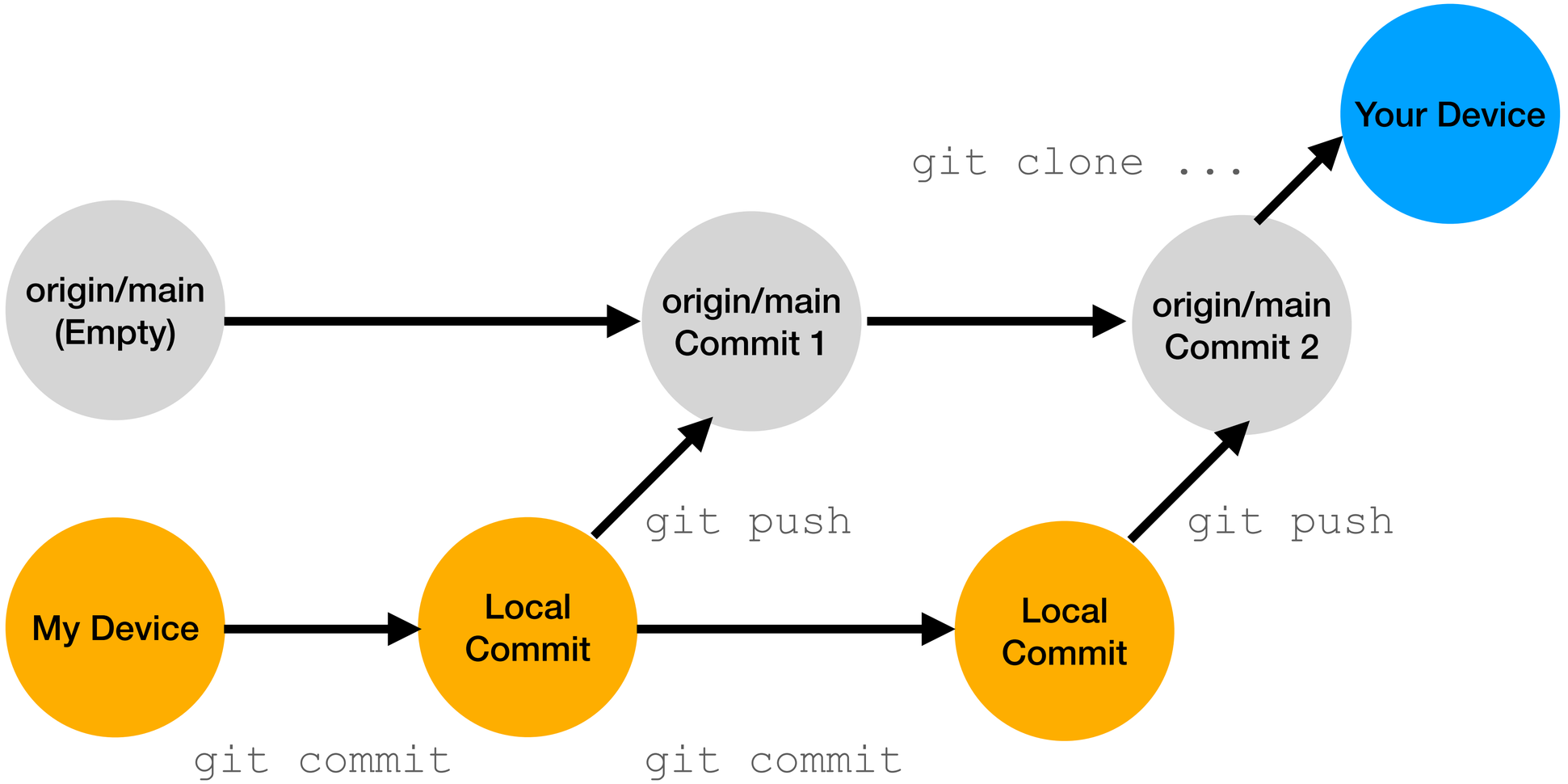
| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
| | | `git clone ...` |



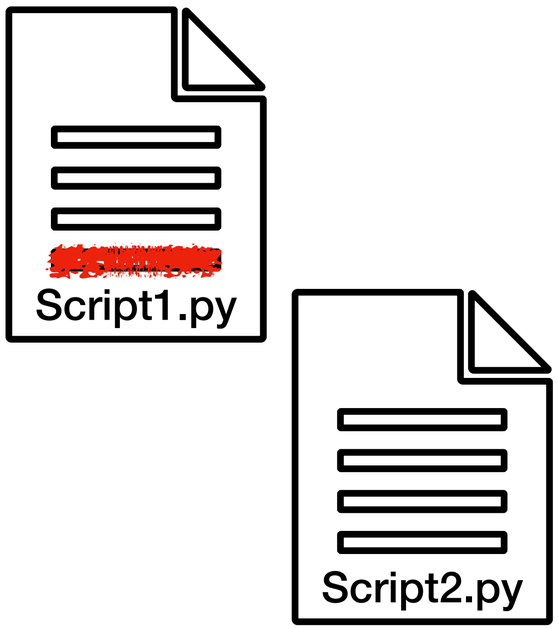
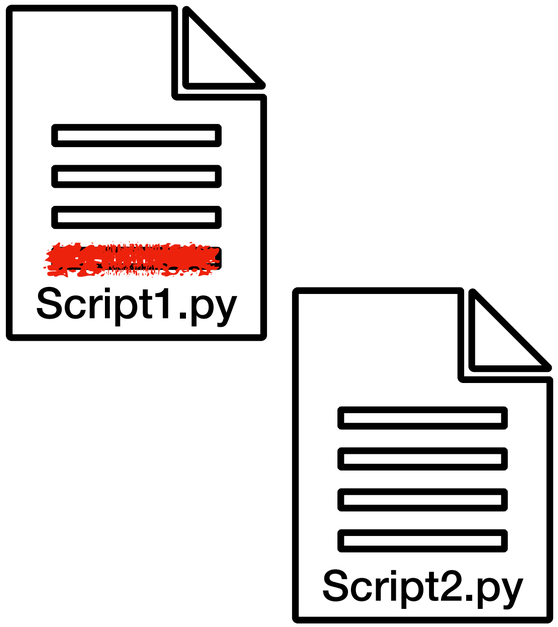
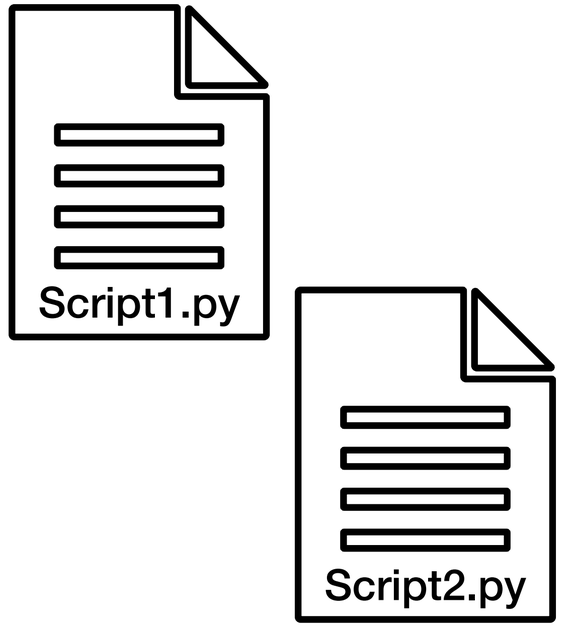



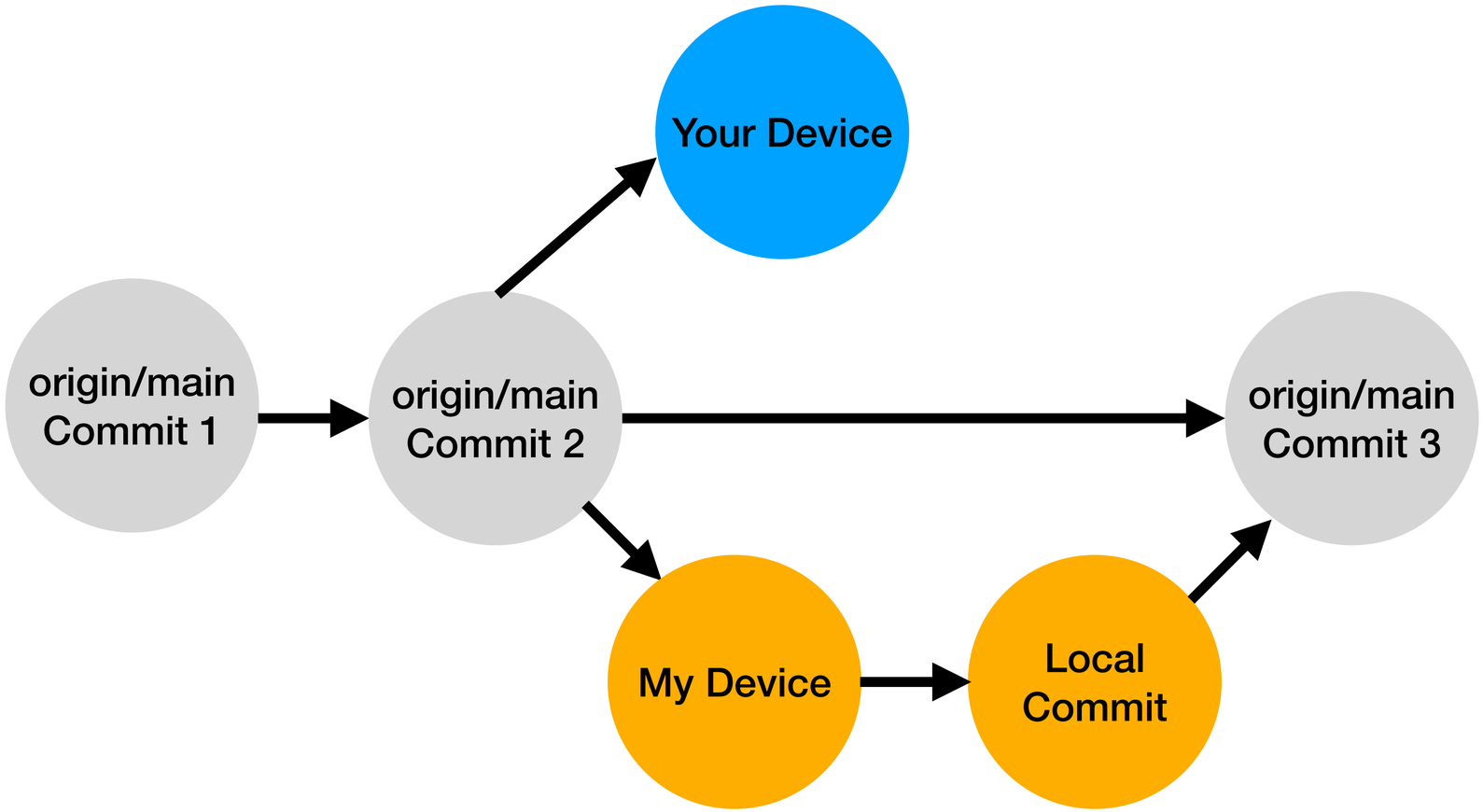
# Collaborating With Github



# Pulling From the Cloud

| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
| `git push` | | |
|  |  |  |

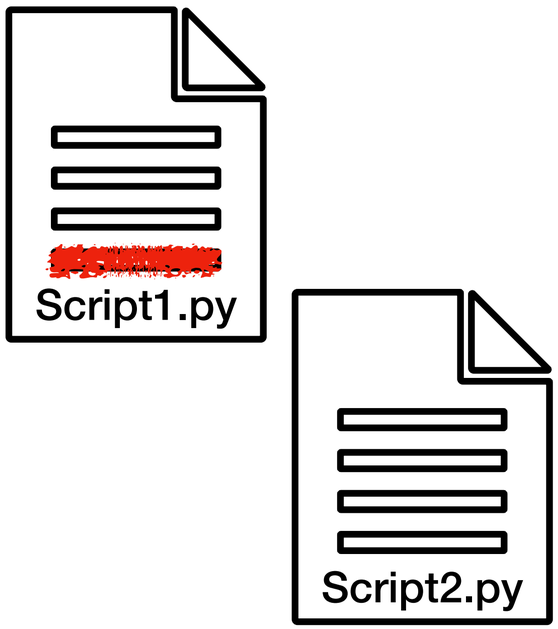
# Pulling From the Cloud

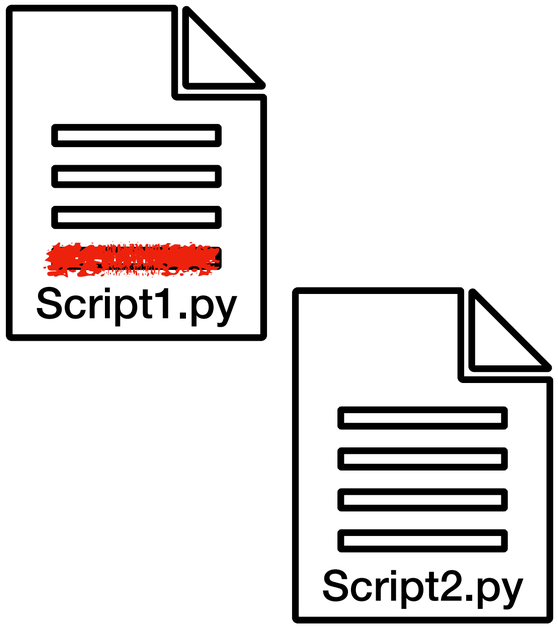


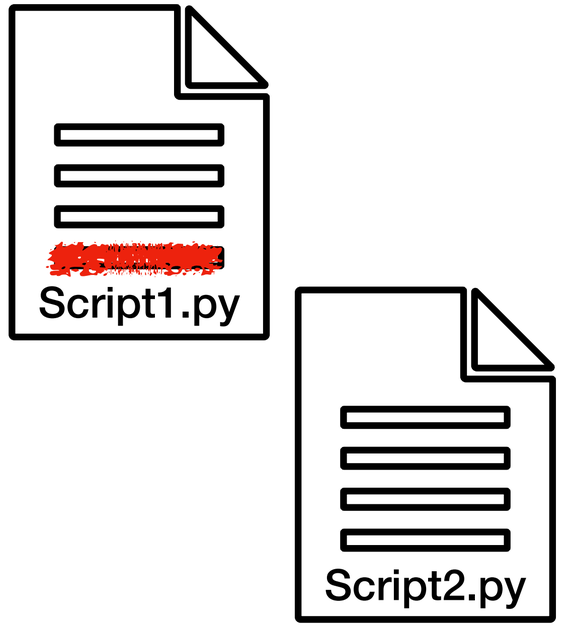
# Pulling From the Cloud

Optional Command: `git fetch`

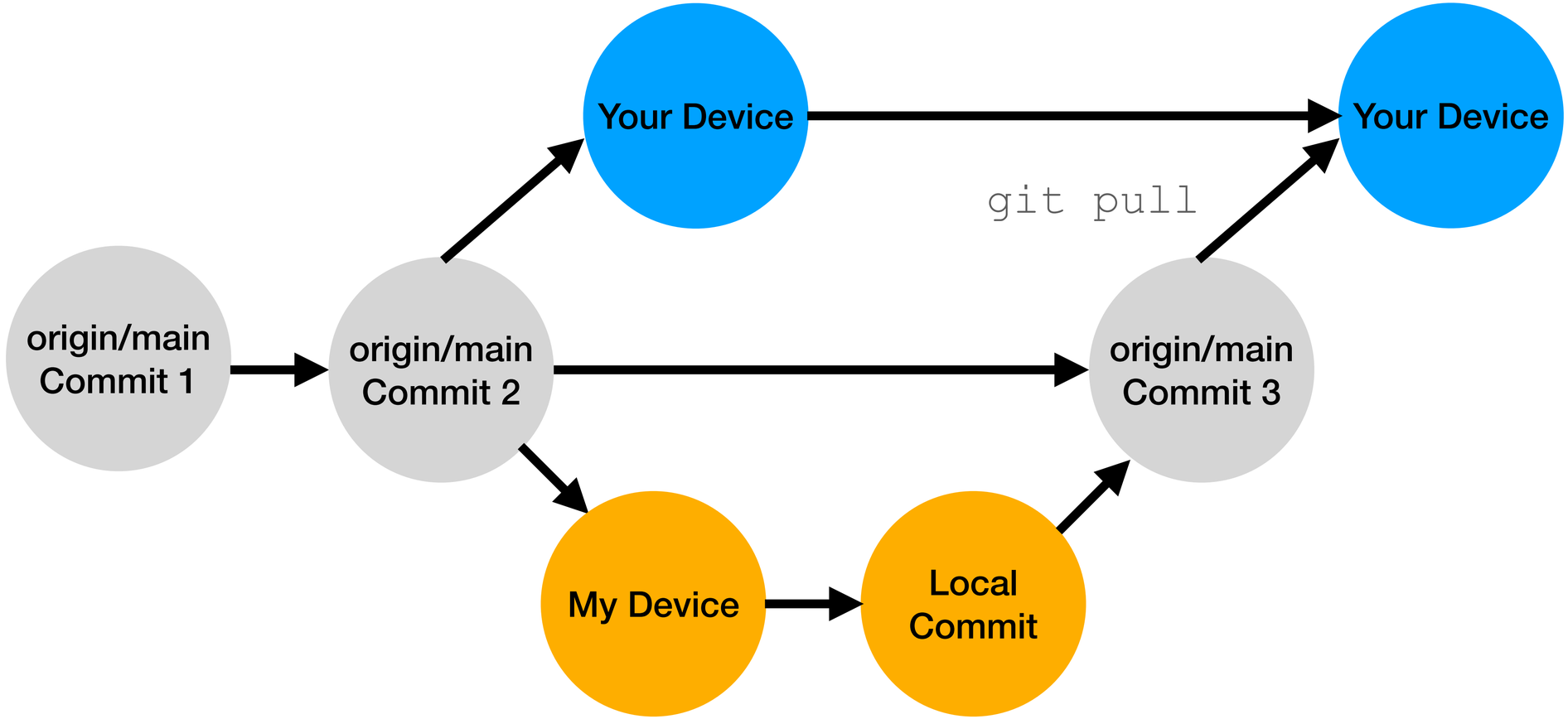
| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
| | | `git pull` |



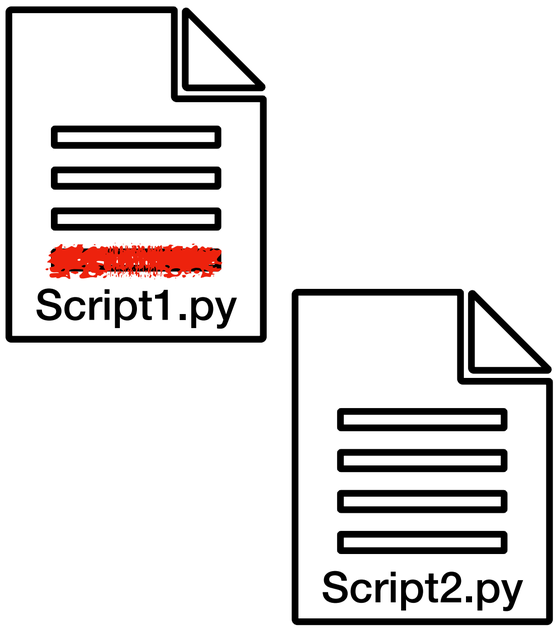
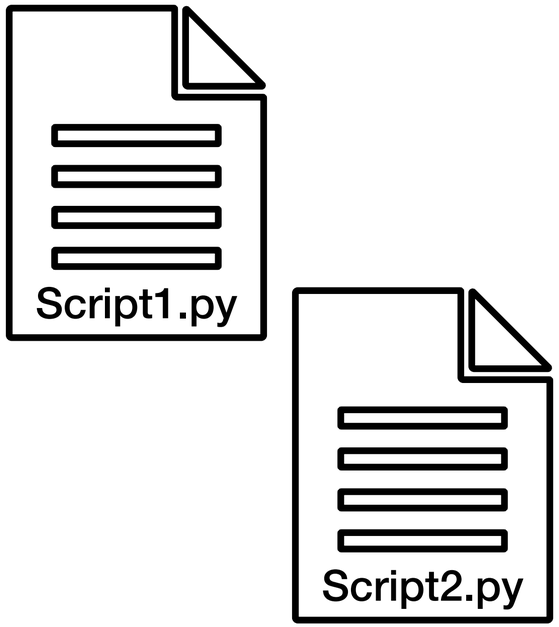
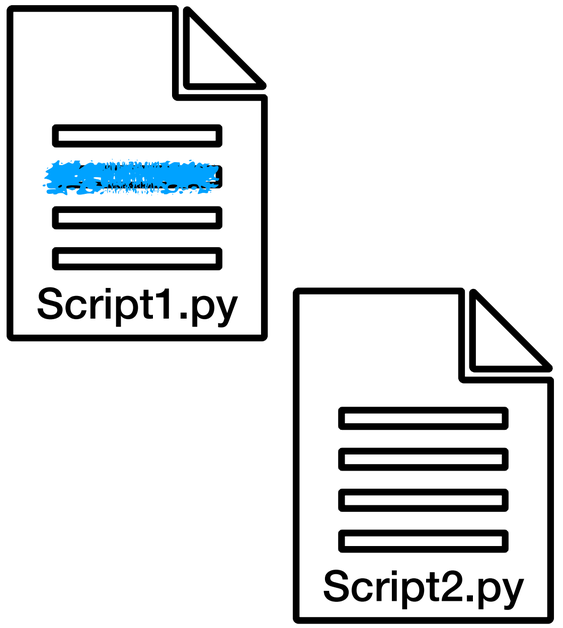



# Collaborating With Github

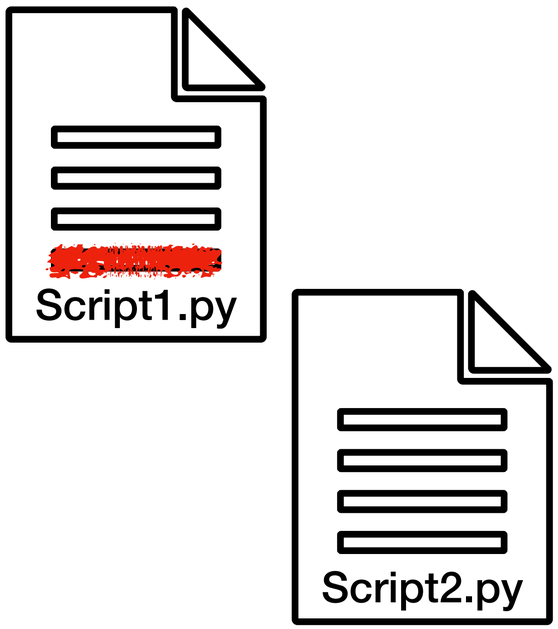
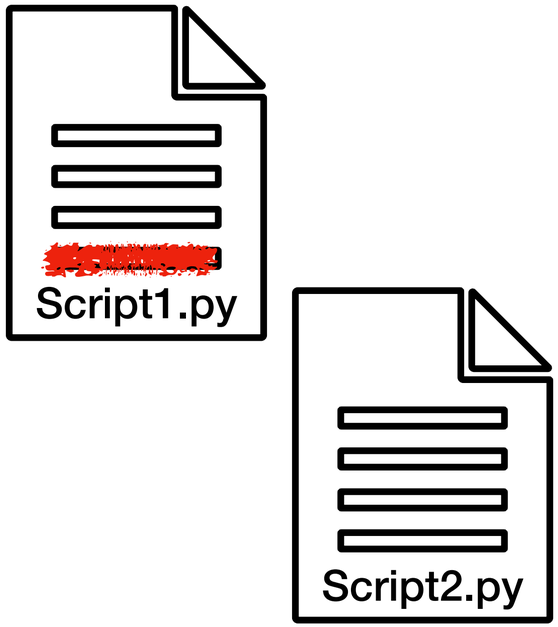
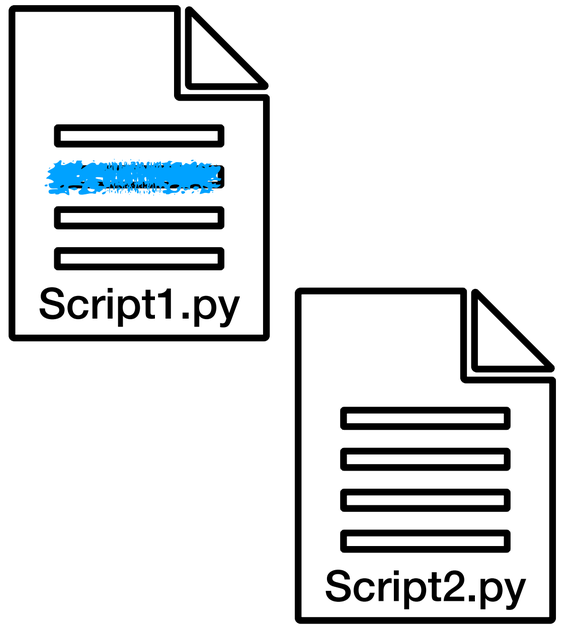


# Merging Different Snapshots

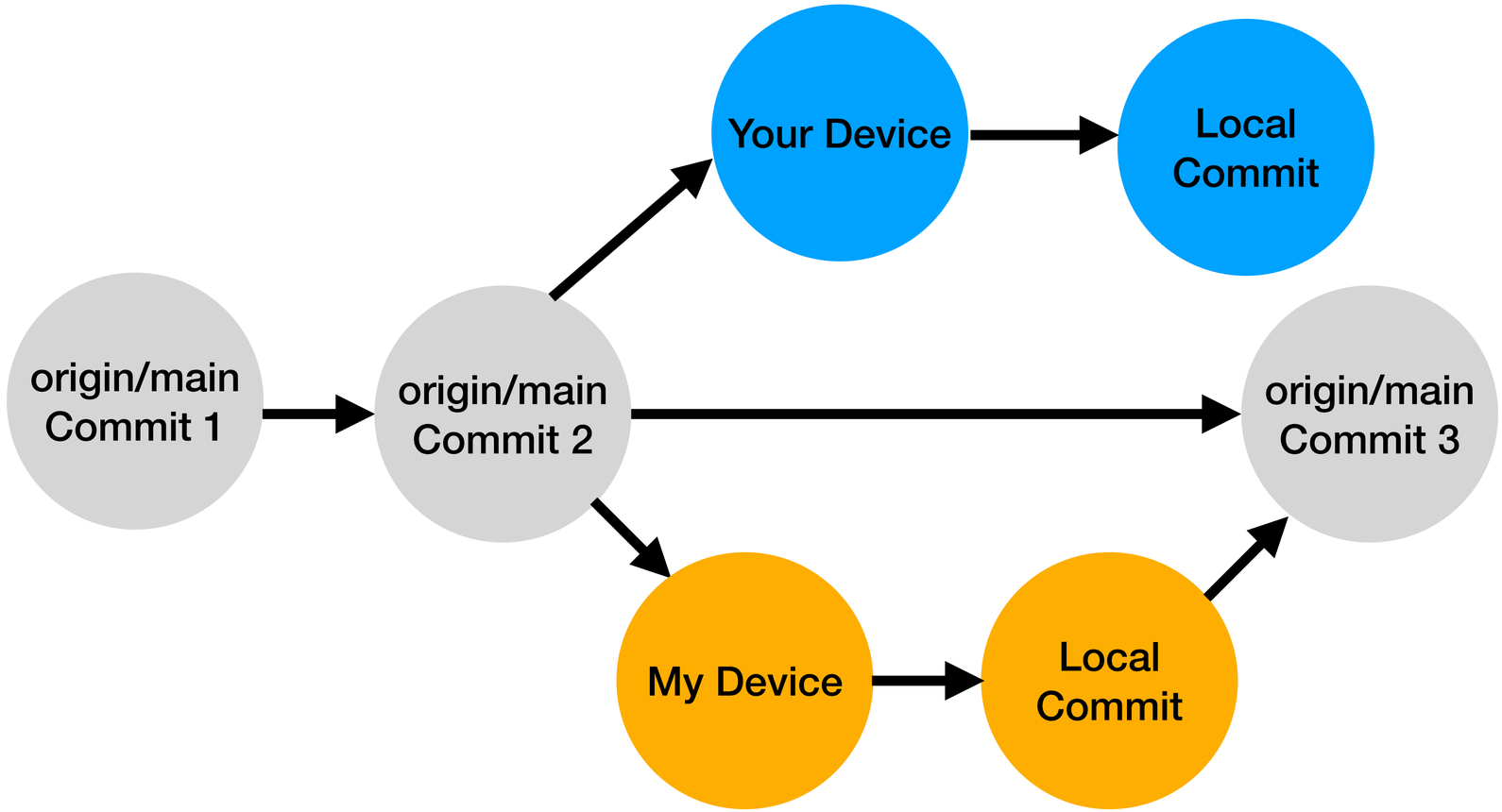
| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
|  |  |  |

If someone else pushes a new snapshot to the cloud first, git will raise an error if I try to push without pulling the new snapshot

# Merging Different Snapshots

| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
| `git push` | | |
|  |  |  |

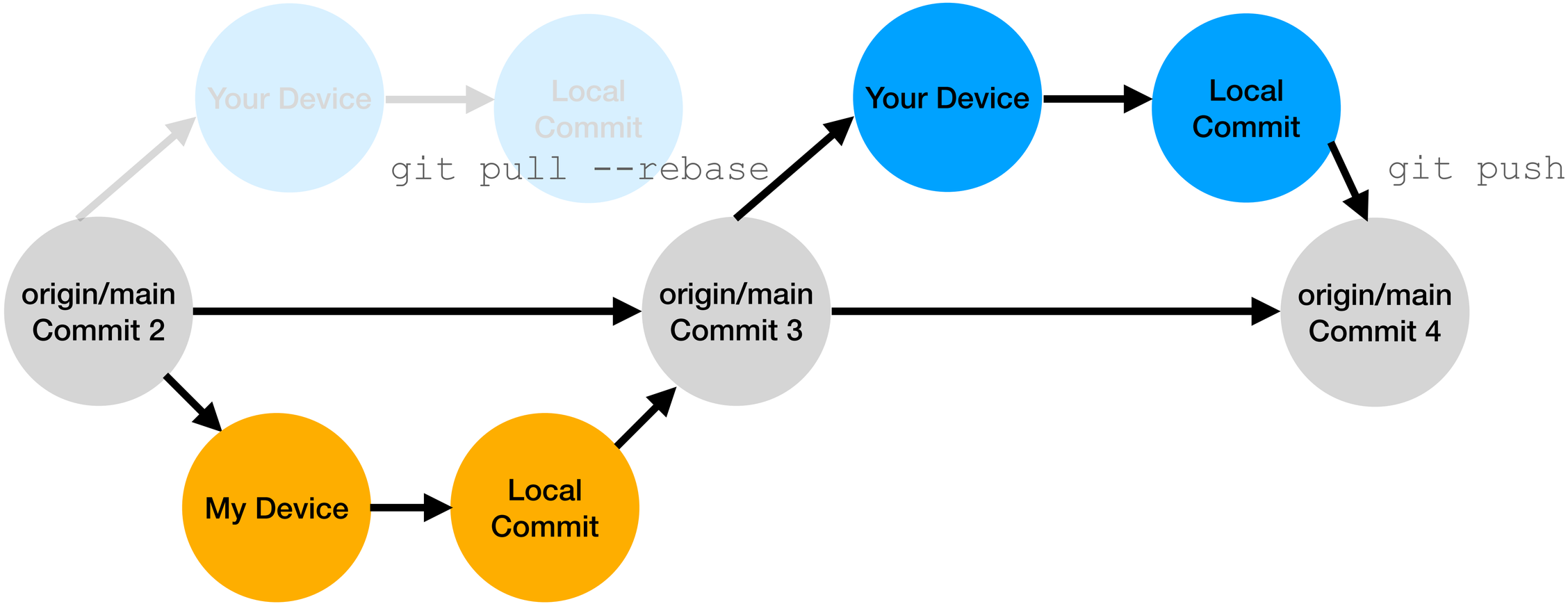
# Collaborating With Github



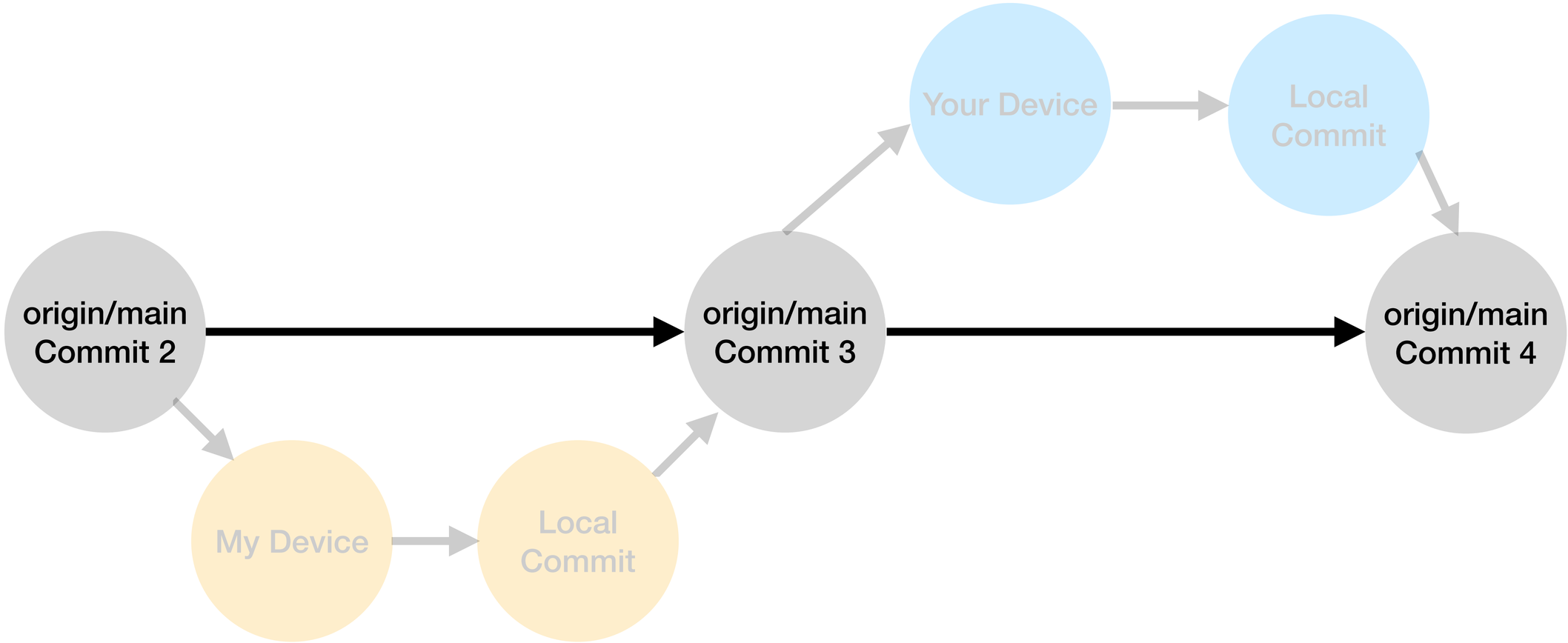
# Merging Different Snapshots

- There are three main ways to resolve diverging branches
  - Rebasing: `git pull --rebase`
  - Merging: After `git pull`, use `git merge`
  - Fast Forward: `git pull --ff-only`
- Ultimately they all do the same thing: merge your branch to the main branch
- They only differ by how they add your commit to the main branch

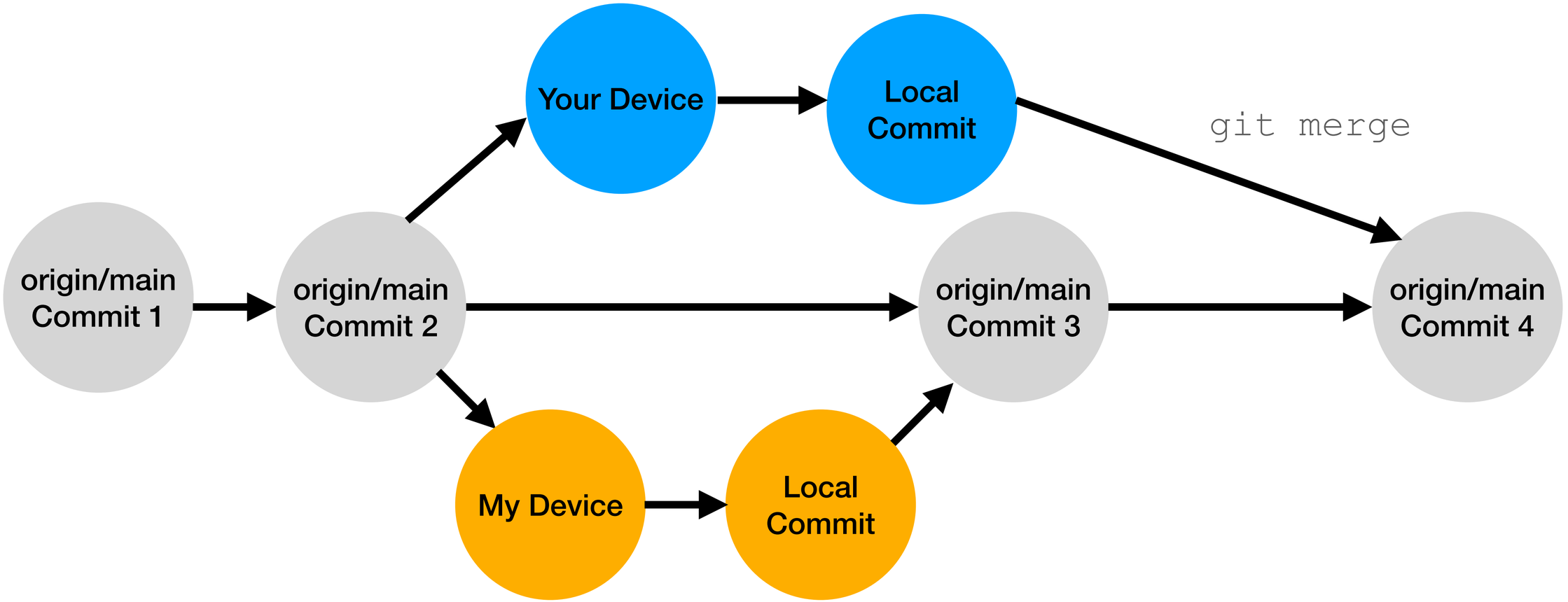
# Rebase


Your Device
Local Commit
git pull --rebase
Your Device
Local Commit
git push
origin/main Commit 2
origin/main Commit 3
origin/main Commit 4
My Device
Local Commit

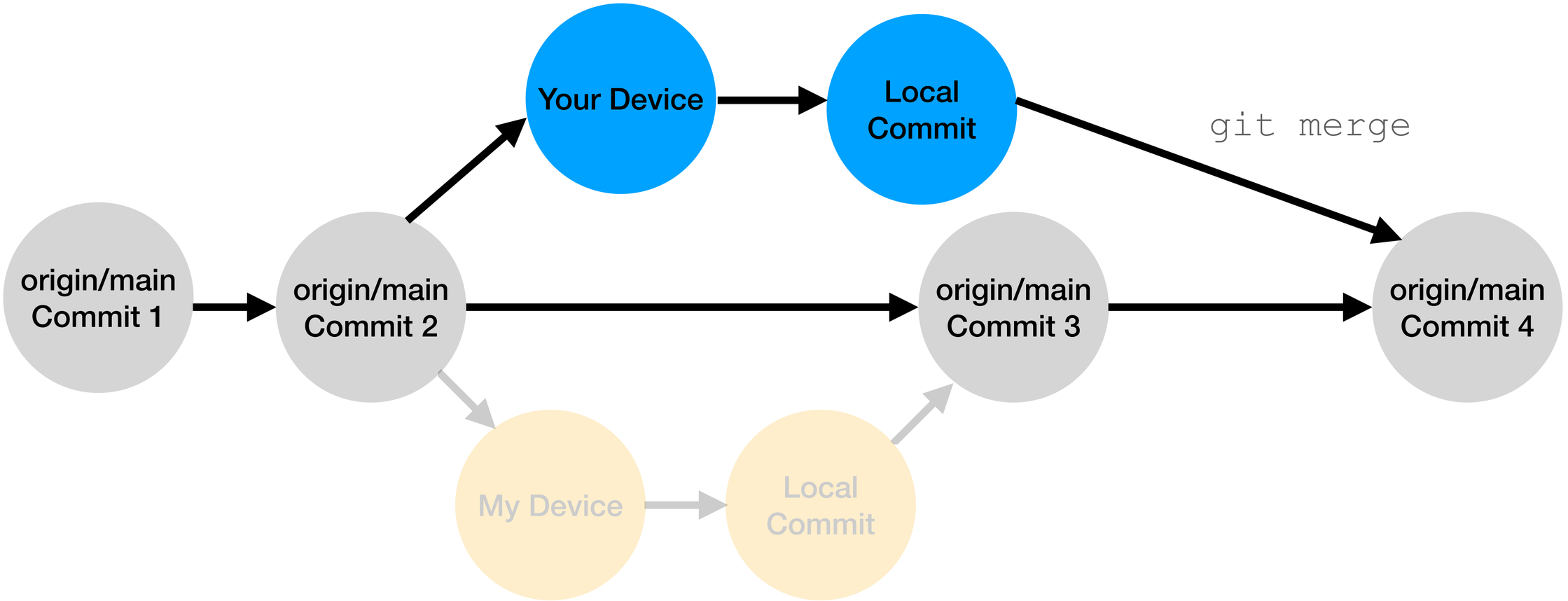
# Rebase (What git sees)



# Merge


Your Device
Local Commit
git merge
origin/main Commit 1
origin/main Commit 2
origin/main Commit 3
origin/main Commit 4
My Device
Local Commit

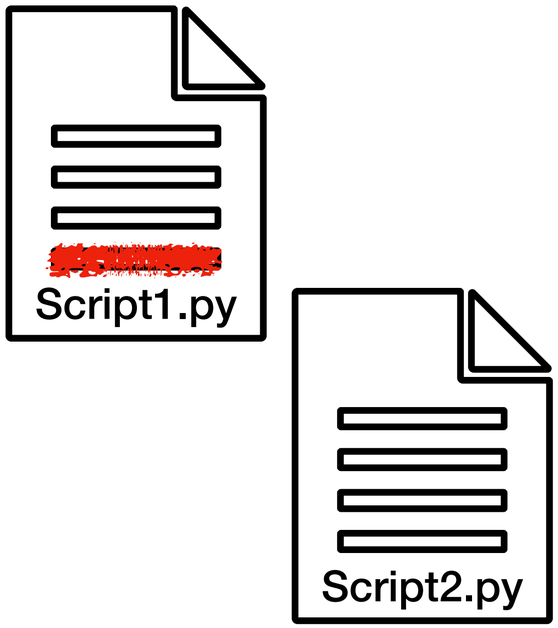
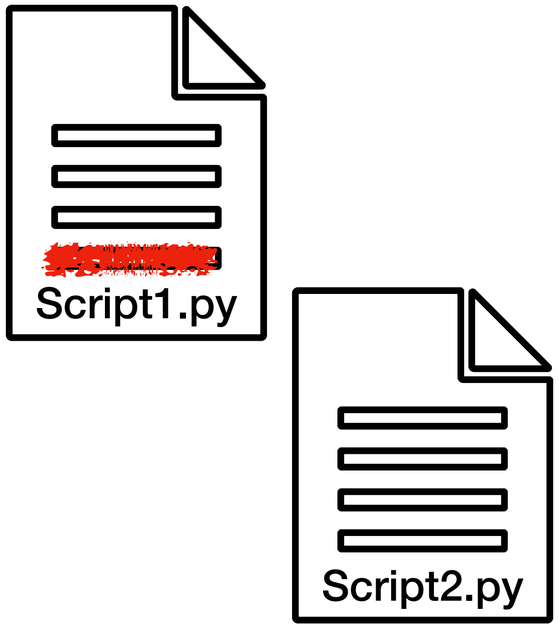
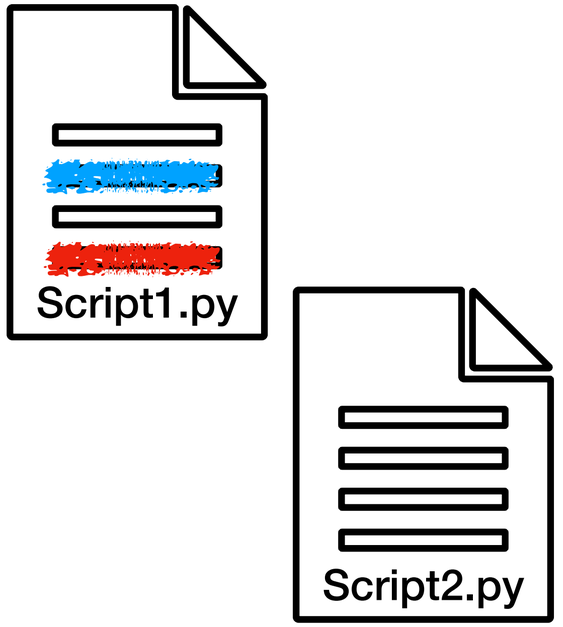
# Merge (What git sees)



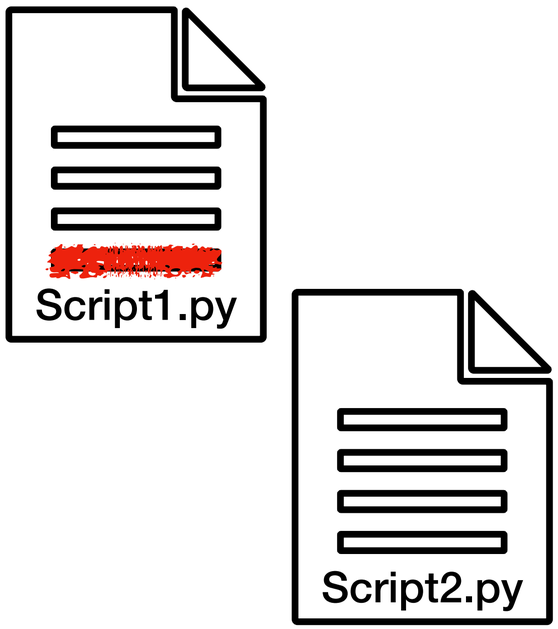
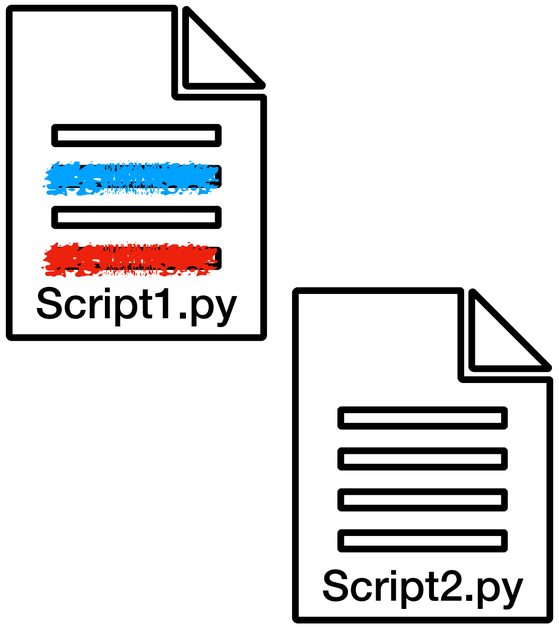
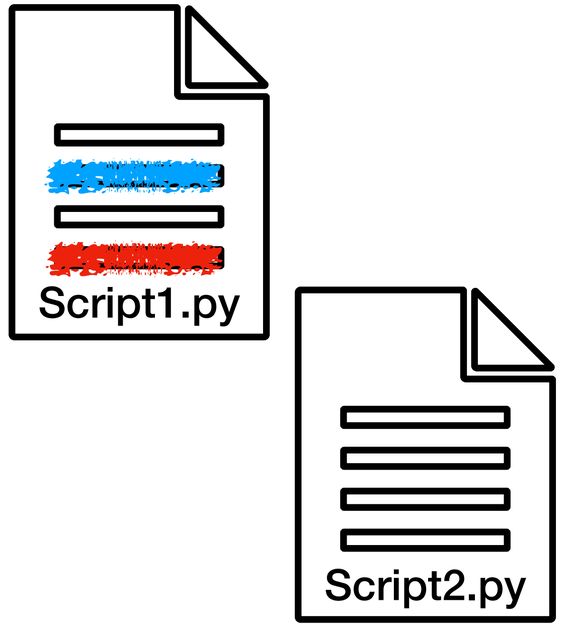
# Fast Forwarding

?????

# Merging Different Snapshots

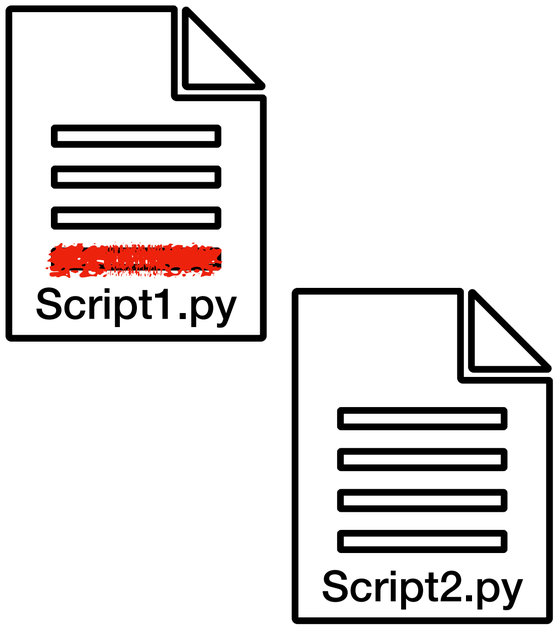
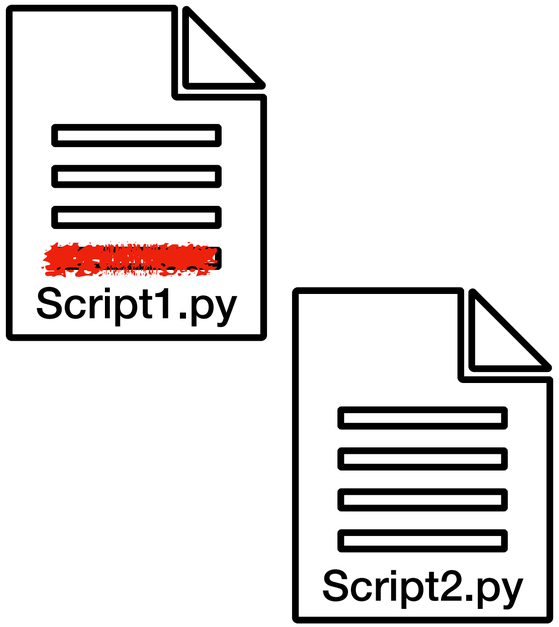
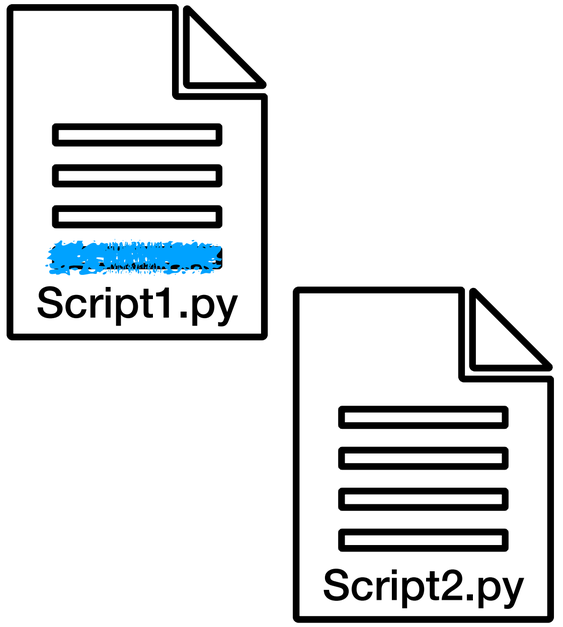
| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
| | | `git pull --rebase`<br>Or<br>`git merge` |
|  |  |  |

# Merging Different Snapshots

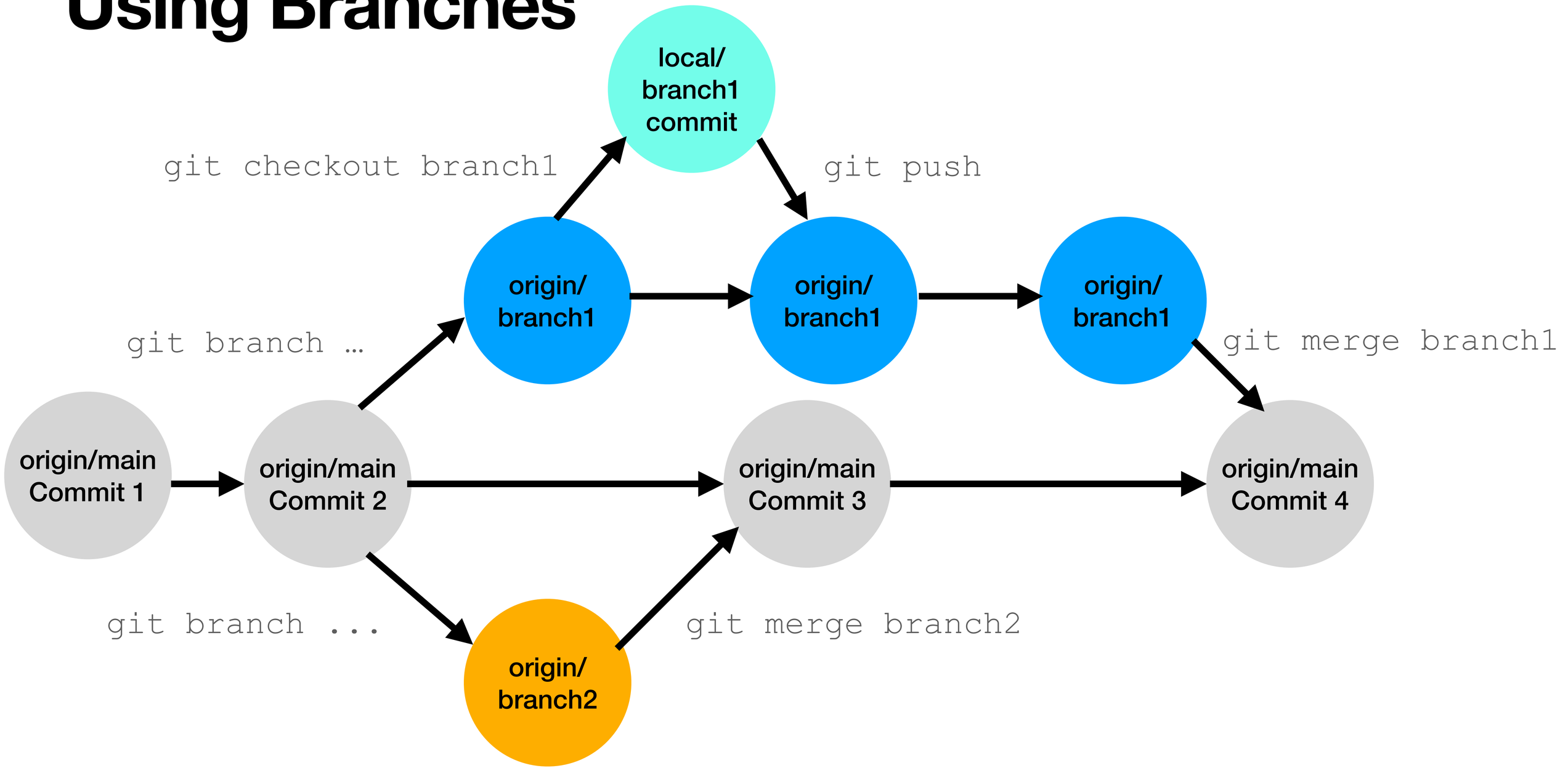
| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
| | | `git push` |
|  |  |  |

# Merging Conflicts

Have to manually resolve conflicts

| My Folder | The Cloud | Your Folder |
| --- | --- | --- |
|  |  | `git pull --rebase`<br>Or<br>`git merge`<br>  |

# Using Branches



# Creating Branches

1) List the local branches:

```
git branch
```

2) Create a new local branch:

```
git branch new_branch main
```

3) Switch to the branch:

```
git checkout new_branch
```

Note that changes that are **committed** to a branch do not appear in any other branch. Any edits in `new_branch` will not be seen when you switch to the main branch (`git checkout main`), even though you are in the same folder!

# Tracking the Branch on GitHub

1) Push the branch to origin (Github):

```
git push -u origin new_branch
```

2) To commit and push to the branch, make sure you are on the `new_branch (git checkout new_branch)` and commit and push as usual

3) Tracking the branch on different computers

a) Fetch Updates from GitHub:

```
git fetch
```

b) Checkout the branch:

```
git checkout new_branch
```

To see more info on on which branch your push are being sent to:

```
git remote show origin
```

# Merging and Deleting Branches

1) Merging branch to main:
    a) Make sure you are on the main branch:
        `git checkout main`
    b) Merge using rebase or merge:
        `git merge new_branch main` or
        `git rebase new_branch main`

2) Delete the local branch:
        `git branch -D new_branch`

3) Delete the origin branch (on server):
        `git push origin --delete new_branch`

# Helpful Resources

- Atlassian has a very helpful and informative guide on using branches and merging (https://www.atlassian.com/git/tutorials/using-branches)

- An Introduction to Git and GitHub (http://www.bioinf.org.uk/teaching/splats/git.pdf)

- Use Git Cheatsheets (https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet)

- Have a bookmark folder just for git commands