GitHub

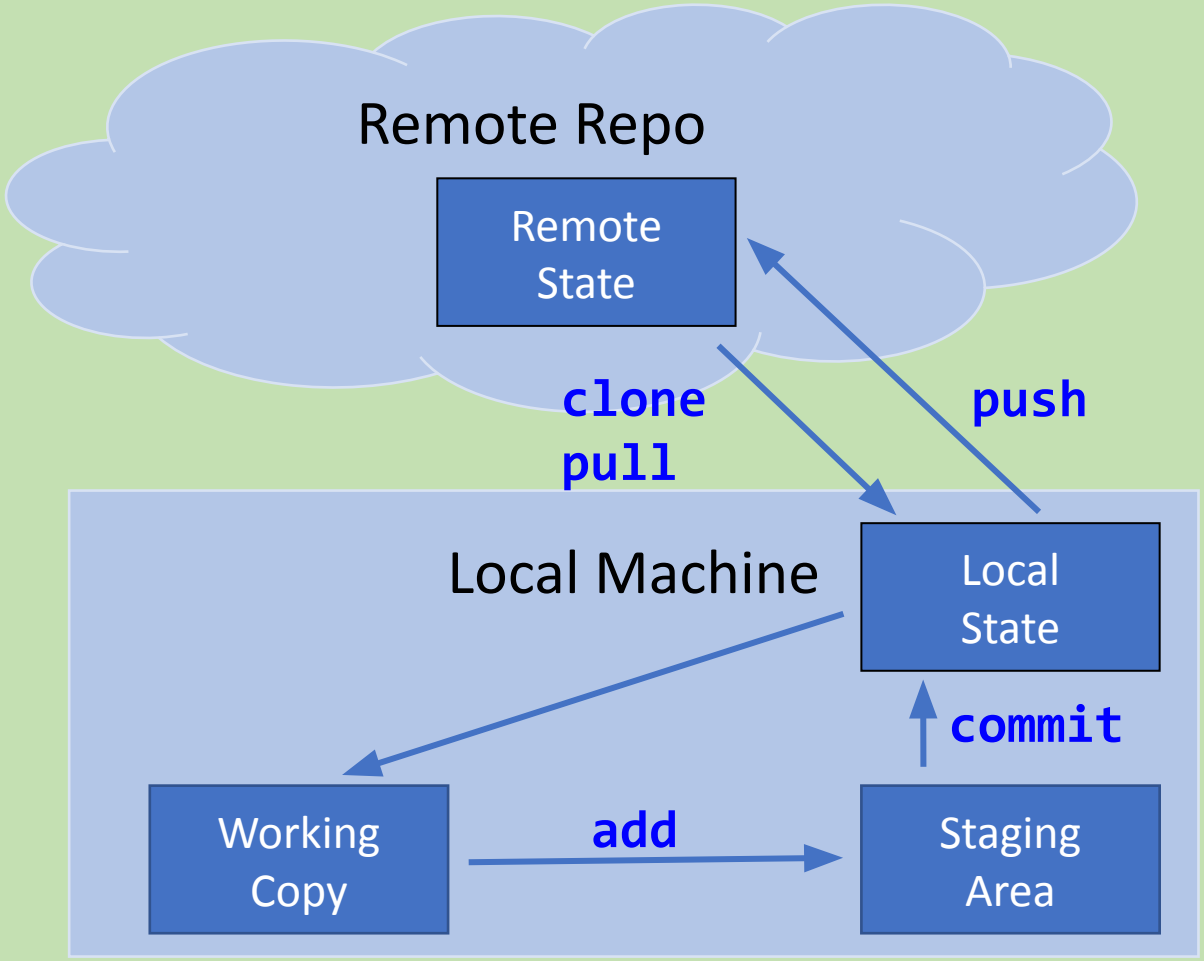
Skylar Gering and Julia Sloan

Part 2

Setup Steps

1. VSCode installed
2. Julia installed
3. Git installed
4. Github account
5. Send us the email account associated with your Github (if you didn’t on Wednesday)

Review


Remote Repo
Remote State
clone
pull
push
Local Machine
Local State
commit
Working Copy
add
Staging Area

1. Open VSCode

2. Open terminal in VSCode

3. If you don’t have the repo from Part 1:
   a. Go to the repo on Github, click the Code button, and copy the HTTPs URL
   b. Run `git clone `url`` from the terminal

4. Navigate to the `drawings/` folder in your terminal

5. Run `git pull`

Review Activity

1. Find a buddy to work with

2. One person in each pair: make a copy of `sample_drawings.jl` in the `drawings/` folder

3. Name this new file something unique to your group

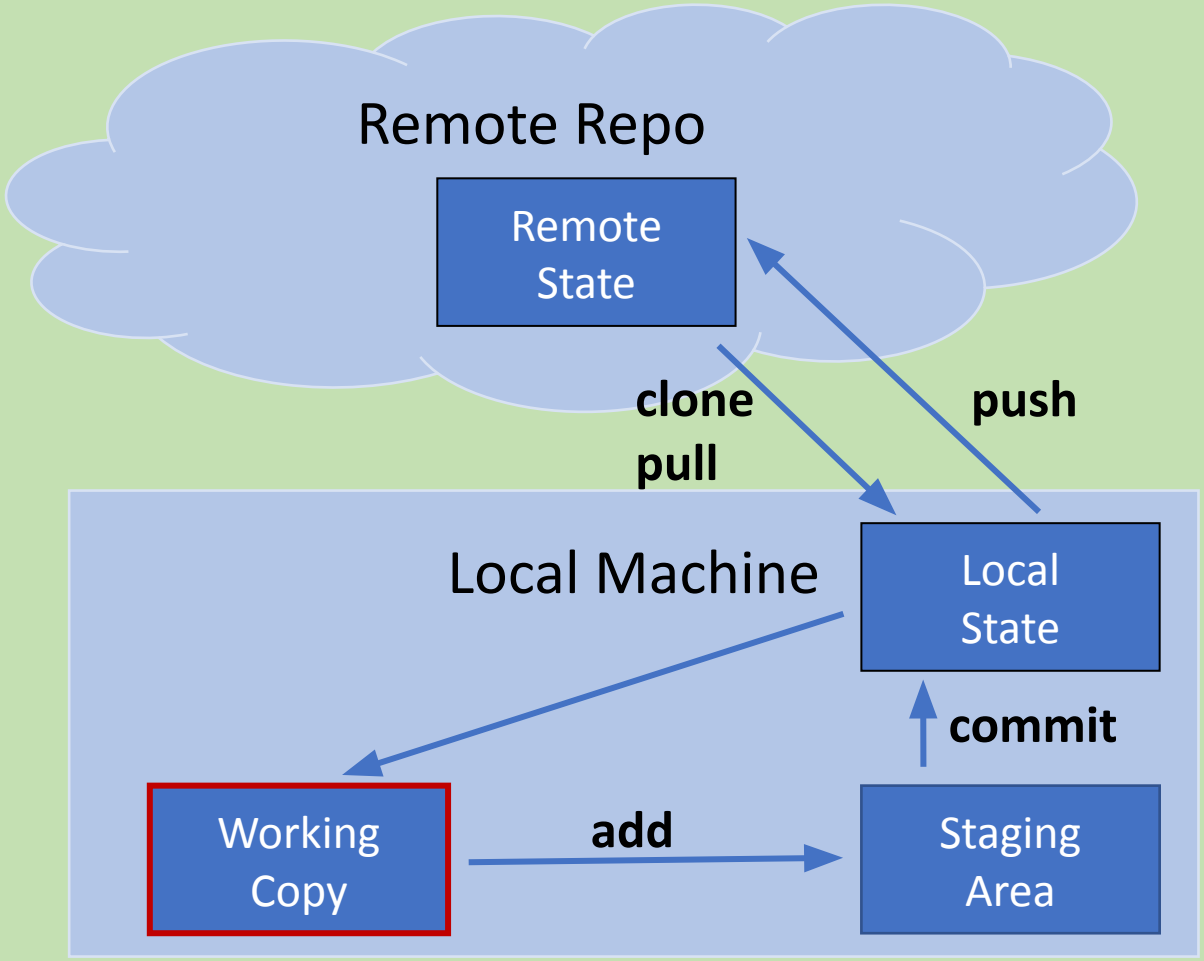
Review Activity

1. WAIT FOR US TO TELL YOU → ONE GROUP AT A TIME
    a. Buddy with the copy of `sample_drawings.jl`:
        i. Add and commit your new file
        ii. Pull any changes from the repo
        iii. Push your updated version of the repo with the new file (one group at a time!!!)
    b. Have your buddy pull so you each have a copy of it

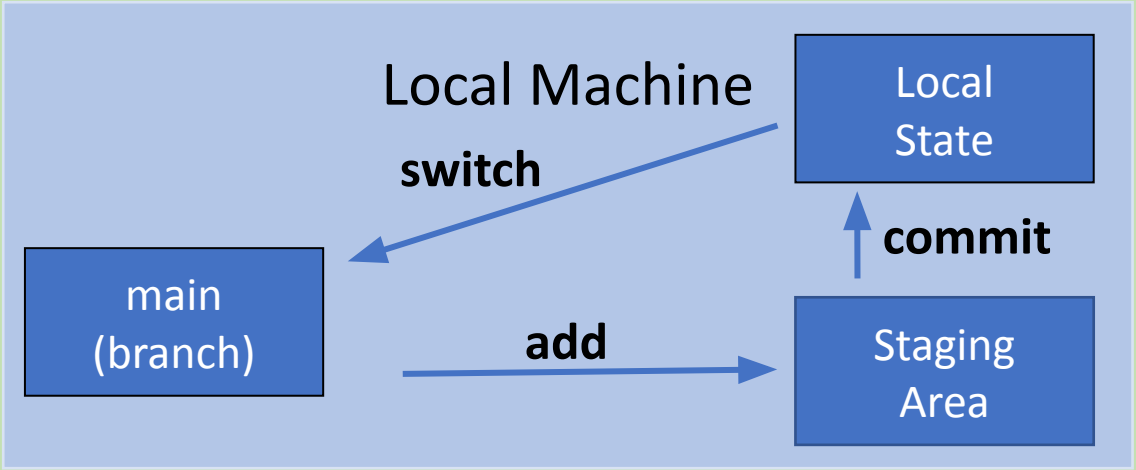
What are Branches?

- Different, and related "working copies" of your code
- Uses:
  - Add a new feature while maintaining original code for other people to use
  - Experiment with new ideas without having to make copies of all your code
  - Collaboration with others
- Main Branch
  - Most up-to-date, stable version of the code
  - Branch that people using your code will clone

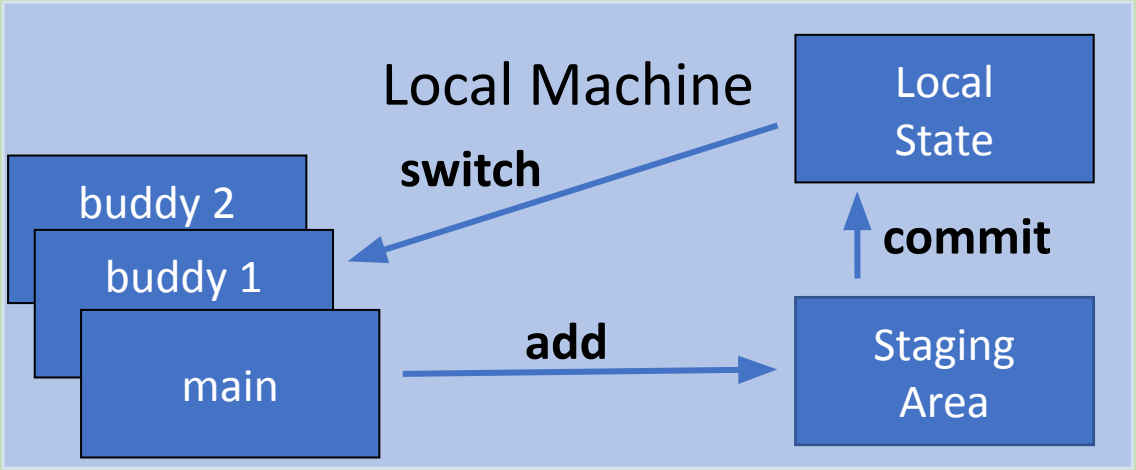
What are branches?



What are branches?

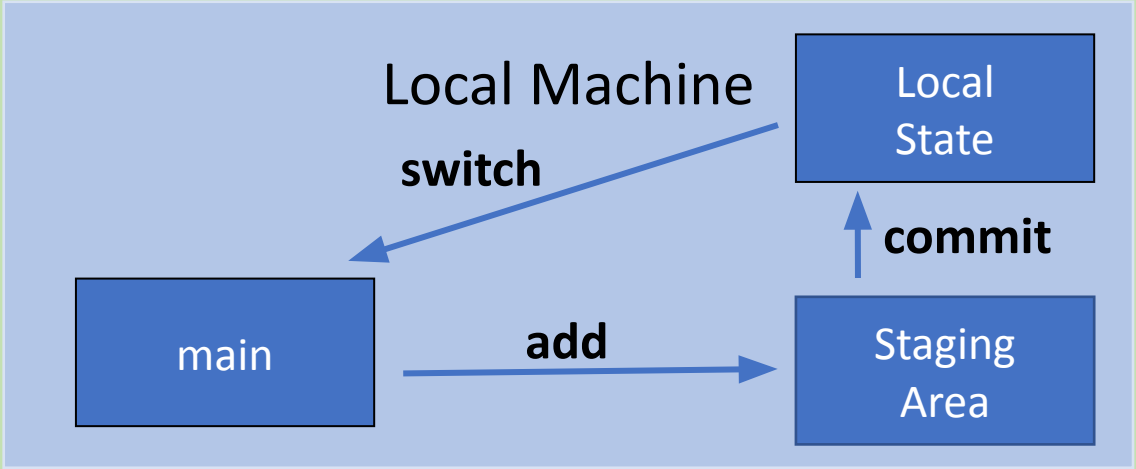


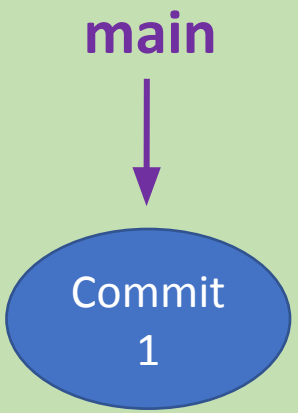
What are branches?



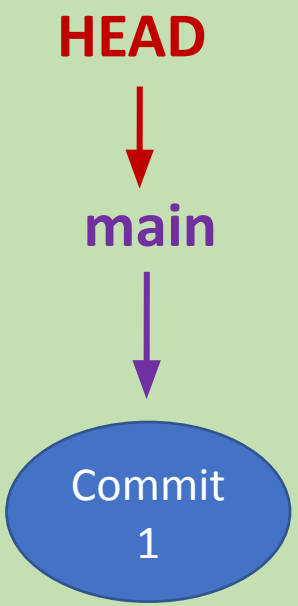
1. Run `git branch `your_name``
2. Run `git branch` ... what do you see?
3. Run `git switch `your_name``
4. Run `git branch` ... what changed?

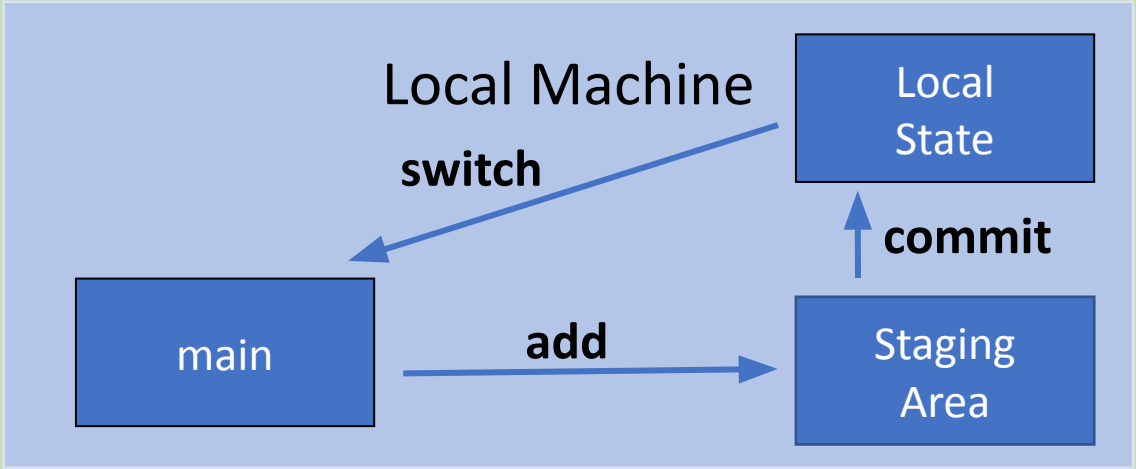
Making a Branch



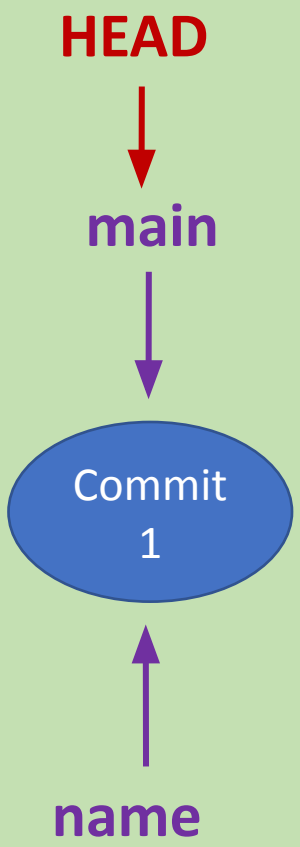


Making a Branch

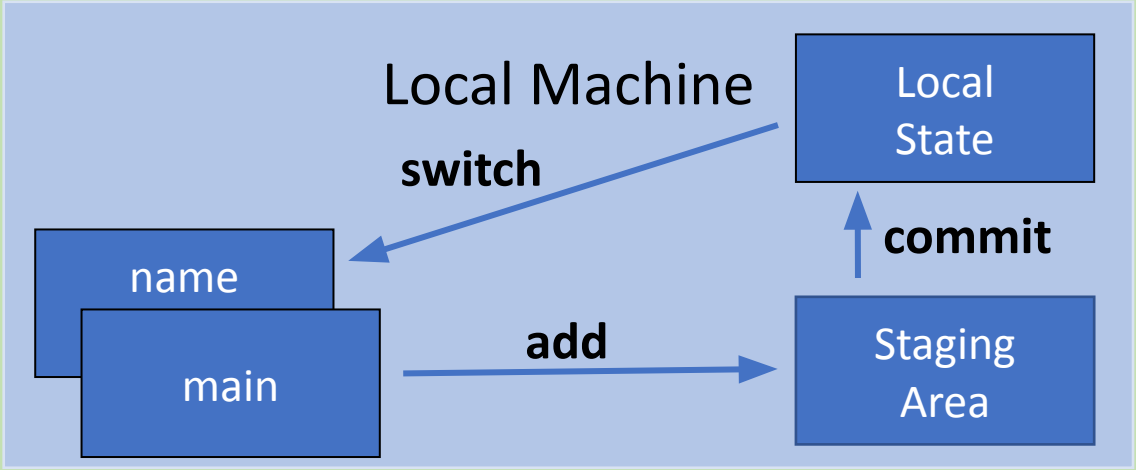




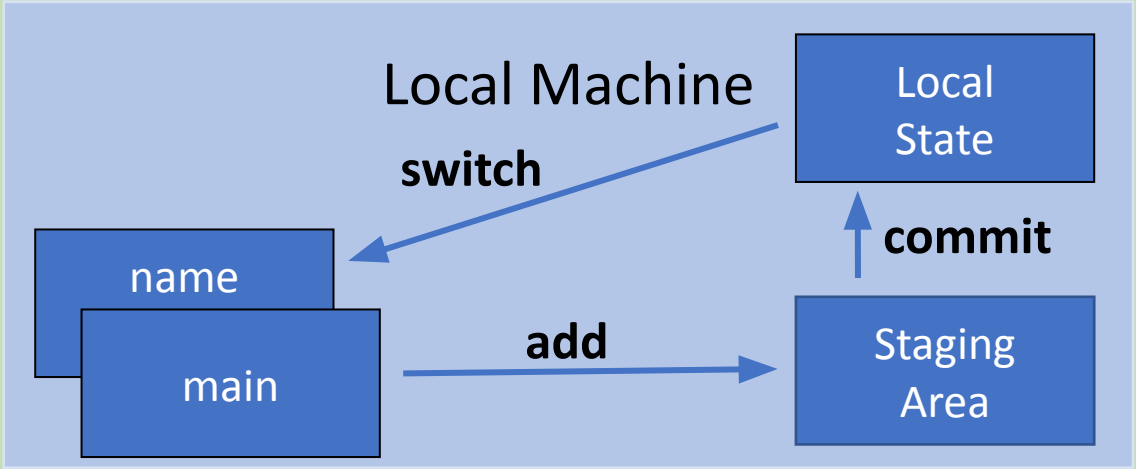
Making a Branch

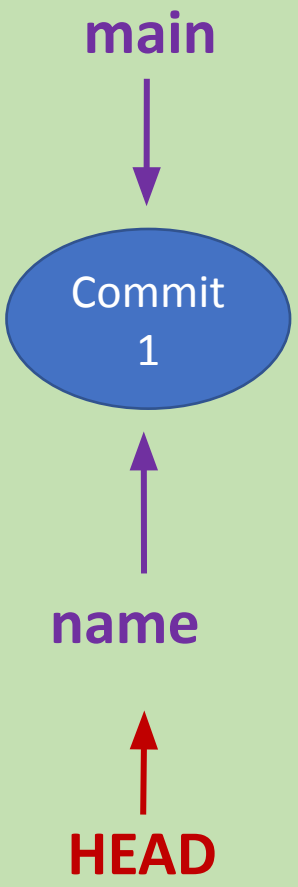



```
git branch name
```



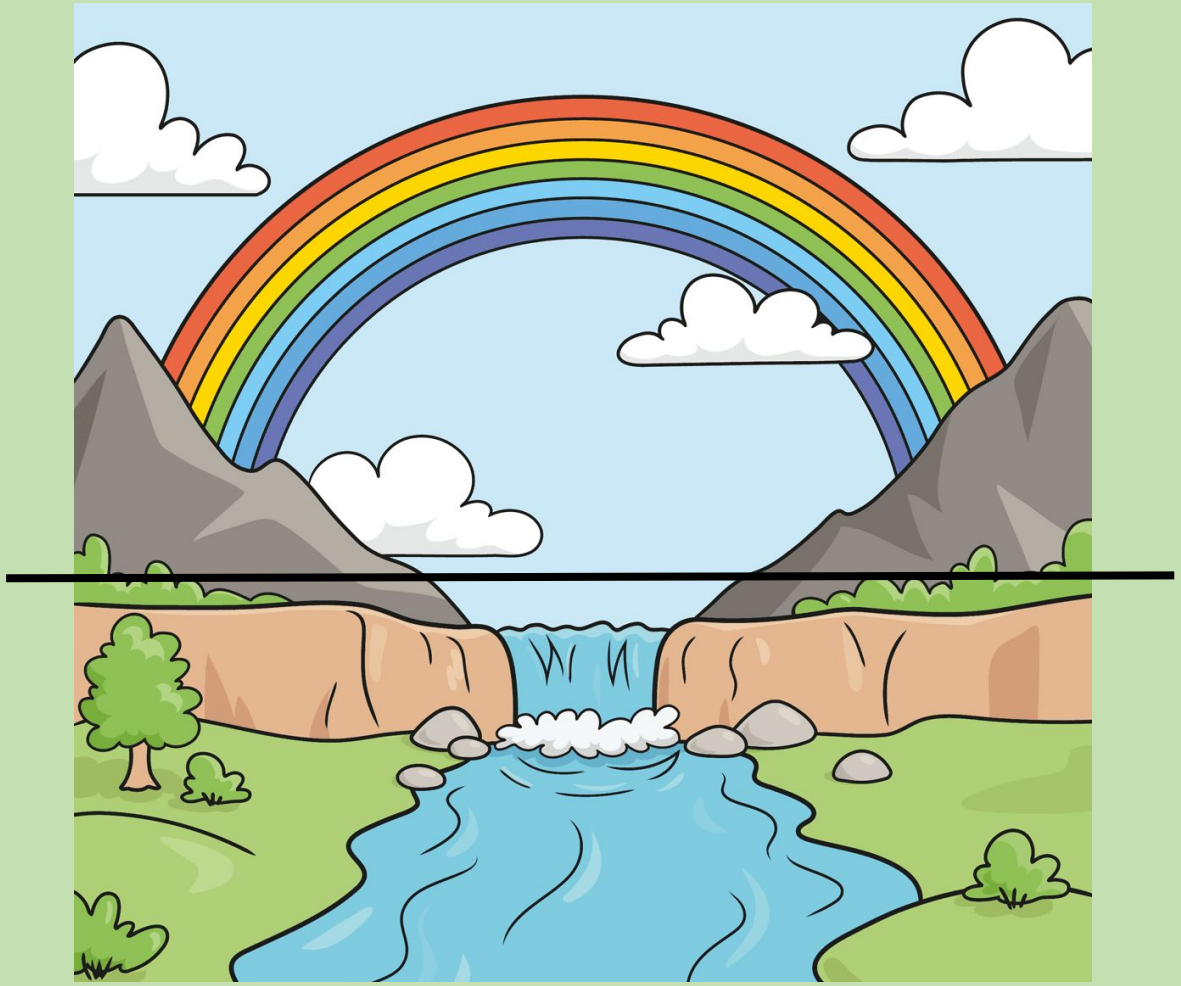
Making a Branch






```
git switch name
```

Make a drawing using emojis!



sample_drawing.jl | octo_gals.jl | sky_julia.jl

drawings > sample_drawing.jl

```
# Use the emojis in the emoji bank to make a drawing.
# Leave the line start comments, but you can remove t

#  ---------------------------
# |                           |
# |                           |
# |                           |
#  ---------------------------
```

emoji_bank.jl

drawings > emoji_bank.jl

```
# 🌨️ 🌪️ 🌩️ 🌧️ 🌦️ 🌤️ 🌞 🌈

# ⭐ ✨ 🌠 🌌 🌃 🌉 🌙 🌁

# 🌱 🎄 🌳 🌴 🌵 🌷 🌻 🍄

# 🎡 🏠 🏘️ 🏢 🏊 🏄 🏇 🏖️

# 🐿️ 🐌 🐝 🕊️ 🦎 🦕 🌳 🐞

# 🐳 🐙 🐟 🐠 🐡 🐬 🦑 🦦

# 🐠 🐊 🚀 🦋 🐦 🦉 🦜 🐝
```

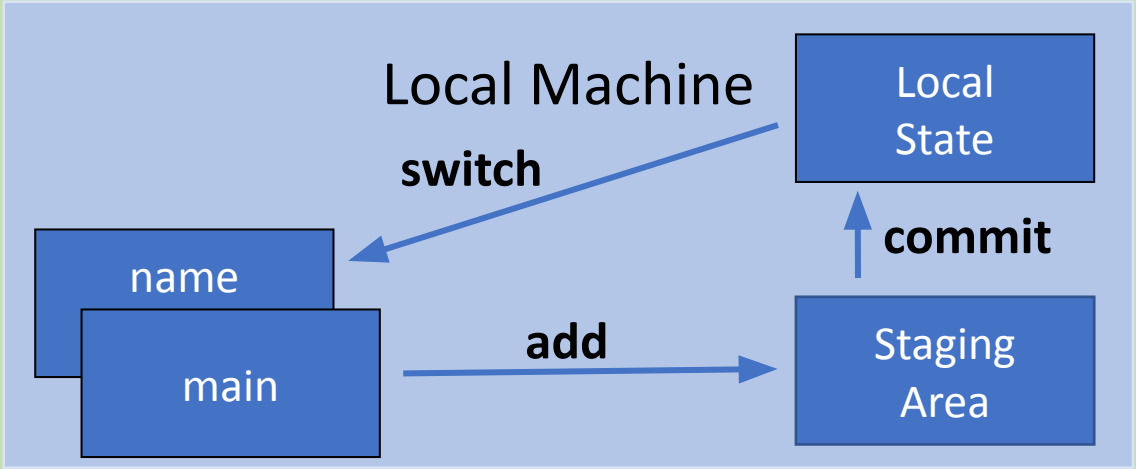
1. Add your first emojis to your drawing
   a. If you're making the top of the drawing, add emojis to represent **time of day**
   b. If you're making the bottom of the drawing, add emojis to represent the **surface type** (land, ocean, etc.)
2. Add and commit your changes with a meaningful commit message

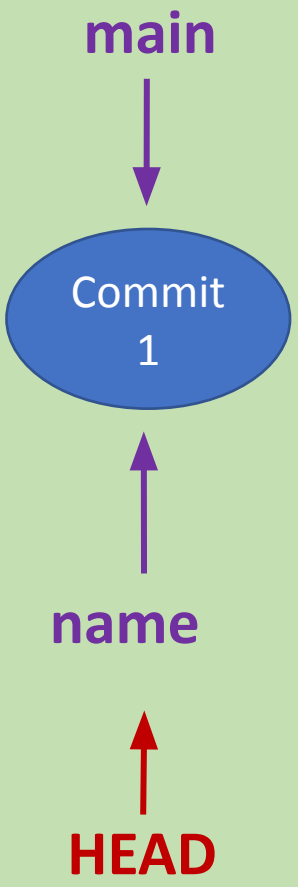
1. Now add a few **animals** to your picture

2. Add and commit your changes with a meaningful commit message

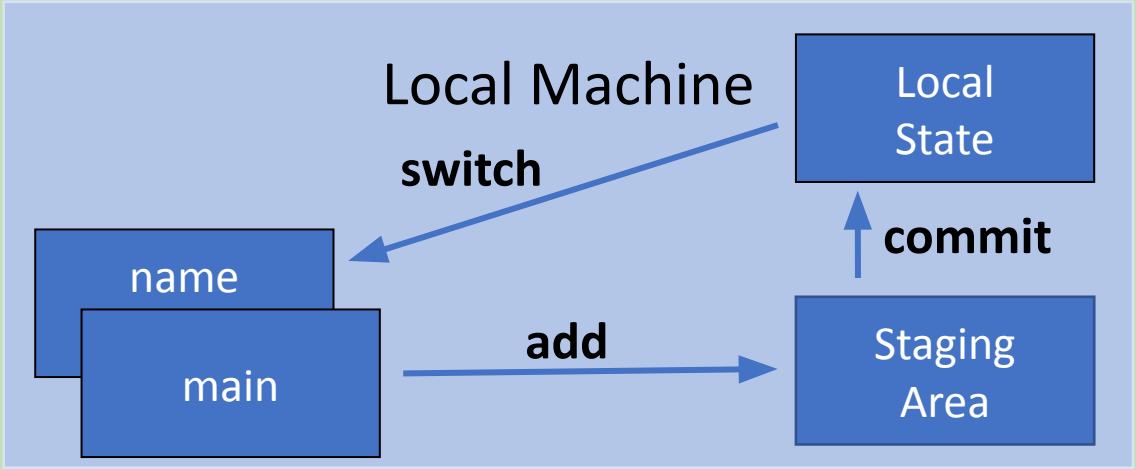
1. Add any other emojis you want to your picture
2. Add and commit your changes with a meaningful commit message
3. Run `git log`

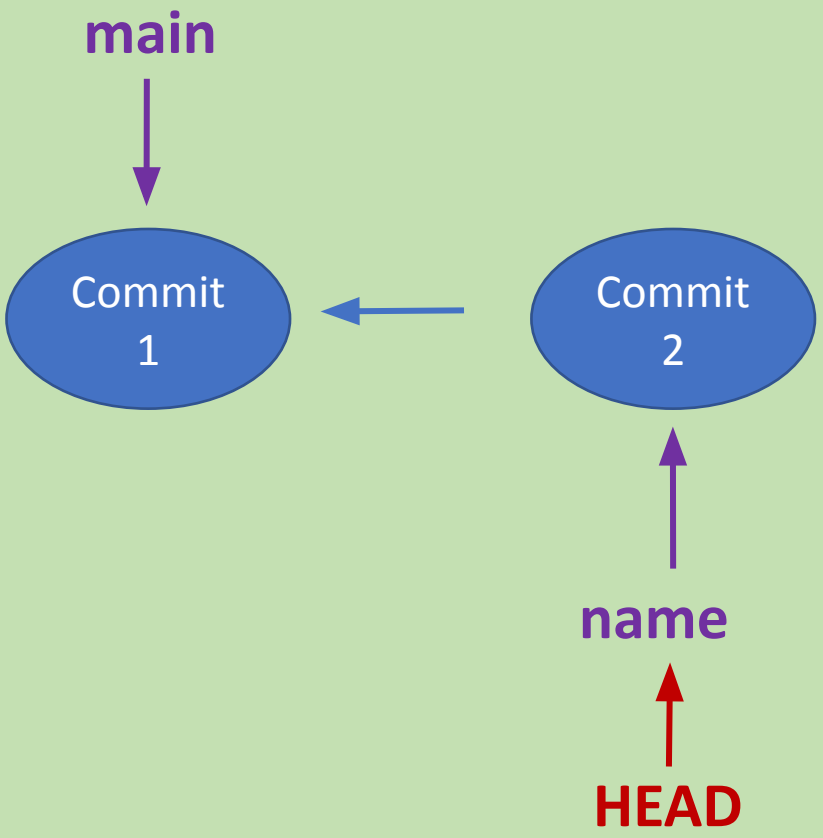
Making a Branch



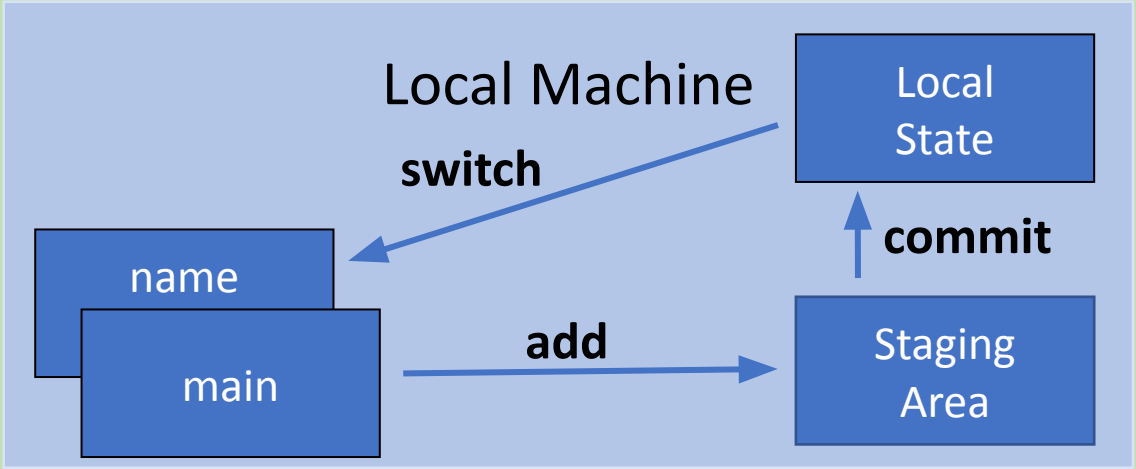


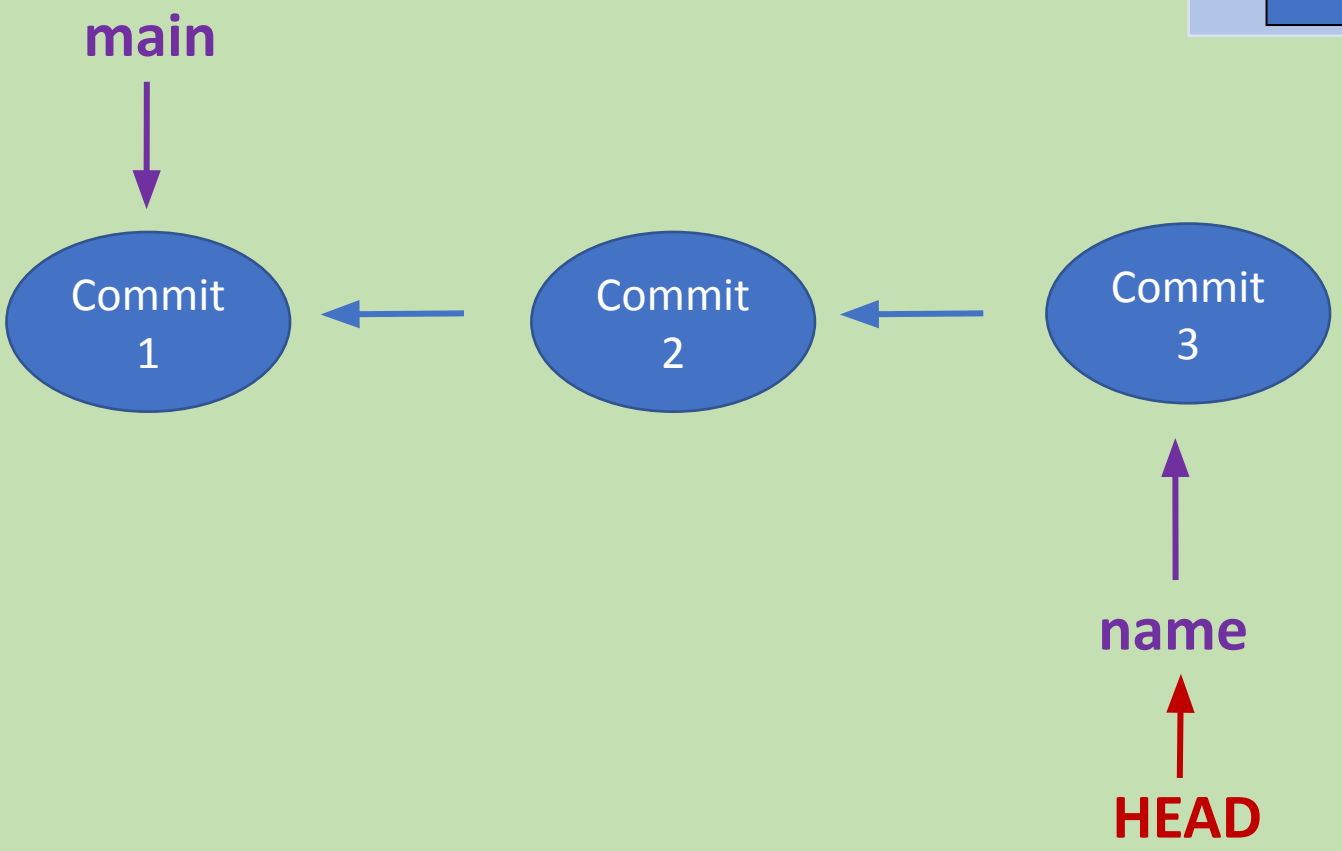
Basic Branch



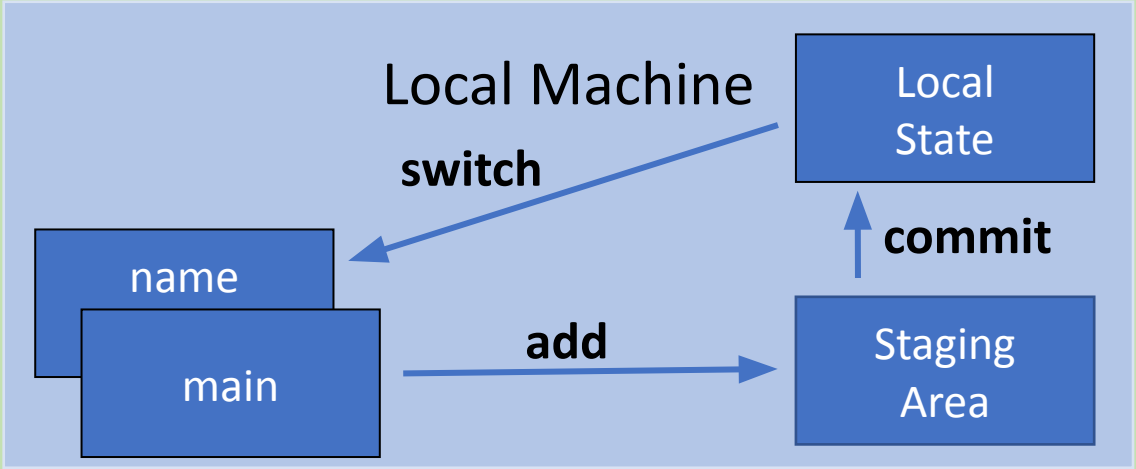


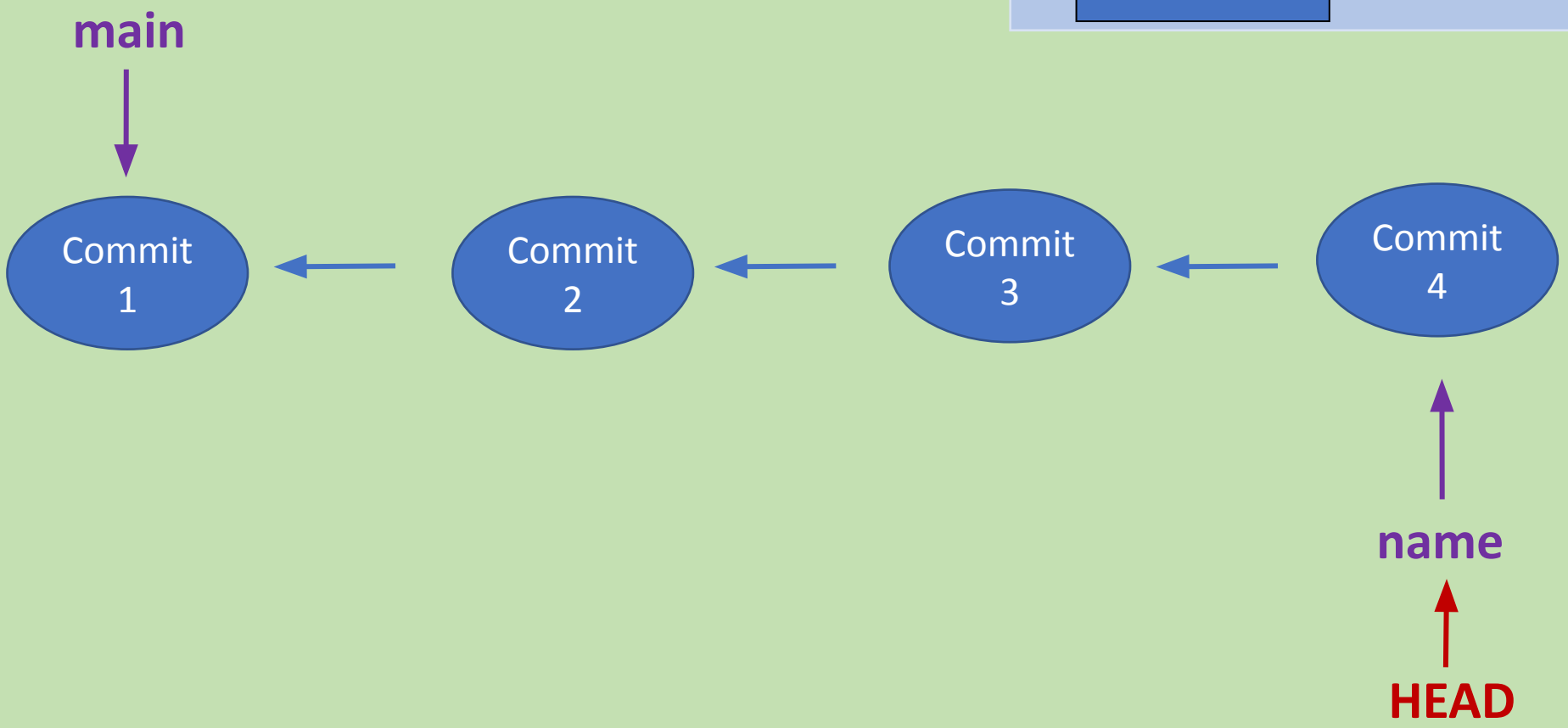
Basic Branch





Basic Branch

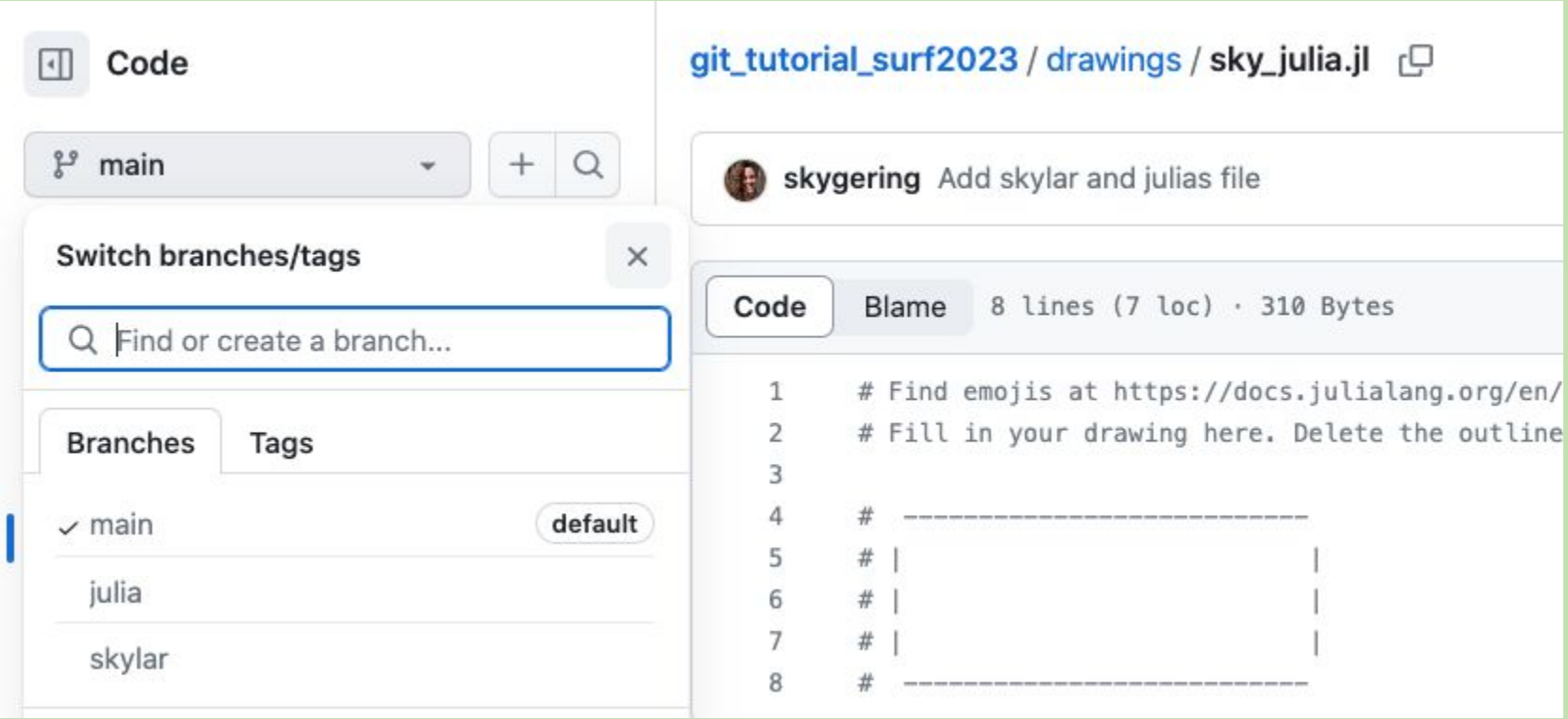




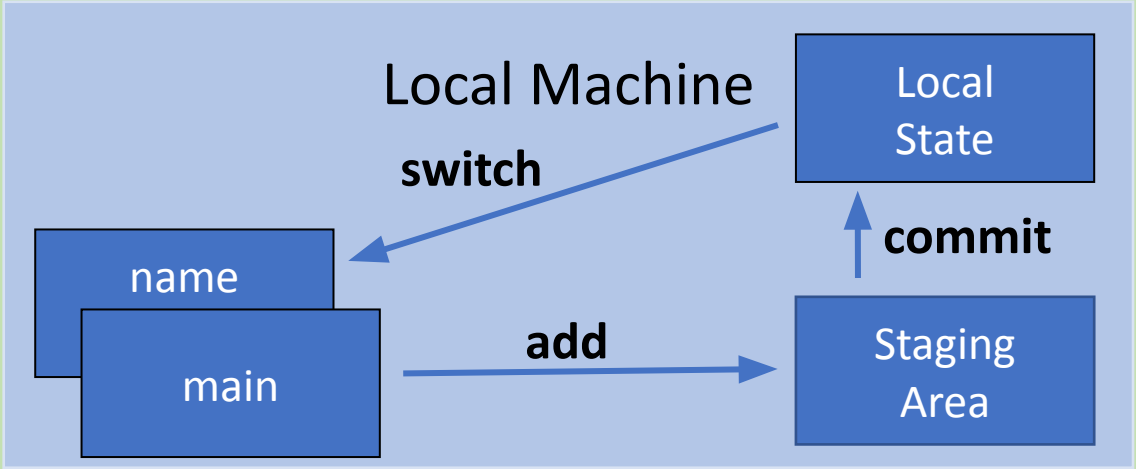
1. Push your code to the remote copy of your branch

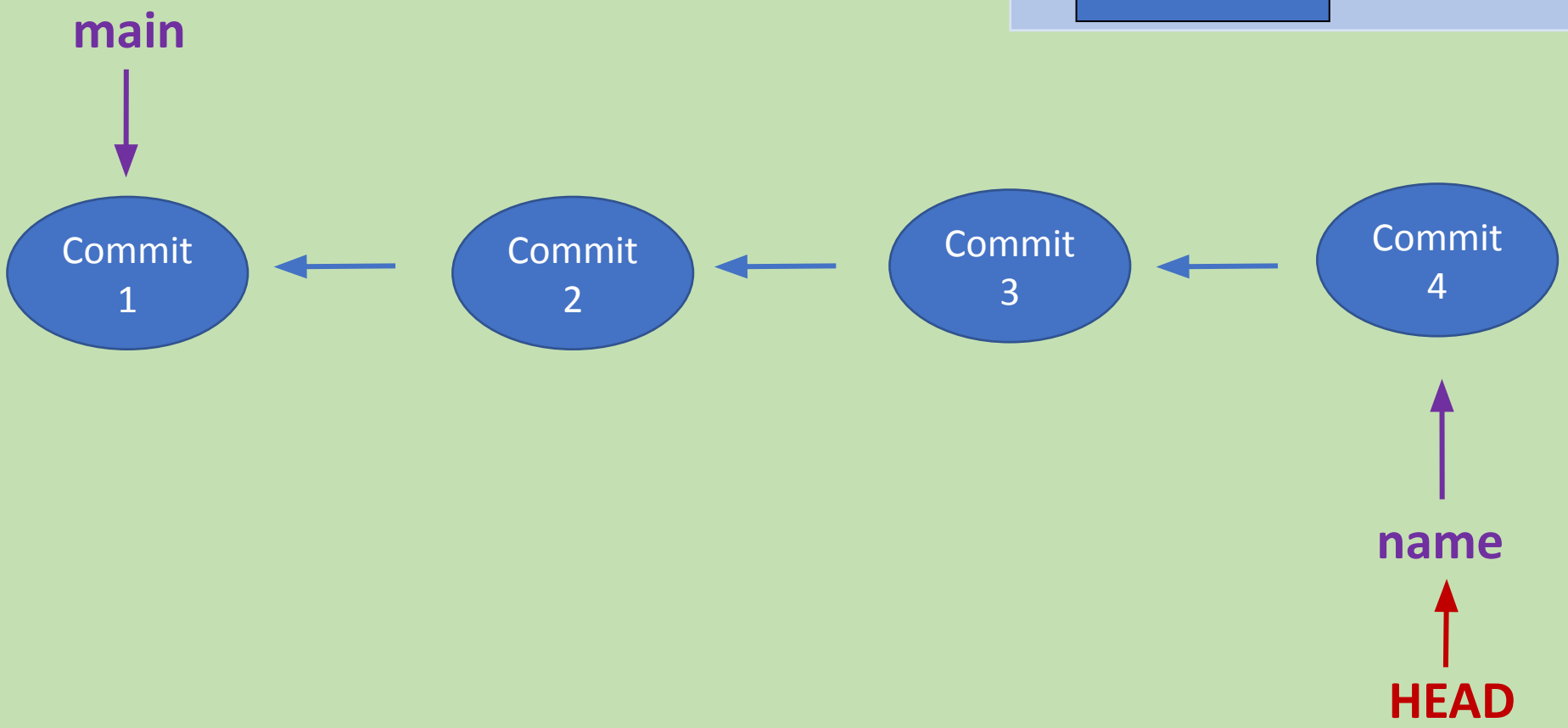
2. When you get an error, run the line it tells you to set the upstream branch
   a. `git push --set-upstream origin `name``

Look at the repository online. Click on your group's file. What do you see if you switch from main to your branches online?

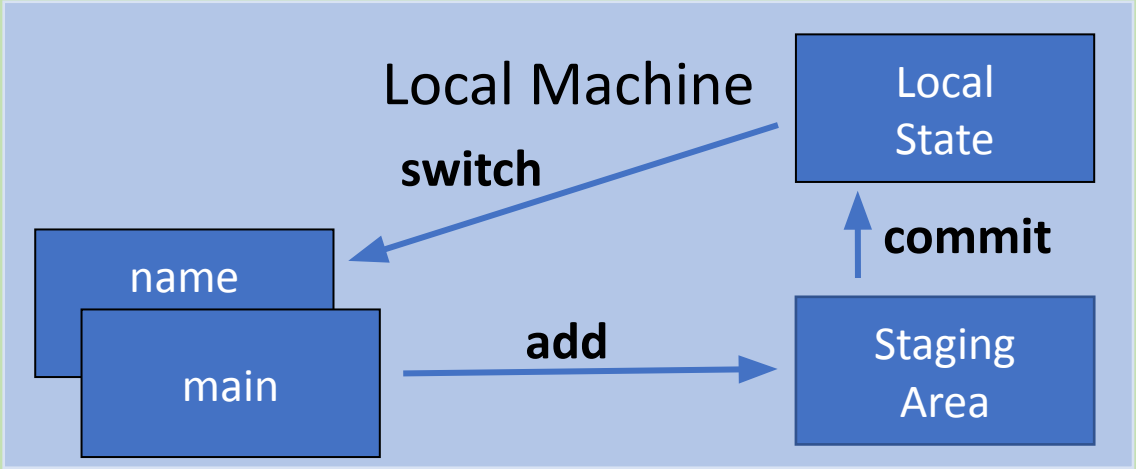


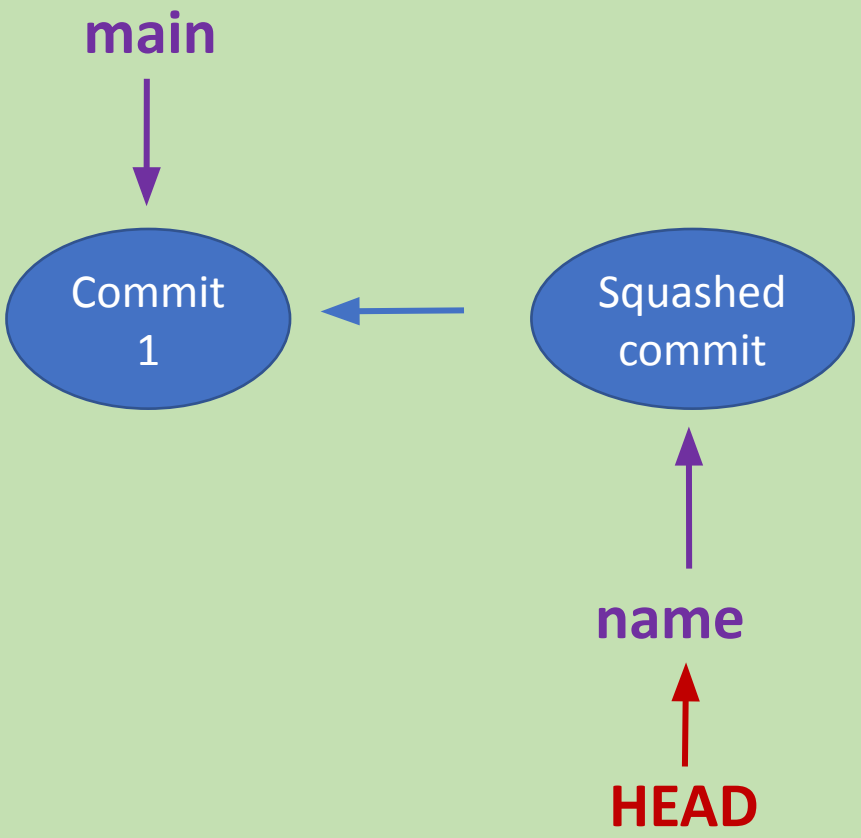
Basic Branch





Basic Branch






```
git rebase -i HEAD~n
```

1. Run `git log` and count how many commits you've made locally (`n`, which should be 3)
    a. **Note: This uses vim**
        i. Type `:q` to exit

1. Squash the commits into one
    a. Run `git rebase -i HEAD~n`
    b. **Note: The rebase editor uses vim**
        i. Type `i` to enter insert mode (so you can write)
        ii. Hit the `esc` key to exit insert mode
    c. Leave `pick` in the first line as it is
    d. For the rest of the lines, change `pick` to `squash`
    e. Exit vim by typing `:wq`

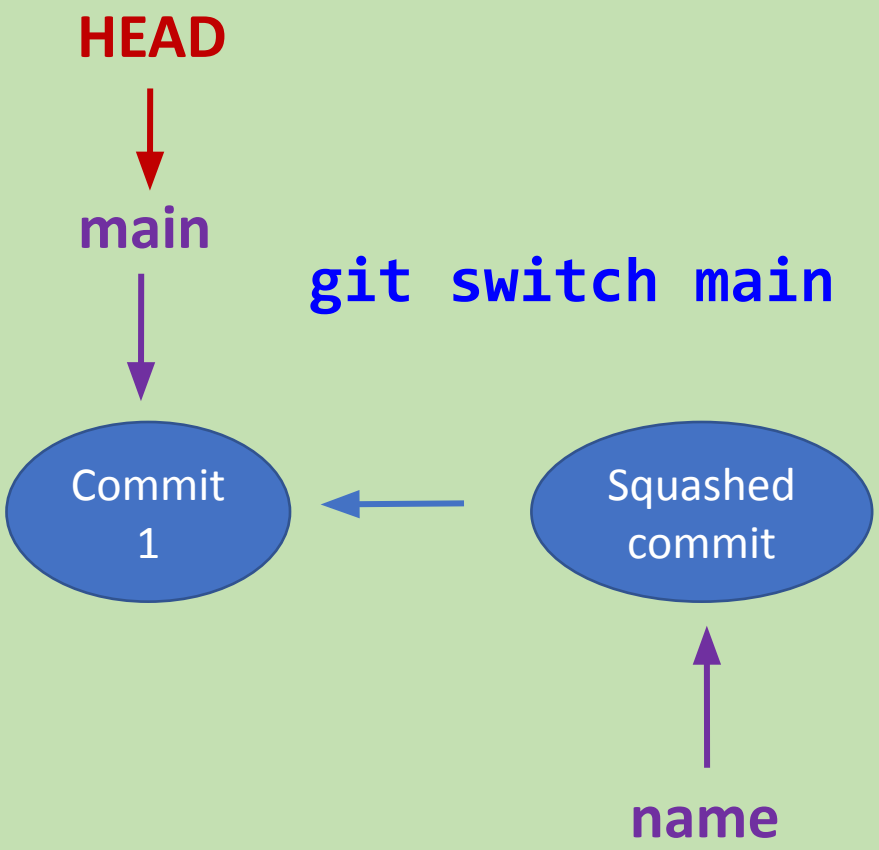
2. On the next page that comes up, exit vim again

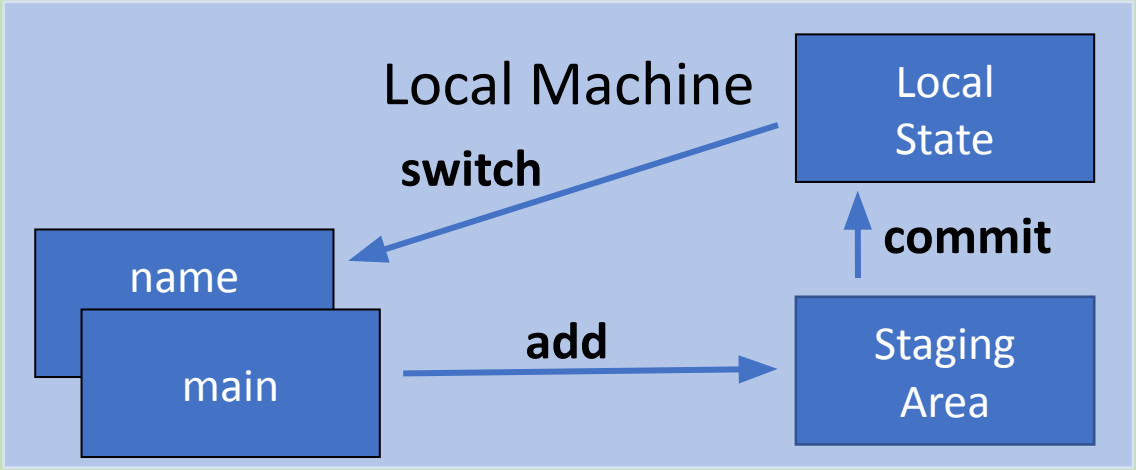
1. Run `git log`

2. How many of your commit messages do you see?

1. Run `git switch main`
2. Look at your file - what looks different?

Basic Branch



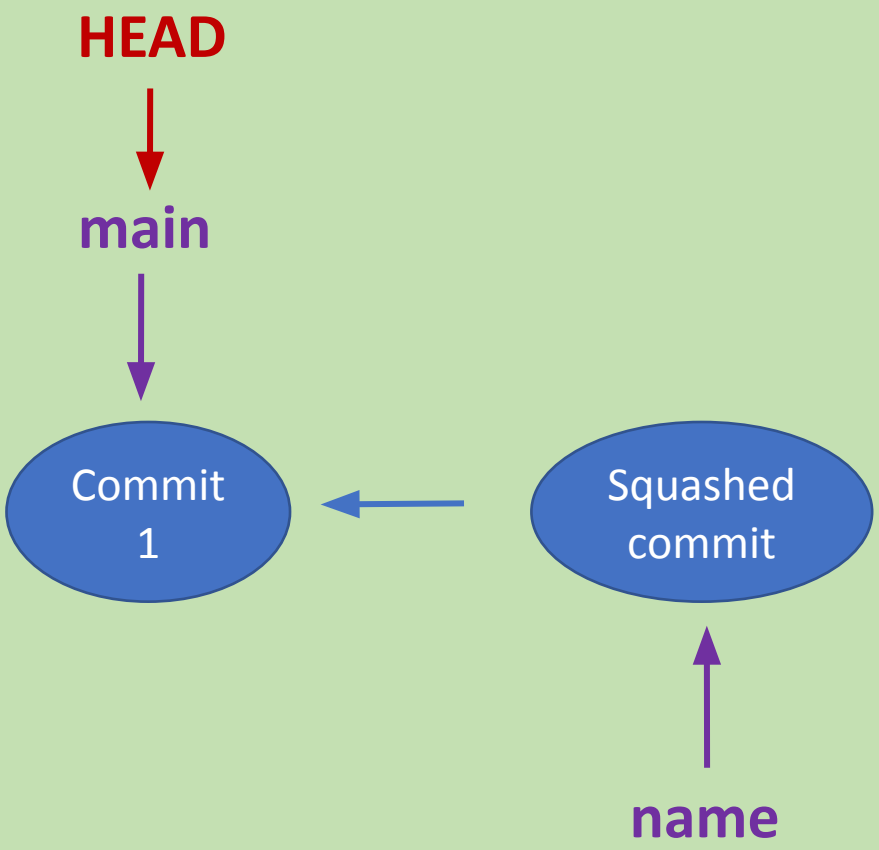


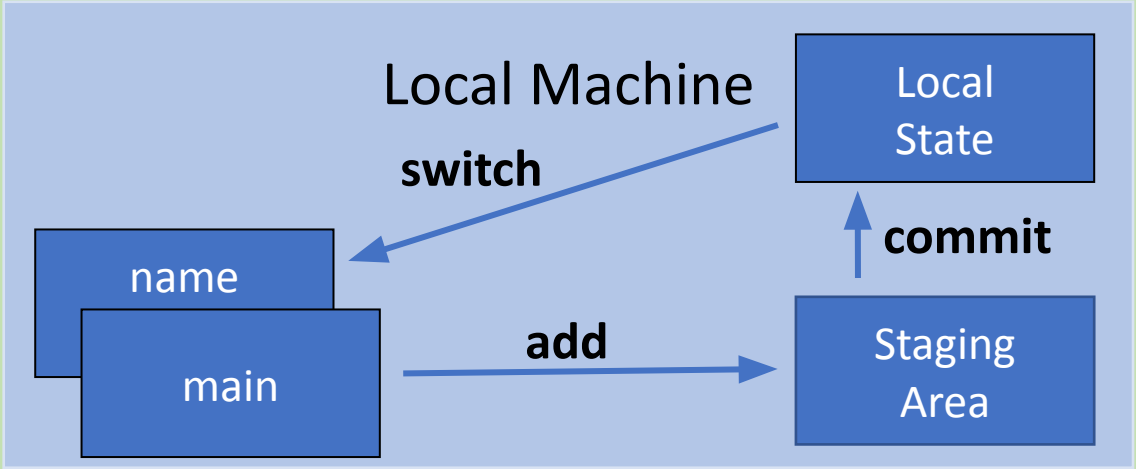
1. Look along with your buddy as you each do the following steps

2. Surface artists: (**one at a time!!!**)
   a. Run `git pull`
   b. Run `git merge `your_branch``
   c. Run `git push` to push the merged commit to main

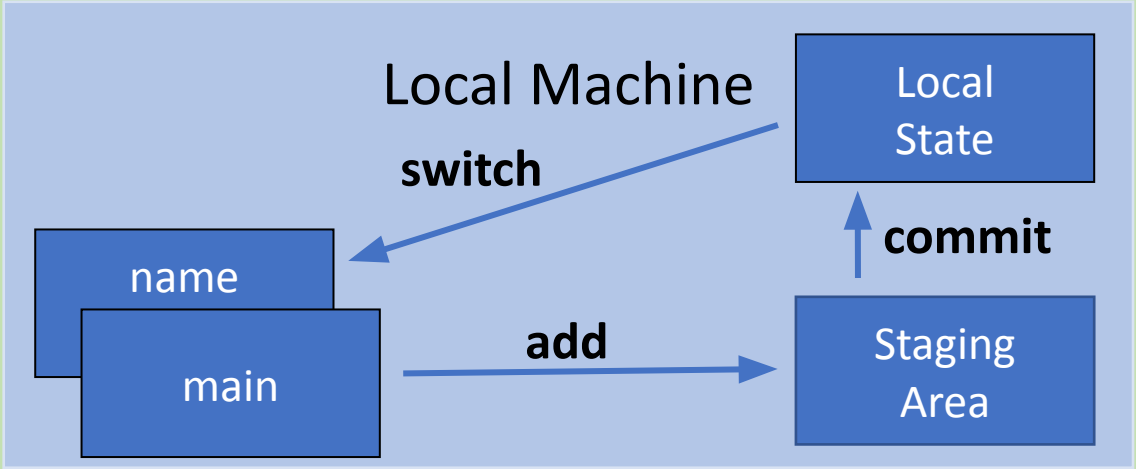
3. Atmosphere artists:
   a. After your buddy finishes their steps, run `git pull` to get their changes
   b. Look at your drawing file - what does it look like?

Basic Branch

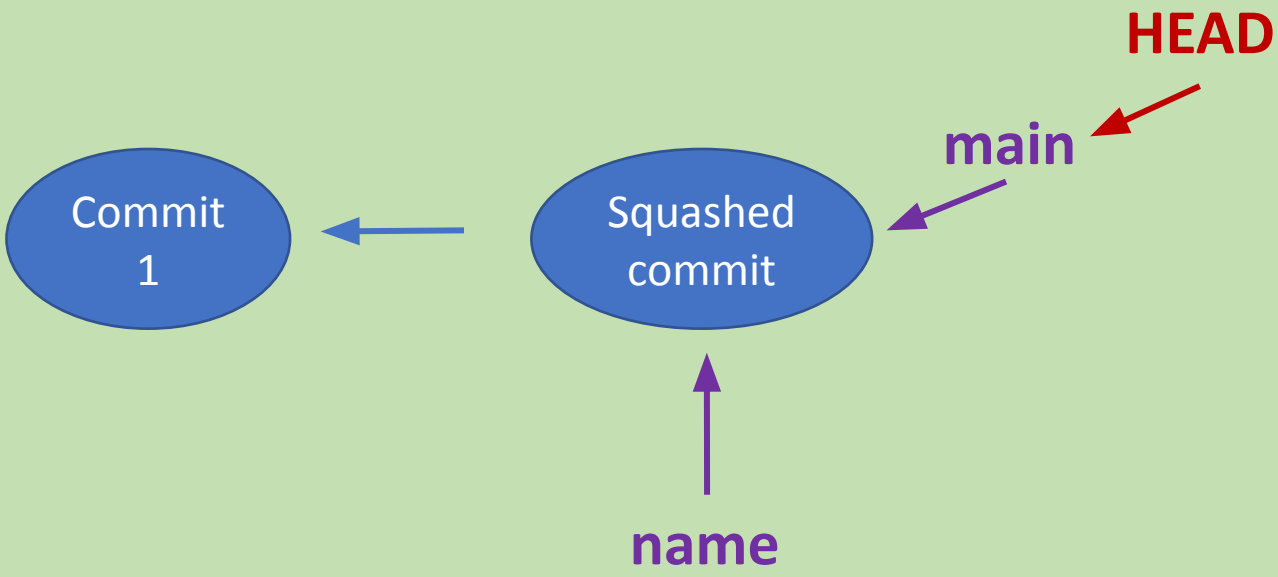




Basic Branch




```
git merge name
```

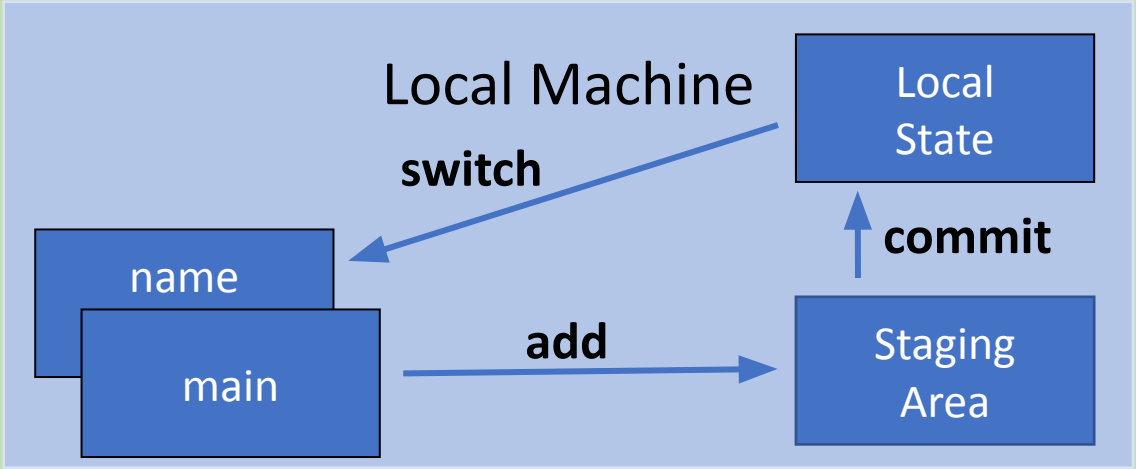


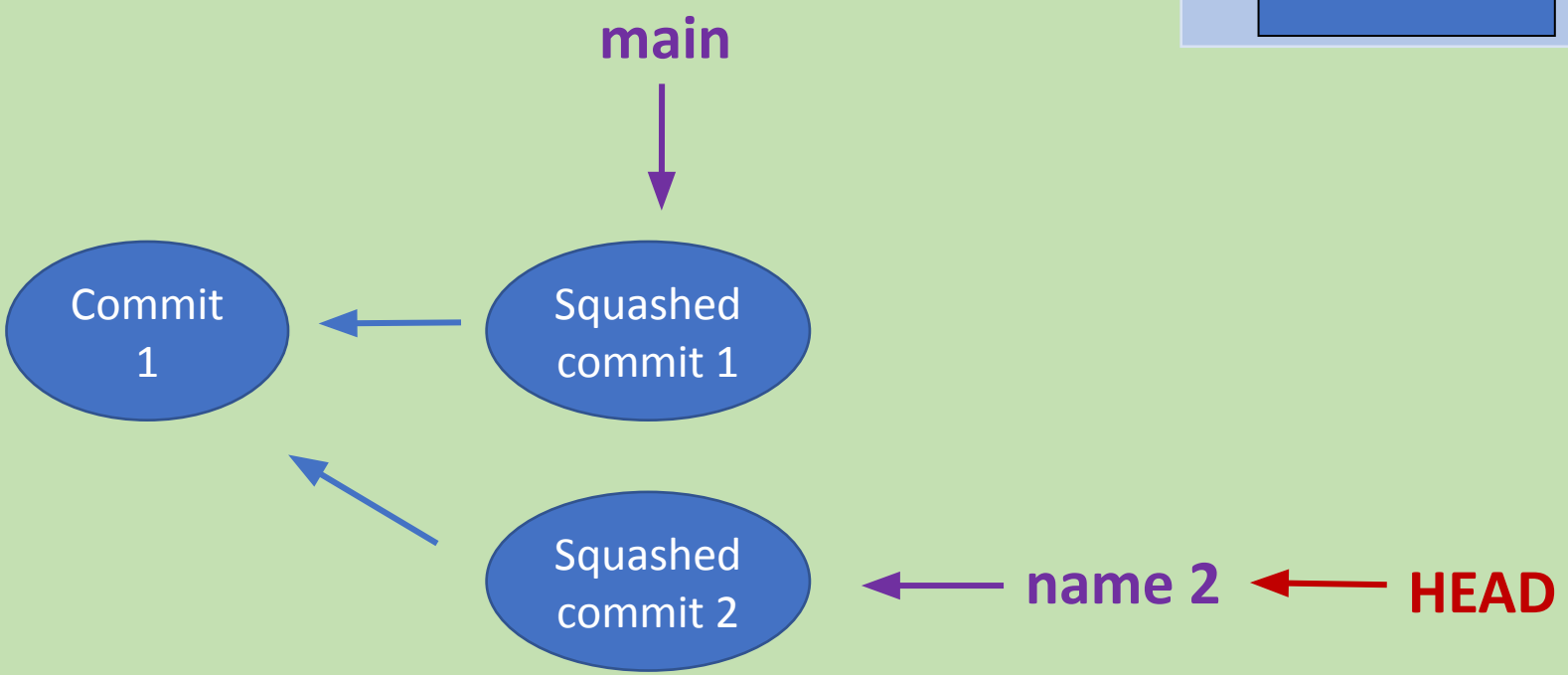
Merges

- Simple Merge
  - Two branches both add new, distinct code
  - They change different parts of the original code
- Merge Conflict
  - Two branches change the **same** piece of code
  - Which branch is correct?
- You will need to manually select which piece of code you want to keep, as there is no hierarchy on which piece of code is correct.

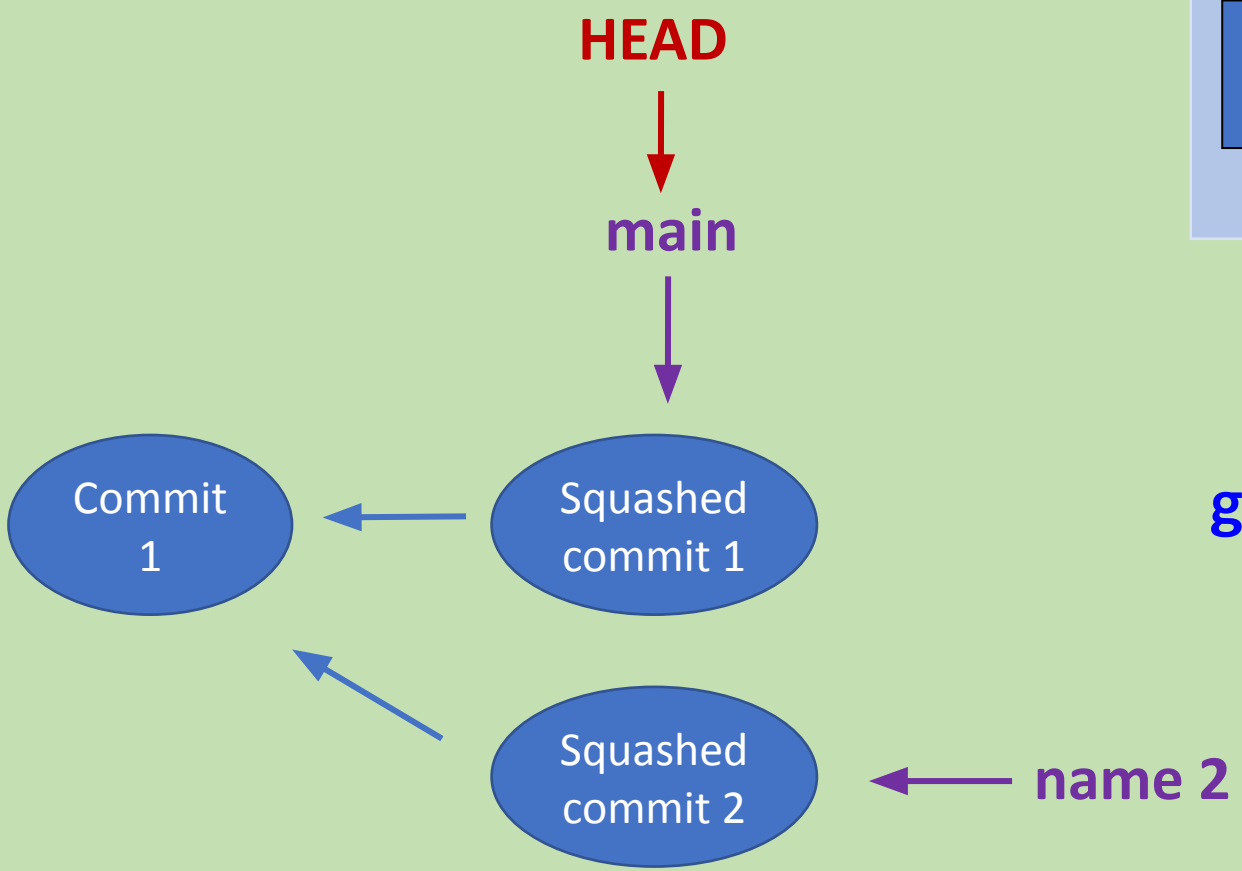
1. Atmosphere artists: (**one at a time!!!**)
    a. Run `git pull`
    b. Run `git merge `your_branch``

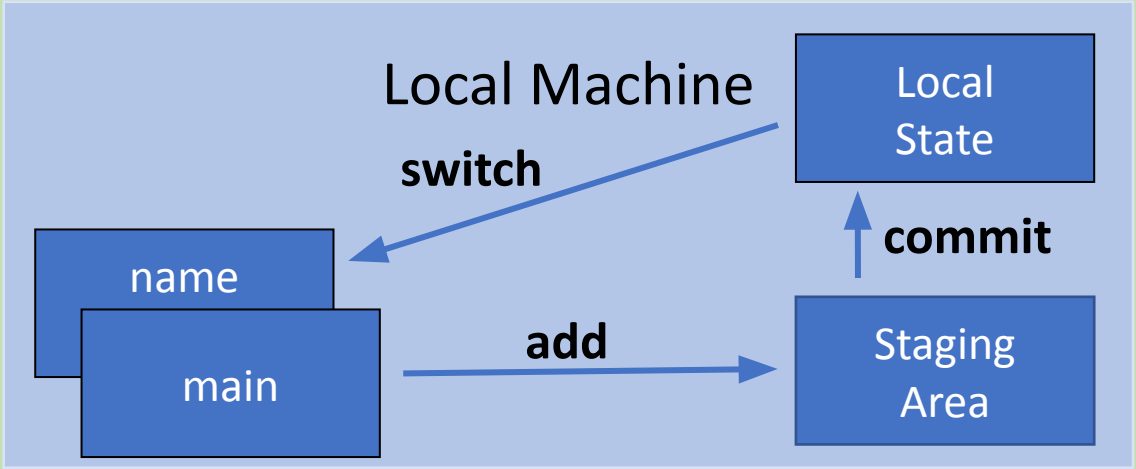
Branch #2


Local Machine
Local State
switch
commit
name
main
add
Staging Area


main
Commit 1
Squashed commit 1
Squashed commit 2
name 2
HEAD

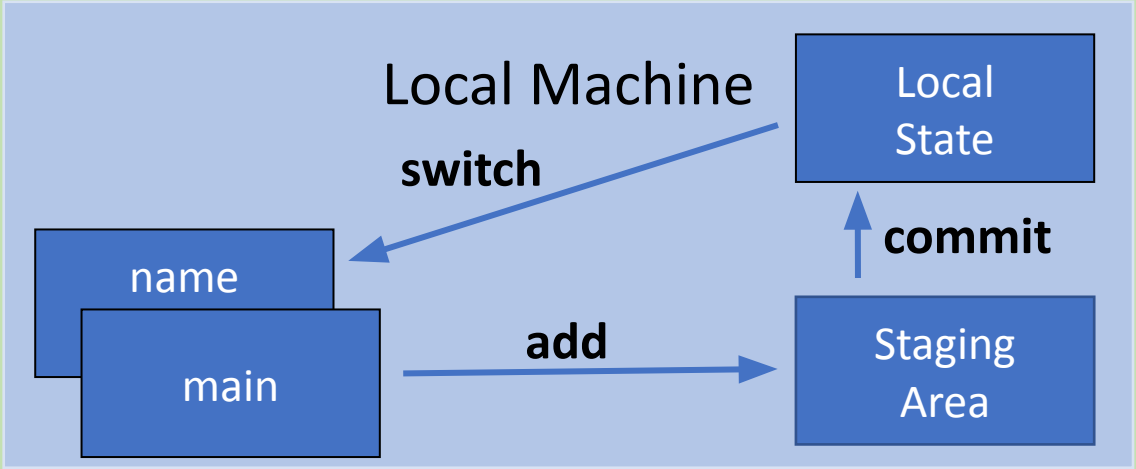
Branch #2

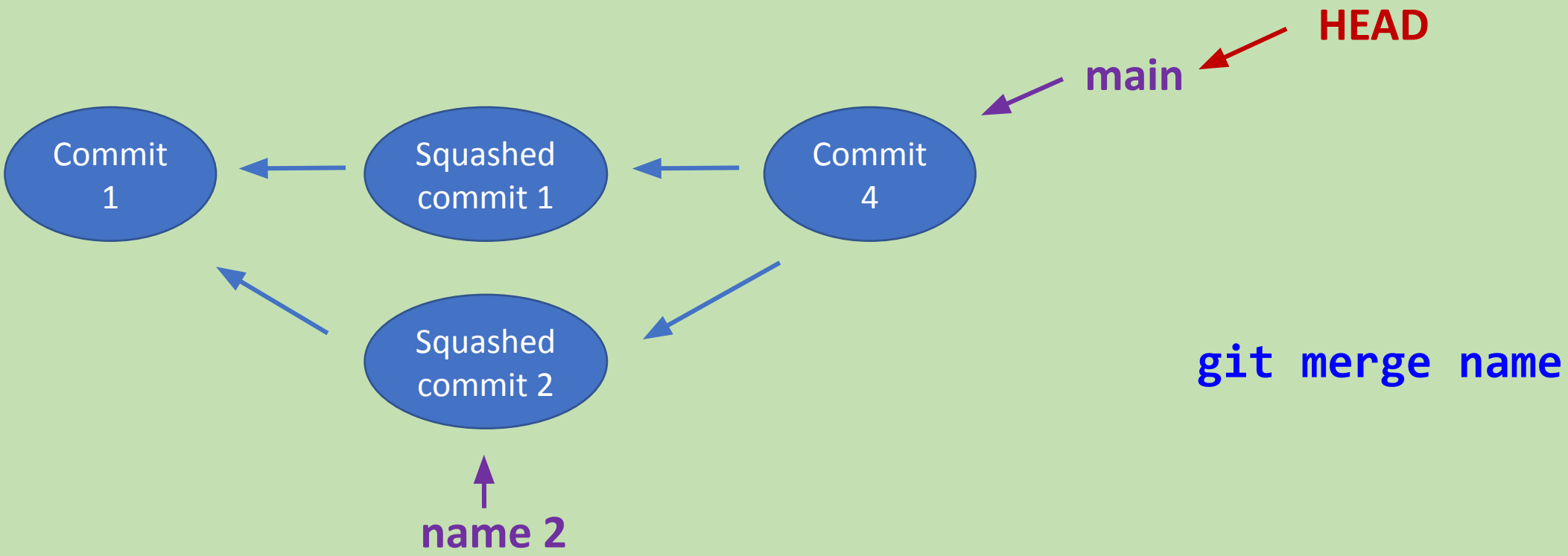





```
git switch main
```

Branch #2





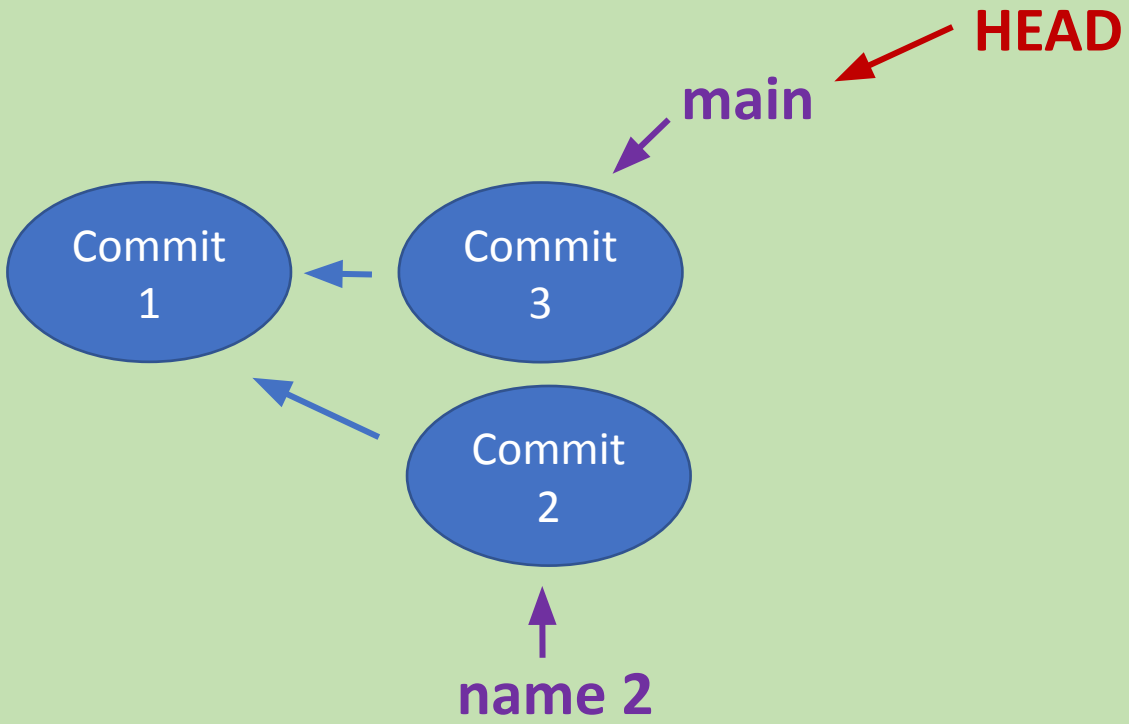

```
git merge name
```

How to resolve Merge Conflict

- Conflict: Git fails during the merge
  - Conflict between the current local branch and the branch being merged
  - Git will leave things for you to resolve manually in conflicted files

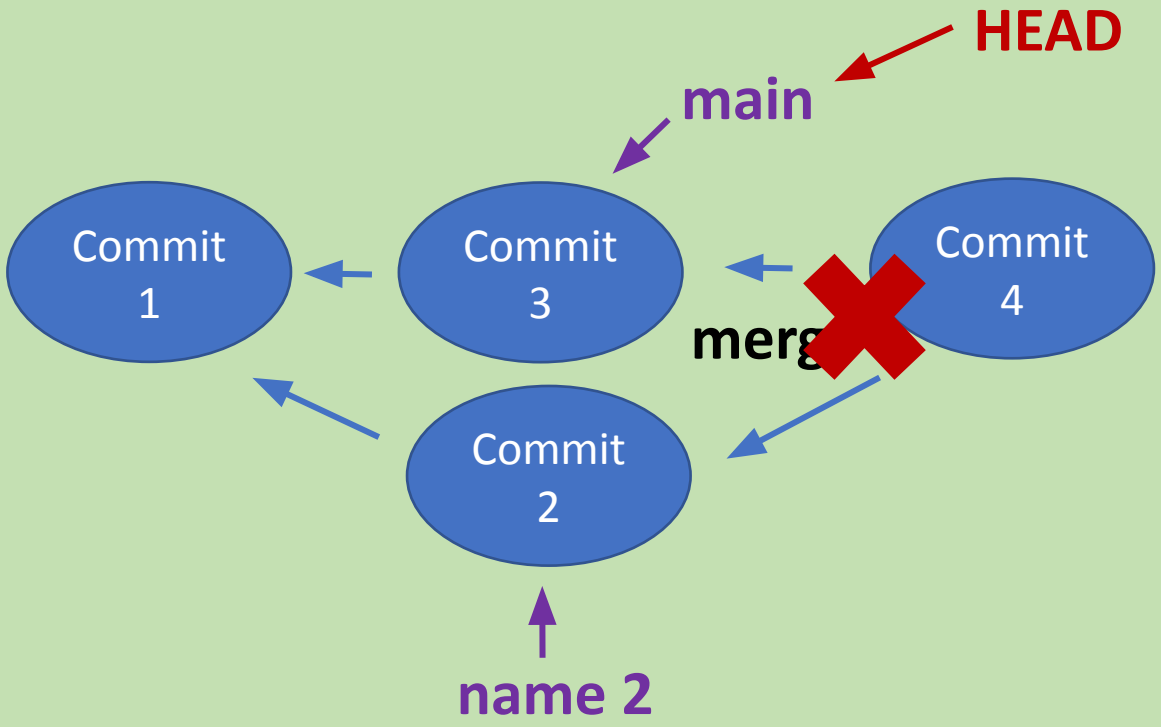
How to resolve Merge Conflict

- Conflict: Git fails during the merge
  - Conflict between the current local branch and the branch being merged
  - Git will leave things for you to resolve manually in conflicted files



How to resolve Merge Conflict

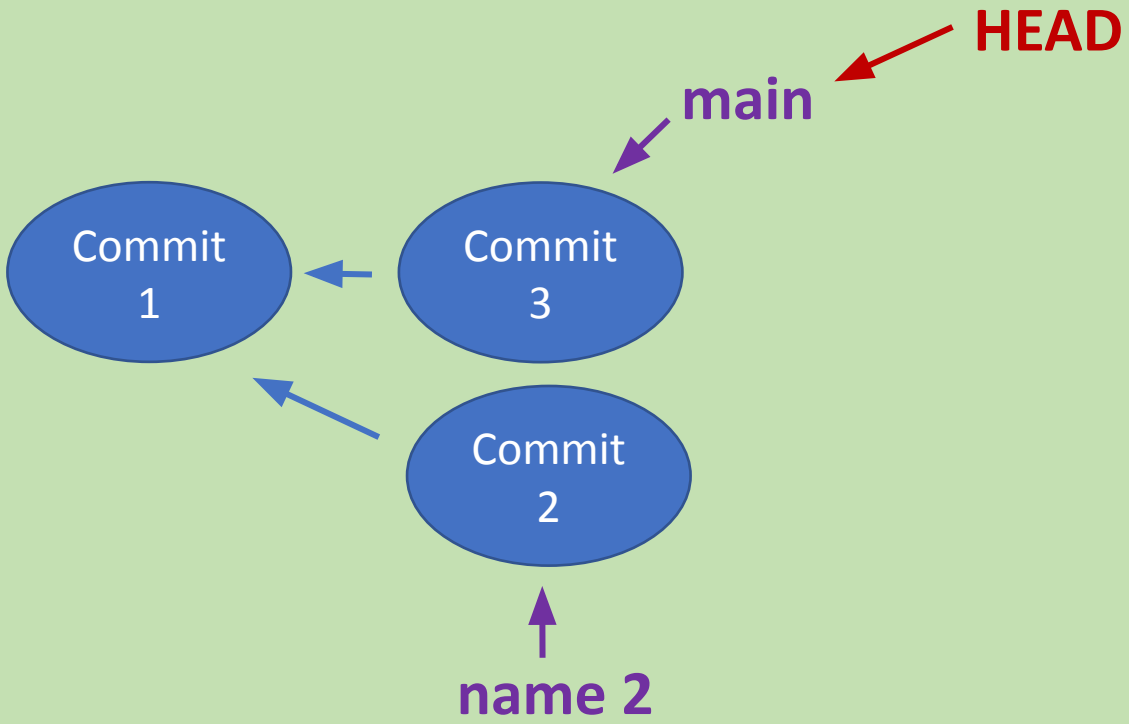
- Conflict: Git fails during the merge
  - Conflict between the current local branch and the branch being merged
  - Git will leave things for you to resolve manually in conflicted files



How to resolve Merge Conflict

- Conflict: Git fails during the merge
  - Conflict between the current local branch and the branch being merged
  - Git will leave things for you to resolve manually in conflicted files

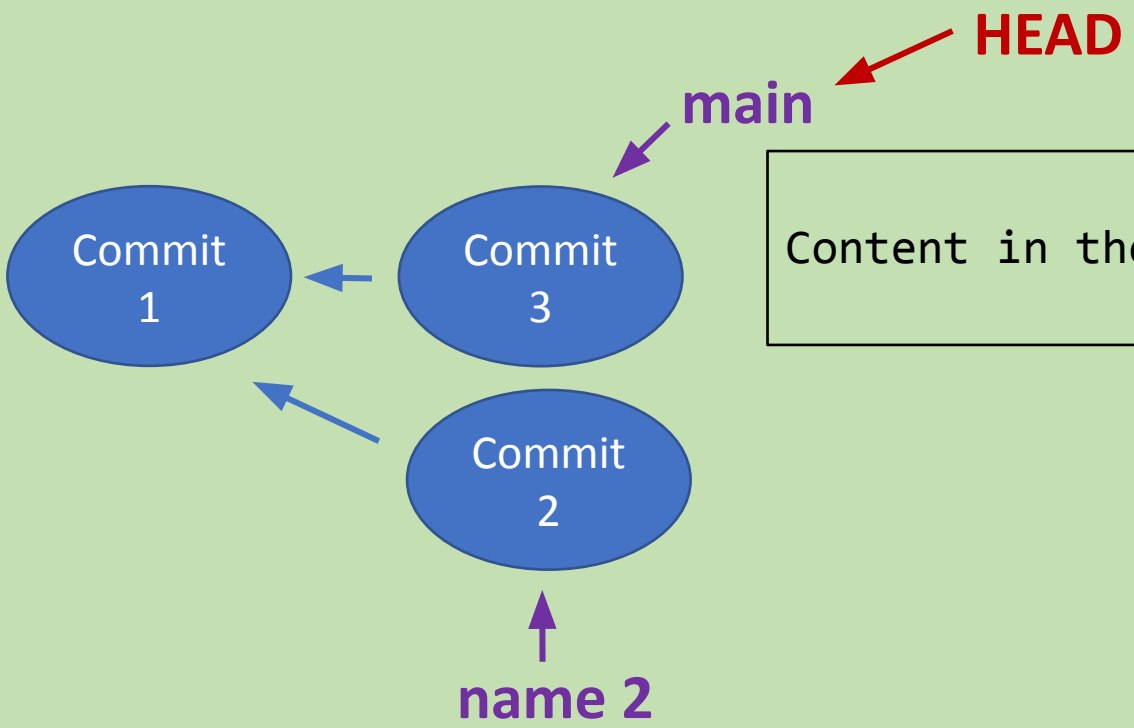
```
<<<<<<<HEAD
Content in the main branch.
=======
Content in the feature branch
>>>>>>>feature
```



How to resolve Merge Conflict

- Conflict: Git fails during the merge
  - Conflict between the current local branch and the branch being merged
  - Git will leave things for you to resolve manually in conflicted files

```
<<<<<<<HEAD
Content in the main branch.
=======
Content in the feature branch.
>>>>>>>feature
```

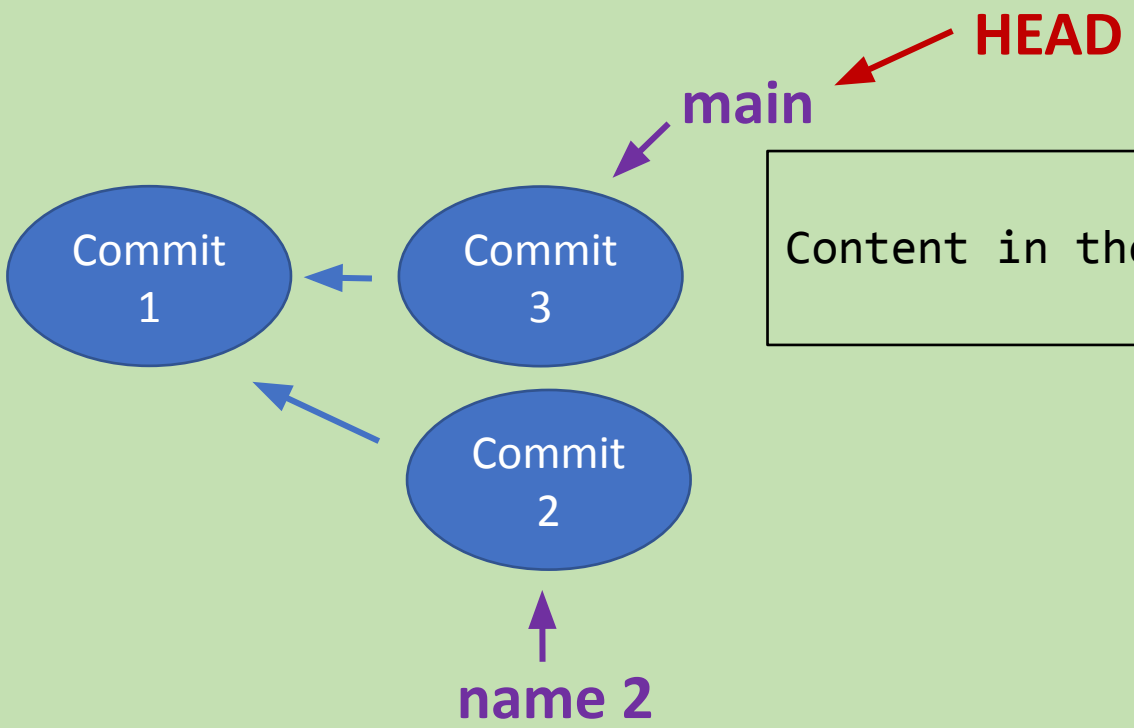



```
Content in the main branch.
```

How to resolve Merge Conflict

- Conflict: Git fails during the merge
  - Conflict between the current local branch and the branch being merged
  - Git will leave things for you to resolve manually in conflicted files

```
<<<<<<<HEAD
Content in the main branch.
=======
Content in the feature branch.
>>>>>>>feature
```




```
Content in the main branch.
```

```
Content in the feature branch.
```

1. Resolve your merge conflict

2. Make sure to remove all the
   <<<<<, =====, >>>>> characters

3. Make sure the final image makes sense :)

1. **One at a time (atmosphere folks)!!!**
    a. Run `git add .`
    b. Run `git commit -m “message”`
    c. Run `git pull`
    d. Run `git push`

What causes merge conflicts?

- In part 1, you edited separate files

  $\longrightarrow$ no merge conflict

- Today, you both tried to change the same part of the same file

  $\longrightarrow$ merge conflict

How to handle merge conflicts:

Use `git status` to check if you need to `add` or `commit` anything

`add` and `commit` your changes

`pull` any changes from the remote branch **(do this often!!)**

Deal with any merge conflicts locally

`push` to remote

Basic Git Commands

- git add – add changes to the staging area
- git commit – move changes from the staging area to local state
- git push – move changes from local state to remote state
- git pull – move changes from remote state to local state
- git clone – create local repo from remote repo
- git status – see what files are being tracked/have changes
- git log – see last few commit messages
- git branch – create a new branch/see branches
- git switch – switch between branches
- More - https://git-scm.com/doc

Any questions?

Getting GitHub Set Up

- GitHub Desktop
  - https://docs.github.com/en/desktop/installing-and-configuring-github-desktop/overview/getting-started-with-github-desktop

- GitHub through terminal:
  - Need to setup SSH:
    - https://docs.github.com/en/authentication/connecting-to-github-with-ssh/about-ssh
  - Switch current repos to SSH:
    - https://docs.github.com/en/get-started/getting-started-with-git/managing-remote-repositories#switching-remote-urls-from-https-to-ssh

More resources:

- Branching Tutorial
  - https://learngitbranching.js.org/
- Merge Conflicts
  - https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts
- Protecting a Branch
  - https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/defining-the-mergeability-of-pull-requests/about-protected-branches
- Pull Requests
  - https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests