

Working with branches

NEH Institute “Advanced Digital Editing”

Week 1, day 5

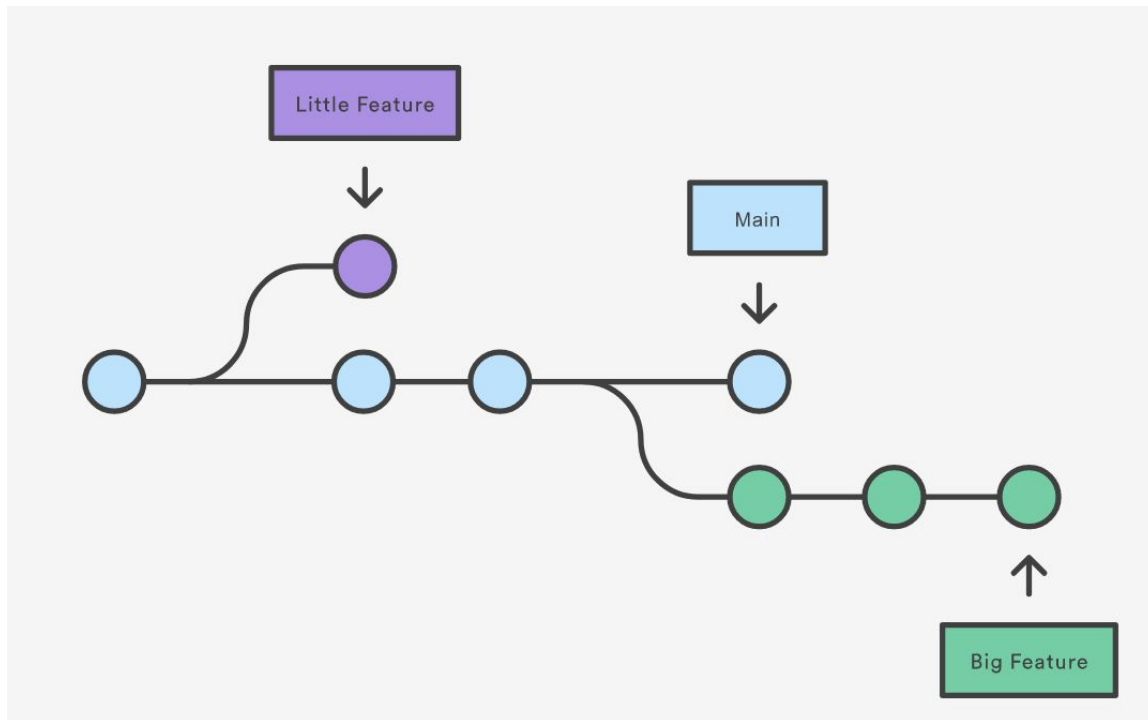
July 15: 11am - 12.30pm

Outcomes

- Understanding the concept of branches
- Creating a branch (of your sample Github repo)
- Pushing to your Github repo from your branch

Recap: branches in git

- A branch is a new/separate version of the main repo;
- With branches you can isolate different acts of development;
- Your work will not affect the main code base.

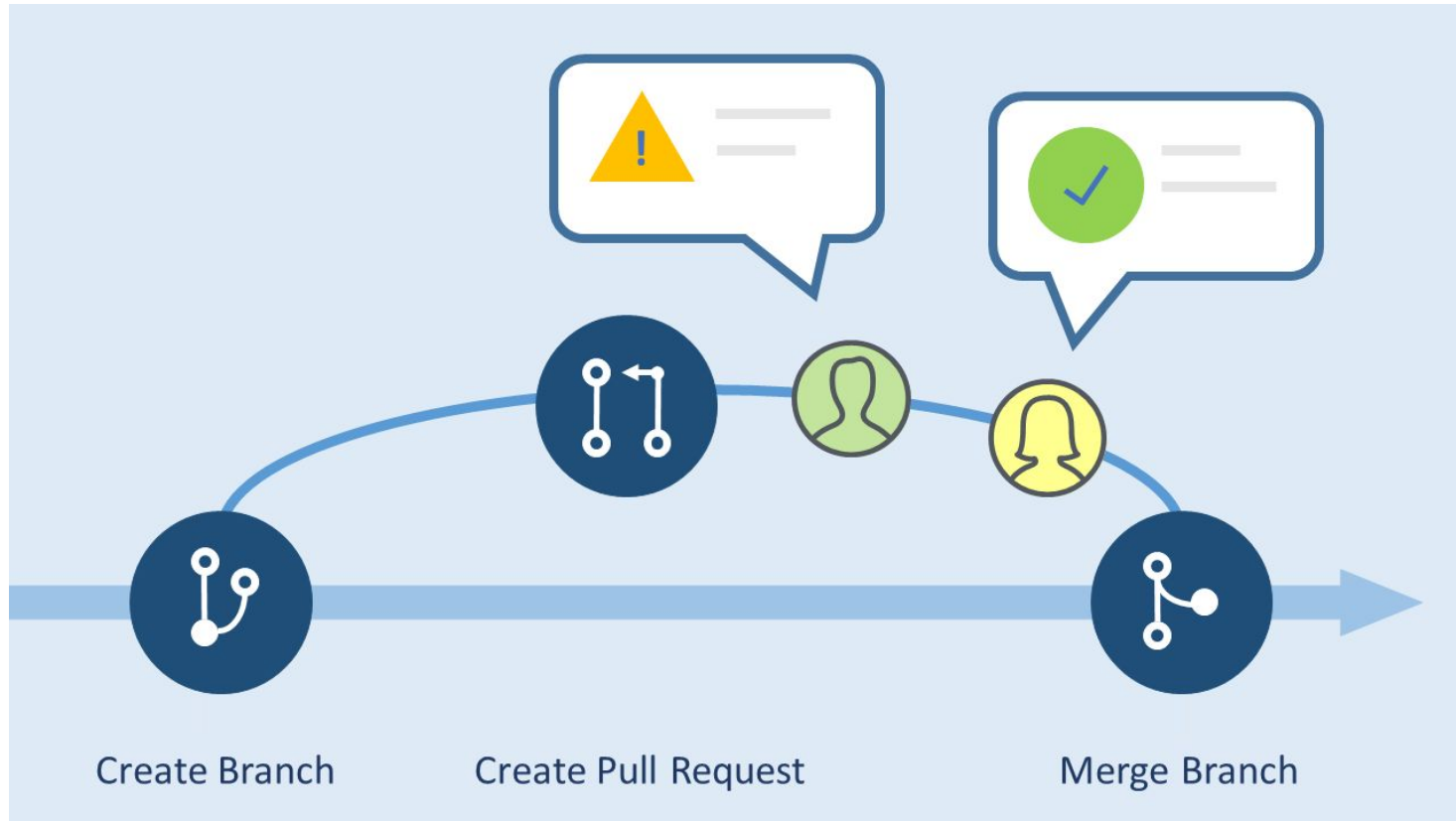


Recap: Github issues

- Issues are used to identify bugs, development tasks, or requesting features for a code base on Github;
- Anyone can create an issue for a public Github repository;
- A Github repo usually has guidelines for creating issues, make sure to read them beforehand;
- Example: the [issues](#) in the pr-app Github repository.

Pull requests

- A pull request (PR) is a proposed change to the codebase in a Git repository;
- Once created, everyone can see and review the PR;
- A PR is stored in isolation of the main branch, so it will not affect the main code base



Initialize a git repository

Workflow:

- Check if you're in your app directory: `$ pwd`
 - If not, navigate to the right directory with `$ ls`
 - With `$ ls -a` you can see all files in the directory, including the hidden files like `.git`
 - When you see a `.git` file in your sample directory, it means it is under git control
- Initialize git: `$ git init`
- Check your status: `$ git status`
- First commit message: `$ git commit -m "Initializing repo"`
- Check your status: `$ git status`

Initialize a git repository (continued)

Go to your browser and head to your Github profile

- Click on the “new” button;
- IMPORTANT: give your repository **the exact same name** as the repo you created;
- IMPORTANT: do **not** check the “Add README” box;
- Click on “create repository”

Go back to the command line

- Run the commands listed under “...or push an existing repository from the command line”
 - `$ git remote add origin [url-of-your-new-repo]`
 - `$ git branch -M main`
 - `$ git push -u origin main`

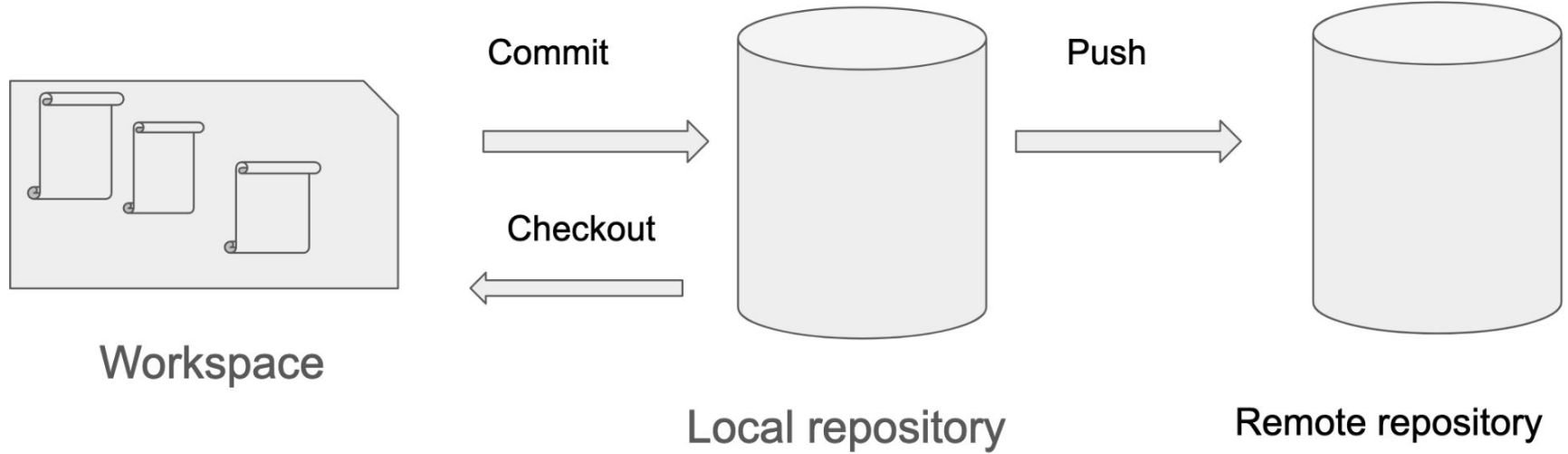
Initialize a git repository

- Switch to your browser and refresh your Github page
- Verify that the repository is no longer empty
- Well done!

Double-check on your command line:

- `$ git remote -v`
- You should see the URL to your Github repository

Recap: push commits from local to remote



Creating a branch

Workflow:

- You should be in the app directory
 - Hit `$pwd` to double-check in which directory you are
 - With `$ ls -a` you can see all files in the directory, including the hidden files like `.git`
 - When you see a `.git` file in your sample directory, it means it is under git control
- List the existing branches
 - `$ git branch -a`
- Create a new branch
 - `$ git branch [name-of-your-branch]`
- Switch to the new branch
 - `$ git checkout [name-of-your-branch]`

Working in a branch

You are now on your newly created branch of the pr-app repository

- List existing branches: `$ git branch -a`
- You should be on your newly created branch

Let's do some work in your branch!

Remember:

- A branch is a separate version of the main code base;
- When you make and save changes, they are stored on your local machine;
- If you commit and push your work to the repository, it will remain in a separate branch;
- Your branch does not affect the main code base in the main repository.

Committing a branch

Let's push your branch to the remote repository

- `$ git status`
- `$ git branch -a`
- `$ git add [file-you-changed]`
- `$ git status`
- `$ git commit -m "[message here]"`
- `$ git push origin [name-of-your-branch]`

Remember:

- By pushing your work to the remote repo, it will remain in a separate branch
- Your branch does not affect the main code base in the remote repository

Committing a branch (continued)

Your branch is now uploaded to the remote repository! Go check it out on your personal Github repository in your browser: [https://github.com/\[your-username\]](https://github.com/[your-username])

Create a pull request

Now that you have done some work that you're proud of, you can create a PR:

- Click on **Pull requests** >> **new pull request**
- Pick your branch from the drop-down menu
 - The base version is the branch to which you want your changes to be applied
 - The compare version is your branch containing the proposed changes
- Click on **Create Pull request**
- Type a title and a description for your PR
- Click **Create Pull request** again
- Optionally: tag people to review your PR

Linking a PR to an issue

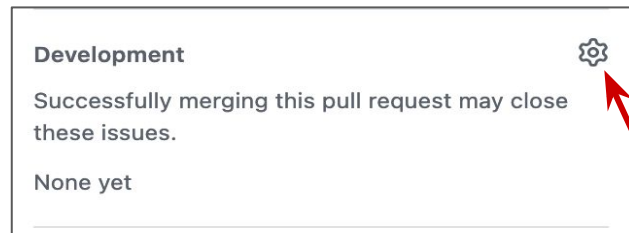
In some cases you will want to link a PR to an existing issue:

- Collaborators can see that someone is working on the issue;
- When the PR is merged, the issue is automatically closed.

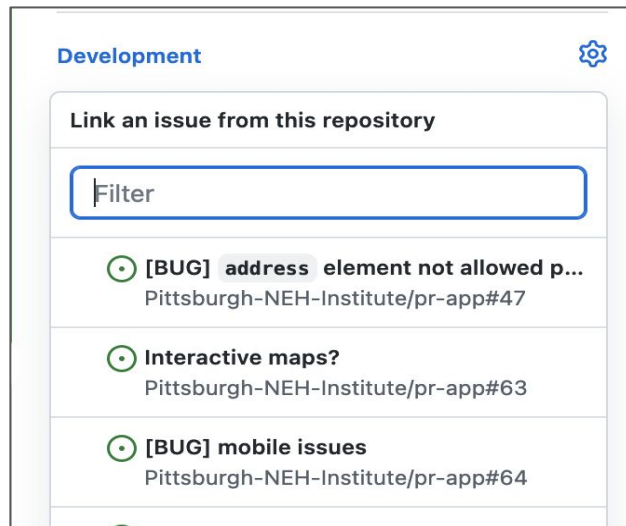
Linking a PR to an issue on Github:

- Click on the PR
- In the right panel, click on **Development**
- Select an issue from the dropdown list

1.



2.





PRs and issues in the TEI Github repo

- The TEI Technical Council maintains the TEI Guidelines and Stylesheets on Github;
- Users of the TEI Guidelines (that means you too!) can submit issues and PRs;
- The issues and PRs are handled and reviewed by the members of the TEI-Council;
- Check out the repo [here](#)!

References

- [Github documentation on issues](#);
- [Github documentation on PRs](#);
- Git cheat sheet (click on the image).

 **Git Cheat Sheet**



Create a Repository

From scratch – Create a new local repository

```
$ git init [project_name]
```

Download from an existing repository

```
$ git clone my_url
```

Observe a Repository

List new or modified files not yet committed

```
$ git status
```

Show the changes to files not yet staged

```
$ git diff
```

Show the changes to staged files

```
$ git diff --cached
```

Show all staged and unstaged file changes

```
$ git diff HEAD
```

Show the changes between two commit ids

```
$ git diff commit1 commit2
```

List the change dates and authors for a file

```
$ git blame [file]
```

Show the file changes for a commit id and/or file

```
$ git show [commit]:[file]
```

Show full change history

```
$ git log
```

Show change history for file/directory including diffs

```
$ git log -p [file/directory]
```

Working With Branches

List all local branches

```
$ git branch
```

List all branches, local and remote

```
$ git branch -av
```

Switch to a branch, my_branch, and update working directory

```
$ git checkout my_branch
```

Create a new branch called new_branch

```
$ git branch new_branch
```

Delete the branch called my_branch

```
$ git branch -d my_branch
```

Merge branch_a into branch_b

```
$ git checkout branch_b  
$ git merge branch_a
```

Tag the current commit

```
$ git tag my_tag
```

Make a Change

Stages the file, ready for commit

```
$ git add [file]
```

Stage all changed files, ready for commit

```
$ git add .
```

Commit all staged files to versioned history

```
$ git commit -m "commit message"
```

Commit all your tracked files to versioned history

```
$ git commit -am "commit message"
```

Unstages file, keeping the file changes

```
$ git reset [file]
```

Revert everything to the last commit

```
$ git reset --hard
```

Synchronize

Get the latest changes from origin (no merge)

```
$ git fetch
```

Fetch the latest changes from origin and merge

```
$ git pull
```

Fetch the latest changes from origin and rebase

```
$ git pull --rebase
```

Push local changes to the origin

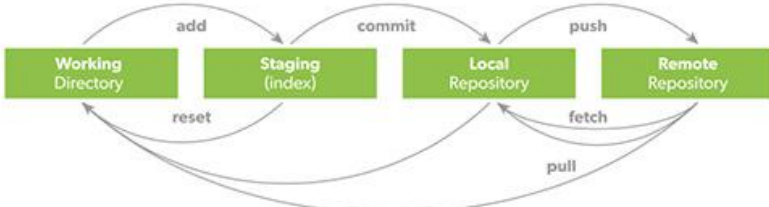
```
$ git push
```

Finally!

When in doubt, use git help

```
$ git [command] --help
```

Or visit training.github.com for official GitHub training.



```
graph LR; WD[Working Directory] -- add --> S[Staging (index)]; S -- commit --> LR[Local Repository]; LR -- push --> RR[Remote Repository]; RR -- fetch --> LR; LR -- pull --> WD; S -- reset --> WD; RR -- pull --> WD;
```

www.jrebel.com

PERFORME