Merge conflicts

INTERMEDIATE GIT

George Boorman
Curriculum Manager, DataCamp

datacamp

Conflicts

- Conflict
  - Inability to resolve differences in the contents of one or more files between branches
- Edit the same file in two branches
- Try to merge
- Git doesn't know what version to keep
- Conflict!

Conflicting versions of README.md

`documentation` branch

```
# Contents and usage

This repo contains source code
for the DataCamp website.

It also contains source code for an
AI-Assistant (recommendation system)
that takes prompts from learners and
returns suggested content
that they might be interested in.

It is for internal use only,
external access is prohibited.
```

`main` branch

```
# Contents and usage

This repo contains source code
for the DataCamp website.

It is for internal use only,
external access is prohibited.
```

Merging

- From the `main` branch

```
git merge documentation
```

```
Auto-merging README.md
CONFLICT (add/add): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Opening the file

```
nano README.md
```

```
# Contents and usage

This repo contains source code for the DataCamp website.

<<<<<<< documentation
It also contains source code for an AI-Assistant (recommendation system)
that takes prompts from learners and returns suggested content that they
might be interested in.

=======
>>>>>>> HEAD
It is for internal use only and external access is prohibited.
```

Git conflict syntax

In both branches

documentation branch

Current branch (main)

In both branches

```
# Contents and usage

This repo contains source code for the DataCamp website.

<<<<<<< documentation
It also contains source code for an AI-Assistant (recommendation system)
that takes prompts from learners and returns suggested content that they
might be interested in.

=======
>>>>>>> HEAD
It is for internal use only and external access is prohibited.
```

Resolving the conflict

```
# Contents and usage

This repo contains source code for the DataCamp website.

<<<<<<< documentation
It also contains source code for an AI-Assistant (recommendation system)
that takes prompts from learners and returns suggested content that they
might be interested in.

=======
>>>>>>> HEAD
It is for internal use only and external access is prohibited.
```

- Save: `Ctrl + O` (not `Ctrl + 0`), then `Enter`
- Exit: `Ctrl + X`

Merging the branches

- Merging now that the conflict is resolved

```
git add README.md
```

```
git commit -m "Resolving README.md conflict"
```

```
git merge documentation
```

```
Already up to date.
```

- Prevention is better than cure!

Let's practice!

INTERMEDIATE GIT

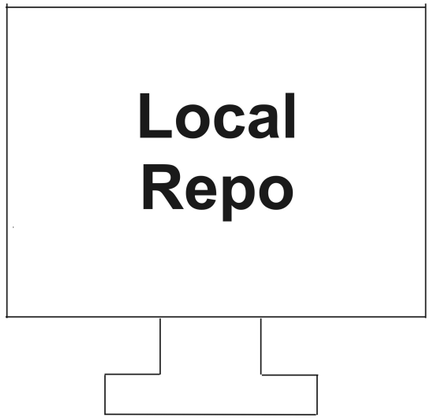
Introduction to remotes

INTERMEDIATE GIT

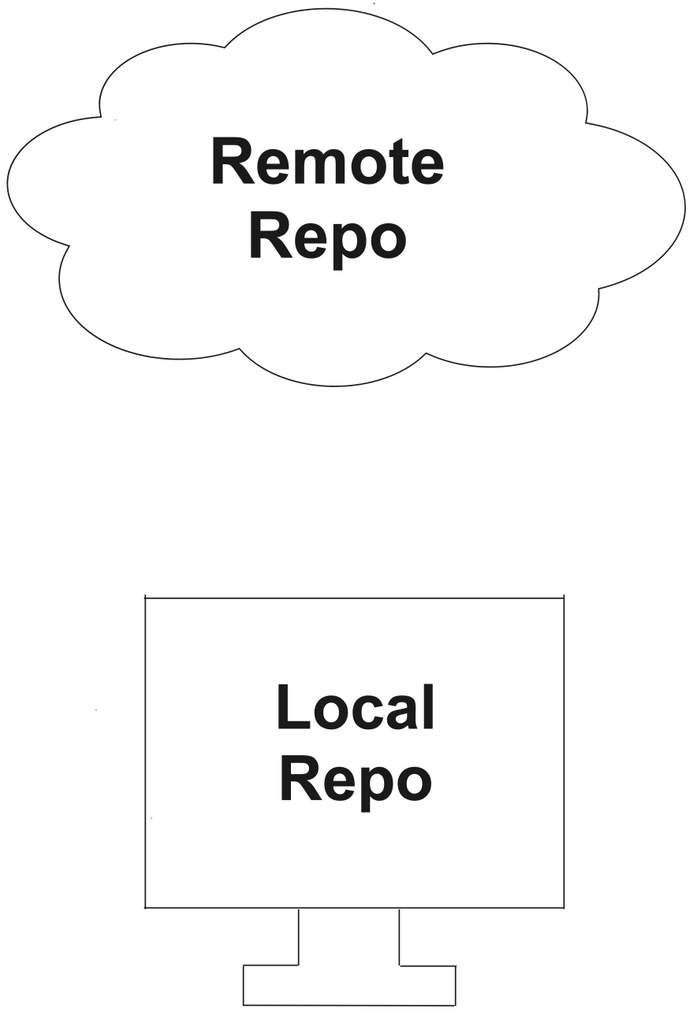
George Boorman
Curriculum Manager, DataCamp

datacamp

Local repo



Remote repo



Why use remote repos?

Benefits of remote repos

- Everything is backed up
- Collaboration, regardless of location

Cloning a repo

- Repo copies on our local computer = local remotes
- Making copies = cloning

```
git clone path-to-project-repo
```

- Cloning a local project

```
git clone /home/george/repo
```

- Cloning and naming a local project

```
git clone /home/george/repo new_repo
```

Cloning a remote

- Remote repos are generally stored in an online hosting service
  - e.g., GitHub, Bitbucket, or GitLab
- If we have an account:
  - We can clone a remote repo on to our local computer

```
git clone URL
```

```
git clone https://github.com/datacamp/project
```

Identifying a remote

- When cloning a repo
  - Git remembers where the original was
- Git stores a remote **tag** in the new repo's configuration
- List all remotes associated with the repo

```
git remote
```

```
datacamp
```

Getting more information

- Get more information about the remote(s)

```
git remote -v
```

```
datacamp    https://github.com/datacamp/project (fetch)
datacamp    https://github.com/datacamp/project (pull)
```

Creating a remote

- When cloning, Git will automatically name the remote `origin`

```
git remote add name URL
```

- Create a remote called `george`

```
git remote add george https://github.com/george_datacamp/repo
```

- Defining remote names is useful for merging

Let's practice!

INTERMEDIATE GIT

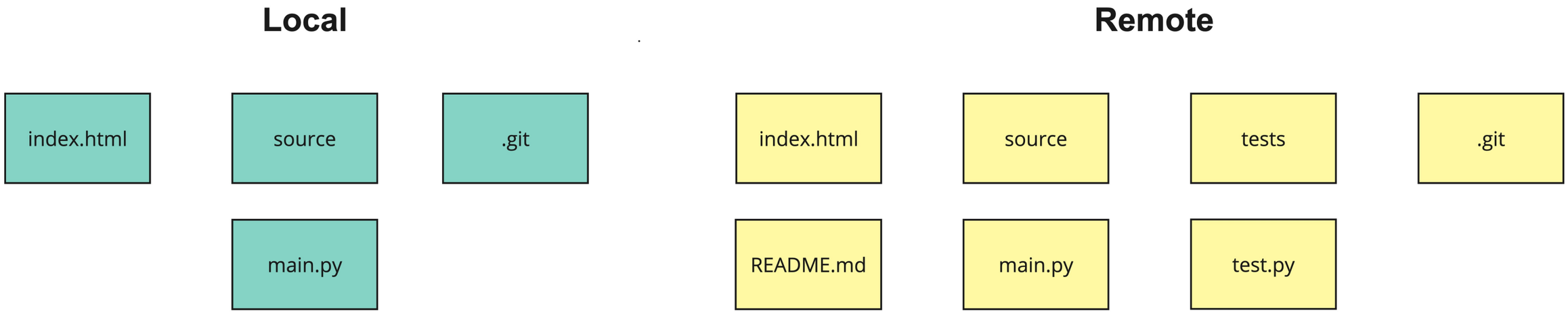
Pulling from remotes

INTERMEDIATE GIT

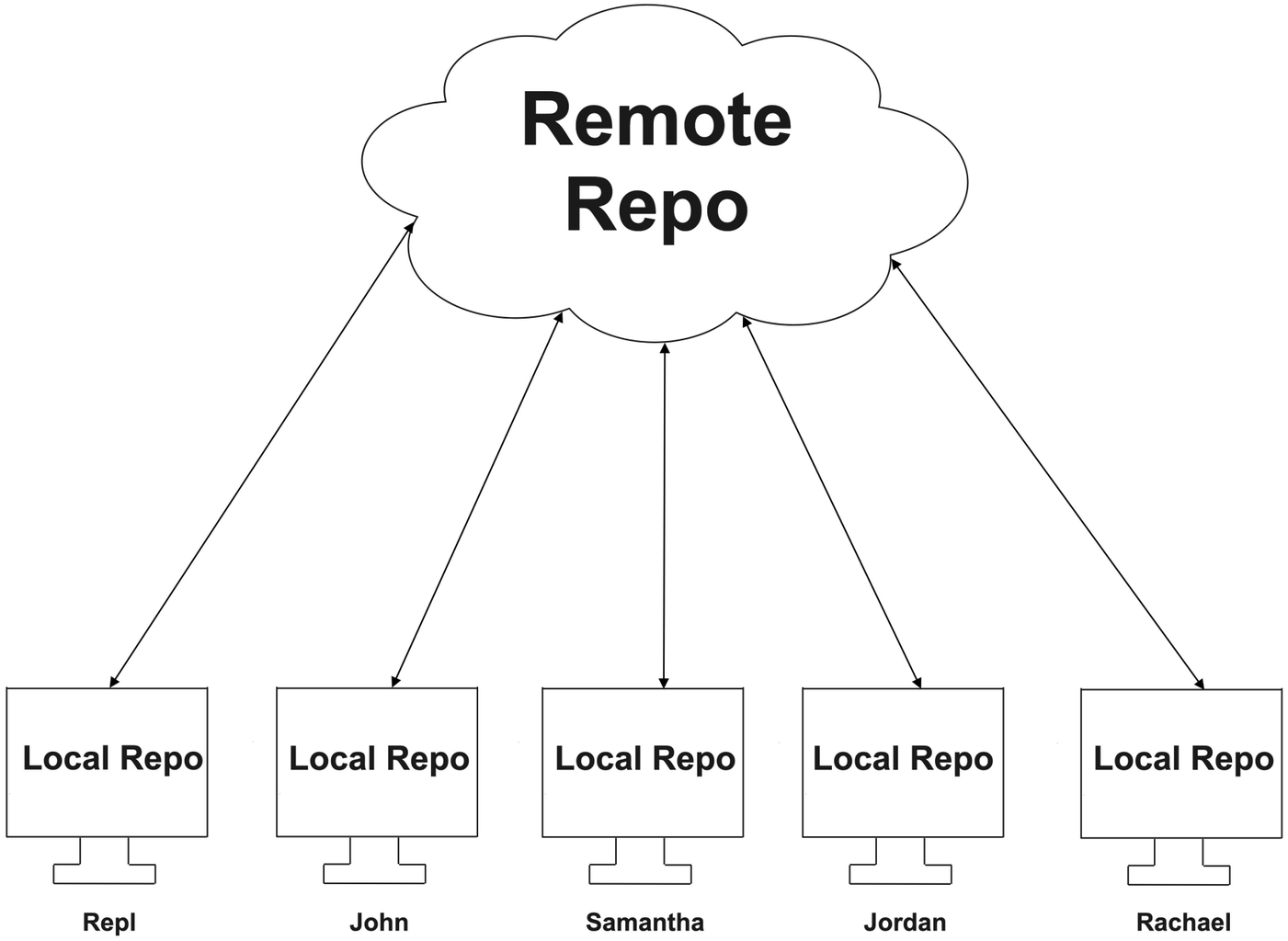
George Boorman
Curriculum Manager, DataCamp

datacamp

Remote vs. local



Collaborating on Git projects



Fetching from a remote

- Fetch from the `origin` remote

```
git fetch origin
```

- Fetch all remote branches
- Might create new local branches if they only existed in the `remote`
- Doesn't merge the remote's contents into local repo

Fetching a remote branch

- Fetch only from the `origin` remote's `main` branch

```
git fetch origin main
```

```
From https://github.com/datacamp/project
 * branch                main     -> FETCH_HEAD
```

Synchronizing content

- Merge `origin` remote's default branch (`main`) into the local repo's current branch

```
git merge origin
```

```
Updating 9dcf4e5..887da2d
Fast-forward
 tests/tests.py | 13 ++++++++++++
 README.md      | 10 ++++++++++
 2 files changed, 23 insertions (+)
```

Pulling from a remote

- Local and remote synchronization is a common workflow
- Git simplifies this process for us!
- Fetch and merge from the remote's default (`main`) into the local repo's current branch

```
git pull origin
```

Pulling a remote branch

- Pull from the `origin` remote's `dev` branch

```
git pull origin dev
```

- Still merges into the local branch we are located in!

Git pull output

```
From https://github.com/datacamp/project
 * branch                dev     -> FETCH_HEAD
Updating 9dcf4e5..887da2d
Fast-forward
 tests/tests.py | 13 ++++++++++++
 README.md      | 10 ++++++++++
 2 files changed, 23 insertions (+)
```

A word of caution

```
git pull origin
```

```
Updating 9dcf4e5..887da2d
error: Your local changes to the following files would be overwritten by merge:
       README.md
Please commit your changes or stash them before you merge.
Aborting
```

- Important to save locally before pulling from a `remote`

Let's practice!

INTERMEDIATE GIT

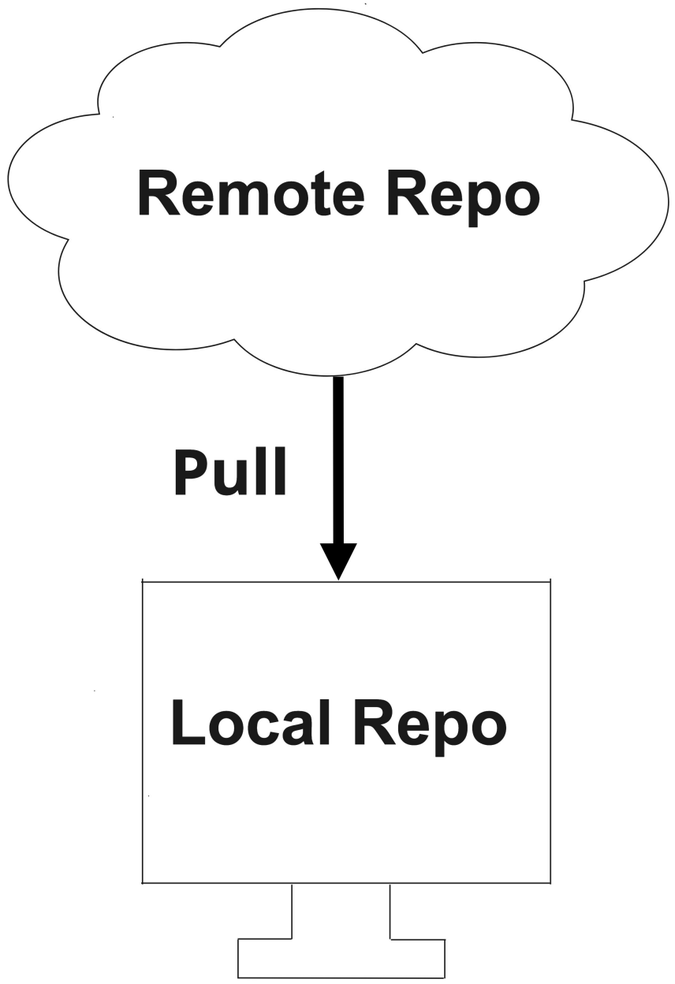
Pushing to remotes

INTERMEDIATE GIT

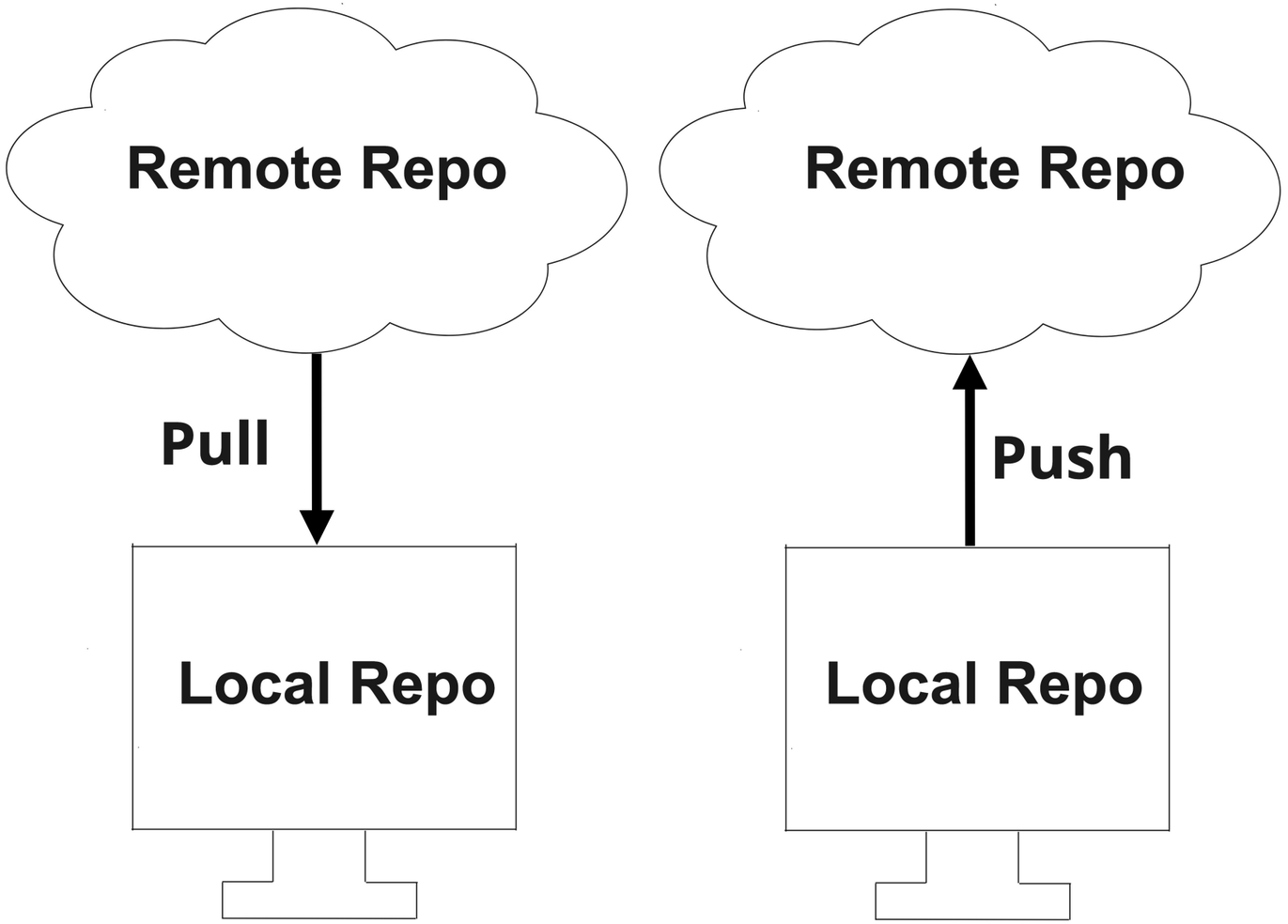
George Boorman
Curriculum Manager, DataCamp

datacamp

Pulling from a remote



Pushing to a remote



git push

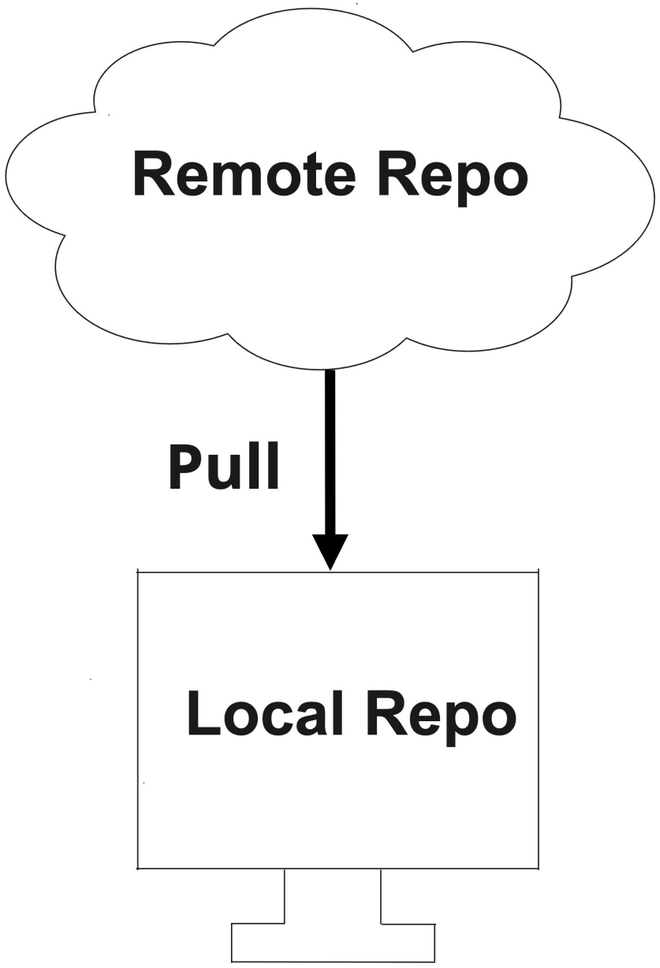
- Save changes locally first!

```
git push remote local_branch
```

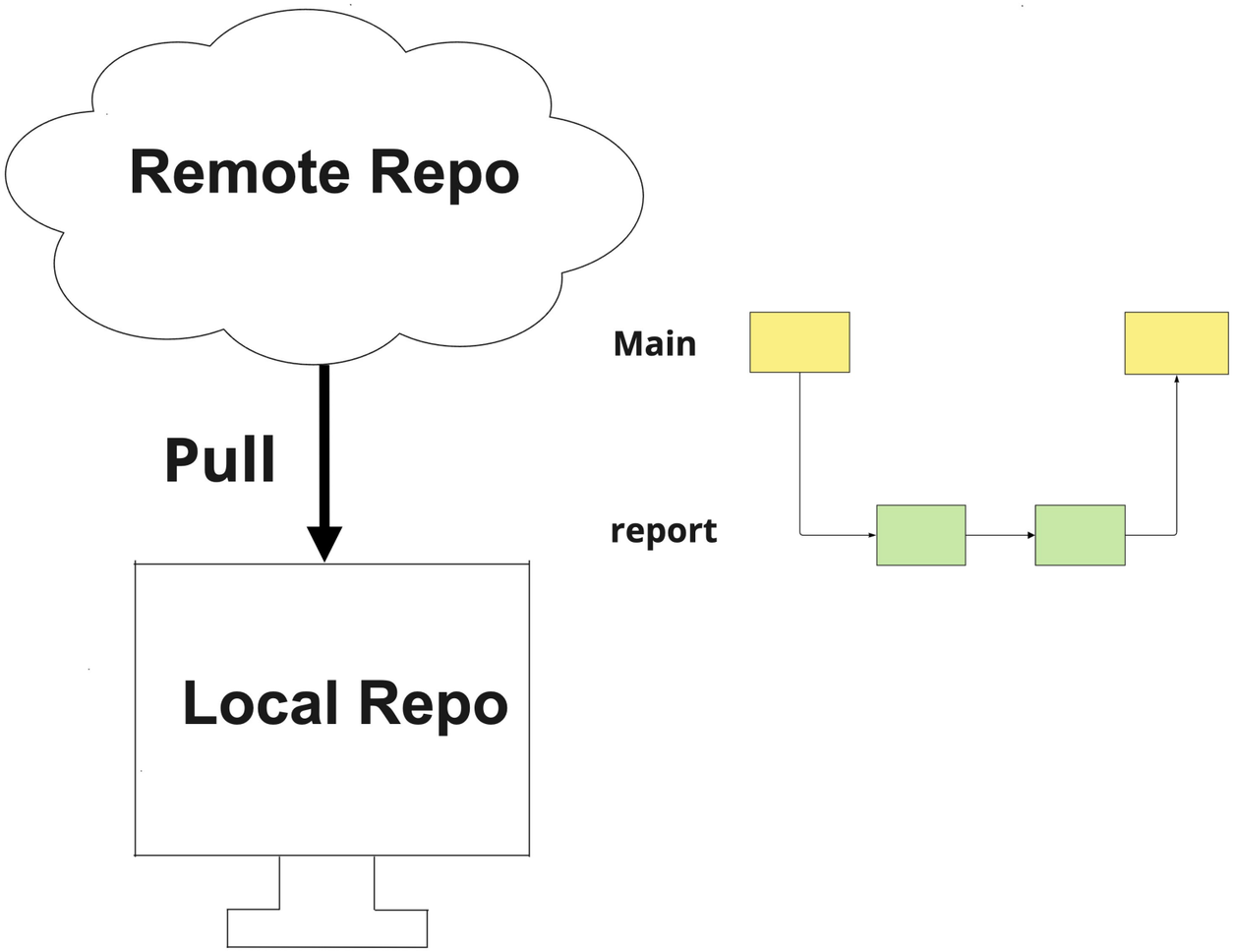
- Push *into* `remote` **from** `local_branch`
- Push changes into `origin` from the local repo's `main` branch

```
git push origin main
```

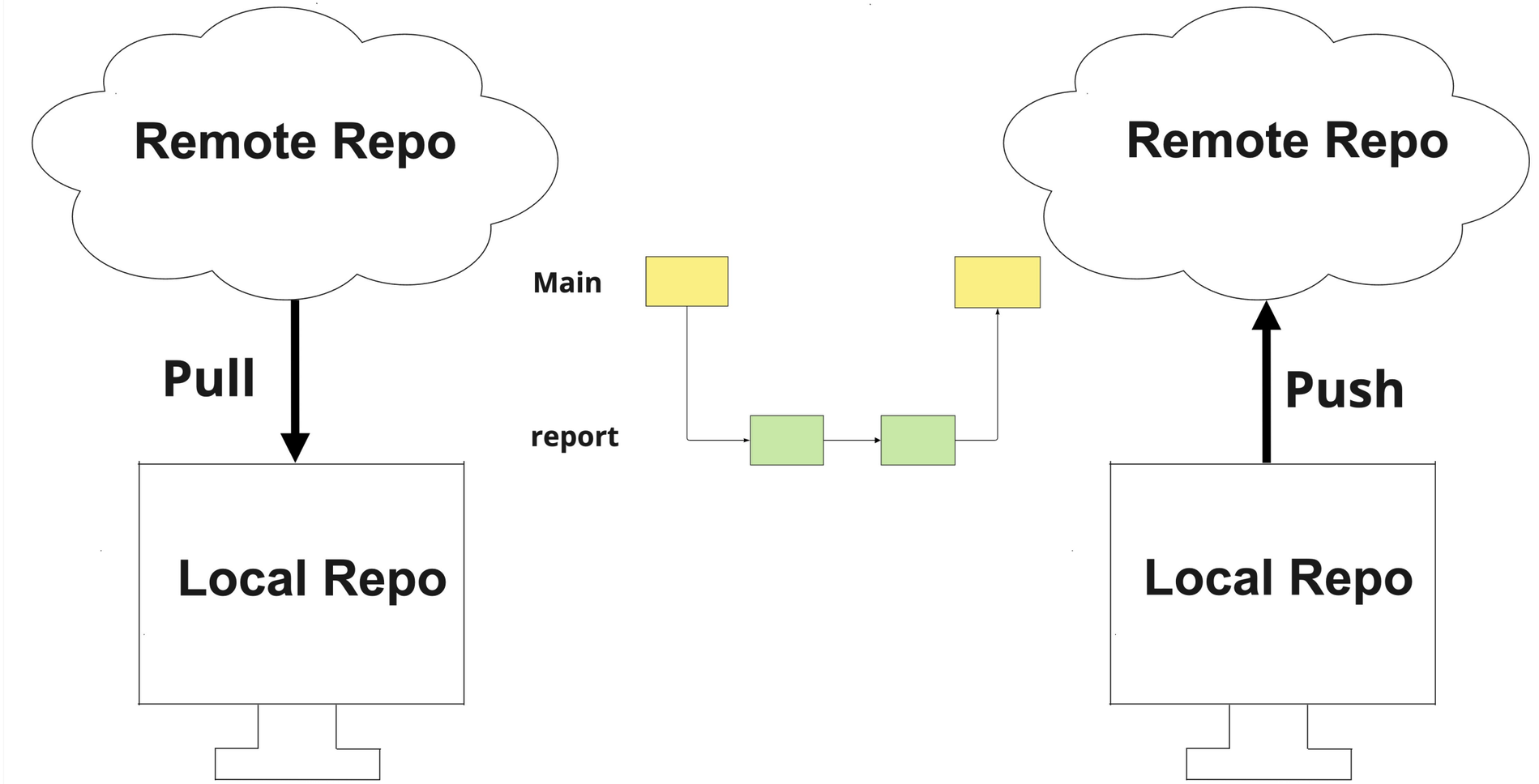
Push/pull workflow



Push/pull workflow



Push/pull workflow



Pushing first

- Pushing `main` to the remote before pulling

```
git push origin main
```

Remote/local conflicts

```
To https://github.com/datacamp/project
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/datacamp/project'
hint: Updates were rejected because the tip of your branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ..') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Remote/local conflicts

Remote

```
To https://github.com/datacamp/project
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/datacamp/project'
hint: Updates were rejected because the tip of your branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ..') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Remote/local conflicts

Outcome

```
To https://github.com/datacamp/project
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/datacamp/project'
hint: Updates were rejected because the tip of your branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ..') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Remote/local conflicts

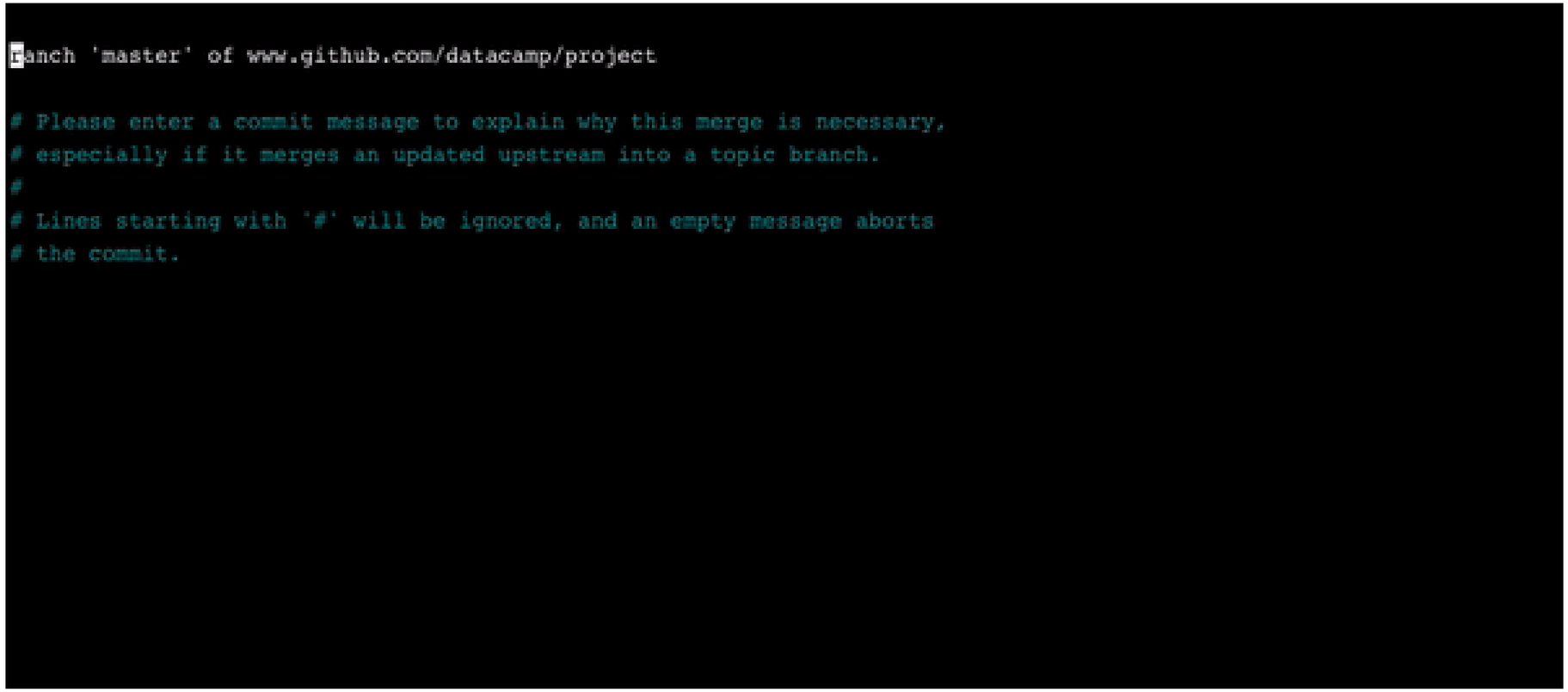
Reason(s) and suggestion(s)

```
To https://github.com/datacamp/project
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/datacamp/project'
hint: Updates were rejected because the tip of your branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ..') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Avoiding a conflict

- Pull from the remote first

```
git pull origin main
```



Pulling without editing

```
git pull --no-edit origin main
```

- Not recommended, unless we are very confident in the history of our project!

Pushing a new local branch

- Working in `hotfix` branch locally
- `hotfix` does not exist in the remote

Creating a new remote branch

- `hotfix` only exists locally

```
git push origin hotfix
```

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 349 bytes | 349.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'hotfix' on GitHub by visiting:
remote:      https://github.com/datacamp/project/pull/new/hotfix
remote:
To https://github.com/datacamp/project
 * [new branch]      hotfix -> hotfix
branch 'hotfix' set up to track 'origin/hotfix'
```

Let's practice!

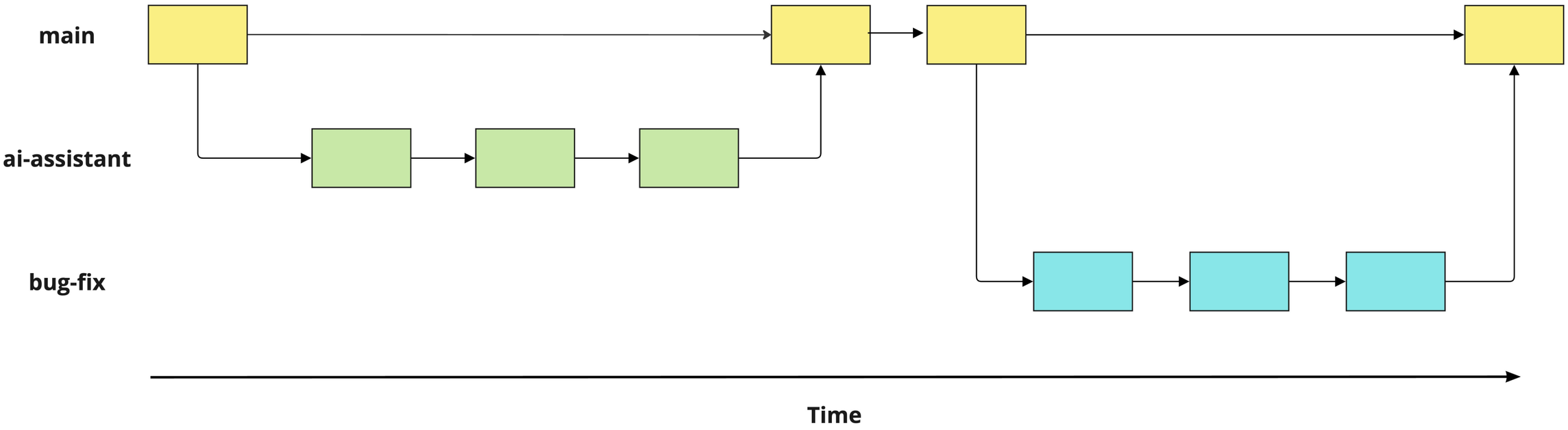
INTERMEDIATE GIT

Congratulations

INTERMEDIATE GIT

George Boorman
Curriculum Manager, DataCamp

Branches



Working with branches

- See all branches

```
git branch
```

- Switch to an existing branch

```
git switch hotfix
```

- Create and switch to a new branch

```
git switch -c hotfix
```

- Compare two branches

```
git diff main hotfix
```

- Rename a branch

```
git branch -m hotfix bugfix
```

- Delete a branch

```
git branch -d hotfix
```

Merging branches

- From `main`, to merge `ai-assistant` into `main`:

```
git merge ai-assistant
```

- From another branch: `git merge source destination`

```
git merge ai-assistant main
```

- Handling merge conflicts

Working with remotes

- Clone a remote repo

```
git clone https://github.com/datacamp/project
```

- Get information about all remotes

```
git remote -v
```

- Add a new remote

```
git remote add george https://github.com/george_datacamp/repo
```

Synchronizing local and remote repos

```
git fetch origin
```

```
git pull origin
```

```
git push origin documentation
```

What next?

- **GitHub Concepts**
- **GitHub and Git Tutorial**
- Starting using Git for your software and data projects!

Let's practice!

INTERMEDIATE GIT