# Setting up Git on a new computer (our RStudio Server)

Childhood Cancer Data Lab

## Setting up Git on a new computer

- 1. Set up a .gitconfig file to tell Git some of your configuration settings
  - At a minimum, the name and email associated with your commits
  - There are *many* settings you can add to your configuration file!
  - We will use the git config command to add settings to this file
- 2. Set up credentials to securely link with your account on GitHub.com

## Approaches to Git credentials: HTTPS or SSH

- HTTPS (we'll be doing this one!)
  - Common choice for beginners
  - Authenticate with username and *token* (Git no longer allows password auth vis https)
    - Personal Access Token (PAT): securely scoped token linked to your GitHub account which you can use instead of a password when working on command line

#### • SSH: Secure Shell

- Connects with paired local private key and remote public keys
- The private key stored locally and encrypted
- Common choice for more advanced Git users

#### We'll use HTTPS with a PAT for this workshop

• Read more about <u>Personal Access Tokens on GitHub</u> and <u>scopes</u>

#### New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.



Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

repo repo_deployment public_repo repo_invite security_events	Full control of private repositories Access commit status Access deployment status Access repositories Access repository invitations Read and write security events	scol
workflow	Update GitHub Action workflows	
write:packages read:packages	Upload packages to GitHub Package Registry Download packages from GitHub Package Registry	
delete:packages	Delete packages from GitHub Package Registry	
□ admin:org	Full control of orgs and teams, read and write org projects	
write:org	Read and write org and team membership, read and write org projects	
read:org	Read org and team membership, read org projects	

Other useful repo scopes

- user: Read/write access to profile info only.
- gist: Write access to GitHub gists ("blog posts"-ish)
- notifications: Read access to notifications, mark as read

#### Let's get set up!

#### In the **Terminal**, we'll begin with these commands:

# Tell GitHub your email and username associated with commits
git config --global user.email "your\_email@example.com"
git config --global user.name "name"

# Cache credentials for 12 hours (aka, type in your PAT fewer times)
git config --global credential.helper "cache --timeout=43200"

# (Optional but useful!) Name default branches "main"

git config --global init.defaultBranch "main"

#### Now we'll set up our PAT and save it securely

On GitHub.com, go to:

Settings > Developer Settings > Personal Access Tokens

Protips!

- Once the PAT is created, you only get *one chance* to save it. Don't close the window before saving!
- Password Managers are your friend for securely storing tokens (and more)

#### Next, we'll create our repository

1. Run git init in Terminal to initialize a repository in ~/training-modules

2. Add and commit an initial set of files

3. Create an empty repository on GitHub.com so there is a remote to push to

4. Tell Git where to push to, and push!

#### Using Git from the command line

# Staging files for commits (briefly...)
# Add a new or modified file to the commit
git add <name of file>

# Remove a file from version control
git rm <name of file>

# Add all files in a directory (DANGER!)
git add .

# Commit your changes

# Provide a commit message on the command line
git commit -m "Informative commit message"

# Stephanie's greatest hits

# What changes did I make?
git diff <name of modified file>

# What's my status?
git status

# Woops! I didn't mean to change that
(unstaged) file
git restore <name of modified file>

# Woops! I didn't mean to stage that
file
git reset <name of staged file>

# Drawbacks of using git commit -m

- You don't get any clear indication of which files you're committing when you run the command
  - Protip: Avoid this mystery by running git status first!

• You can only provide a short message, which is usually fine, but more involved situations may merit a more detailed commit message

#### You can just use git commit without a flag, but...

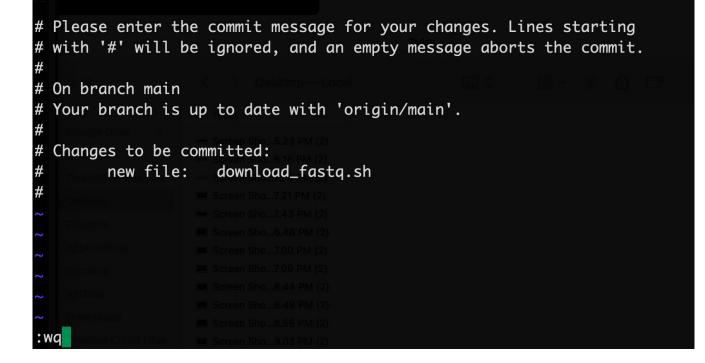
• ...you'll also get thrown into the command line text editor vi which may not be a pleasant experience!

• Don't want to get stuck in vi?

git config --global core.editor <editor-i-like-more>

```
# for example:
git config --global core.editor nano
git config --global core.editor emacs
```

#### Escaping the vi text editor



Did you end up here by accident and now you're stuck????

Type the following:

#### :wq

And hit enter!

→ Aborting commit due to empty commit message.

Then, you'll have to redo the commit.

## Pushing to origin

• Before we can push, we have to tell Git where to push to

# Tell git about the repo's remote URL. # We use the literal URL since we're using HTTPS auth git remote add origin {REMOTE-URL}

• The *first time* you push to a new branch, you have to tell Git which branch (remember, even the main branch doesn't exist yet on GitHub!)

```
# The first time you push to a new branch, use -u origin BRANCH
git push -u origin main
# Moving forward in this branch, we can just do...
git push
```

#### Protip!

You can skip the -u origin main bit it you add this to your config file:

# Always automatically create the remote branch upon pushing
git config --global push.autoSetupRemote "true"

Learn more about what you can specify in your config file!

- <u>Atlassian docs</u>
- <u>Git docs</u>

#### Helpful commands for working with branches

# create new branch (-c) and switch to it
git switch -c new-feature-branch

# unless you set your config as in the last slide, the first push requires `-u origin <name of remote branch to create>` git push -u origin new-feature-branch

# note that you can switch back to any existing branch, e.g. main, with: git switch main