

Intro to Git

Welcome to the pit.

A presentation heavily borrowed from datacamp and cobbled together
with some other things

Git Agenda

Git Background

Git Setup

Git Workflow

Git Collaboration

I've never heard of git

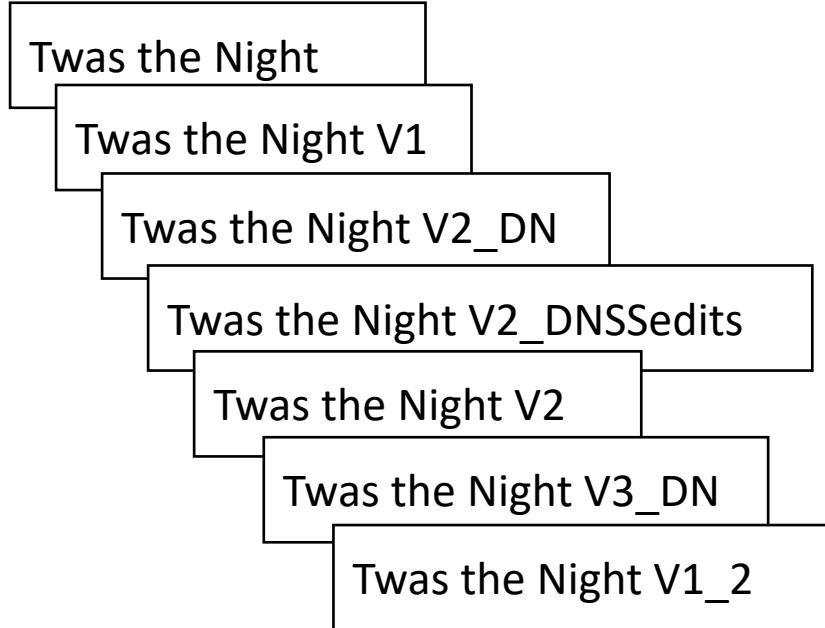
THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZIZE THESE SHELL
COMMANDS AND TYPE THEM TO SYNC UP.
IF YOU GET ERRORS, SAVE YOUR WORK
ELSEWHERE, DELETE THE PROJECT,
AND DOWNLOAD A FRESH COPY.



What even is git?



Twas the Night Before Christmas.doc - Word

REVIEW

Track Changes

'Twas the Night ~~B~~bbefore Christmas: A Visi~~on~~sion by Clement Clarke Moore (1779-1863)

Twas the night before Christmas, when all through the house
-Not a ~~creature-teacher~~ was stirring, not even a~~mouse-spouse~~.
-The stockings were ~~flung hung~~ by the chimney~~with care I swear~~,
In hopes that St ~~Nicholas-Nick seen would be there~~is on Medicare.

The children were nestled all snug in the ~~shedir beds~~,
While ~~millions visions~~ of ~~beanie babies sugar plums~~ danced ~~in-on~~ their ~~sledheads~~.
And mamma in her ~~spurs-kerchief~~, and I in my ~~chapseap~~,
Had just settled ~~a quarrel over reggae or rapour brains for a long winter's nap.~~

How do git repositories work?

- Git repositories are basically project folders containing:
 - project directory: directory containing all the files and folders associated with your project
 - git directory: directory containing all the extra information required to capture changes between versions.

The diagram illustrates the structure of a Git repository. It shows a file browser window on the right and a terminal window on the left.

File Browser (Right):

- Shows a list of project folders: DSPG2020, dspg21ari, facilityaccess, git_training, and sdad_data_commons.
- The **dspg21ari** folder is selected.
- The file browser interface includes tabs: Files, Plots, Packages, Help, Viewer.
- The path bar shows: Home > git > **dspg21ari**.
- The contents of the **dspg21ari** folder are listed:

 - ..
 - .gitignore (81 B, Jun 11, 2021, 10:00 AM)
 - .Rhistory (3.2 KB, Jul 27, 2021, 4:40 PM)
 - data
 - dspg21ari.Rproj (258 B, Jun 2, 2022, 11:17 AM)
 - functions
 - output
 - README.md (50 B, Jun 10, 2021, 3:52 PM)
 - src

Terminal (Left):

- Shows the command being run: `git pull <remote> <branch>`
- Output of the command:

```
remote: Compressing objects: 100% (14/14), done.  
remote: Total 688 (delta 1), reused 1 (delta 1), pack-reused 673  
Receiving objects: 100% (688/688), 20.72 MiB / 53.31 MiB/s, done.  
Resolving deltas: 100% (390/390), completed with 1 local object.  
From https://github.com/DSPG-Young-Scholars-Program/dspg21ari  
 1fe389d..a5b8691 main      -> origin/main  
There is no tracking information for the current branch.  
Please specify which branch you want to merge with.  
See git-pull(1) for details.
```
- Help text for setting tracking information:

```
git pull <remote> <branch>  
  
If you wish to set tracking information for this branch you can do so with:  
  
git branch --set-upstream-to=origin/<branch> jns
```
- Current working directory and file listing:

```
bash-4.4$pwd  
/home/js2mr/git/dspg21ari  
bash-4.4$ls -a  
. . . data 1_31_i_P_j_S_t_i .git .gitignore output README.md .Rhistory .Rproj.use src  
bash-4.4$
```

What even is git?



Gizem – Version 1

A screenshot of a Java IDE (NetBeans) showing a single file named "Main.java". The code implements a KNN classifier with a fixed k-value of 3. It loads training and testing data from CSV files, normalizes the data, and then iterates through the testing set to find the three nearest neighbors in the training set based on Euclidean distance. It then predicts the class of each test sample based on the majority class of its neighbors and prints the results to the console.

Devika – Version 2

A screenshot of a Java IDE (NetBeans) showing the same "Main.java" file as version 1. The code is identical, implementing a KNN classifier with k=3. It loads data from CSV files, normalizes it, and finds the three nearest neighbors for each test sample to predict their class.

Teja – Version 3

A screenshot of a Java IDE (NetBeans) showing the same "Main.java" file as versions 1 and 2. The code is identical, implementing a KNN classifier with k=3. It loads data from CSV files, normalizes it, and finds the three nearest neighbors for each test sample to predict their class.

What even is git?



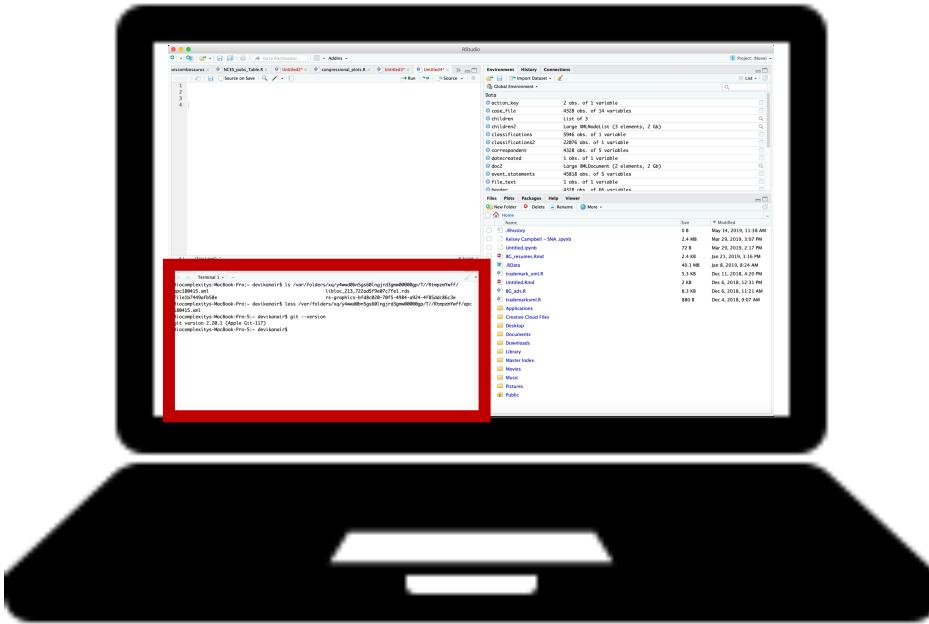
A GitHub logo featuring its black cat icon inside a thought bubble. Below the bubble is the word "GitHub".

lib/compose.js

View file @e71062d

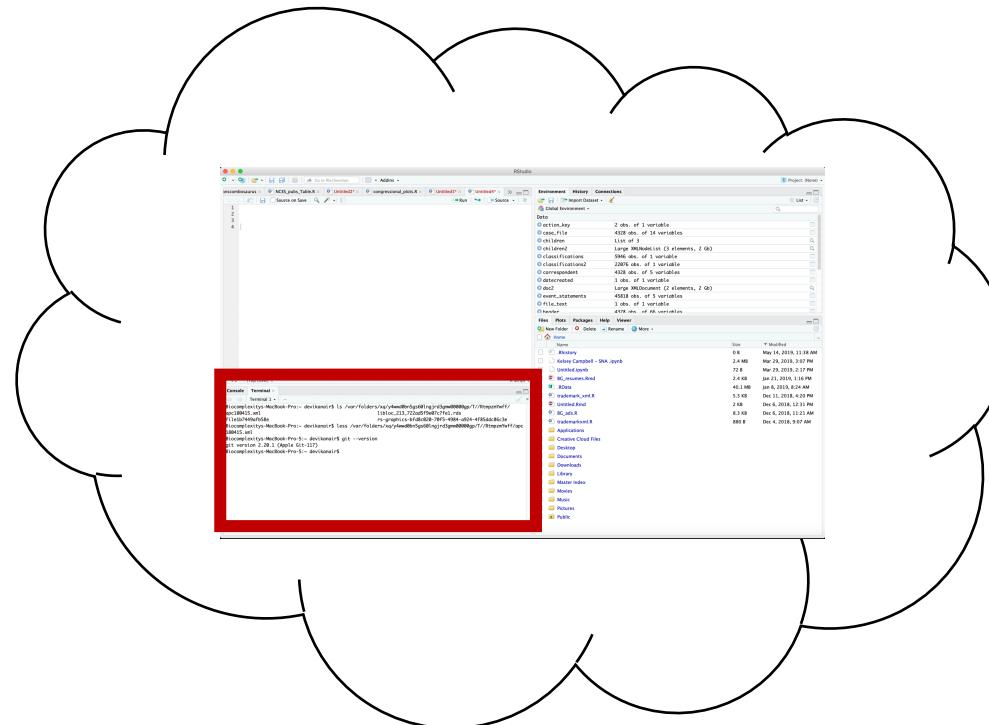
```
@@ -12,19 +12,27 @@ function setWritability(obj, writable) {  
12   }));  
13 }  
14  
origin//their changes  
15 function mixin(base, mixins) {  
16   base.mixedIn = base.hasOwnProperty('mixedIn') ? base.mixedIn :  
17   [];  
  
18   for (var i = 0; i < mixins.length; i++) {  
19     if (base.mixedIn.indexOf(mixins[i]) == -1) {  
20       module.exports = {  
21         mixin: mixin  
22     };  
23   }  
24 }  
25  
HEAD//our changes  
26 function mixin(base, mixins) {  
27   base.mixedIn = Object.prototype.hasOwnProperty.call(base, 'mi  
28 xedIn') ? base.mixedIn : [];  
29  
30   for (var i = 0; i < mixins.length; i++) {  
31     if (base.mixedIn.indexOf(mixins[i]) == -1) {  
32       setWritability(base, false);  
33       mixins[i].call(base);  
34       base.mixedIn.push(mixins[i]);  
35     }  
36   }  
37 }  
38  
39 module.exports = {  
40   mixin: mixin  
41 };
```

Where are we doing git?



Your personal laptop and RStudio application is for practice, personal projects.

All project-related code must be written to the RStudio server.

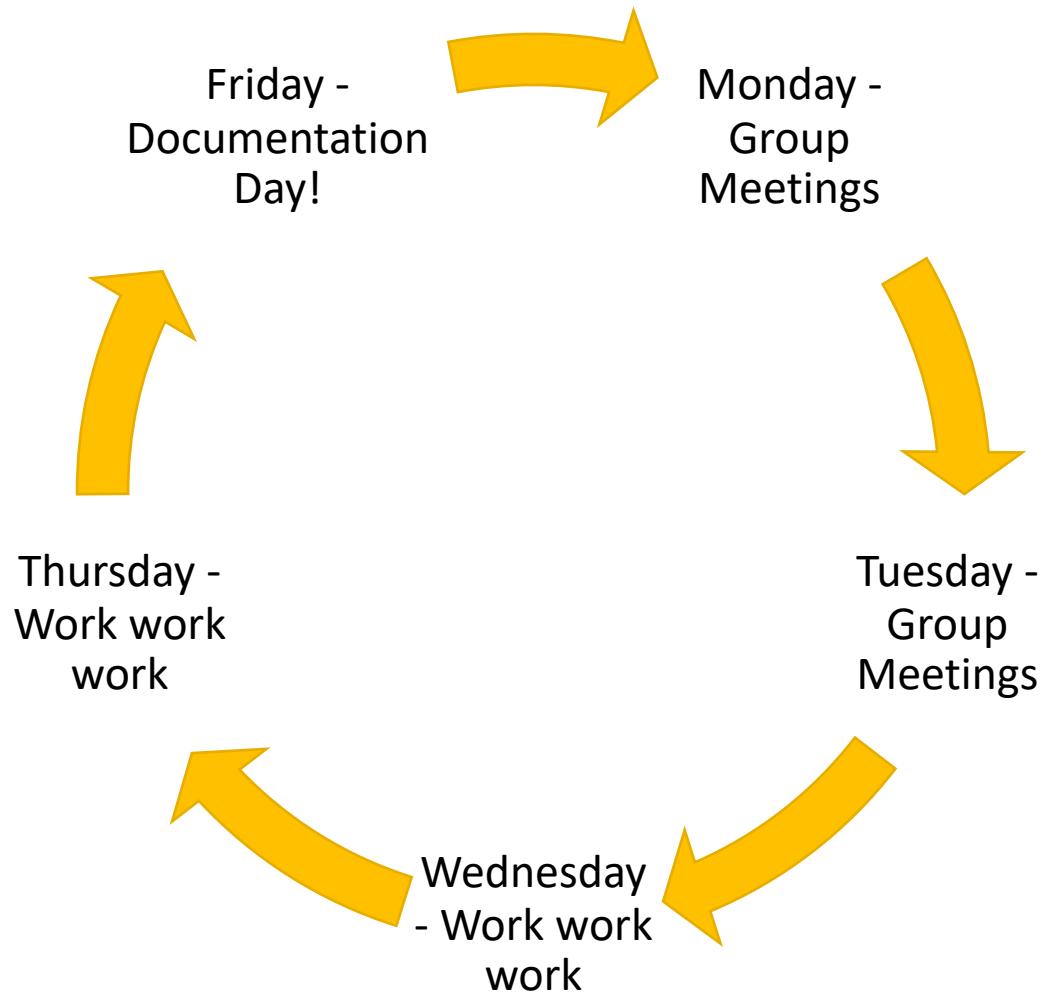


We will always do git in the terminal.

Your Role

Fellows responsible for reviewing and merging code?

- Does it run?
- No datasets are checked in?
- Are there loops that can be written more simply with an apply or map function?
- Is there repetitive code that could be written more simply using a custom function?
- Are your assumptions about the data tested in code?



Let's Git It Started

Git Setup

Git Installation

Git Configuration

Git Repositories

Git On Up

Git Installation

- Step 1: Check to see if you already have git installed.

Open up a terminal window

Enter the following

```
git --version
```

- Results: If you see this, then:

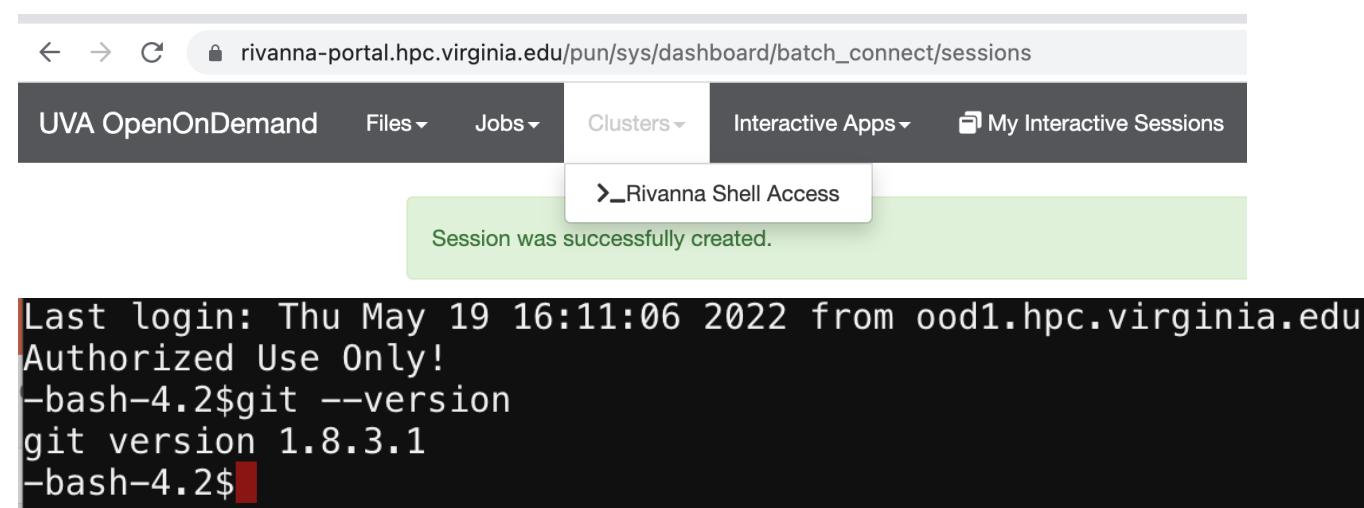
```
git version 1.8.3.1
```

✓ YOU'RE GOOD

'git' is not recognized as an internal or
external command, operable program or batch file

✗ Do not pass go, instead go here:

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>



The screenshot shows a web browser window with the URL `rivanna-portal.hpc.virginia.edu/pun/sys/dashboard/batch_connect/sessions`. The page has a dark header with tabs for "UVA OpenOnDemand", "Files", "Jobs", "Clusters", "Interactive Apps", and "My Interactive Sessions". A dropdown menu for "Clusters" is open, showing "Rivanna Shell Access". Below the header, a green success message box says "Session was successfully created.". The main content area is a black terminal window displaying the following text:

```
Last login: Thu May 19 16:11:06 2022 from ood1.hpc.virginia.edu
Authorized Use Only!
-bash-4.2$git --version
git version 1.8.3.1
-bash-4.2$
```

(I Can't Git No) Satisfaction

Git Configuration

- Step 2: If you don't already have a [GitHub account](#), create one. If you do, log into it.

- Step 3: Check git setup on your computer

```
git config --list
```

- Step 4: Setup git on your computer

Add your name and email to your settings
using your GitHub email account

```
git config --global user.name "Cesar Montalvo"  
git config --global user.email cpm9w@virginia.edu
```

Double check this change went through by using:

```
git config --global user.name  
git config --global user.email
```

- Step 5: Navigate on GitHub:
 - User icon in top right -> Settings -> Developer settings -> Personal access tokens

- Step 6: Generate new token
 - Note: dspgkey
 - Expiration: 90 days
 - Scope: repo

Generate token

- Step 7: Copy & save the key
 - Password manager
 - Environmental variable

The image shows two screenshots illustrating the process of generating a personal access token.

Top Screenshot: A screenshot of the GitHub user menu. The user is signed in as "jo-schroeder". The "Settings" option is highlighted with a red box. Other options visible include "Set status", "Your profile", "Your repositories", "Your codespaces", "Your organizations", "Your projects", "Your stars", "Your gists", "Upgrade", "Feature preview", "Help", and "Sign out".

Bottom Screenshot: A screenshot of the "Generate token" form. It includes fields for "Note" (set to "dspgkey"), "Expiration" (set to "90 days"), and a "Select scopes" section. The "repo" scope is selected. Other available scopes listed are "repo:status", "repo_deployment", "public_repo", "repo:invite", and "security_events". A note at the bottom states: "Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)"

How does git work?

Git Workflow

1. Write your code

2. Stage your code

3. Commit your code

4. Push your code

5. Merge your code

How does git work?

The screenshot shows the RStudio interface with several windows open:

- Editor:** Shows an R script named `vis_script.R` containing code related to DataExplorer and dplyr packages.
- Console:** Shows the execution of R code, including ggplot2 and Hmisc packages, to generate plots for value distribution.
- Terminal:** Shows the command `git add .` being run in the current directory (`/git/stem_edu`).
- Environment:** Shows the global environment with various objects like `BGpoint`, `BGshape`, `con`, etc.
- History:** Shows the history of changes made to the repository.
- Connections:** Shows database connections.
- Git:** Shows the local repository structure with files like `3_cip_profiling_final.html`, `3_cip_profiling_final.Rmd`, etc.

A large yellow arrow points from the code in the Editor window towards the Terminal window.

```
library(DataExplorer)
melt(DataExplorer::introduce(ads_cip_0717))
DataExplorer::profile_missing(ads_cip_0717)
### Uniqueness of Unique ID
#IDs that appear more than once
dupejobids <- ads_cip_0717 %>% dplyr::group_by(bgtjobid) %>% dplyr::summarise(count = dplyr::n()) %>% dplyr::filter(count > 1)
#Multi-appearance ID
dupecount <- ads_cip_0717 %>% select(-bgtjobid) %in% dupejobids$bgtjobid, c("bgtjobid", "jobdate")) %>% group_by(bgtjobid)
nonuniqueIDs <- names(dupejobids) - nrow(dupecount)
print(paste(nonuniqueIDs, "unique job ids have multiple dates associated with them."))
### Histograms
# datable[fil Select, BY]
### Job Date
values <- ads_cip_0717[!is.na(lubridate::as_date(jobdate)), lubridate::as_date(jobdate)]
(BT Profiling)
```

```
git add .
```

```
ggplot(plotfreq, aes(x=names(plotfreq[,1]), y=names(plotfreq[,2])), main="Top Value Distribution") +  
  xlab(Hmisc::capitalize(names(plotfreq[,1]))) + ylab("N") +  
  theme(axis.text.x=element_text(angle = 90, hjust = 1)) +  
  coord_flip()  
  print(plot)  
}  
remove(i)  
for (i in seq_along(plotfreq)) {  
  plot <- ggplot(plotfreq[i,], aes(V1, N)) +  
    geom_bar(stat = "identity", fill = "paleturquoise4", width=.7) +  
    ggtitle(paste(Hmisc::capitalize(names(plotfreq[i,])), ": Top Value Distribution")) +  
    xlab(Hmisc::capitalize(names(plotfreq[,1]))) + ylab("N") +  
    theme(axis.text.x=element_text(angle = 90, hjust = 1)) +  
    coord_flip()  
  print(plot)  
}  
Hmisc::capitalize(names(plotfreq[5]))  
[1] "County"  
Hmisc::capitalize(names(plotfreq[4]))  
[1] "City"
```

1. Write your code

How does git work?



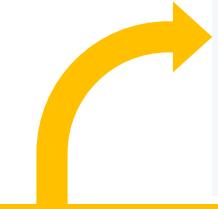
```
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git status
On branch devika_bgt_val
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   src/burn_glass_validation/bgt_technical_doc.Rmd

no changes added to commit (use "git add" and/or "git commit -a")
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git add .
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$
```

2. Stage your code

How does git work?



```
Console Terminal x
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git status
On branch devika_bgt_val
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   src/burn_glass_validation/bgt_technical_doc.Rmd

no changes added to commit (use "git add" and/or "git commit -a")
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git add .
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git commit -m "edits to technical doc"
[devika_bgt_val a27d8fa] edits to technical doc
 1 file changed, 1 insertion(+), 1 deletion(-)
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$
```

2. Commit your code

How does git work?

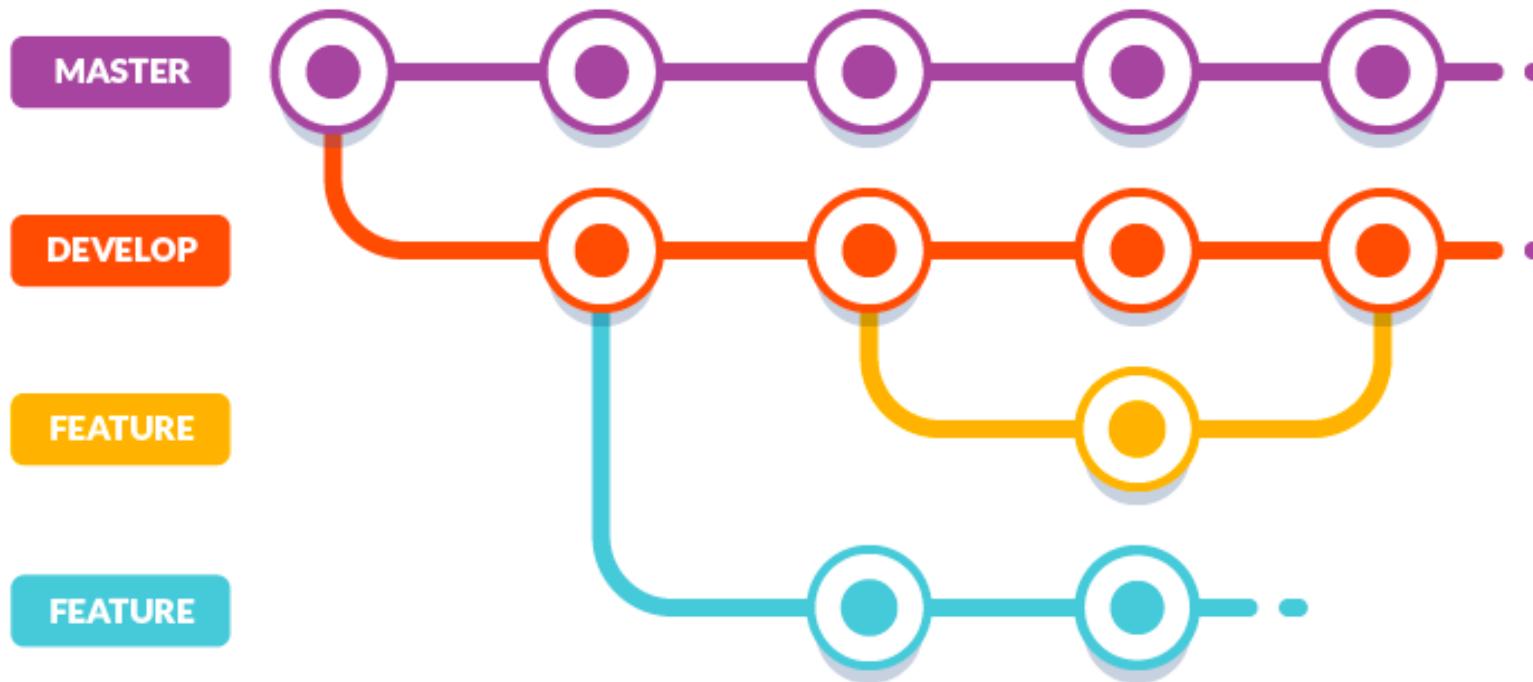


```
Console Terminal x
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu
modified:   src/burn_glass_validation/bgt_technical_doc.Rmd

no changes added to commit (use "git add" and/or "git commit -a")
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git add .
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git commit -m "edits to technical doc"
[devika_bgt_val a27d8fa] edits to technical doc
 1 file changed, 1 insertion(+), 1 deletion(-)
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$ git push origin devika_bgt_val
Password for 'http://dnair@sdad.policy-analytics.net:30080':
Counting objects: 5, done.
Delta compression using up to 12 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 473 bytes | 473.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0)
remote:
remote: To create a merge request for devika_bgt_val, visit:
remote:   http://sdad.policy-analytics.net:30080/sdad/stem_edu/merge_requests/new?merge_request%5Bsource_branch%5D=devika_bgt_val
remote:
To http://sdad.policy-analytics.net:30080/sdad/stem_edu.git
 17cd8dc..a27d8fa  devika_bgt_val -> devika_bgt_val
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/stem_edu$
```

4. Push your code

Git Branches



Master Branch:
Functional combination of all team members' code



Branch:
A copy of the code that is undergoing development



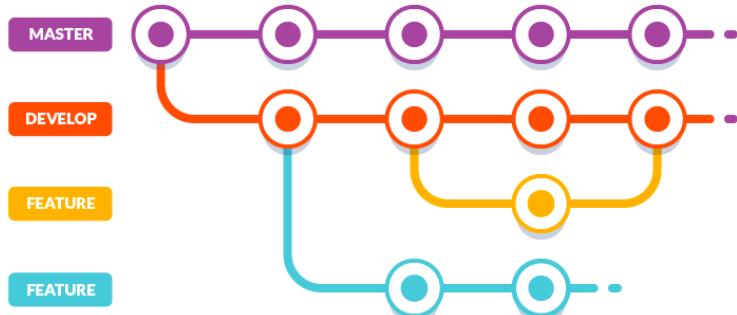
Branch:
A copy of the code that is undergoing development



Branch:
A copy of the code that is undergoing development



Git Branches



Master Branch:

Functional combination of all team members' code



Branch:

A copy of the code that is undergoing development



Branch:

A copy of the code that is undergoing development



Branch:

A copy of the code that is undergoing development



Example:

Business Innovation – detecting innovation in text sources



Gizem Task:

Gizem working to import journal articles text data



Devika Task:

Devika working to articulate data structure and data types of articles



Teja Task:

Teja working to import and clean FDA data for subsequent validation



Git Branches

Example:

Business Innovation – detecting innovation in text sources



Gizem Task:

Gizem working to import journal articles text data



Devika Task:

Devika working to articulate data structure and data types of articles



Teja Task:

Teja working to import and clean FDA data for subsequent validation



Branch Name:

factiva_gizem

Scripts:

import_factiva.R
import_factiva_meta.R

Branch Name:

factiva_devika

Scripts:

import_factiva.R
clean_factiva.R

Branch Name:

fda_teja

Scripts:

import_FDA.R
clean_FDA.R

Save your work!

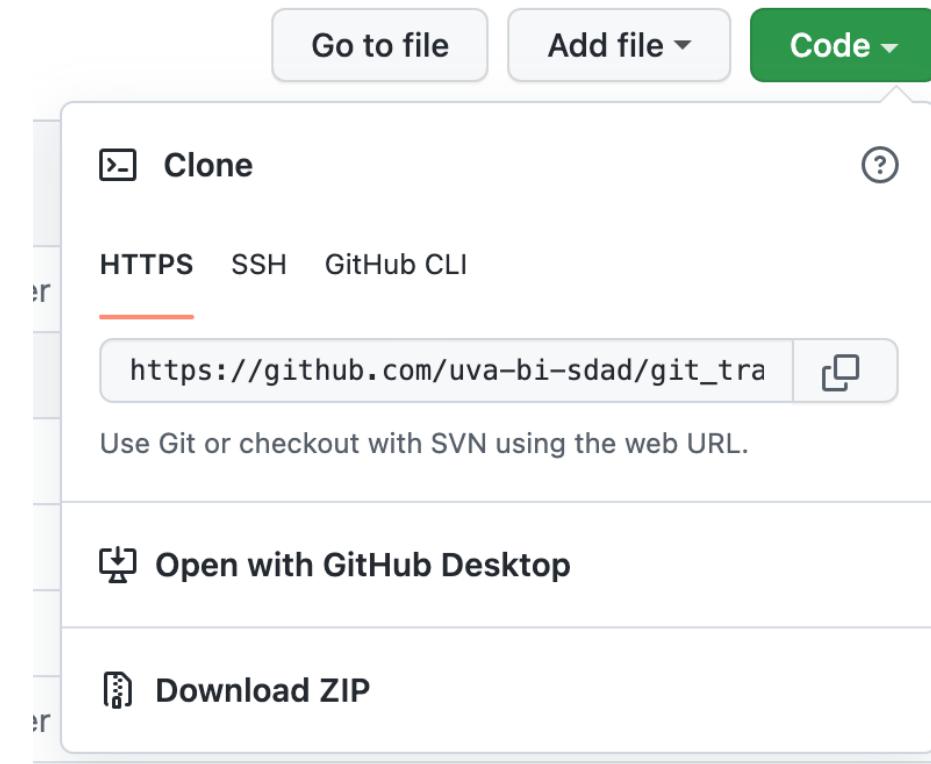
Branches are pushed up to GitHub for code review.

(git status, git add, git commit, git push)

Exercise

Git Test Code

- Step 1: Find the git_training repository on GitHub (uva-bi-sdad organization)
 - Copy the SSH link
https://github.com/uva-bi-sdad/git_training.git
- Step 2: Clone the project into your Rivanna RStudio container using Rstudio window controls
 - In your Home directory – create a new folder called ‘git’
 - File
 - New project
 - Version Control
 - Git
 - Repository URL <- https://github.com/uva-bi-sdad/git_training.git
 - Create project as a subdirectory of your new ‘git’ folder within your home directory



Exercise

Git Test Code

- Step 3: What branch are you on?
 - In terminal Run git status
 - Create your branch git checkout –b *your_branch*
- Step 4: Create a new file in /dspg2022 and save it as *your_name.txt*
- Step 5: In terminal
 - Run git status <- What has been changed?
 - Run git add . <- Stage new changes
 - Run git commit –m “*your message*” <- commit your changes
 - Run git push origin *your_branch*

Exercise

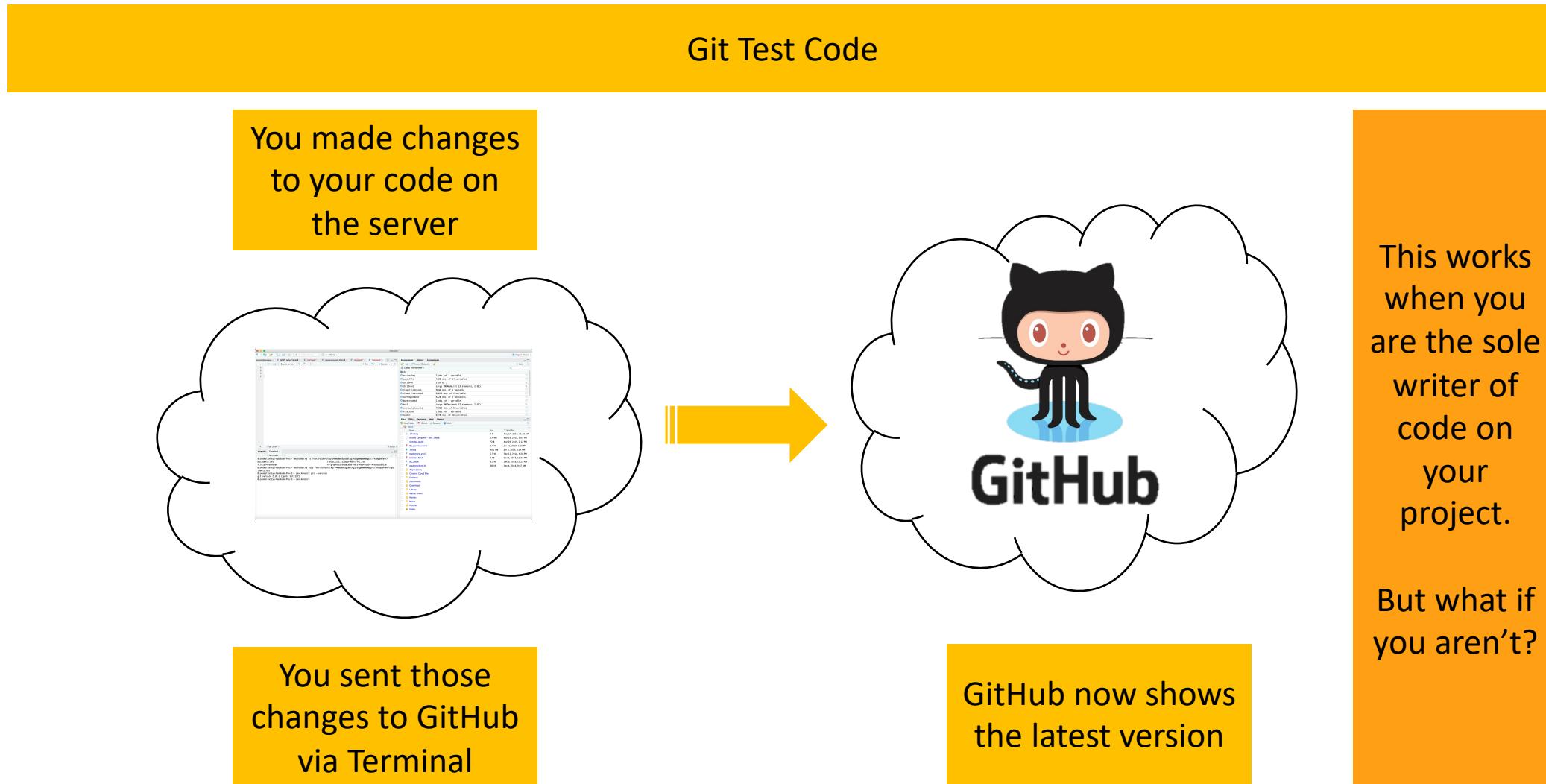
Git Test Code

- Step 6: Navigate to GitHub
 - Open a pull request
- Step 7: Delete your branch

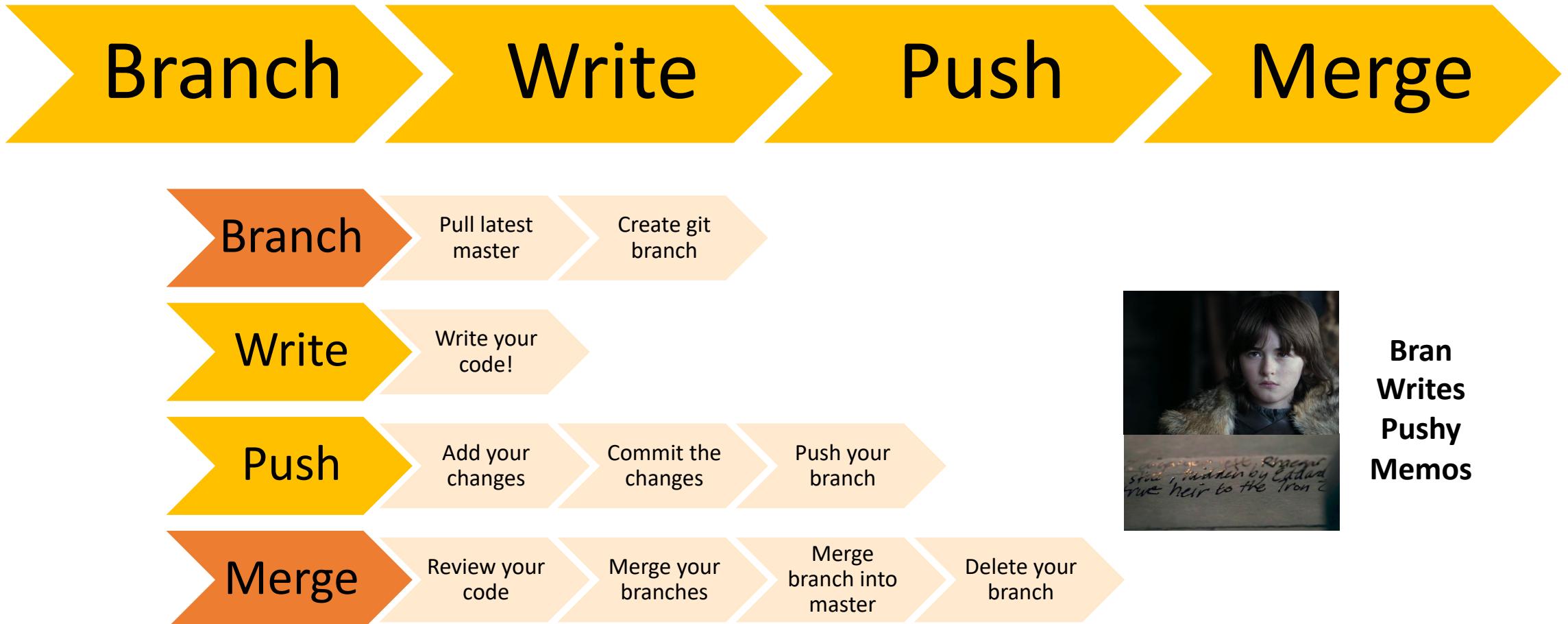
PAUSE

- Step 8: Git pull everyone's changes

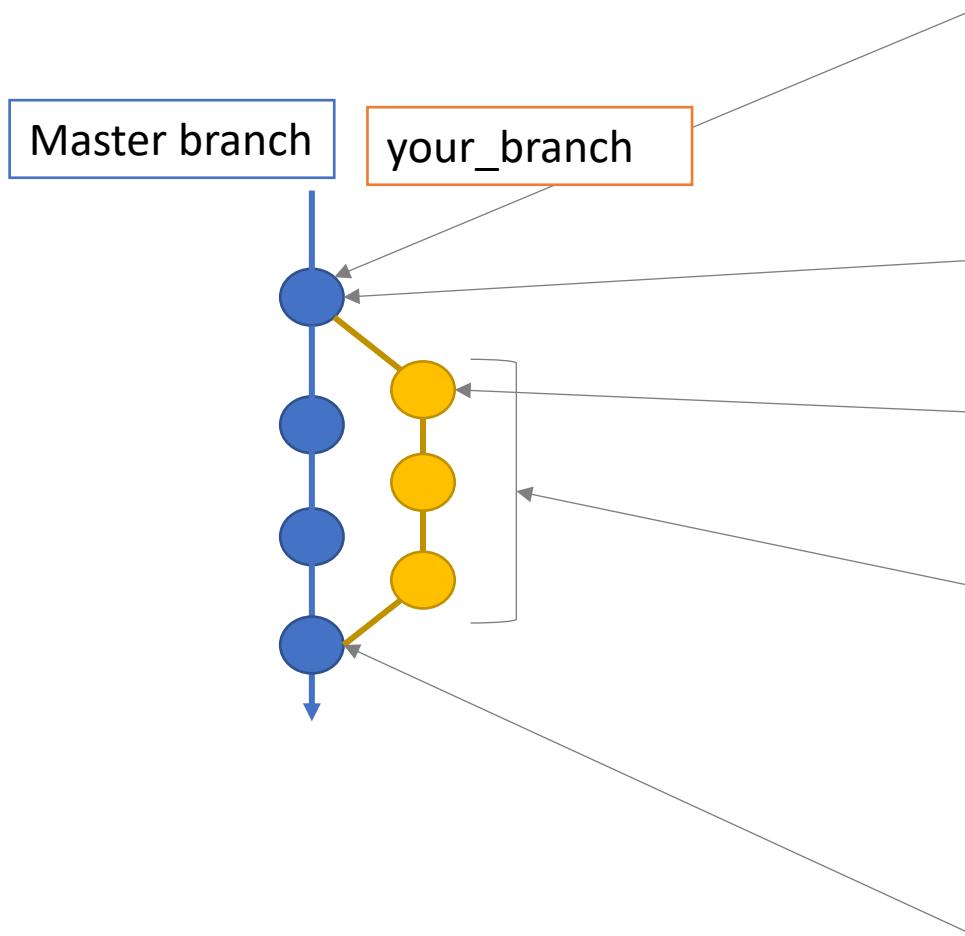
Exercise – What just happened?



Just remember – BWPM

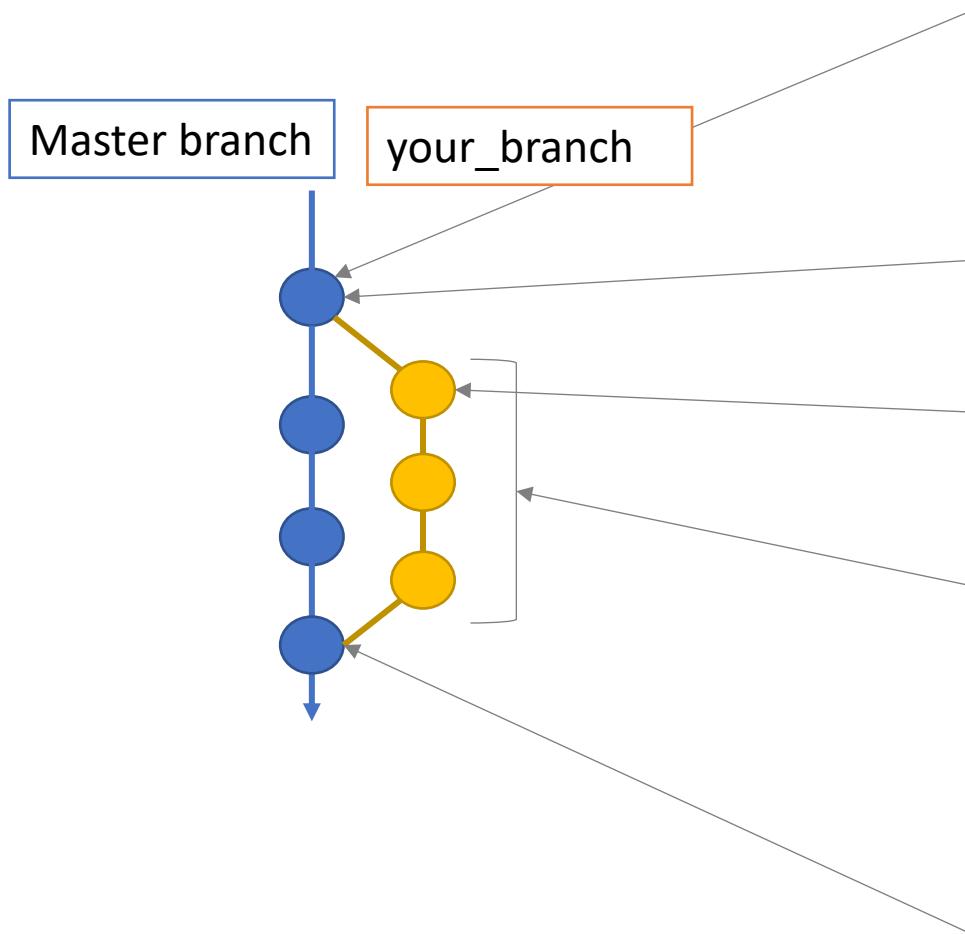


GIT CHEAT SHEET



0. Make sure you are in your project repository, either of these two ways:
 - Navigate to your project repo by opening the .Rproj file in Files  `dspg21ari.Rproj`
 - Use Terminal command line to:
 - To check your working directory: `pwd`
 - To see contents of folder: `ls -a`
 - To move into folder below: `cd foldername`
 - To move into folder above: `cd ..`
1. Make sure your master branch is up to date
 1. Go to the master branch: `git checkout master`
 2. Update your master branch: `git pull origin master`
2. Create a new branch: `git checkout -b your_branch`
 1. Add your name or initials as the branch name
 `dspg21ari.Rproj`
3. Go code and write commits!
 1. `git add <my file>`
 2. `git commit -m 'this is what I did!'`
4. Push your branch `git push origin your_branch`
5. Issue a pull request
6. Have someone (maintainers) review your code
7. (Maintainers) merge the pull request
 1. You can also checkoff a box that will also delete the branch on the remote
 2. Or delete the branch manually under the branches view
8. Go back to master: `git checkout master`
9. Pull down your merged code: `git pull origin master`
10. Delete your branch: `git branch -d your_branch` (note it is a **lower case d**)
11. Clean up your branches: `git fetch --prune`

GIT CHEAT SHEET



0. Make sure you are in your project repository, either of these two ways:
 - Navigate to your project repo by opening the .Rproj file in Files  `dspg21ari.Rproj`
 - Use Terminal command line to:
 - To check your working directory: `pwd`
 - To see contents of folder: `ls -a`
 - To move into folder below: `cd foldername`
 - To move into folder above: `cd ..`
1. Make sure your master branch is up to date
 1. Go to the master branch: `git checkout master`
 2. Update your master branch: `git pull origin master`
2. Create a new branch: `git checkout -b your_branch`
 1. Add your name or initials as the branch name
3. Go code and write commits!
 1. `git add <my file>`
 2. `git commit -m 'this is what I did!'`
4. Push your branch `git push origin your_branch`
5. Issue a pull request
6. Have someone (maintainers) review your code
7. (Maintainers) merge the pull request
 1. You can also checkoff a box that will also delete the branch on the remote
 2. Or delete the branch manually under the branches view
8. Go back to master: `git checkout master`
9. Pull down your merged code: `git pull origin master`
10. Delete your branch: `git branch -d your_branch` (note it is a **lower case d**)
11. Clean up your branches: `git fetch --prune`

Git Lucky

Git Collaboration

Git Branches

Git Push

Git Merge

Git Pull

Git Branch Workflow

Git Branching

Check git branch

Create git branch

Write your code

Push your code

Merge your code

Note: you already know how to write and push your code!

How does git work?



```
Console Terminal x
Terminal 1 dnair1@docker-s-12vcpu-48gb-nyc1-01: ~/git/business_innovation
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/business_innovation$ git status
On branch master
nothing to commit, working tree clean
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/business_innovation$
```

1. Check git branch

How does git work?



```
Console Terminal x
Terminal 1 dnair1@docker-s-12vcpu-48gb-nyc1-01: ~/git/business_innovation
```

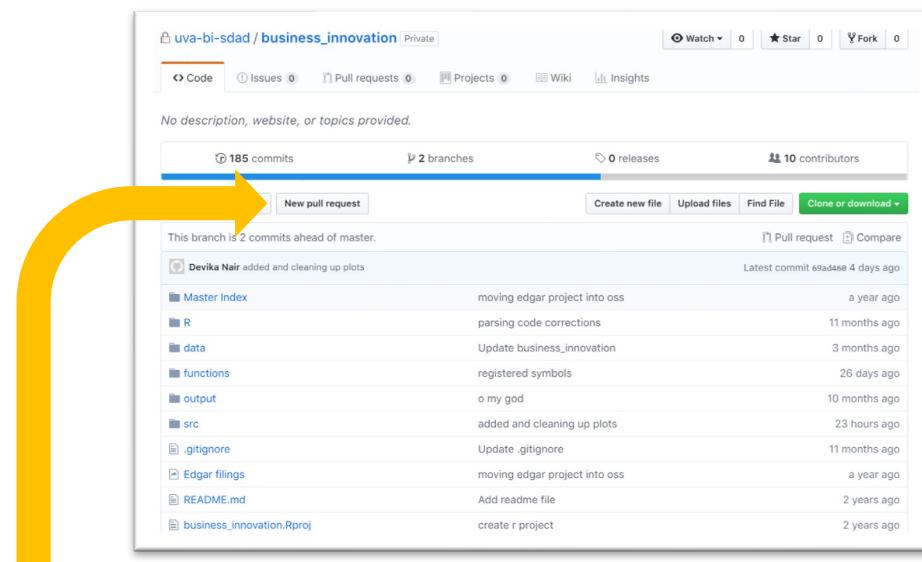
```
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/business_innovation$ git status
On branch master
nothing to commit, working tree clean
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/business_innovation$ git checkout -b partner_branch
Switched to a new branch 'partner_branch'
dnair1@docker-s-12vcpu-48gb-nyc1-01:~/git/business_innovation$
```

2. Create git branch

How does git work?

3. Write your code

4. Push your code



5. Merge your code

Exercise

Git Test Code

Check git branch

- Teams of 2
- Clone your partner's personal project (see Come Togither slide)
- Make partner collaborator on your project

Create git branch

- In Terminal
 - Check what branch you're on using `git status`
 - Create a new branch for yourself using `git checkout -b partner_branch`

Write your code

- In Console
 - Open your partner's `test.R` script
 - Make a change to the script, like changing `mtcars` to `head(mtcars)`
- In Terminal

- Check that the `test.R` script appears as modified using `git status`
 - `git add .`
 - `git commit -m "updated partner's test script"`
 - `git push origin branchname`

Enter GitHub password

Push your code

- `git status`
- Navigate to GitHub – Is your newest branch available?

When your coworker asks you which git branch you're currently working on



Exercise - what did we just do?

Example:
Personal project



Partner A Task: Partner A looking at mtcars data	Branch Name: master	Scripts: test.R	Save your work! Branches are pushed up to GitHub for code review. (git status, git add, git commit, git push)
Partner B Task: Partner B decided we only needed to look at first 6 rows of mtcars	Branch Name: partner_branch	Scripts: test.R	

Git Merge

Screenshot of a GitHub repository page for "uva-bi-sdad / business_innovation". The repository is private and has 185 commits, 2 branches, 0 releases, and 10 contributors. A yellow arrow points from a large yellow button at the bottom to the "New pull request" button in the top right of the commit list.

No description, website, or topics provided.

185 commits · 2 branches · 0 releases · 10 contributors

New pull request

This branch is 2 commits ahead of master.

Commit	Message	Date
Devika Nair added and cleaning up plots	Latest commit 69ad460 4 days ago	
Master Index	moving edgar project into oss	a year ago
R	parsing code corrections	11 months ago
data	Update business_innovation	3 months ago
functions	registered symbols	26 days ago
output	o my god	10 months ago
src	added and cleaning up plots	23 hours ago
.gitignore	Update .gitignore	11 months ago
Edgar filings	moving edgar project into oss	a year ago
README.md	Add readme file	2 years ago
business_innovation.Rproj	create r project	2 years ago

Create pull request



(pull request = merge request)

Git Merge

uva-bi-sdad / business_innovation [Private]

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Security Insights

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: master ▾ compare: devika_sec ▾ ✓ Able to merge. These branches can be automatically merged.

Devika sec

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request ▾

Reviewers
No reviews

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
None yet



Create pull request

Git Merge

Devika sec #1

 Open DevikaNair90 wants to merge 4 commits into `master` from `devika_sec` 

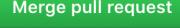
Conversation 0 Commits 4 Checks 0 Files changed 4

 DevikaNair90 commented just now + ...
No description provided.

 Devika Nair added some commits 9 days ago
updates to tech doc 01057d4
added and cleaning up plots 69ad460
adding labels and notes to EDA plots 80b9e0a
changes to fuzzymatching company names 9d595ac

Add more commits by pushing to the `devika_sec` branch on [uva-bi-sdad/business_innovation](#).

 This branch has no conflicts with the base branch
Merging can be performed automatically.

 Merge pull request  You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



Check for conflicts

Git Merge

Devika sec #1

 Open DevikaNair90 wants to merge 4 commits into `master` from `devika_sec` 

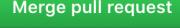
Conversation 0 Commits 4 Checks 0 Files changed 4

 DevikaNair90 commented just now + ...
No description provided.

 Devika Nair added some commits 9 days ago
updates to tech doc 01057d4
added and cleaning up plots 69ad460
adding labels and notes to EDA plots 80b9e0a
changes to fuzzymatching company names 9d595ac

Add more commits by pushing to the `devika_sec` branch on [uva-bi-sdad/business_innovation](#).

 This branch has no conflicts with the base branch
Merging can be performed automatically.

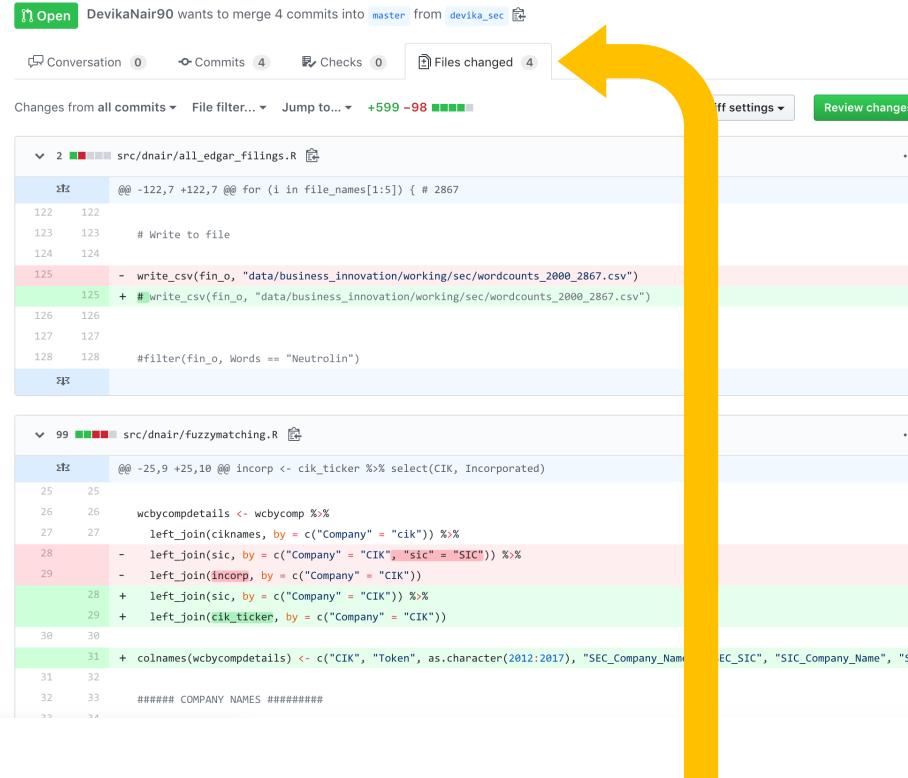
 Merge pull request  You can also [open this in GitHub Desktop](#) or view [command line instructions](#).



Check for conflicts

Git Merge

Devika sec #1



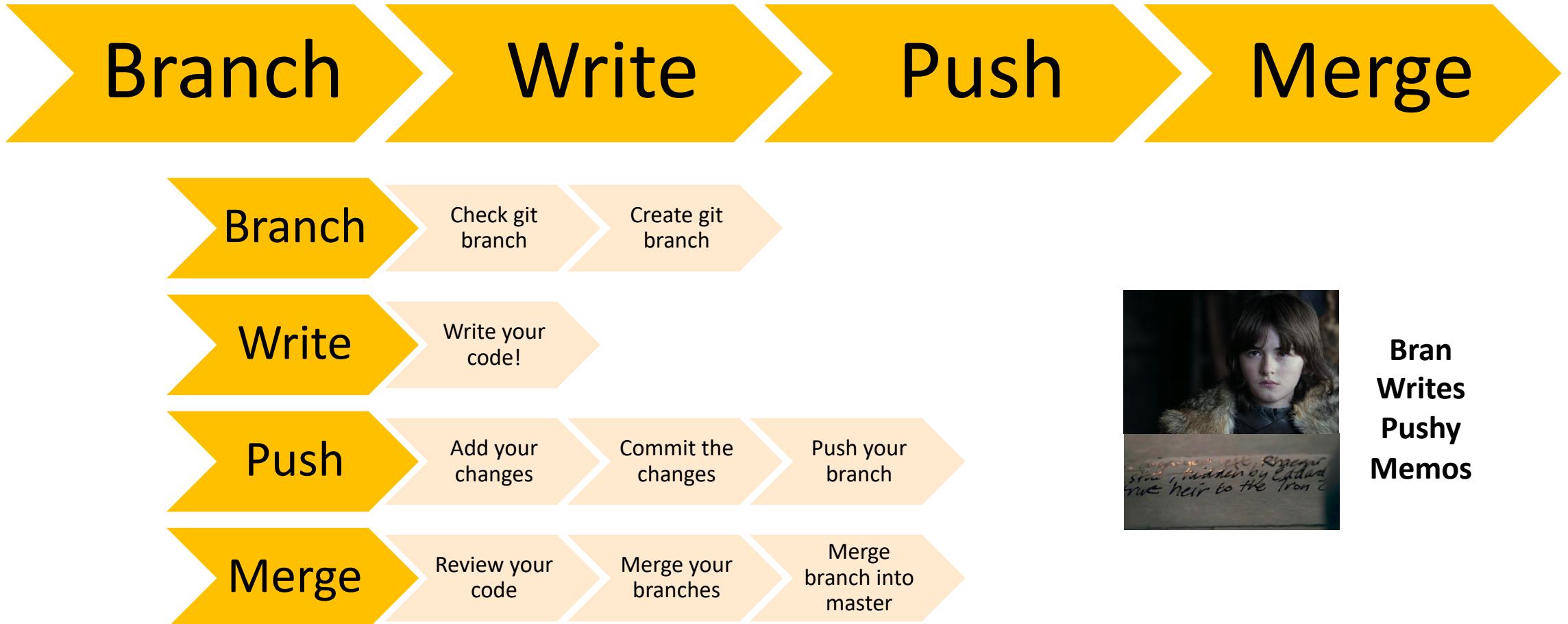
The screenshot shows a GitHub merge pull request titled "Devika sec #1" by DevikaNair90. The pull request merges four commits from the "devika_sec" branch into the "master" branch. The "Files changed" tab is selected, showing two files: "src/dnair/all_edgar_filings.R" and "src/dnair/fuzzymatching.R". The code diff highlights changes made in the "fuzzymatching.R" file, specifically in lines 125 through 30. A large yellow arrow points from the "Review changes" button at the top right towards the code diff area.

```
diff --git a/src/dnair/all_edgar_filings.R b/src/dnair/all_edgar_filings.R
@@ -122,7 +122,7 @@ for (i in file_names[1:5]) { # 2867
 122 122
 123 123     # Write to file
 124 124
-125 - write_csv(fin_o, "data/business_innovation/working/sec/wordcounts_2000_2867.csv")
+125 + #write_csv(fin_o, "data/business_innovation/working/sec/wordcounts_2000_2867.csv")
 126 126
 127 127
 128 128     #filter(fin_o, Words == "Neutrolin")
```

```
diff --git a/src/dnair/fuzzymatching.R b/src/dnair/fuzzymatching.R
@@ -25,9 +25,10 @@ incorp <- cik_ticker %>% select(CIK, Incorporated)
 25 25
 26 26     wcbycopdetails <- wcbycop %>%
 27 27         left_join(ciknames, by = c("Company" = "cik")) %>%
-28 -     left_join(sic, by = c("Company" = "CIK", "sic" = "SIC")) %>%
-29 -     left_join(incorp, by = c("Company" = "CIK"))
+28 +     left_join(sic, by = c("Company" = "CIK")) %>%
+29 +     left_join(cik_ticker, by = c("Company" = "CIK"))
 30 30
 31 31     colnames(wcbycopdetails) <- c("CIK", "Token", as.character(2012:2017), "SEC_Company_Name", "SEC_SIC", "SIC_Company_Name", "SIC_SIC")
 32 32     ##### COMPANY NAMES #####
```

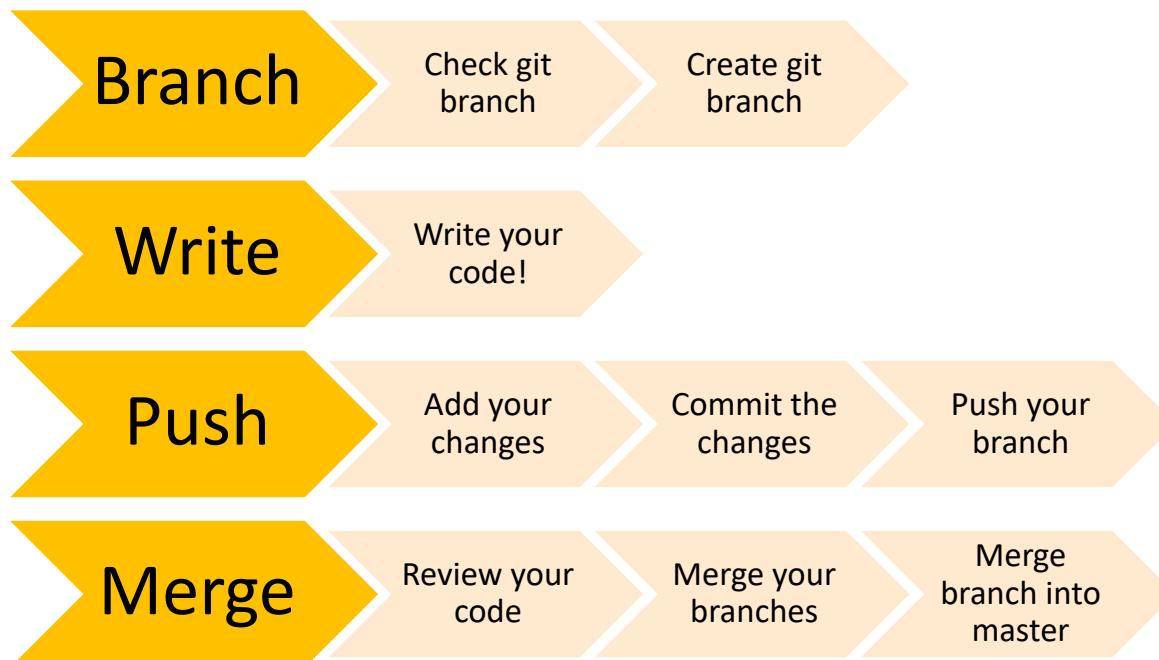
Review code changes

Just remember – BWPM



COVER YOUR EARS

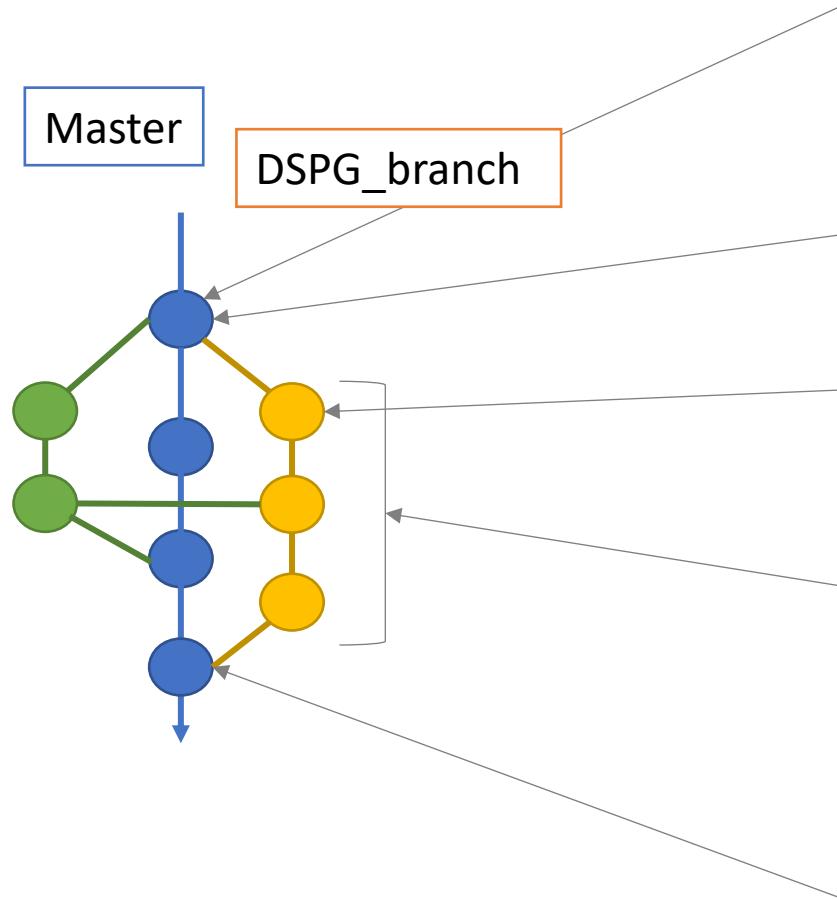
Just remember – BWPM



*...Raven still, broken by Eddard
true heir to the Iron Throne*



Bran
Writes 0 ravens?
gets **Pushed**
Emerges winner



0. Make sure you are in your project repository, either of these two ways:
 - Navigate to your project repo by opening the .Rproj file in Files
 - Use Terminal command line to:
 - To check your working directory: `pwd`
 - To see contents of folder: `ls -a`
 - To move into folder below: `cd foldername`
 - To move into folder above: `cd ..`

1. Make sure your master branch is up to date
 1. Go to the master branch: `git checkout master`
 2. Update your master branch: `git pull origin master`

2. Create a new branch (give it a useful name): `git checkout -b my-awesome_task`
 1. Give your branch a sensible name about what you are working on
 2. Add your name or initials to branch name

3. Go code and write commits!
 1. `git add <my file>`
 2. `git commit -m 'look at all this cool stuff'`
4. Push your branch `git push origin my-awesome_task`

5. Issue a pull request
6. Have someone (maintainers) review your code
7. (Maintainers) merge the pull request
 1. You can also check off a box that will also delete the branch on the remote
 2. Or delete the branch manually under the branches view
8. Go back to master: `git checkout master`
9. Pull down your merged code: `git pull origin master`
10. Delete your branch: `git branch -d my-awesome_task` (note it is a **lower case d**)
11. Clean up your branches: `git fetch --prune`

