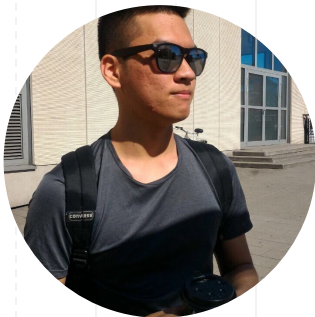




GEEKOUT 2022 GIT WORKSHOP

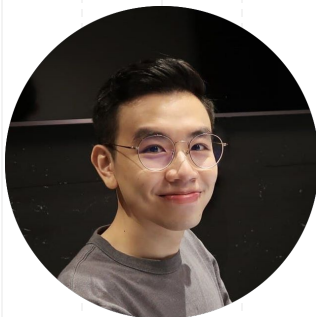
HELLO! MEET YOUR PRESENTERS



Lucas Yap

Current Project - Notarise

Hobbies - Elden Ring, Gunpla,
Lego



Ernest Boey

Current Project - Container Stack

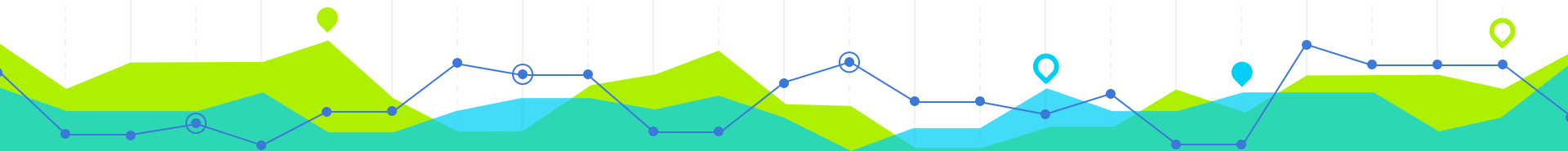
Hobbies -  



Darryl Sw

Current Project - Container
Stack, GCC 2.0

Hobbies - Biking, Hiking

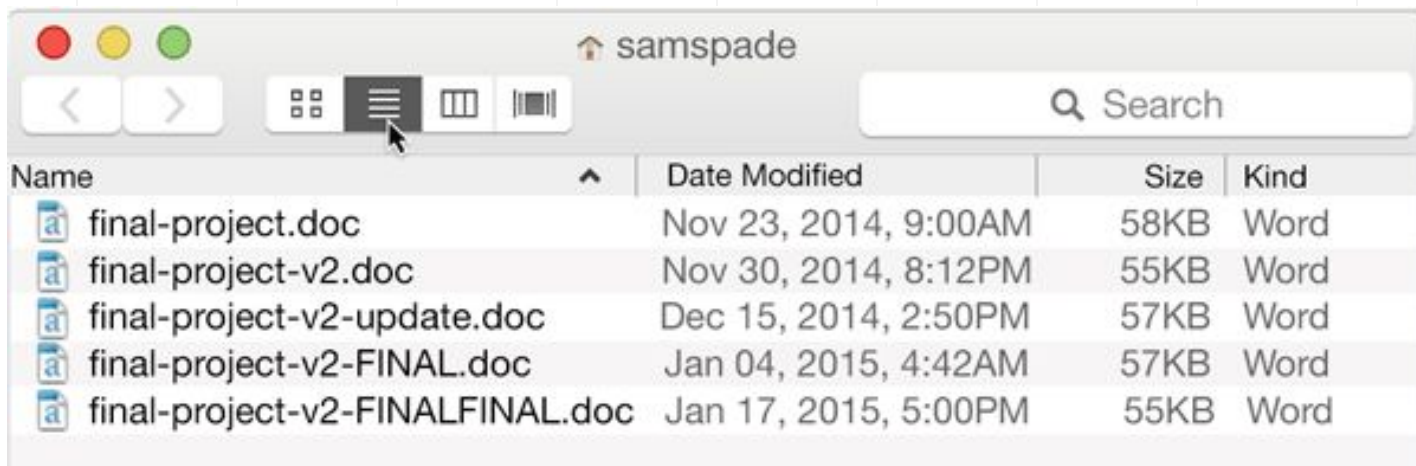


What is Git?

Background on Version Control

1

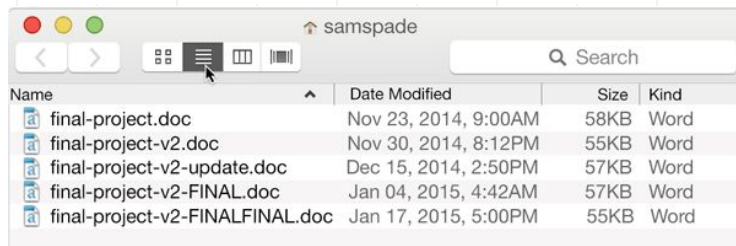
PROBLEM



Source: https://fundamentals.generalassemb.ly/02_unit/version-control-and-git.html

WHAT'S THE INTENT?

- Keep version of work that is satisfactory
- Keep a copy before sending to collaborators
- Reference copies of work at a specific point in time

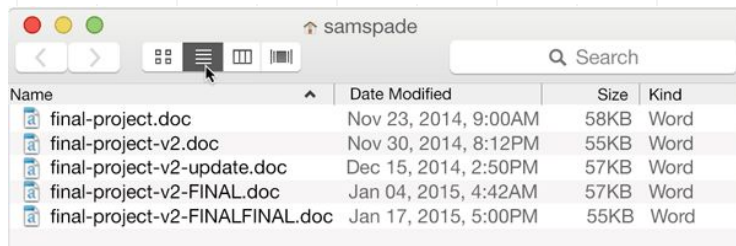


Name	Date Modified	Size	Kind
final-project.doc	Nov 23, 2014, 9:00AM	58KB	Word
final-project-v2.doc	Nov 30, 2014, 8:12PM	55KB	Word
final-project-v2-update.doc	Dec 15, 2014, 2:50PM	57KB	Word
final-project-v2-FINAL.doc	Jan 04, 2015, 4:42AM	57KB	Word
final-project-v2-FINALFINAL.doc	Jan 17, 2015, 5:00PM	55KB	Word

Source: https://fundamentals.generalassembly.ly/02_unit/version-control-and-git.html

WHAT'S WRONG?

- Lots of files with duplicate content
- Difficult to compare changes
- Difficult to integrate when work is done in parallel



Source: https://fundamentals.generalassembly.ly/02_unit/version-control-and-git.html

ONLINE DOCUMENT EDITORS

- Work concurrently on the same file
- Versioning
 - Compare changes
 - Revert changes
- Works well for single-file documents



CODE PROJECTS ARE COMPLICATED!

1. (Who?) wrote this?
2. (When?) was this code written?
3. (What?) is the code trying to do?
4. (Why?) was it written the way that it was written?
5. (Where and What?) are the changes that broke the app?
6. (How?) do I revert the code back to a working state?

GIT SOLVES THAT: WHEN DID WHO DO WHAT?

✓ master	origin/master	Merge...	Christopher Haar	51f6b67	Today at 3:54 PM
		split GenerateUpdateClusterConfigIn...	Adrien Zieba	bcc823e	8 Jun 2022 at 3:01 AM
		Merge pull request #1340 from mu...	muva	9e1ce16	8 Jun 2022 at 6:03 PM
		providerconfig: make sure v1 auth...	Muvaffak Onus	7d227a5	5 Jun 2022 at 4:20 PM
		Merge pull request #1234 from c...	Christopher Haar	7a66a14	8 Jun 2022 at 3:52 AM
		added default tag management t...	Cecilia Bernardi	585fc9a	7 Jun 2022 at 7:51 PM
		Merge pull request #1336 fro...	Christopher Haar	12876a9	5 Jun 2022 at 3:00 PM
		Added LogGroupNamePrefix so desc...	Petter Jacobsen	d9916f0	1 Jun 2022 at 8:55 PM
		upgrade aws sdk v2 (#1329)	Ben	c115d12	28 May 2022 at 1:37 PM
		Merge pull request #1331 from adrie...	Christopher Haar	e614638	28 May 2022 at 6:32 AM
		update trust relationship in assumeR...	Adrien Zieba	4be1b4b	28 May 2022 at 5:18 AM
		Merge pull request #1327 from ne...	Christopher Haar	5af968e	27 May 2022 at 2:45 PM
		Update various build files to represe...	Nic Cope	efe238a	27 May 2022 at 9:23 AM
		Rename crossplane/provider-aws to...	Nic Cope	aef7ee3	27 May 2022 at 9:14 AM

Source: <https://github.com/crossplane-contrib/provider-aws>

GIT SOLVES THAT: WHAT AND WHY WAS IT DONE?

```
50 pollInterval = app.Flag("poll", "Poll interval controls how often an individual
51 leaderElect
52 maxReconcil
53
54 namespace
55 enableExter
56 )
57 kingpin.MustP
58
59 zl := zap.New
60 log := loggin
61 if *debug {
62     // The cont
63     // *very* v
64     // logger w
65     ctrl.SetLogger(zl)
66 }
67
```

 **Nic Cope**, 2 years ago (January 30th, 2020 3:24 PM)

Plumb up logging and event recording implementations
<https://github.com/crossplaneio/crossplane-runtime/pull/108>

This commit updates all stack-aws controllers to use the logging and event recording functionality added in the above crossplane-runtime PR. Legacy controllers are minimally updated to avoid package scoped loggers.

Signed-off-by: Nic Cope <negz@rk0n.org>

 |  Connect to GitHub... |  Team... | ...

Loading...

Source: <https://github.com/crossplane-contrib/provider-aws>

GIT SOLVES THAT: WHAT CHANGED? HOW DO I FIX IT?

```

32  xpv1 "github.com/crossplane/crossplane-runtime/apis/common
33  xpcontroller "github.com/crossplane/crossplane-runtime/pkg
34  "github.com/crossplane/crossplane-runtime/pkg/feature"
35  "github.com/crossplane/crossplane-runtime/pkg/logging"
36  "github.com/crossplane/crossplane-runtime/pkg/ratelimiter"
37  "github.com/crossplane/crossplane-runtime/pkg/resource"
38
39  "github.com/crossplane/provider-aws/apis"
40  "github.com/crossplane/provider-aws/apis/v1alpha1"
41  "github.com/crossplane/provider-aws/pkg/controller"
42  "github.com/crossplane/provider-aws/pkg/features"
43  )
44
45  func main() {
46      var (
47          app      = kingpin.New(filepath.Base(os.Args[0])
48          debug    = app.Flag("debug", "Run with debug log
49          syncInterval = app.Flag("sync", "Sync interval contr

```

```

32  xpv1 "github.com/crossplane/crossplane-runtime/apis/common
33  xpcontroller "github.com/crossplane/crossplane-runtime/pkg
34  "github.com/crossplane/crossplane-runtime/pkg/feature"
35  "github.com/crossplane/crossplane-runtime/pkg/logging"
36  "github.com/crossplane/crossplane-runtime/pkg/ratelimiter"
37  "github.com/crossplane/crossplane-runtime/pkg/resource"
38
39+ "github.com/crossplane-contrib/provider-aws/apis"
40+ "github.com/crossplane-contrib/provider-aws/apis/v1alpha1"
41+ "github.com/crossplane-contrib/provider-aws/pkg/controller
42+ "github.com/crossplane-contrib/provider-aws/pkg/features"
43  )
44
45  func main() {
46      var (
47          app      = kingpin.New(filepath.Base(os.Args[0])
48          debug    = app.Flag("debug", "Run with debug log
49          syncInterval = app.Flag("sync", "Sync interval contr

```

Source: <https://github.com/crossplane-contrib/provider-aws>

WHAT IS GIT?

Version Control System (VCS) that addresses:

1. Versioning
2. Audit
3. Communication and Collaboration



POPULAR GIT PROVIDERS



GitHub



GitLab



Bitbucket

COMMON MISCONCEPTIONS

1. Git = GitHub
2. Git is for code only
 - What is code?
3. Git is for teamwork - individual work won't benefit from Git
 - Version backup / checkpoints
 - Diffs checking
 - Commit messages



GitHub

Student Developer Pack

<https://education.github.com/pack>



About Microsoft Azure

Access to Microsoft Azure cloud services and learning resources – no credit card required

Benefit

Free access to 25+ Microsoft Azure cloud services plus \$100 in Azure credit.



About Namecheap

Affordable registration, hosting, and domain management

Benefit

1 year domain name registration on the .me TLD.



About Canva

With Canva, anyone can create professional looking graphics and designs. Featuring thousands of templates and an easy to use editor.

Benefit

Free 12 month subscription of Canva's Pro tier.

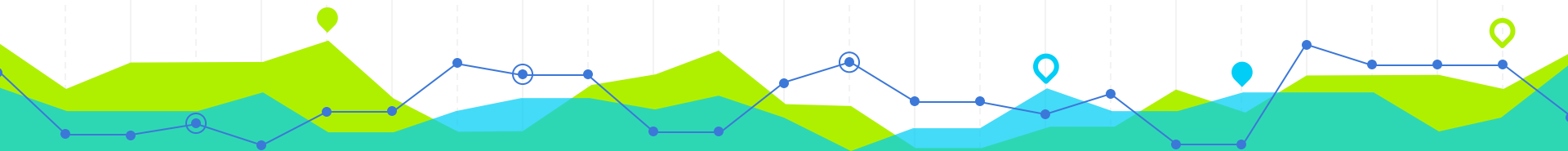
JETBRAINS

About JetBrains

Professional desktop IDEs: IntelliJ IDEA, PyCharm, and more.

Benefit

A free subscription for students, to be renewed annually.

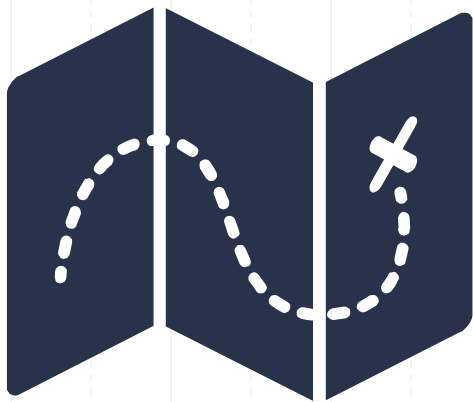


Hands-on Workshop

Guided Workshop and Advance Challenges

2

TRACKS



Guided Workshop



Advance Challenges
go.gov.sg/geekout2022-git



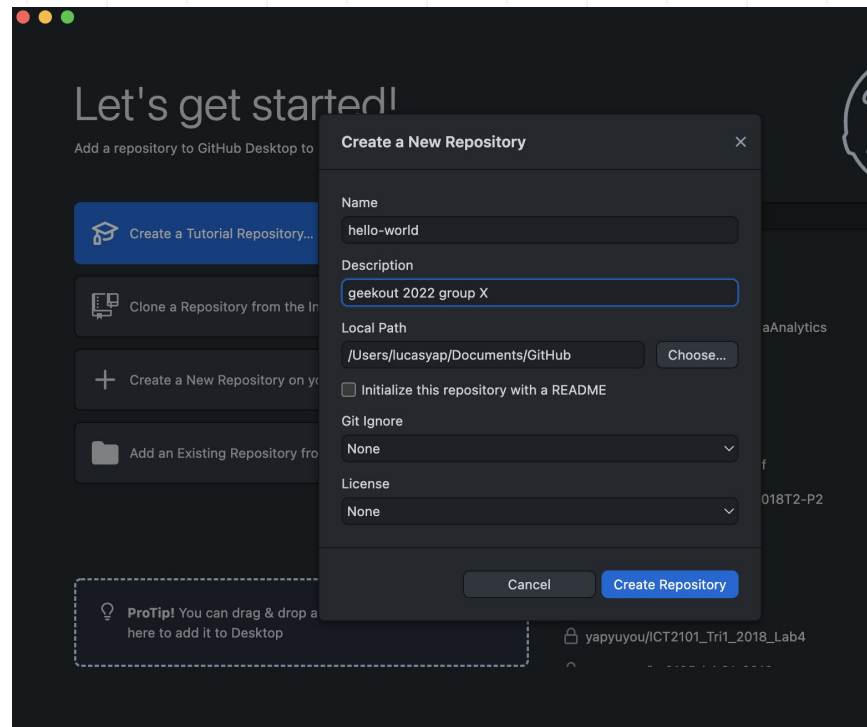
Setting Up a Git Repository

Hello World!

3

ONE person per group:

1. Click “Create a New Repository on your Hard Drive...”
2. Fill in the name and description
3. Click “Create Repository”

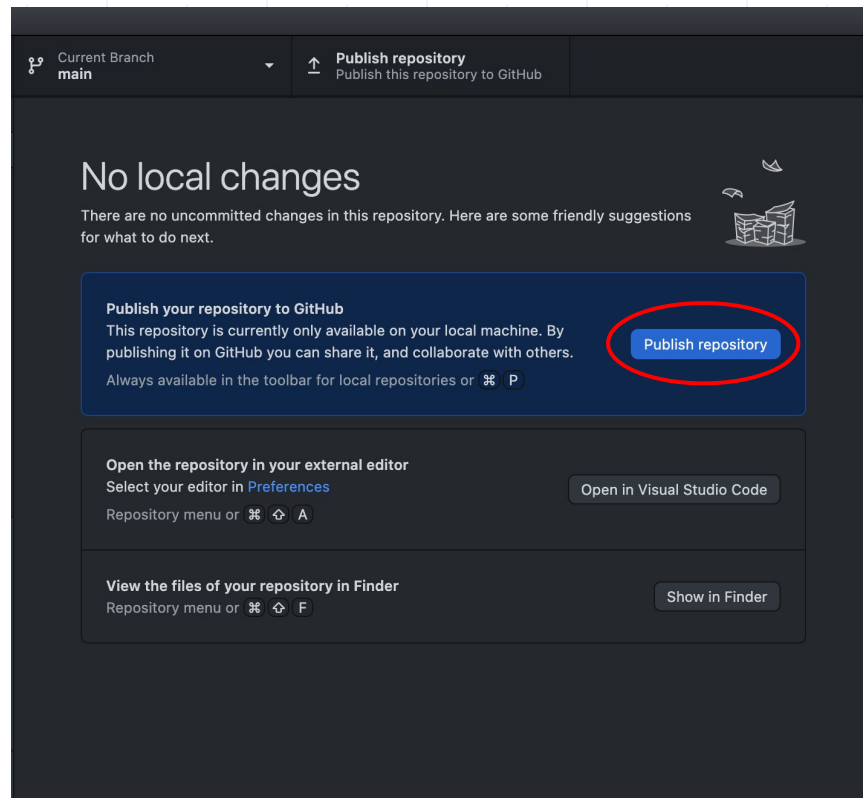


```
# Within project root directory
> git init
> git remote add <remote_name> <remote_repo_url>
> git push <remote_name>
```

ONE person per group:

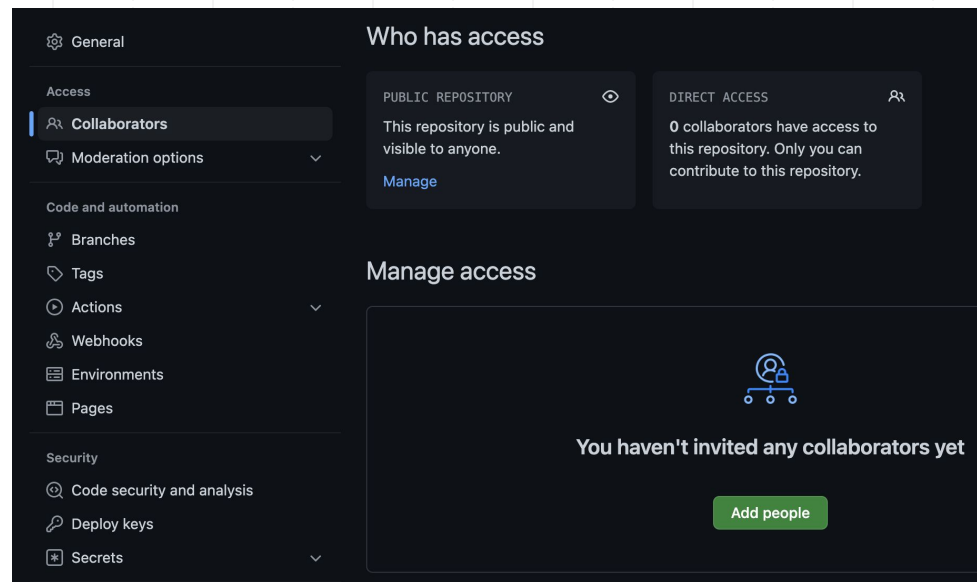
1. Click “Create a New Repository on your Hard Drive...”
2. Fill in the name and description
3. Click “Create Repository”
4. Click “Publish Repository”
5. Uncheck “Keep this code private”
6. View your repository at github.com/<username>!

```
# Within project root directory
> git init
> git remote add <remote_name> <remote_repo_url>
> git push <remote_name>
```



That same person per group:

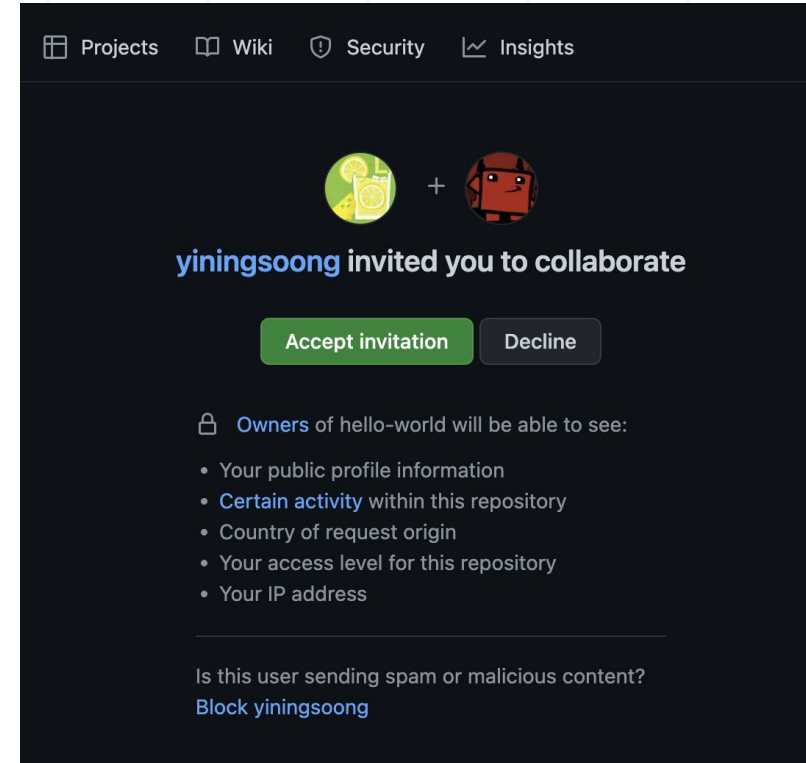
1. On your browser, go to the newly created repository and click on the “Settings” tab
2. Under “Collaborators”, click “Add people” and add your group mates via their github usernames



CLONE THE REPOSITORY

Everyone else within the group:

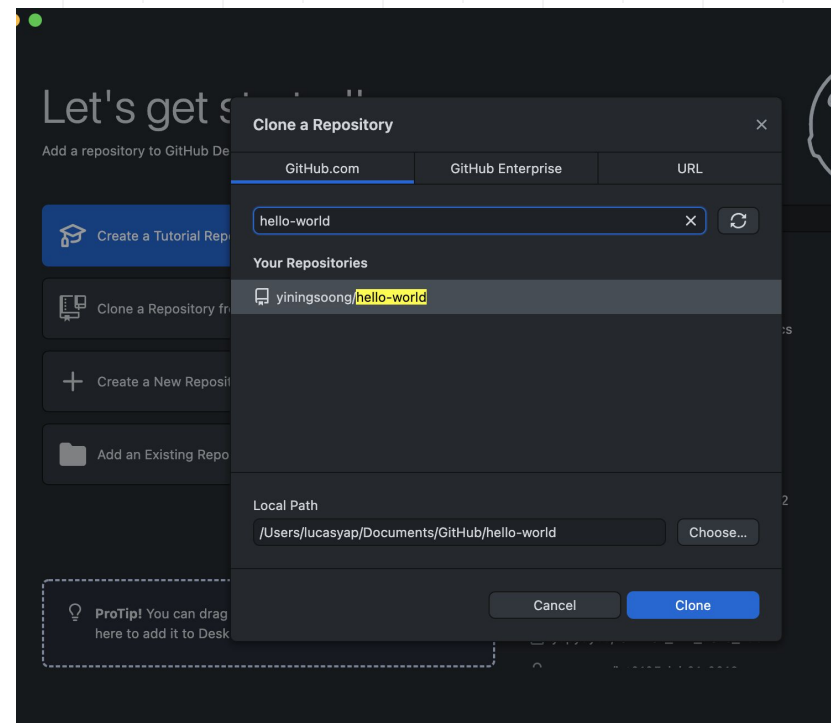
1. Either check your corresponding email
OR sign into github and visit
[github.com/<owner's
username>/<repositoryname>/invitations](https://github.com/<owner's_username>/<repositoryname>/invitations)



```
# After step 1 is completed  
> git clone <remote_repo_url>
```

Everyone else within the group:

1. Either check your corresponding email
OR sign into github and visit
github.com/<owner's_username>/<repositoryname>/invitations
2. Open GitHub Desktop and click “Clone a Repository from the Internet...”
3. Look for the newly created repository and click “clone” (you may need to refresh)



```
# After step 1 is completed  
> git clone <remote_repo_url>
```



20 MINS



Advance Challenges
go.gov.sg/geekout2022-git





Using Your Git Repository

How to fetch and make changes

4

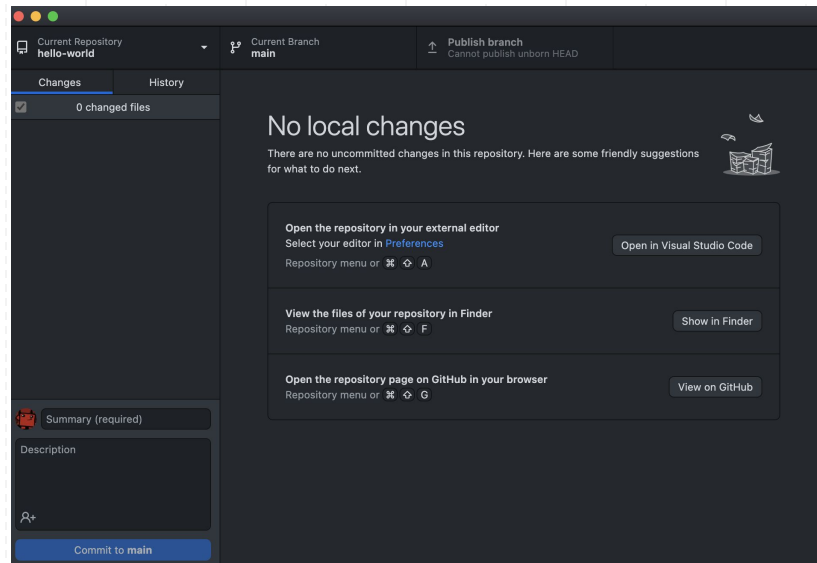
MAKING CODE CHANGES

Now you are all synced up!

But what happens if someone makes a change?

ONE person per group:

1. Create a new file called README.md with the contents “initial commit”



```
> nano README.md
# Enter contents and save
> git add README.md
> git commit -m "Create README.md"
> git push
```

PUSHING CODE CHANGES

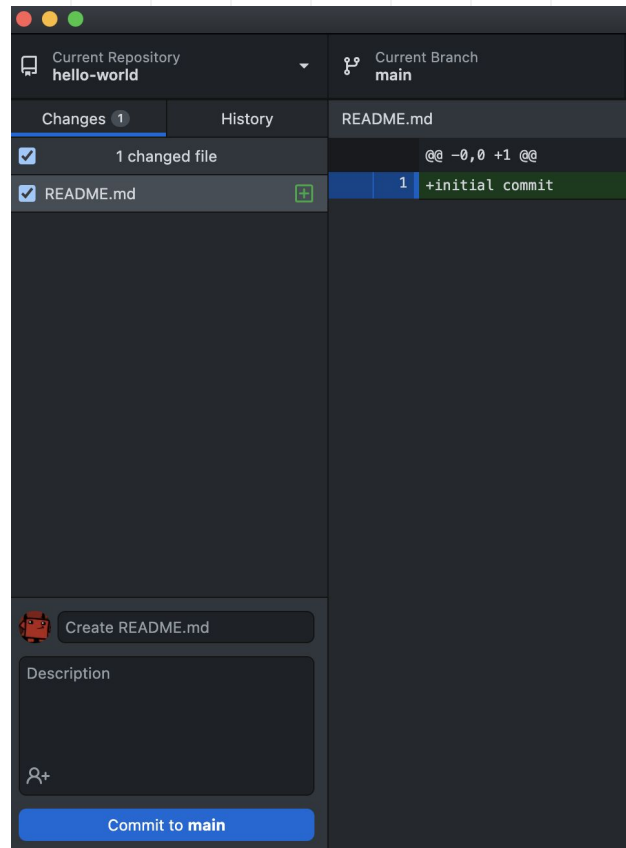
Now you are all synced up!

But what happens if someone makes a change?

ONE person per group:

1. Create a new file called README.md with the contents “initial commit”
2. Click “Commit to main” followed by “push origin”

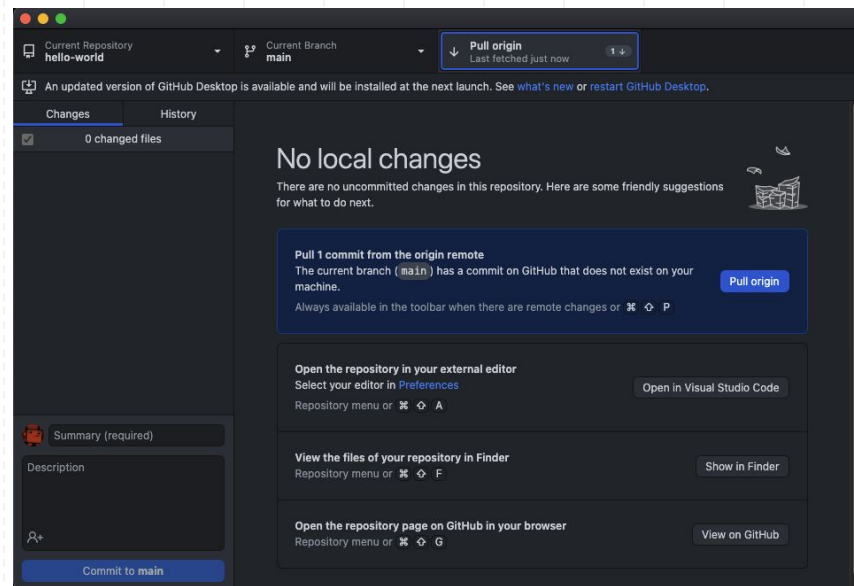
```
> nano README.md
# Enter contents and save
> git add README.md
> git commit -m "Create README.md"
> git push
```



PULLING CODE CHANGES

Everyone else within the group:

1. Click “Fetch origin” on the top right
2. You should then see that a change has been made
3. Click “Pull origin” to update your local repository with the latest change



```
> git fetch  
> git pull
```

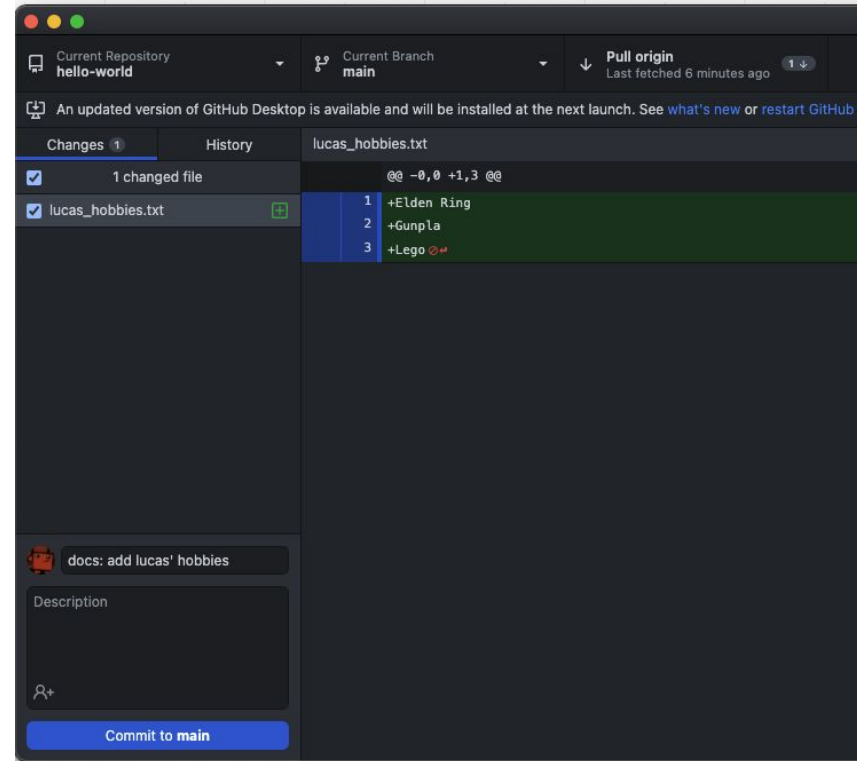
PUSHING CODE CHANGES FROM MULTIPLE TEAM MEMBERS

Now for everyone to try making changes simultaneously.

Everyone in the group:

1. Create a new file called *YOUR_NAME_hobbies.txt* and type in some of your hobbies!
2. Click “Commit to main” followed by “Push origin” (you may first have to click “Pull origin”)

```
> nano example_hobbies.txt
# Enter contents and save
> git add example_hobbies.txt
> git commit -m "docs: add example's hobbies"
> git push
```



Writing “good” commit messages:

- Have a scope e.g. feat, fix, test, docs
- Keep it short, not more than one sentence
 - ✗ change number of hobbies in file because actually I don't really cycle anymore so it might not be relevant
 - ✓ fix: change number of hobbies
- Don't be too vague
 - ✗ update README.md
 - ✓ docs: add info on gitworkshop to README.md
- Use the imperative mood (if applied, this commit will...)
 - ✗ adding one more hobby
 - ✗ added one more hobby
 - ✓ docs: add one more hobby

Useful links:

<https://www.conventionalcommits.org/en/v1.0.0/>

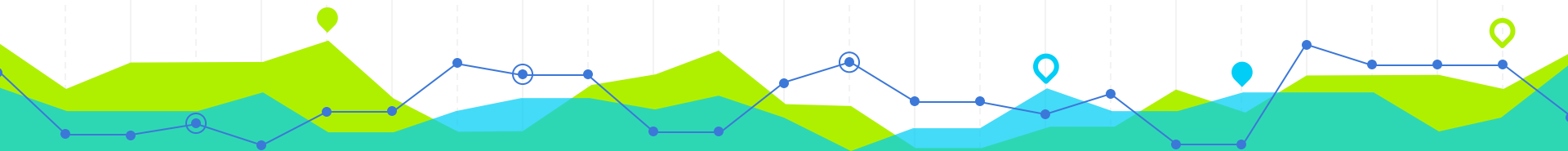
<https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages/>



Merge Conflicts Demo

What happens if we both edit the same file?

5



Making Pull Requests

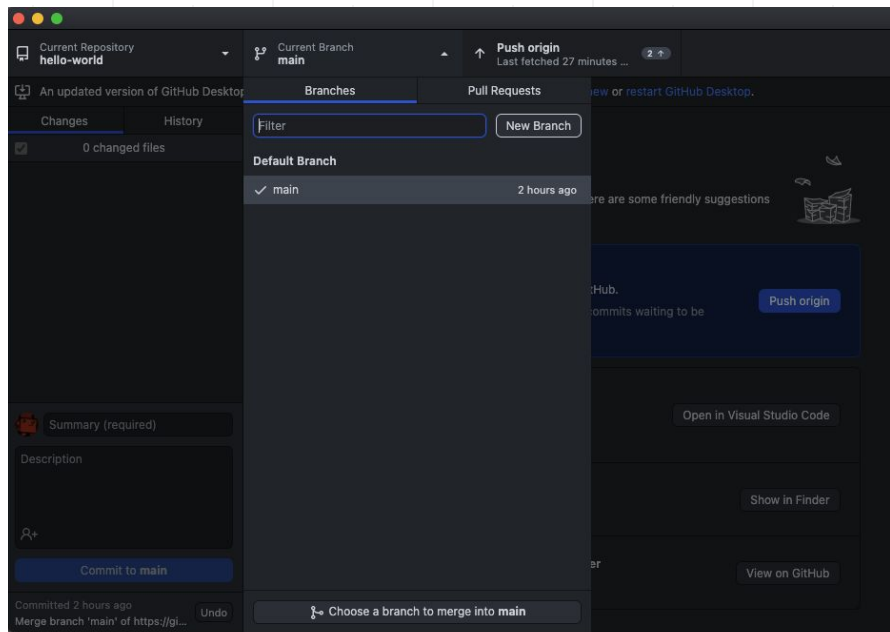
Avoid making direct commits to master/main!

6

PUSHING TO BRANCHES

Everyone in the group:

1. Click “Current Branch”, “New Branch”, enter your name as the branch name and then click “Create Branch”



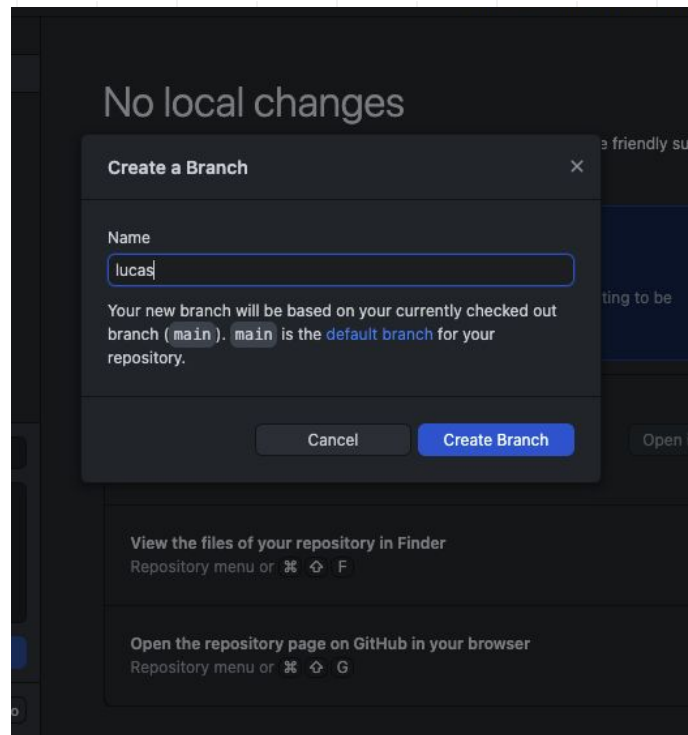
```
> git checkout -b example_branch
> nano README.md
# Edit contents and save
> git add README.md
> git commit -m "Create README.md"
> git push --set-upstream origin example_branch
```

PUSHING TO BRANCHES

Everyone in the group:

1. Click “Current Branch”, “New Branch”, enter your name as the branch name and then click “Create Branch”

```
> git checkout -b example_branch  
> nano README.md  
# Edit contents and save  
> git add README.md  
> git commit -m "Create README.md"  
> git push --set-upstream origin example_branch
```



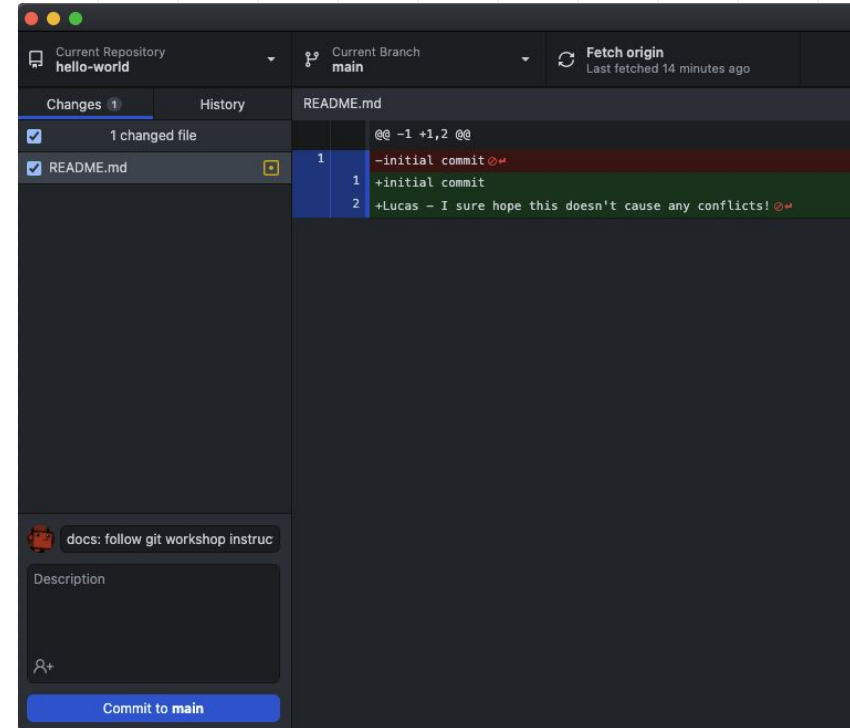
EXPERIENCING MERGE CONFLICTS

Now let's all experience conflicts!

ONE person in the group:

1. Checkout the main branch
2. Open up README.md within any text editor and add any change to a new line
3. Commit and push to origin
4. Checkout your own branch

```
> nano README.md
# Edit content and save
> git add README.md
> git commit -m "Create README.md"
> git push
```

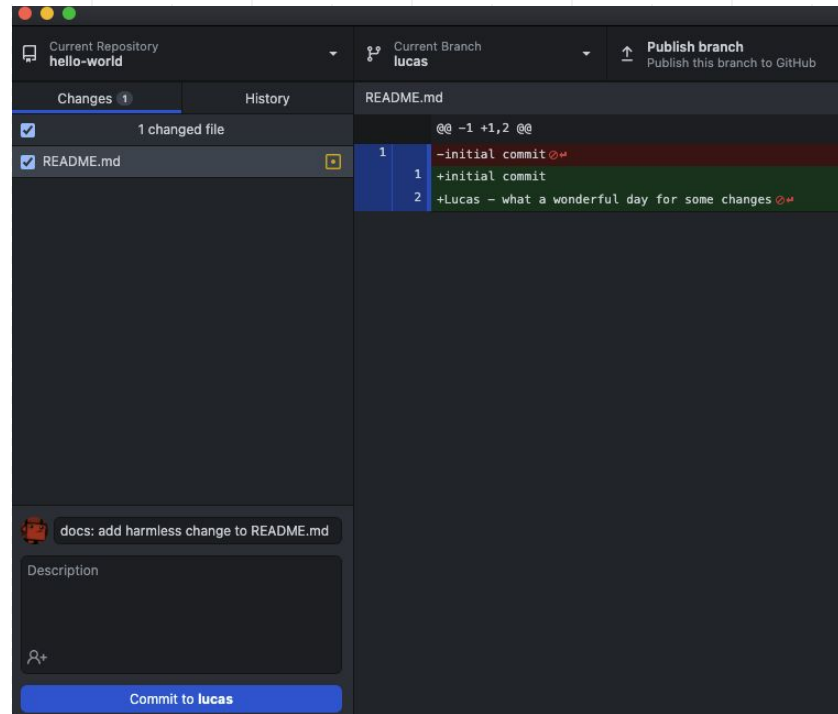


PUSHING TO BRANCHES

Everyone in the group:

1. Open up README.md within any text editor and add any change to a new line (different from the one made previously!)
2. Commit and publish the branch (try writing a good commit message!)

```
> git checkout -b example_branch
> nano README.md
# Edit contents and save
> git add README.md
> git commit -m "Create README.md"
> git push --set-upstream origin example_branch
```

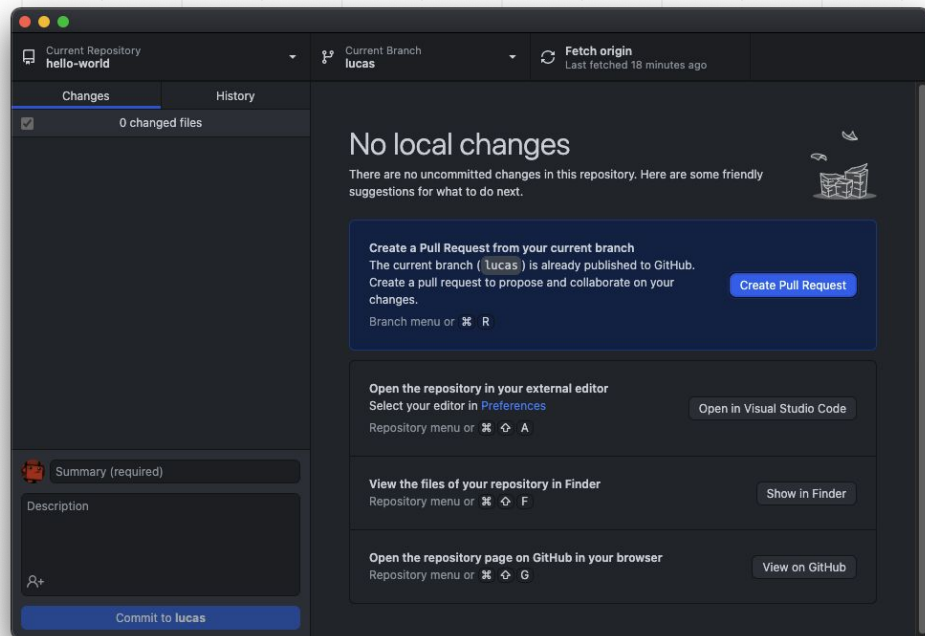


CREATING PULL REQUESTS TO MAIN BRANCH

Now let's go through the review process

ONE person in the group:

1. Click “Create Pull Request”

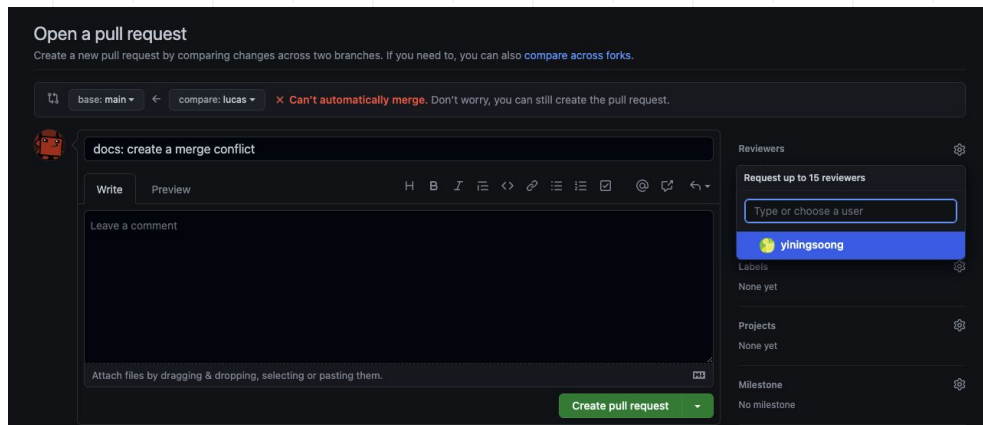


CREATING PULL REQUESTS TO MAIN BRANCH

Now let's go through the review process

ONE person in the group:

1. Click “Create Pull Request”
2. Enter a title and create your Pull Request, you may also assign a reviewer at this point

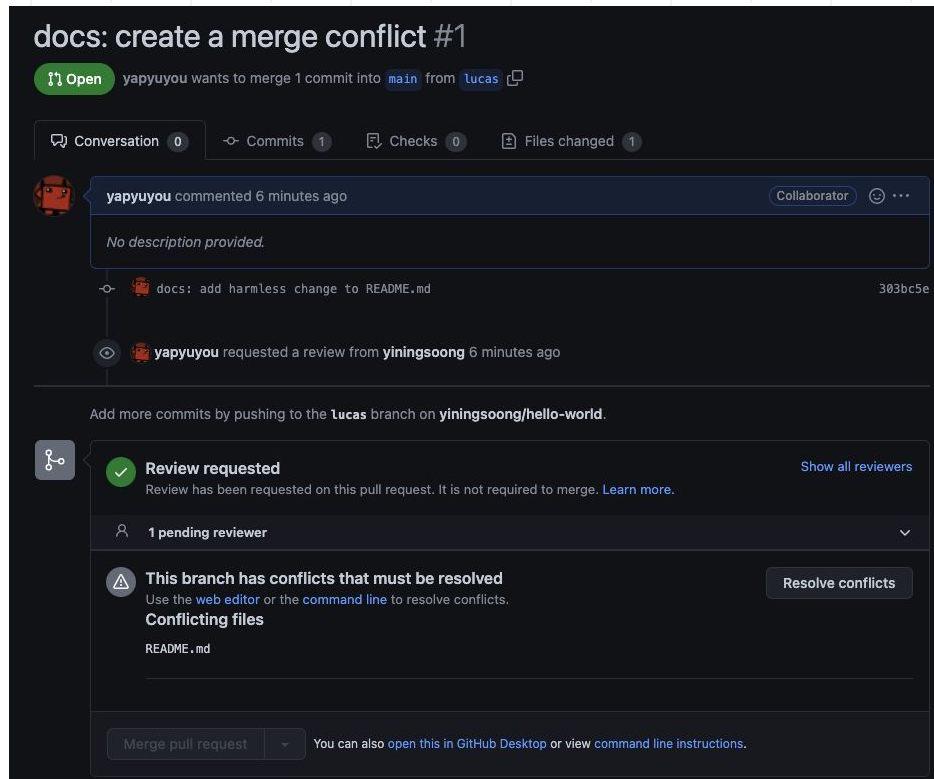


CREATING PULL REQUESTS TO MAIN BRANCH

Now let's go through the review process

ONE person in the group:

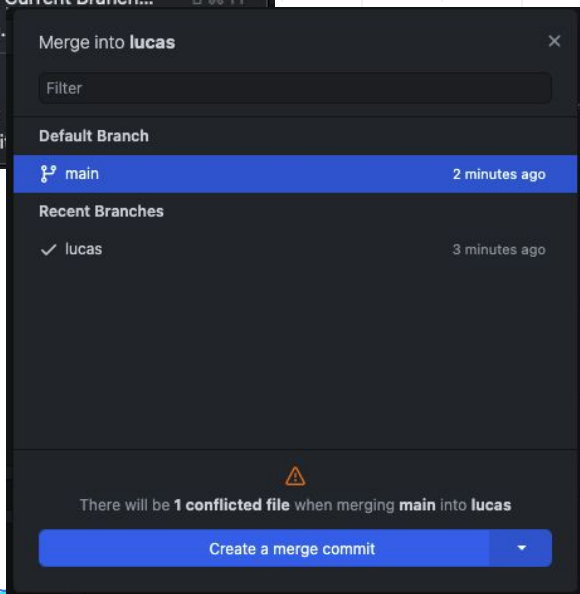
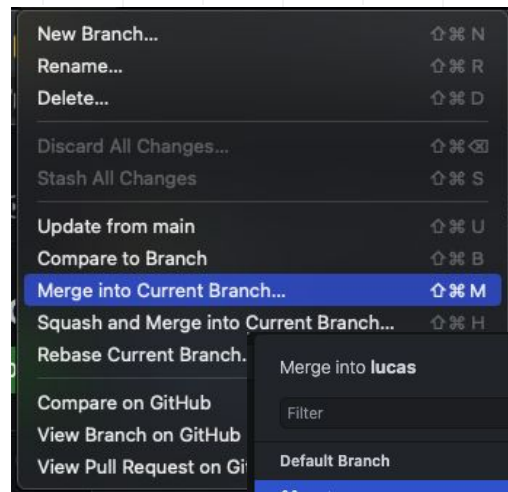
1. Click “Create Pull Request”
2. Enter a title and create your Pull Request, you may also assign a reviewer at this point



RESOLVING MERGE CONFLICTS IN PULL REQUESTS

ONE person in the group:

1. Branch -> Merge into Current Branch
2. Select the main branch

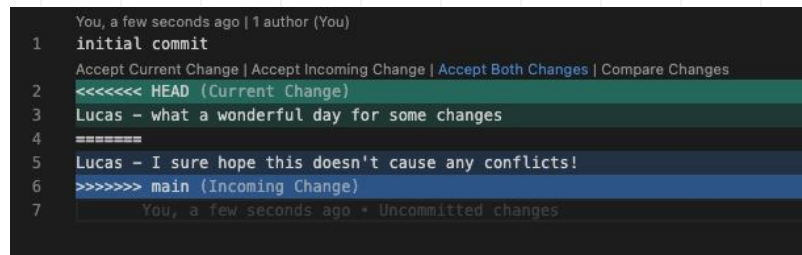
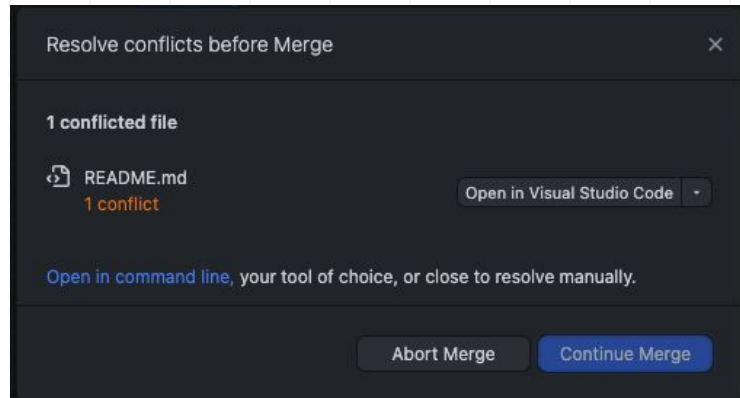


```
> git checkout main
> git pull
# Switch back to your own branch
> git checkout example_branch
> git merge main
```


RESOLVING MERGE CONFLICTS IN PULL REQUESTS

ONE person in the group:

1. Branch -> Merge into Current Branch
2. Select the main branch
3. A prompt should appear informing you of the conflict
4. Resolve the conflict in an external editor

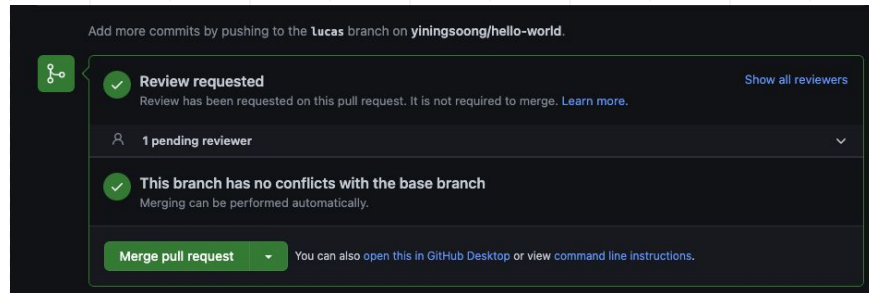
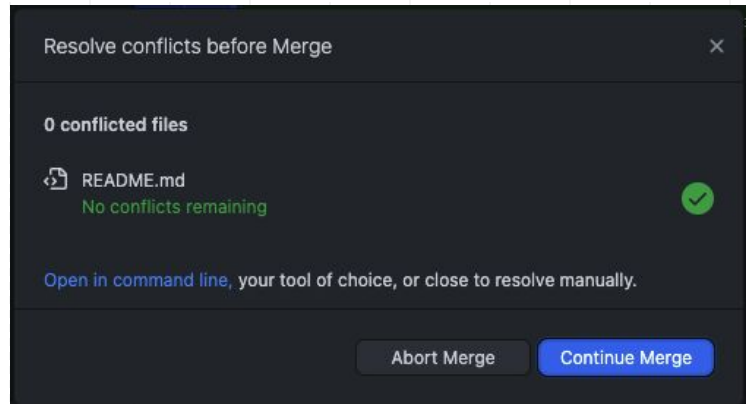


```
# You should be prompted to resolve the conflict
> nano README.md
# Edit contents and save'
> git add README.md
> git merge --continue
```

RESOLVING MERGE CONFLICTS IN PULL REQUESTS

ONE person in the group:

1. Branch -> Merge into Current Branch
2. Select the main branch
3. A prompt should appear informing you of the conflict
4. Resolve the conflict in an external editor
5. Click “Continue merge”
6. Push the branch
7. Your Pull Request should now be updated!



```
> git push  
# Check your Pull Request in your web browser
```

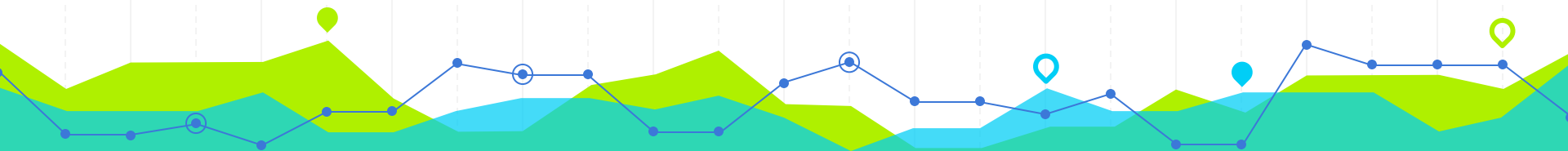
PULL REQUESTS FROM MULTIPLE TEAM MEMBERS

Everyone in the group:

1. Raise a pull request
2. Merge main into current and resolve the conflicts
3. Get another person to review and approve your changes
4. Merge your Pull Request into Main

To find out more about merging strategies -

<https://www.atlassian.com/git/tutorials/merging-vs-rebasing>



Forking vs Cloning Demo

What's the difference?

7

FORK THE ADVANCE CHALLENGES REPOSITORY



Advance Challenges
go.gov.sg/geekout2022-git

THANKS!

Any questions?



Advance Challenges

go.gov.sg/geekout2022-git