Git and GitHub

Dr. John Businge

john.businge@unlv.edu

Anouncements

- Books for the class – the books are recommended

Overview Page

Literature

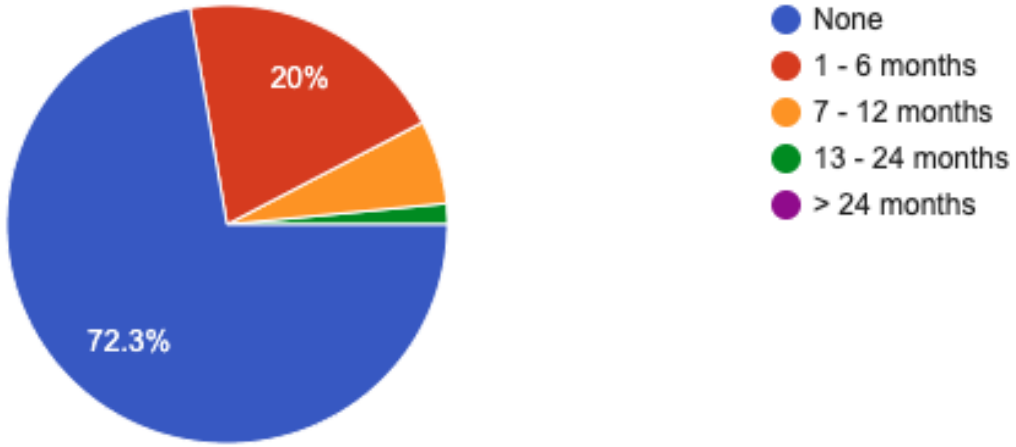
The following book is recommended as general background information to the course. They are available in the library.

1. Design Patterns: Elements of Reusable Object-Oriented Software, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
2. Design Patterns in Modern C++: Reusable Approaches for Object-Oriented Software Design
3. Object-Oriented Analysis, Design and Implementation
4. Using UML : Software Engineering with Objects and Components
5. Pro Git, available for free at: https://git-scm.com/book/en/v2
6. Dathan, B., Ramnath, S. (2015). Introduction. In: Object-Oriented Analysis, Design and Implementation. Undergraduate Topics in Computer Science. Springer, Cham. https://doi.org/10.1007/978-3-319-24280-4_1
7. Cooperative Software Development, Amy Ko, available for free: Book Site

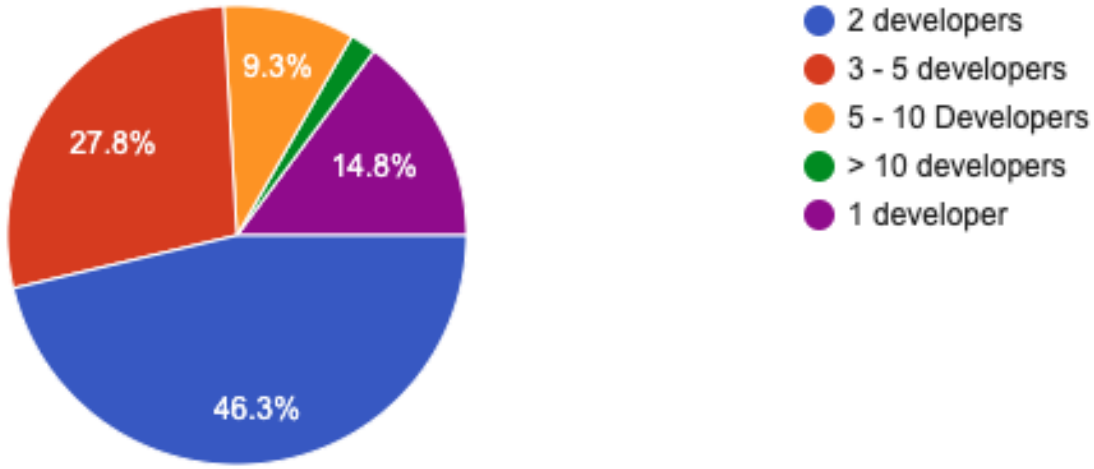
- You should be able to do Part 1 of the "**Git and GitHub Lab**"
- Begin discussing your project ideas - Precondition report due (01/31)
- Weekly meetings - 12 – 15 meetng minutes are expected (Minutes template)
- At least 50% of meetings should be physical
- Team discussions on Discord and Meeting minutes are part of the evaluation

Teams

Industry software development experience



Software development team size

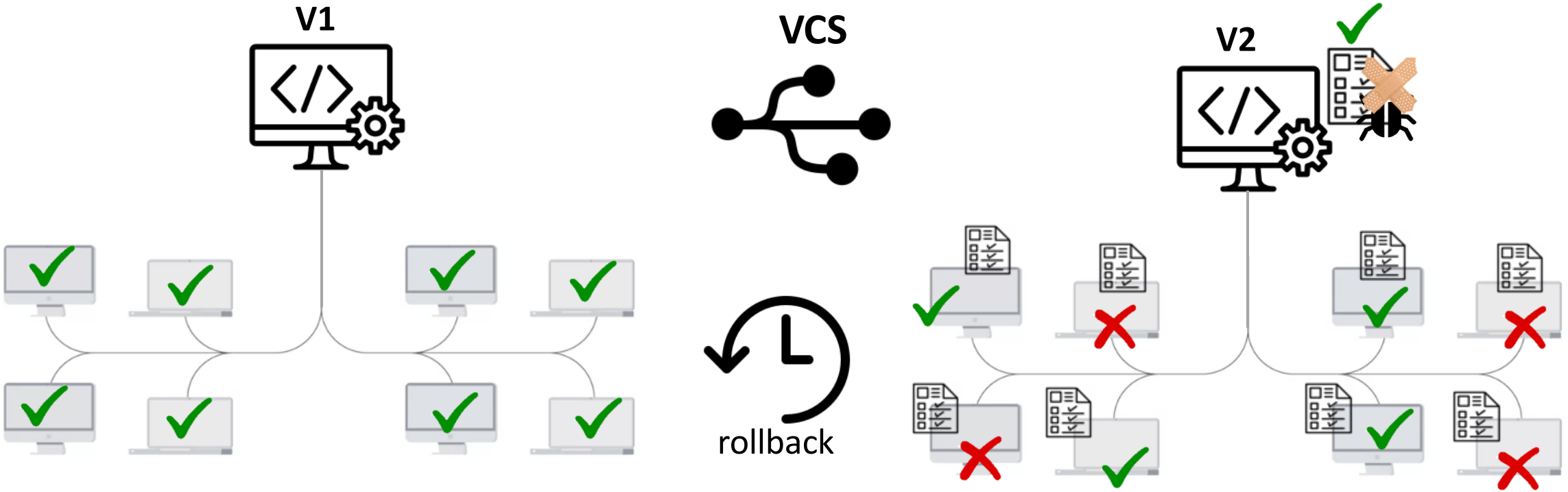


- 1 in 5 have some experience
- Great ideas shared
- Do not focus on how easy it is to develop the idea, challenge yourselves
- We are aiming at the MVP – learning process and not delivery

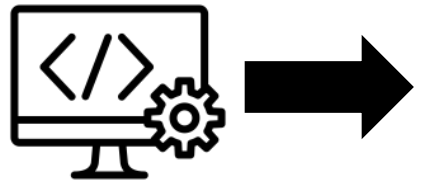
Git and GitHub

- These are tools that everyone in IT can benefit from, even if it’s not just for programming.
- The tools will allow one to roll back when mistakes happen and help teams collaborate easily.
- Nowadays, lots of employers will ask to see your GitHub portfolio when you're interviewing for IT roles.
- GitHub portfolios give companies an idea of what projects you've worked on and what kind of code you can write.

Git & GitHub - version Control



git



Software project development

Project Files

- Code
- Tests
- Documentation
- Configuration

What is Git?

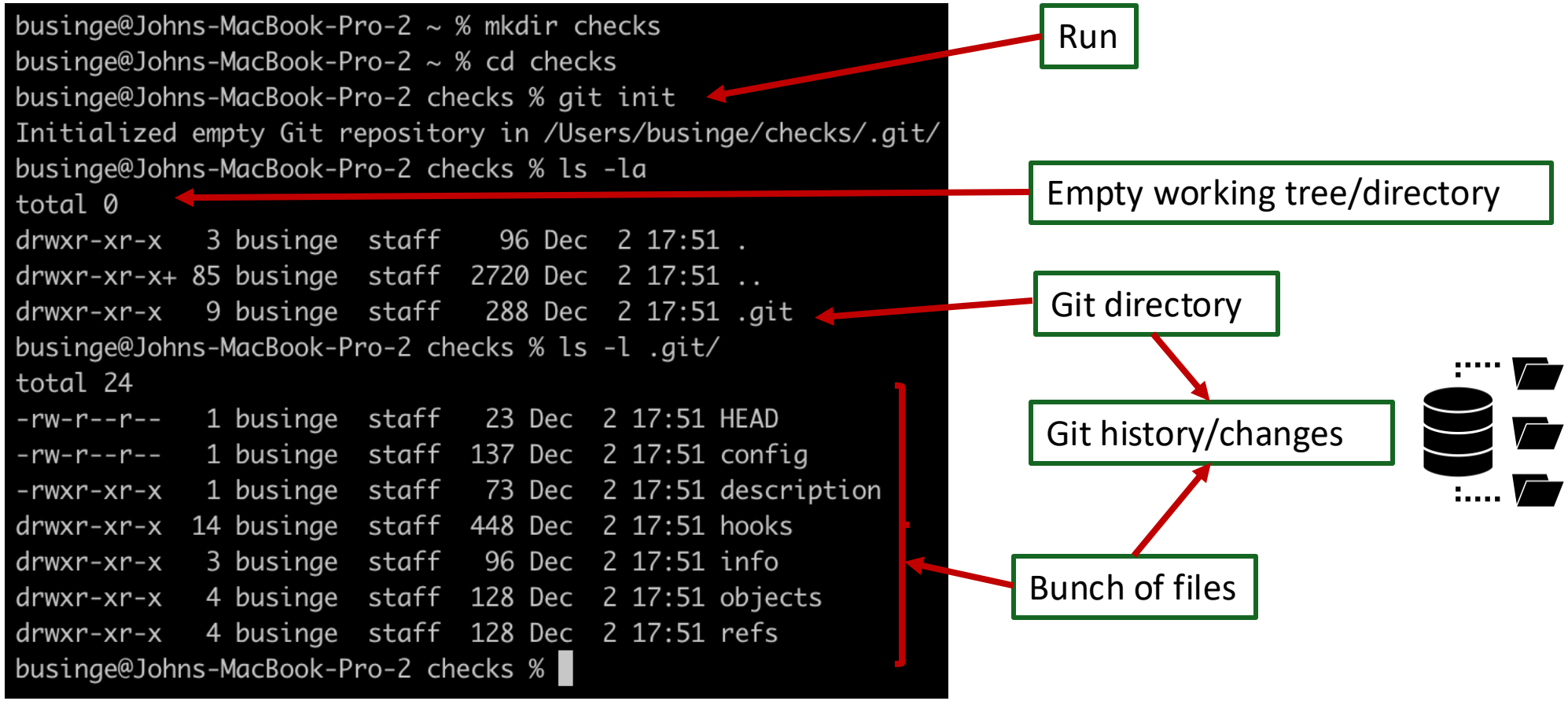
- Git is a free open-source system available for Installation on Unix-based platforms, Windows, and macOS.
- Git is now one of the most popular version control systems out there and is used in millions of projects.
- One resource I like, and I recommend digging through is this free book: http://git-scm.com/book/en/v2
- Another is Stack Overflow
- And if you are really, really stuck, the formal documentation: http://git-scm.com/documentation

Working with Git

There are two ways to start working with a git repository:

- git init - creating a repository from scratch
- git clone - a repository already exists somewhere else on a remote site

```
businge@Johns-MacBook-Pro-2 ~ % mkdir checks
businge@Johns-MacBook-Pro-2 ~ % cd checks
businge@Johns-MacBook-Pro-2 checks % git init
Initialized empty Git repository in /Users/businge/checks/.git/
businge@Johns-MacBook-Pro-2 checks % ls -la
total 0
drwxr-xr-x   3 businge  staff    96 Dec  2 17:51 .
drwxr-xr-x+ 85 businge  staff  2720 Dec  2 17:51 ..
drwxr-xr-x   9 businge  staff   288 Dec  2 17:51 .git
businge@Johns-MacBook-Pro-2 checks % ls -l .git/
total 24
-rw-r--r--   1 businge  staff   23 Dec  2 17:51 HEAD
-rw-r--r--   1 businge  staff  137 Dec  2 17:51 config
-rwxr-xr-x   1 businge  staff   73 Dec  2 17:51 description
drwxr-xr-x  14 businge  staff  448 Dec  2 17:51 hooks
drwxr-xr-x   3 businge  staff   96 Dec  2 17:51 info
drwxr-xr-x   4 businge  staff  128 Dec  2 17:51 objects
drwxr-xr-x   4 businge  staff  128 Dec  2 17:51 refs
businge@Johns-MacBook-Pro-2 checks %
```

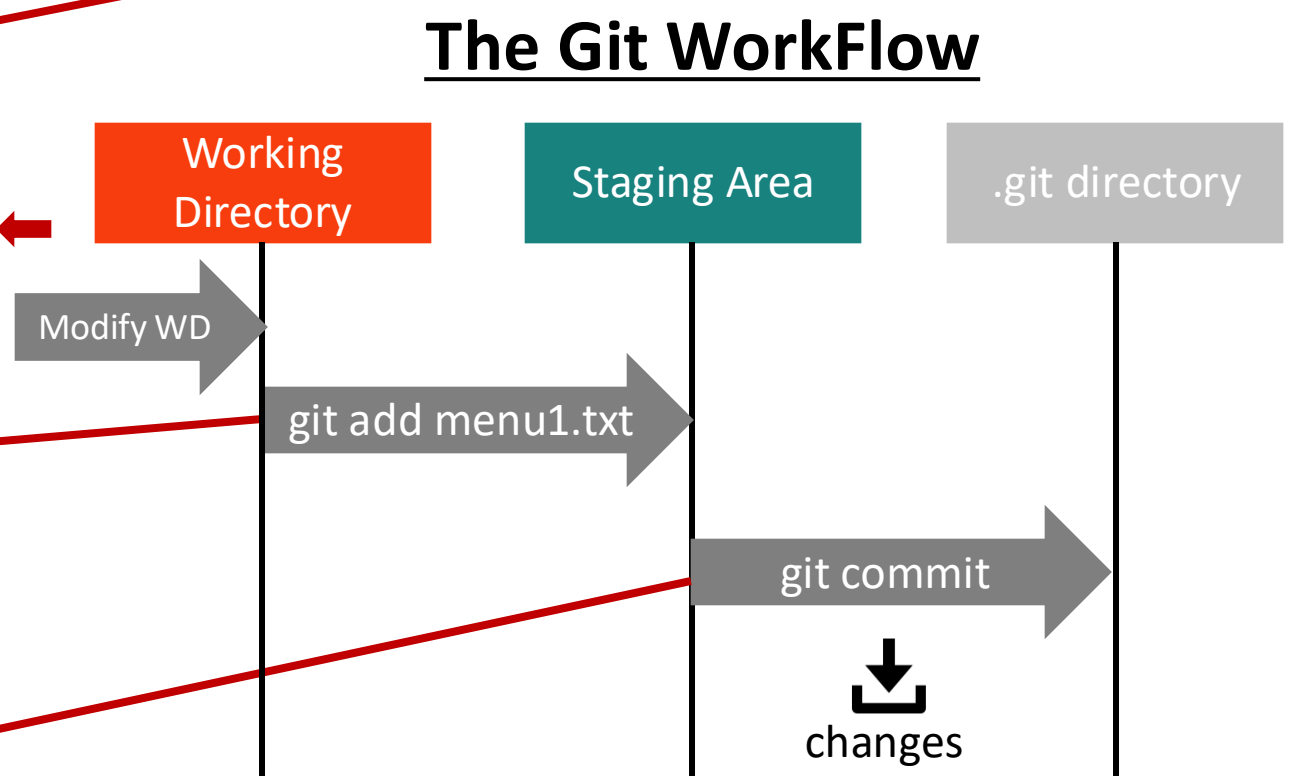


Working with Git

```
businge@Johns-MacBook-Pro-2 checks % cp ../menu1.txt .
businge@Johns-MacBook-Pro-2 checks % ls -l
total 8
-rw-r--r--  1 businge  staff  41 Dec  2 19:19 menu1.txt
businge@Johns-MacBook-Pro-2 checks % git add menu1.txt
businge@Johns-MacBook-Pro-2 checks % git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   menu1.txt
```

```
add new menu.txt
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   menu1.txt
#
```

Working directory – add menu1.txt



What makes a good commit message?

```
* 7d0fc3e typo
* 8fc509a more changes
* efe5fc5 add test
* 447c5a0 updates
* 1189cf0 update condition
* 68abca0 updates
* 29f73ed more changes
* cefaa18 add file
```

https://www.freecodecamp.org/news/a-beginners-guide-to-git-how-to-write-a-good-commit-message/

```
1  Present Tense Summary of Code Change
2
3  One or two paragraphs explaining what was tried and why
4  the changes I've made where needed.  Maybe even some
5  links to resources where more information can be found.
6  The main thing to think about is **why**.  Pretend you
7  will need to look at this in 20 years to answer that
8  question, it will help.
```

http://benfalk.com/blog/2015/10/20/commits-that-matter/

Short Commit Message
- Under 50 characters

Detailed Commit Message
- One or more paragraphs, if needed
- Lines kept under 72 characters

```
user@ubuntu:~$ cat example_commit.txt
Provide a good commit message example

The purpose of this commit is to provide an example of a hand-crafted,
artisanal commit message. The first line is a short, approximately 50 character
summary, followed by an empty line. The subsequent paragraphs are jam-packed
with descriptive information about the change, but each line is kept under 72
characters in length.

If even more information is needed to explain the change, more paragraphs can
be added after blank lines, with links to issues, tickets, or bugs. Remember
that future you will thank current you for your thoughtfulness and foresight!

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Changes to be committed:
# new file:   super_script.py
# new file:   cool_config.txt
#
```

https://initialcommit.com/blog/git-commit-messages-best-practices

Getting More Information About Our Changes

- git log
- git log – p
- git log --stat
- git show <commit-ID>

git log

```
create mode 100644 menu1.txt
businge@Johns-MacBook-Pro-2 checks % git log
commit 71518969420f647dc932ac5fd613c01b3b9142d1 (HEAD -> master)
Author: johnxu21 <johnxu21@gmail.com>
Date:   Fri Dec 2 19:26:00 2022 -0800

    add new menu.txt
```

git log -p

```
businge@Johns-MacBook-Pro-2 checks % git log -p
commit 71518969420f647dc932ac5fd613c01b3b9142d1 (HEAD -> master)
Author: johnxu21 <johnxu21@gmail.com>
Date:   Fri Dec 2 19:26:00 2022 -0800

    add new menu.txt

diff --git a/menu1.txt b/menu1.txt
new file mode 100644
index 0000000..d12a6b5
--- /dev/null
+++ b/menu1.txt
@@ -0,0 +1,6 @@
+Menu:
+
+Apples
+Bananas
+Grapes
+Strawberries
```

git log --stat

```
businge@Johns-MacBook-Pro-2 checks % git log --stat
commit 71518969420f647dc932ac5fd613c01b3b9142d1 (HEAD -> master)
Author: johnxu21 <johnxu21@gmail.com>
Date:   Fri Dec 2 19:26:00 2022 -0800

    add new menu.txt

 menu1.txt | 6 ++++++
 1 file changed, 6 insertions(+)
```

menu1.txt

Menu:

Apples
Bananas
Grapes
Strawberries

Recall that we have one commit in our git repository

Getting More Information About Our Changes

- git log
- git log – p
- git log --stat
- git show <commit-ID>

Changes
- Staged/
- committed

- git diff (changes not yet staged)

menu1.txt

Menu1:

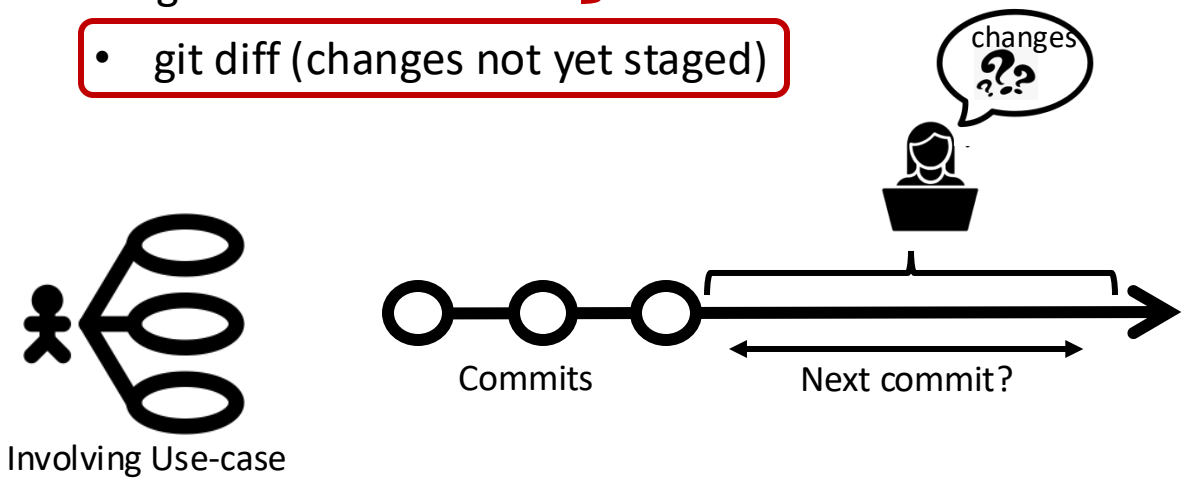
Apples
Bananas
Oranges
Pears

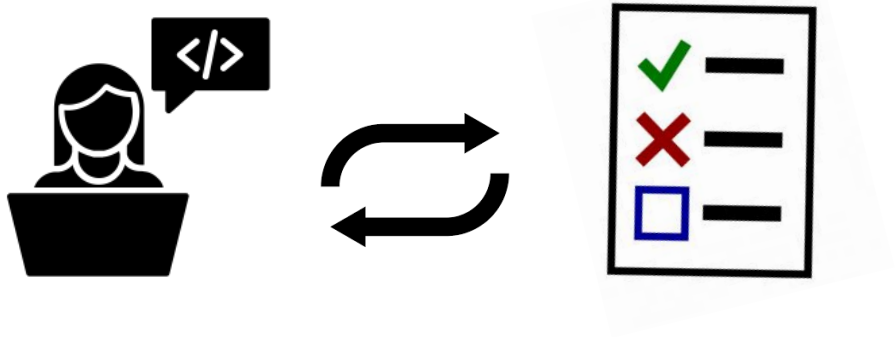
menu1.txt

Menu1:

Apples
Bananas
Oranges
Strawberry ~~Pears~~

Unstaged changes





```
businge@Johns-MacBook-Pro-2 checks % git diff
diff --git a/menu1.txt b/menu1.txt
index e7c5c1c..5a429ff 100644
--- a/menu1.txt
+++ b/menu1.txt
@@ -3,4 +3,4 @@ Menu:
 Apples
 Bananas
 Grapes
-Pears
+Strawberries
```

Deleting and Renaming Files

```
businge@Johns-MacBook-Pro-2 checks % ls -l
total 16
-rw-r--r--  1 businge  staff  42 Dec  3 15:42 menu1.txt
-rw-r--r--  1 businge  staff  41 Dec  3 16:30 menu2.txt
businge@Johns-MacBook-Pro-2 checks % git rm menu1.txt
rm 'menu1.txt'
businge@Johns-MacBook-Pro-2 checks % ls -l
total 8
-rw-r--r--  1 businge  staff  41 Dec  3 16:30 menu2.txt
businge@Johns-MacBook-Pro-2 checks % git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    menu1.txt

businge@Johns-MacBook-Pro-2 checks % git commit -m 'deleted the uneeded menu1.txt file'
[master 735008a] deleted the uneeded menu1.txt file
 1 file changed, 6 deletions(-)
 delete mode 100644 menu1.txt
```

menu1.txt

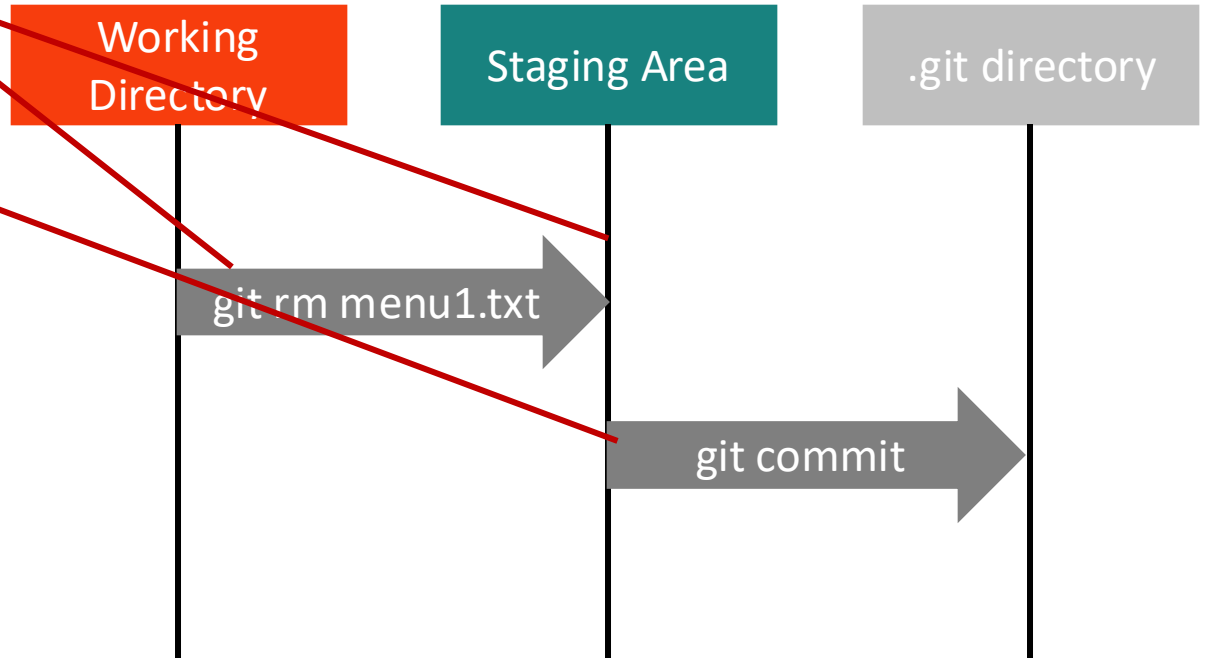
Menu1:

Apples
Bananas
Oranges
Strawberry

menu2.txt

Menu:

Apples
Bananas
Grapes
Strawberries



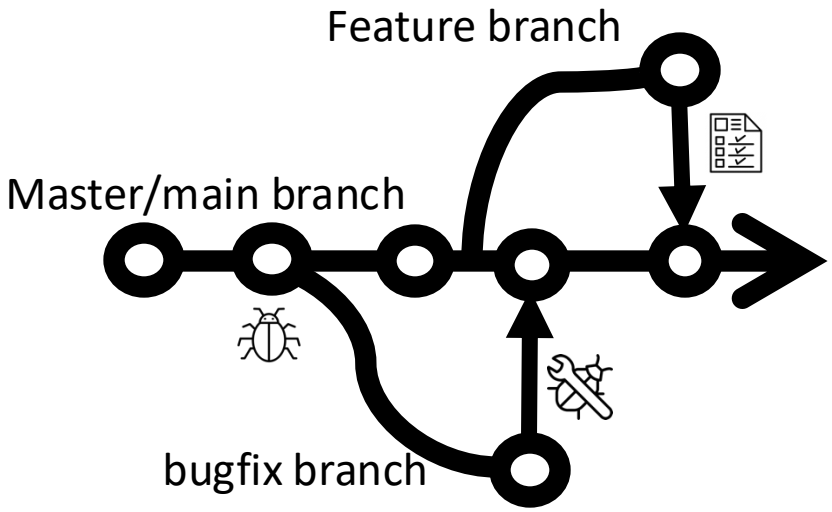
Linux
- rm
- mv

Git
- git rm
- git mv

More git commands resources

Command	Explanation & Link
git commit -a	Stages files automatically
git log -p	Produces patch text
git show	Shows various objects
git diff	Is similar to the Linux `diff` command, and can show the differences in various commits
git diff --staged	An alias to --cached, this will show all staged files compared to the named commit
git add -p	Allows a user to interactively review patches to add to the current commit
git mv	Similar to the Linux `mv` command, this moves a file
git rm	Similar to the Linux `rm` command, this deletes, or removes a file

Branching and Merging



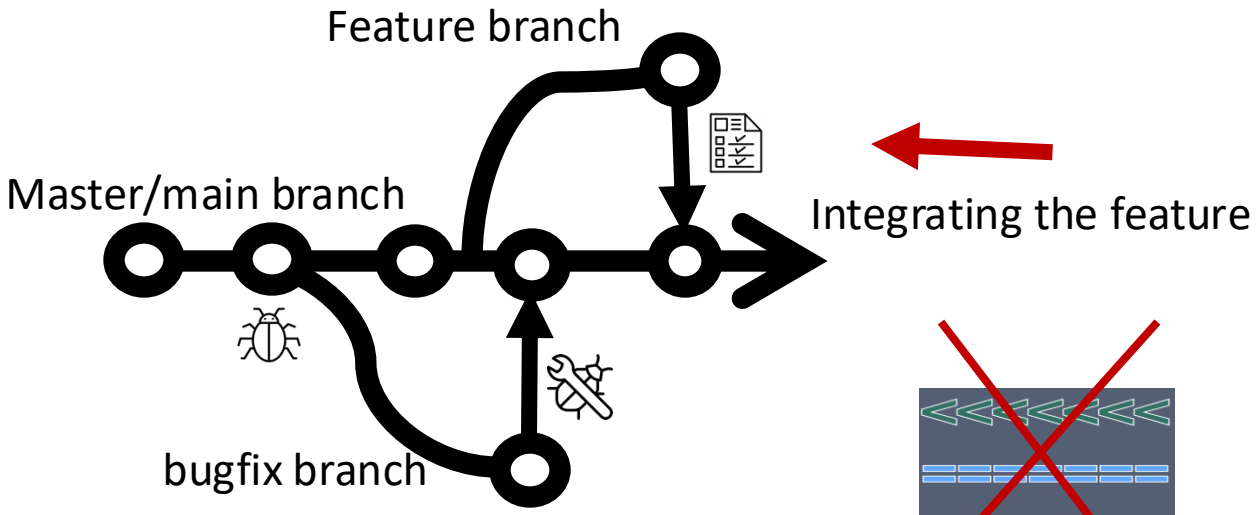

```
businge@Johns-MacBook-Pro-2 checks % git branch
* master
businge@Johns-MacBook-Pro-2 checks % git branch new-feature
businge@Johns-MacBook-Pro-2 checks % git branch
* master
  new-feature
businge@Johns-MacBook-Pro-2 checks % git checkout new-feature
Switched to branch 'new-feature'
businge@Johns-MacBook-Pro-2 checks % git branch
  master
* new-feature
businge@Johns-MacBook-Pro-2 checks % git checkout -b bug-fixing
Switched to a new branch 'bug-fixing'
businge@Johns-MacBook-Pro-2 checks % git branch
* bug-fixing
  master
  new-feature
```

```
businge@Johns-MacBook-Pro-2 checks % git checkout new-feature
Switched to branch 'new-feature'
businge@Johns-MacBook-Pro-2 checks % atom memory.py
businge@Johns-MacBook-Pro-2 checks % git add memory.py
businge@Johns-MacBook-Pro-2 checks % git commit -m 'Add an empty memory.py'
[new-feature a957a46] Add an empty memory.py
 1 file changed, 4 insertions(+)
 create mode 100644 memory.py
businge@Johns-MacBook-Pro-2 checks % git log -2
commit a957a462b5f9bb2ab8dc979f2c5eaf689f8fbfa7 (HEAD -> new-feature)
Author: johnxu21 <johnxu21@gmail.com>
Date:   Sun Dec 4 12:43:35 2022 -0800

    Add an empty memory.py

commit 2894d366f59bc7000e3736f8022faad3af003bc5 (master, bug-fixing)
Author: johnxu21 <johnxu21@gmail.com>
Date:   Sat Dec 3 19:07:02 2022 -0800
```

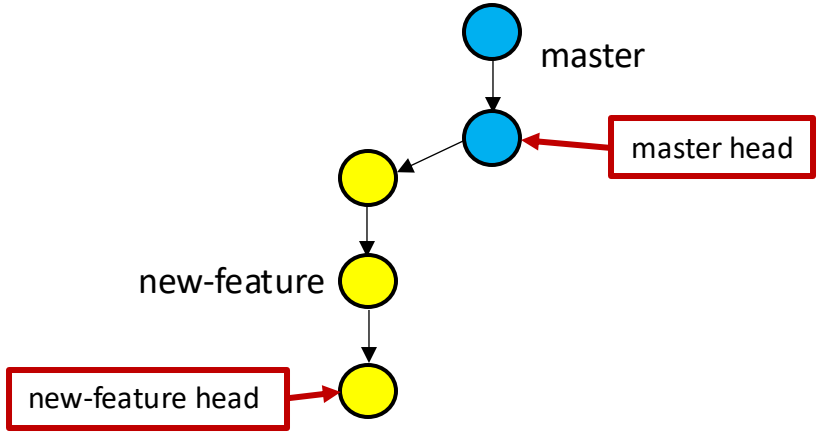
Branches are an important part of the Git workflow.
What is a branch?

- A branch represents an independent line of development in a project.
- Branching is a common practice in software development.

Branching and Merging

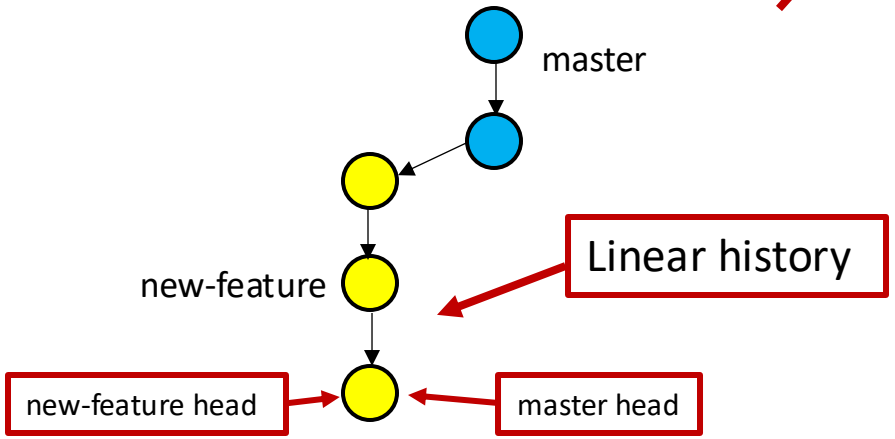


Integrating the feature



Merge Algorithms

- Fast-forward – not diverged
- Three-way merge - diverged

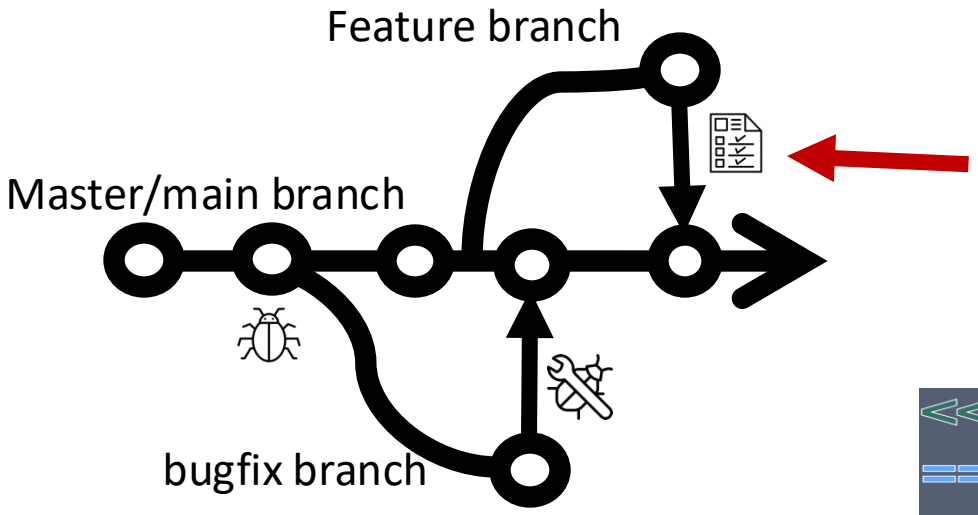


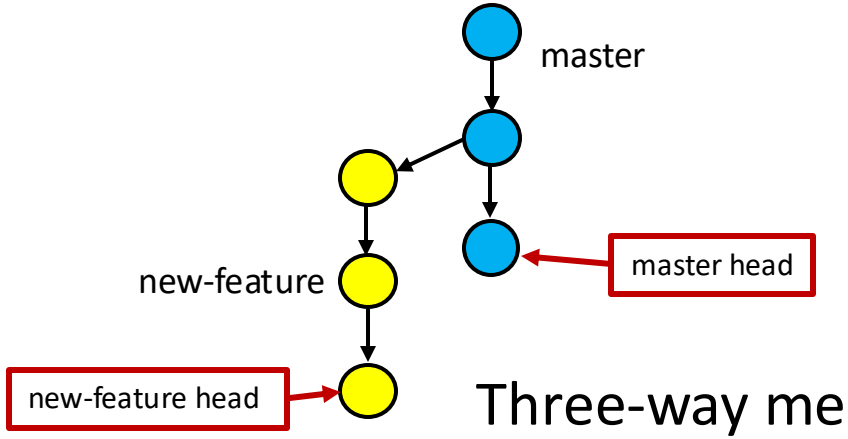
After Fast-forward merge

Fast-forward

```
businge@Johns-MacBook-Pro-2 checks % git checkout master
Switched to branch 'master'
businge@Johns-MacBook-Pro-2 checks % git merge new-feature
Updating 2894d36..a957a46
Fast-forward
 memory.py | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 memory.py
businge@Johns-MacBook-Pro-2 checks % git log
commit a957a462b5f9bb2ab8dc979f2c5eaf689f8fbfa7 (HEAD -> master, new-feature)
```

Branching and Merging

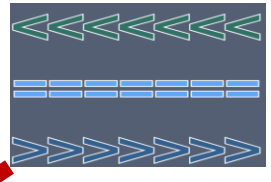


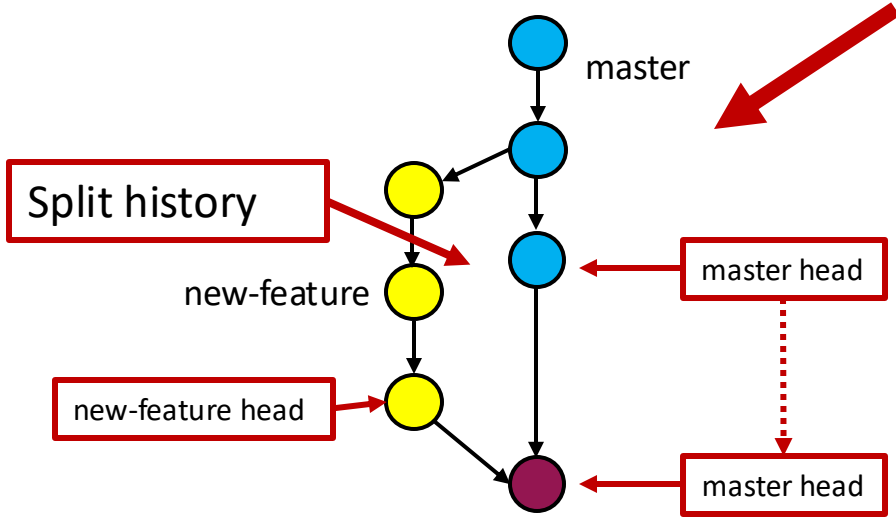


Three-way merge

Merge Algorithms

- Fast-forward – not diverged
- Three-way merge - diverged





After Three-way merge

Three-way merge

```
businge@Johns-MacBook-Pro-2 checks % git checkout master
Switched to branch 'master'
businge@Johns-MacBook-Pro-2 checks % git merge new-feature
Updating 2894d36..a957a46
 memory.py | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 memory.py
businge@Johns-MacBook-Pro-2 checks % git log --graph --oneline
*   a6ad632 (HEAD -> master) Merge branch 'new-feature'
|\
| * bf88595 (new-feature) Print everhting is OK
* | 0a92c31 Add comment to main()
|/
* a957a46 Add an empty memory.py
```

Merge Conflicts

```
businge@Johns-MacBook-Pro-2 checks % atom memory.py
businge@Johns-MacBook-Pro-2 checks % git commit -a -m 'Add comment to main()'
[master 0a92c31] Add comment to main()
 1 file changed, 1 insertion(+), 1 deletion(-)
businge@Johns-MacBook-Pro-2 checks % git checkout new-feature
Switched to branch 'new-feature'
businge@Johns-MacBook-Pro-2 checks % atom memory.py
businge@Johns-MacBook-Pro-2 checks % git commit -a -m 'Print everhting is OK'
[new-feature bf88595] Print everhting is OK
 1 file changed, 1 insertion(+), 1 deletion(-)
businge@Johns-MacBook-Pro-2 checks % git checkout master
Switched to branch 'master'
businge@Johns-MacBook-Pro-2 checks % git merge new-feature
Auto-merging memory.py
CONFLICT (content): Merge conflict in memory.py
Automatic merge failed; fix conflicts and then commit the result.
```

```
businge@Johns-MacBook-Pro-2 checks % git status
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   memory.py

no changes added to commit (use "git add" and/or "git commit -a")
```

memory.py (at master)

```
Use me ... our changes
<<<<<<< HEAD
    "Check if there is enough memory"
=======
    print("Everthing is OK!")
>>>>>>> new-feature
Use me ... their changes
```

Keep both

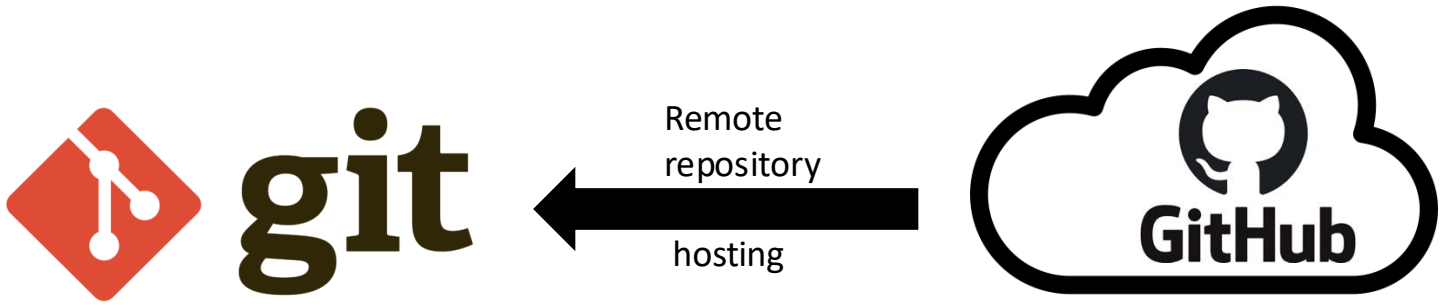
```
Use me ... our changes
    "Check if there is enough memory"
    print("Everthing is OK!")
```

```
businge@Johns-MacBook-Pro-2 checks % git add memory.py
businge@Johns-MacBook-Pro-2 checks % git status
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   memory.py

businge@Johns-MacBook-Pro-2 checks % git commit
[master a6ad632] Merge branch 'new-feature'
businge@Johns-MacBook-Pro-2 checks % git log --graph --oneline
*   a6ad632 (HEAD -> master) Merge branch 'new-feature'
|\
| * bf88595 (new-feature) Print everhting is OK
* | 0a92c31 Add comment to main()
|/
* a957a46 Add an empty memory.py
```

More Resources on Branches and Merging

Command	Explanation & Link
git branch	Used to manage branches
git branch <name>	Creates the branch
git branch -d <name>	Deletes the branch
git branch -D <name>	Forcibly deletes the branch
git checkout <branch>	Switches to a branch.
git checkout -b <branch>	Creates a new branch and switches to it.
git merge <branch>	Merge joins branches together.
git merge --abort	If there are merge conflicts (meaning files are incompatible), --abort can be used to abort the merge action.
git log --graph --oneline	This shows a summarized view of the commit history for a repo.
git rebase	You can also use git rebase branchname to change the base of the current branch to be branchname

Intro to GitHub



Bitbucket

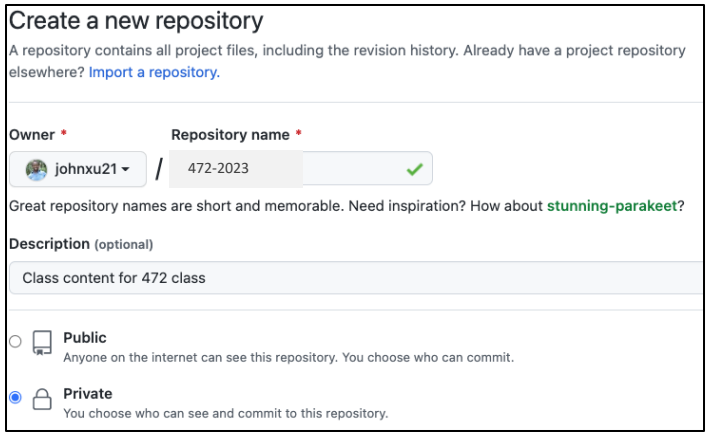
GitLab

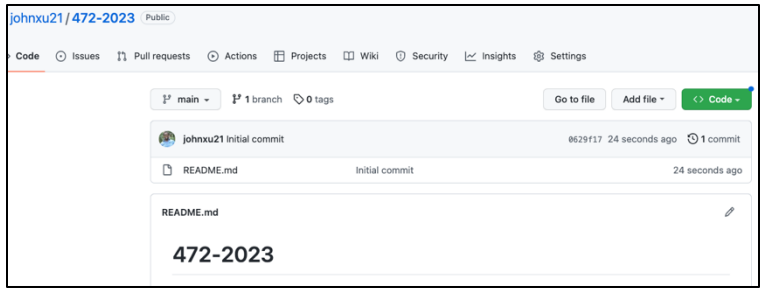
Announcements

- Discord Conversations – Group 1 please get up to speed
- Weekly group meetings on the project: Template for the minutes
- Start working on your Precondition report ASAP

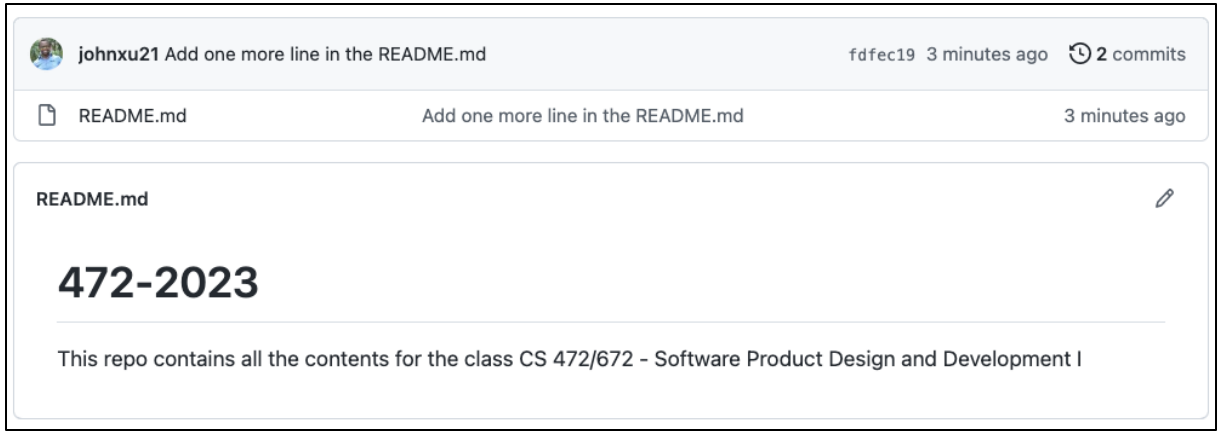
Interacting with GitHub

- git init
- git clone





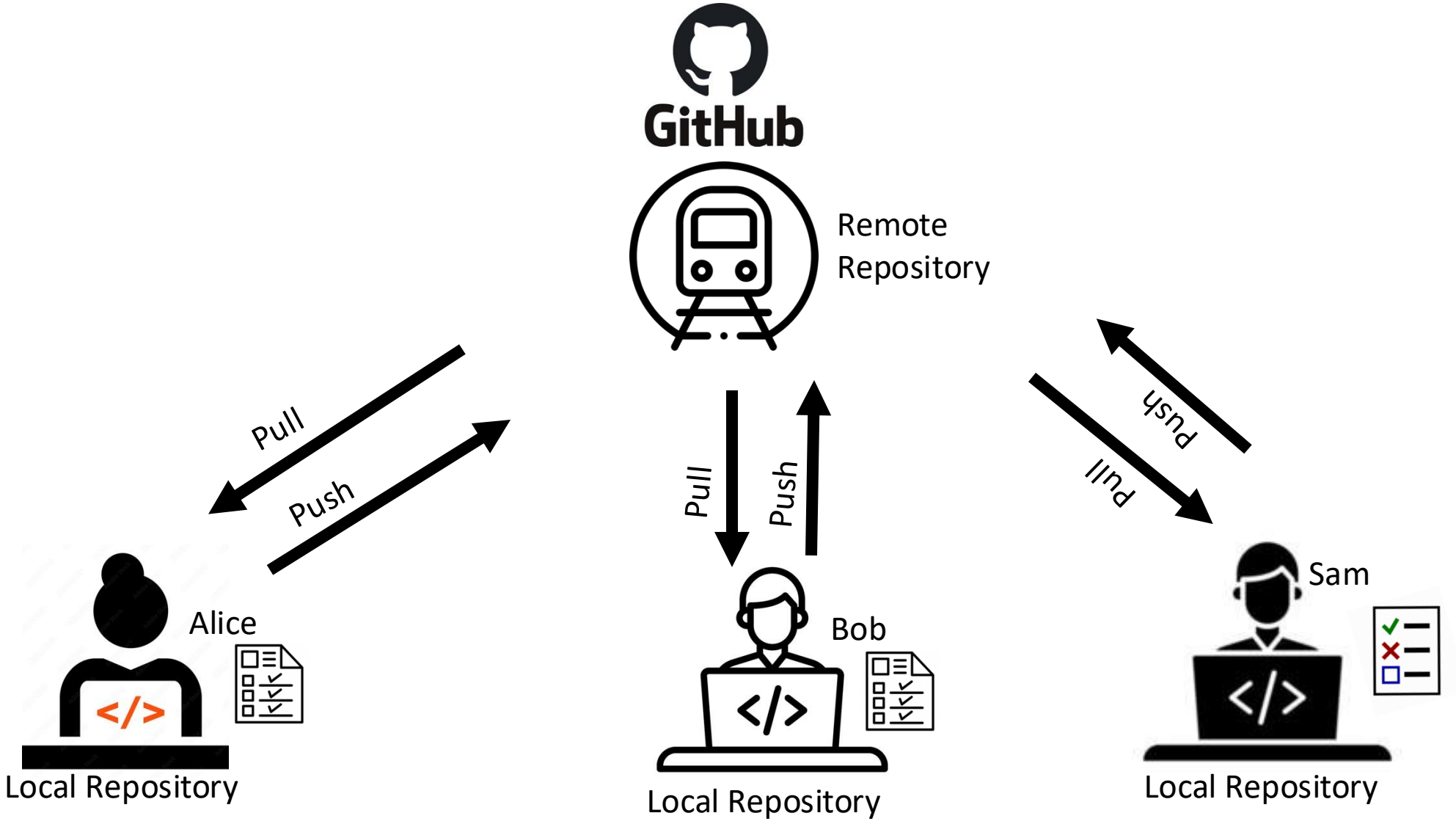

```
businge@Johns-MacBook-Pro-2 checks % git clone https://github.com/johnxu21/472-2023.git
Cloning into '472-2023'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
businge@Johns-MacBook-Pro-2 checks % cd 472-2023
businge@Johns-MacBook-Pro-2 472-2023 % ls -l
total 8
-rw-r--r--  1 businge  staff  10 Dec  5 15:44 README.md
businge@Johns-MacBook-Pro-2 472-2023 % atom README.md
businge@Johns-MacBook-Pro-2 472-2023 % git commit -a -m 'Add one more line in the README.md'
[main fdfec19] Add one more line in the README.md
 1 file changed, 3 insertions(+), 1 deletion(-)
businge@Johns-MacBook-Pro-2 472-2023 % git push
info: please complete authentication in your browser...
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), 361 bytes | 361.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/johnxu21/472-2023.git
   0629f17..fdfec19  main -> main
```



More Resources - Interacting with GitHub

Command	Explanation & Link
git clone URL	Git clone is used to clone a remote repository into a local workspace
git push	Git push is used to push commits from your local repo to a remote repo
git pull	Git pull is used to fetch the newest updates from a remote repository

Working with Remote and Fetching Changes



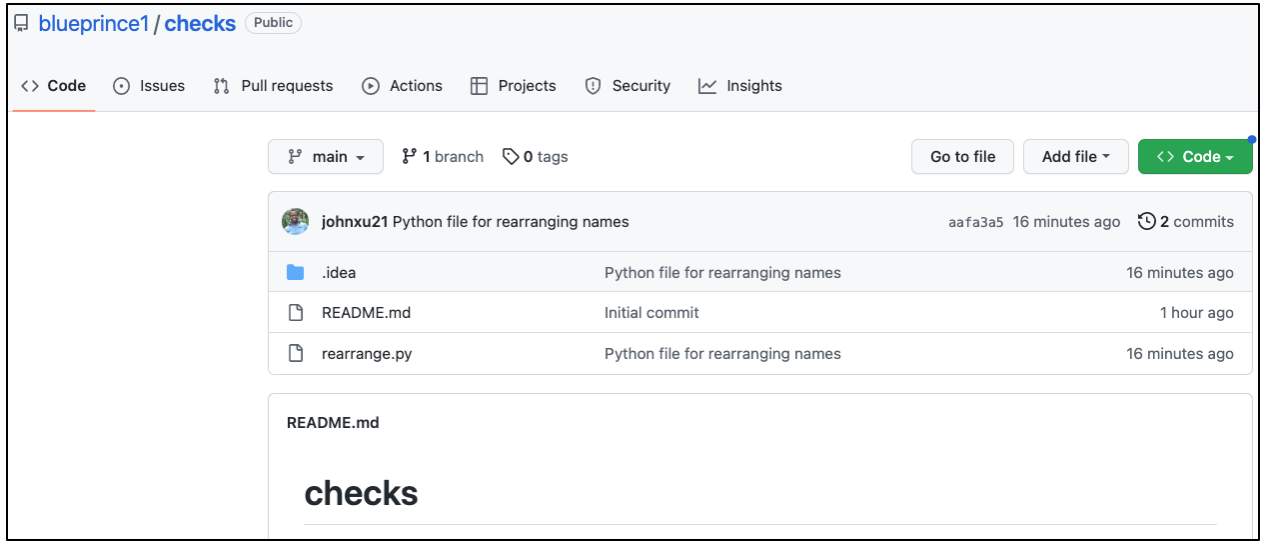
Working with Remote and Fetching Changes

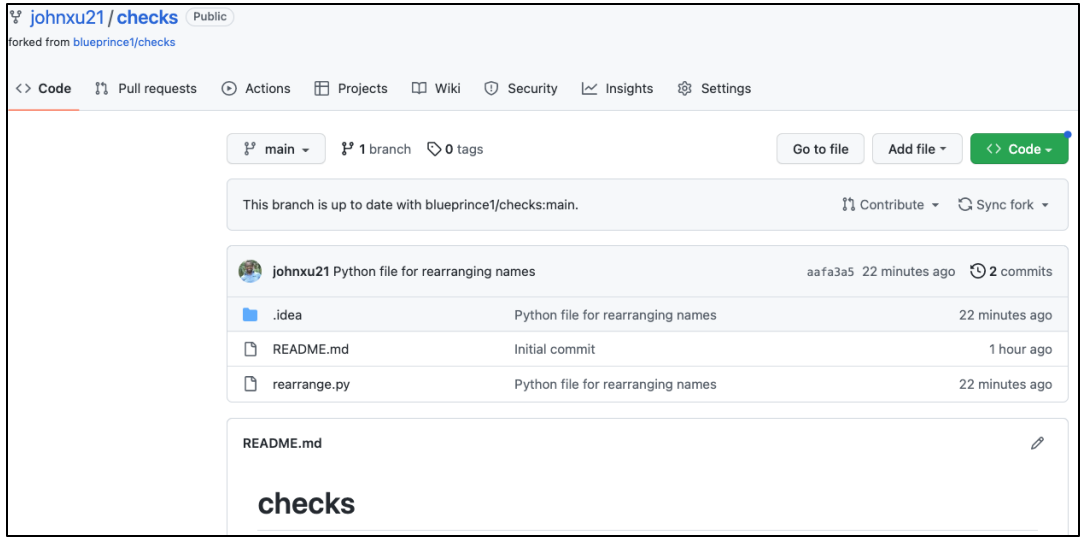
Command	Explanation & Links
git remote	Lists remote repos
git remote -v	List remote repos verbosely
git remote show <name>	Describes a single remote repo
git remote update	Fetches the most up-to-date objects
git fetch	Downloads specific objects
git branch -r	Lists remote branches; can be combined with other branch arguments to manage remote branches

Conflict Resolution with Remotes

- https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-merge-conflicts
- https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line
- The git rebase command is a lot more powerful. Check out this link for more information.

Collaboration on GitHub - Pull request – Fork Workflow





Collaboration on GitHub - Pull request – Fork Workflow

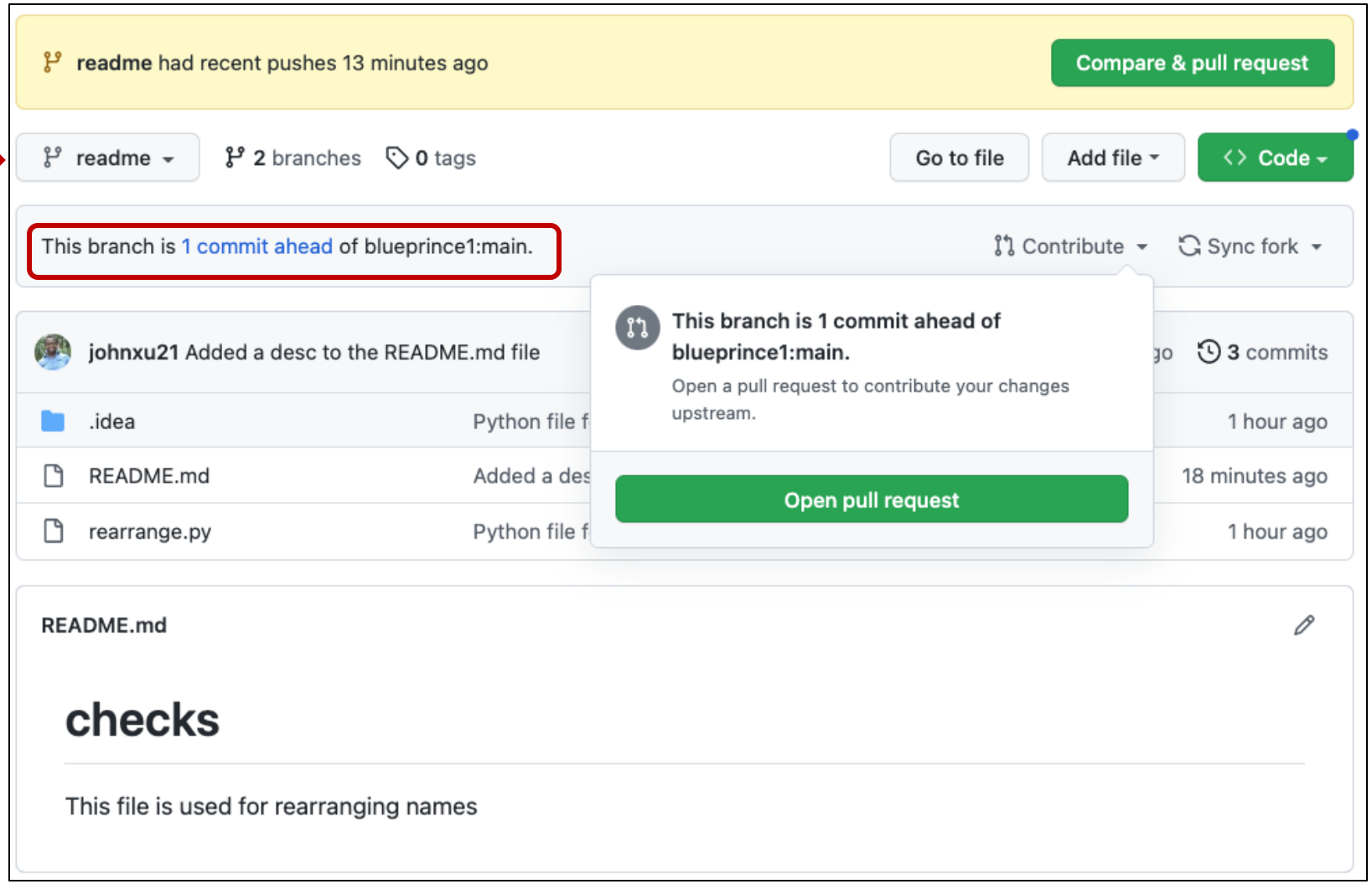
```
businge@Johns-MacBook-Pro-2 472-2023 % git clone https://github.com/johnxu21/checks.git
Cloning into 'checks'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 15 (delta 0), reused 12 (delta 0), pack-reused 0
Unpacking objects: 100% (15/15), done.
```

```
businge@Johns-MacBook-Pro-2 checks % git checkout -b readme
Switched to a new branch 'readme'
businge@Johns-MacBook-Pro-2 checks % atom README.md
businge@Johns-MacBook-Pro-2 checks % git add README.md
businge@Johns-MacBook-Pro-2 checks % git commit -m 'Added a desc to the README.md file'
[readme d7373be] Added a desc to the README.md file
 1 file changed, 3 insertions(+), 1 deletion(-)
businge@Johns-MacBook-Pro-2 checks % git push -u origin readme
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 374 bytes | 374.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'readme' on GitHub by visiting:
remote:      https://github.com/johnxu21/checks/pull/new/readme
remote:
To https://github.com/johnxu21/checks.git
 * [new branch]      readme -> readme
Branch 'readme' set up to track remote branch 'readme' from 'origin'.
```

Create local branch

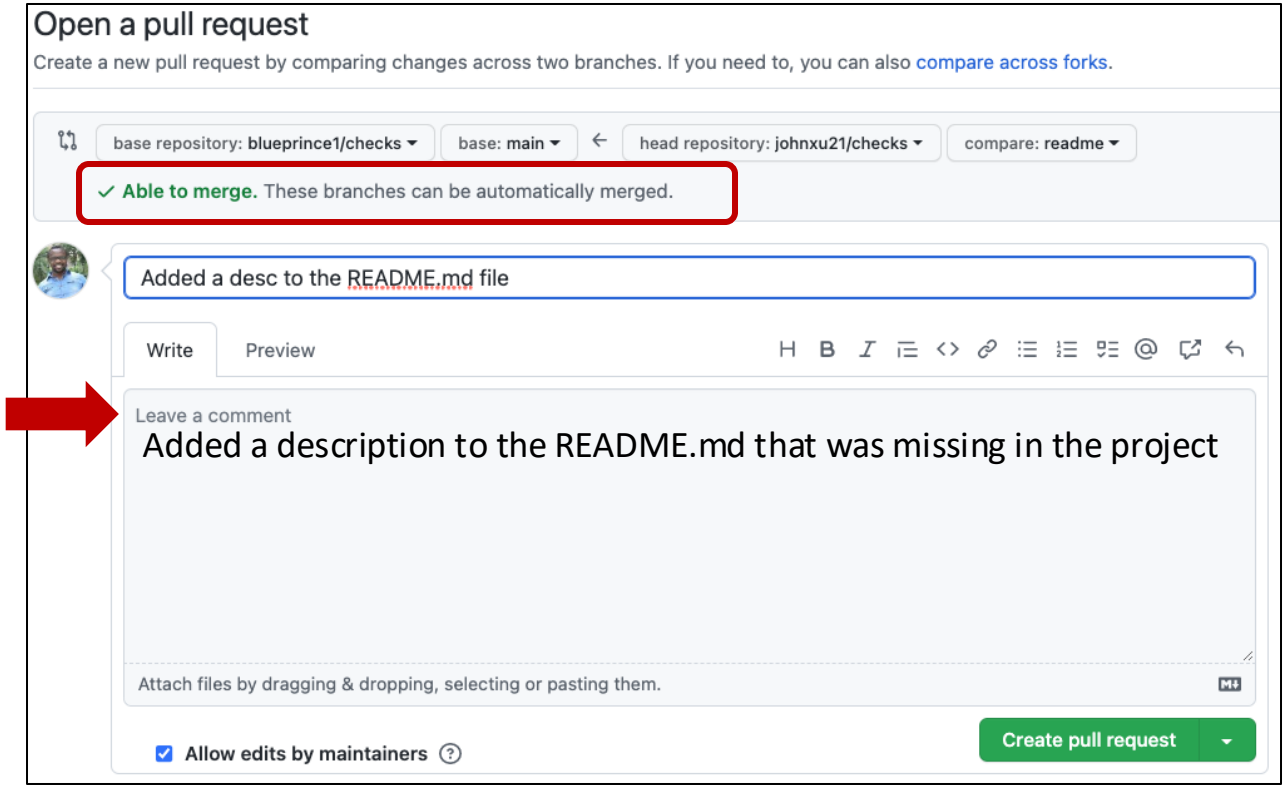
Create remote branch -> push

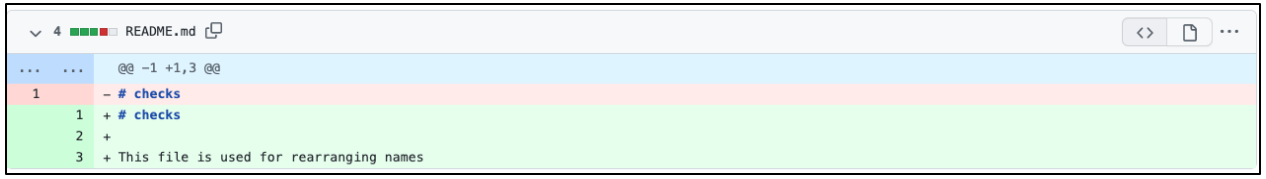
Collaboration on GitHub - Pull request – Fork Workflow

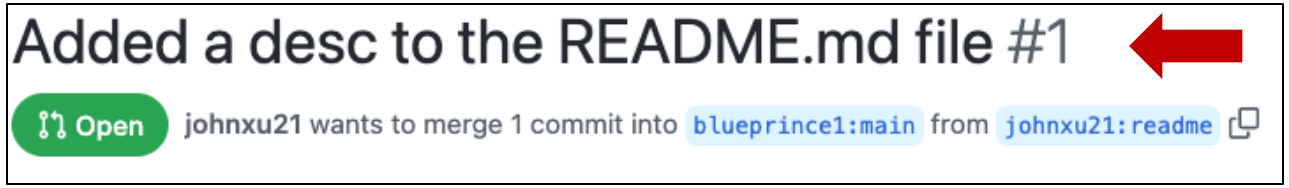


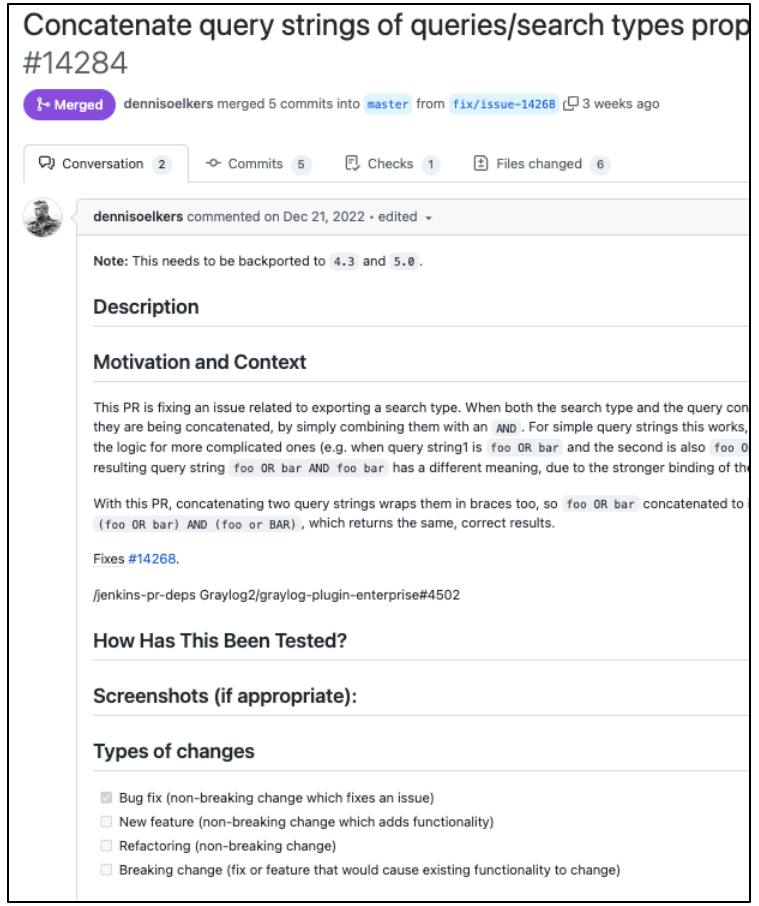
- A pull request is a commit or series of commits that you send to the owner of the repository so that they incorporate it into their tree.

Collaboration on GitHub - Pull request – Fork Workflow

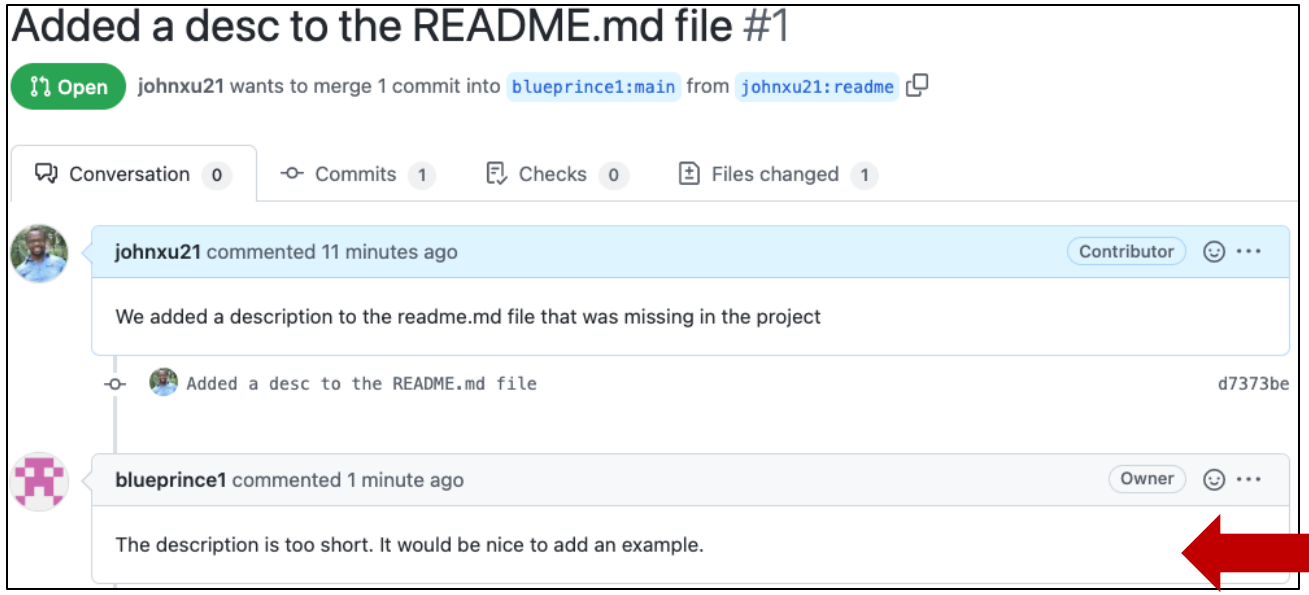






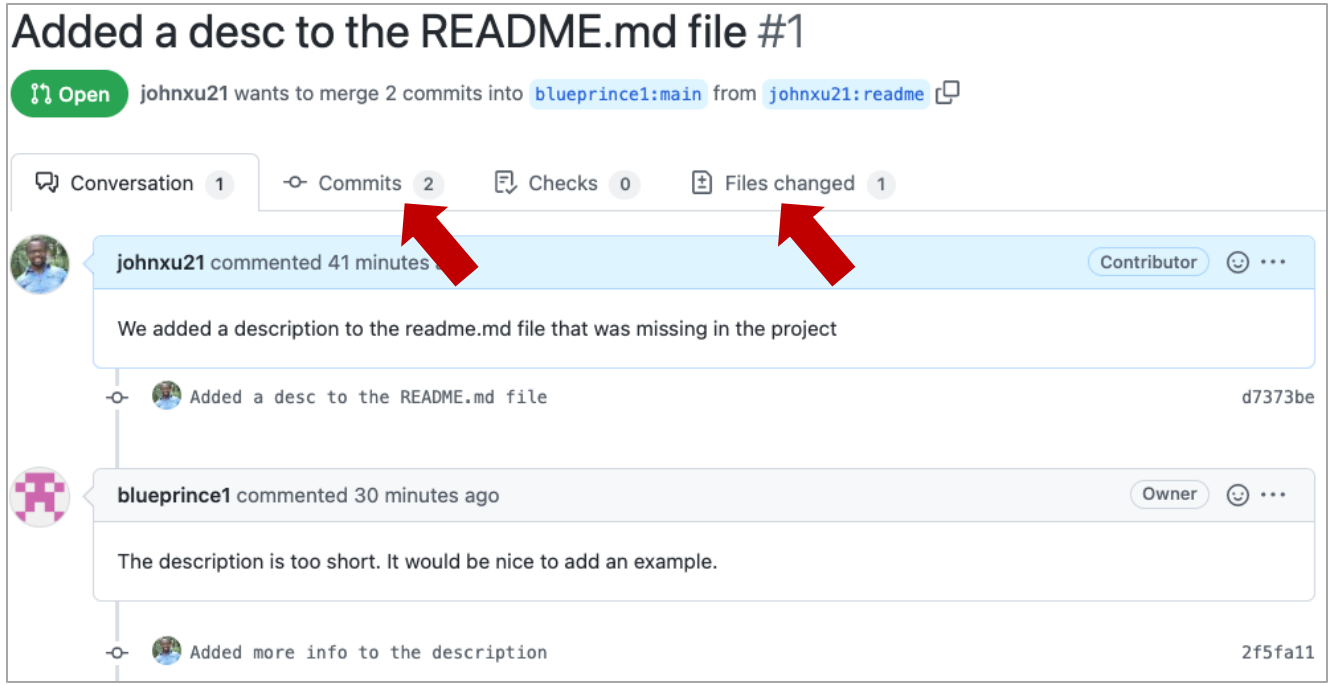


Updating an Existing Pull Request




```
businge@Johns-MacBook-Pro-2 checks % git commit -a -m 'Added more info to the description'
[readme 2f5fa11] Added more info to the description
 1 file changed, 8 insertions(+), 1 deletion(-)
businge@Johns-MacBook-Pro-2 checks % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 462 bytes | 462.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/johnxu21/checks.git
   d7373be..2f5fa11  readme -> readme
```

Updating an Existing Pull Request



Some projects may ask you to have only one commit in your pull requests

Squashing Changes

```
% git rebase -i master
```

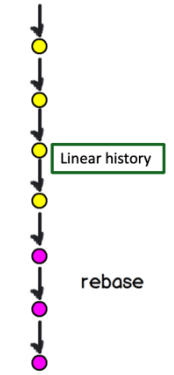
```
pick d7373be Added a desc to the README.md file
pick 2f5fa11 Added more info to the description

# Rebase aafa3a5..2f5fa11 onto aafa3a5 (2 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

```
pick d7373be Added a desc to the README.md file
squash 2f5fa11 Added more info to the description
```

```
# This is a combination of 2 commits.
# This is the 1st commit message:

Added a desc to the README.md file

# This is the commit message #2:

Added more info to the description

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
```

```
# This is a combination of 2 commits.
# This is the 1st commit message:

Added a desc to the README.md file, including an example use case



# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
```

```
businge@Johns-MacBook-Pro-2 checks % git rebase -i main
[detached HEAD 78bc0e1] Added a desc to the README.md file including an example use case
 Date: Tue Dec 6 13:16:42 2022 -0800
 1 file changed, 10 insertions(+), 1 deletion(-)
Successfully rebased and updated refs/heads/readme.
```



git rebase - rewrites the history of commits

Public Repositories ✗

WAIVED

pull request

Squashing Changes

```
businge@Johns-MacBook-Pro-2 checks % git show
commit 78bc0e16b559e38220debcd9e2c3da70f980ebdf (HEAD -> readme)
Author: johnxu21 <johnxu21@gmail.com>
Date:   Tue Dec 6 13:16:42 2022 -0800

    Added a desc to the README.md file including an example use case

diff --git a/README.md b/README.md
index 4f4688e..9b2b7bc 100644
--- a/README.md
+++ b/README.md
@@ -1 +1,10 @@
-# checks
\ No newline at end of file
+# checks
+
+This file is used sort a list of names by their surnames.
+
+
+# Example
+
+Passing sortSur(nameList = ['John Wick', 'Jason Voorhees'])
+
+will return: nameList = ['Jason Voorhees', 'John Wick']
```

```
businge@Johns-MacBook-Pro-2 checks % git status
On branch readme
Your branch and 'origin/readme' have diverged,
and have 1 and 2 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

nothing to commit, working tree clean
```

```
businge@Johns-MacBook-Pro-2 checks % git log --graph --oneline -4
* 78bc0e1 (HEAD -> readme) Added a desc to the README.md file including an example use case
* aafa3a5 (origin/main, origin/HEAD, main) Python file for rearranging names
* e8130f1 Initial commit
businge@Johns-MacBook-Pro-2 checks % git push
```

```
businge@Johns-MacBook-Pro-2 checks % git push
To https://github.com/johnxu21/checks.git
 ! [rejected]        readme -> readme (non-fast-forward)
error: failed to push some refs to 'https://github.com/johnxu21/checks.git'
hint: Updates were rejected because the tip of your current 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.
```

Squashing Changes

```
businge@Johns-MacBook-Pro-2 checks % git push -f
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 493 bytes | 493.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/johnxu21/checks.git
 + 2f5fa11...78bc0e1 readme -> readme (forced update)
businge@Johns-MacBook-Pro-2 checks % git log --graph --oneline -4
* 78bc0e1 (HEAD -> readme, origin/readme) Added a desc to the README.md file including an exam
ple use case
* aafa3a5 (origin/main, origin/HEAD, main) Python file for rearranging names
* e8130f1 Initial commit
```

https://help.github.com/en/articles/about-pull-request-merges

Best Practices for Collaboration

- It's a good idea to always synchronize your branches before starting any work on your own.
- Avoid having very large changes that modify a lot of different things.
- When working on a big change, it makes sense to have a separate feature branch.
  - To make the final merge of the feature branch easier, it makes sense to regularly merge changes made on the master branch back onto the feature branch.
- Avoid rebasing on public repository

- https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/about-merge-conflicts
- https://help.github.com/en/github/collaborating-with-issues-and-pull-requests/resolving-a-merge-conflict-using-the-command-line

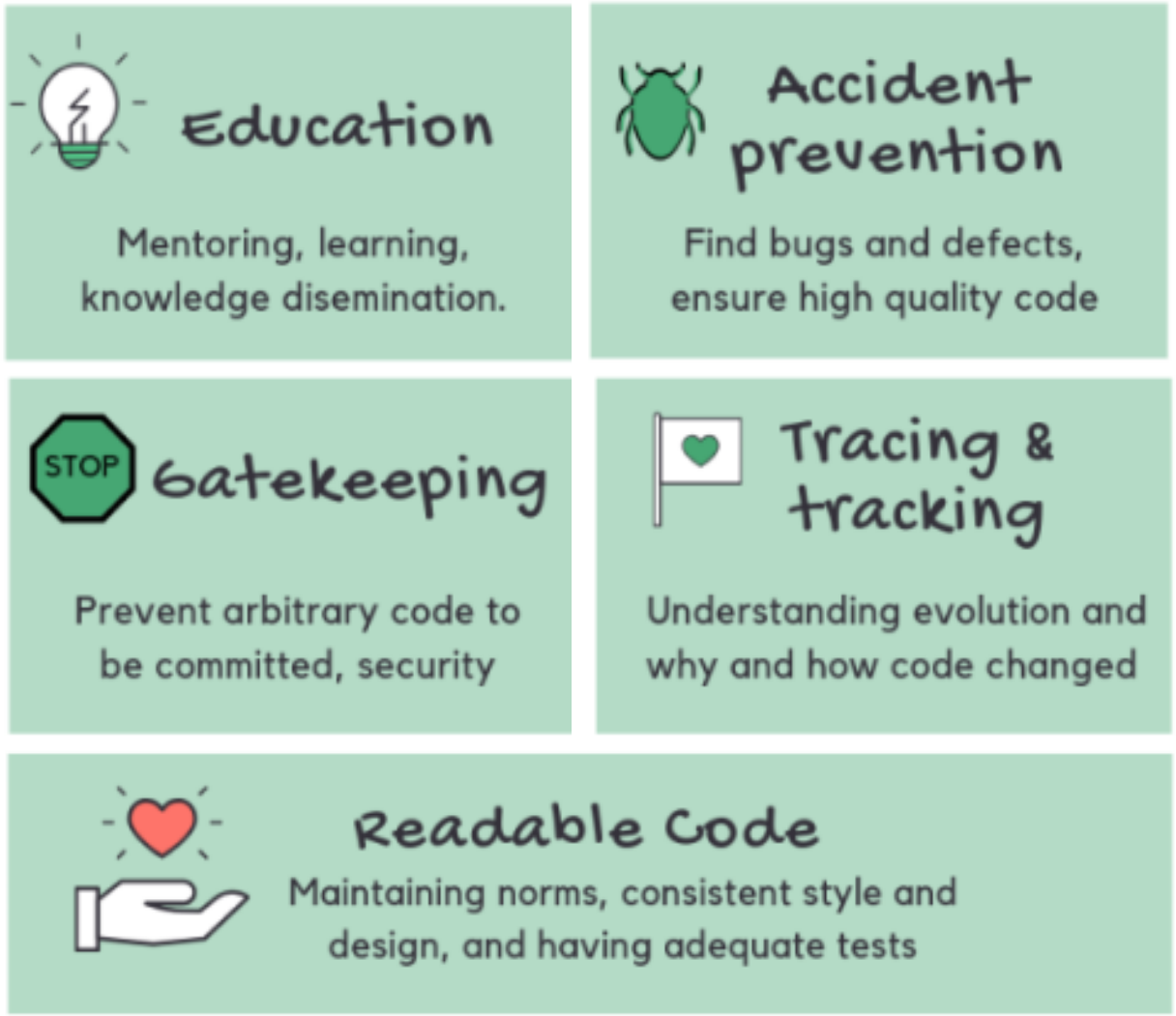
Code Reviews

Code review (sometimes referred to as peer review) is a software quality assurance activity in which **one** or **several people** check a program mainly by viewing and reading parts of its source code, and they do so after implementation or as an interruption of implementation.

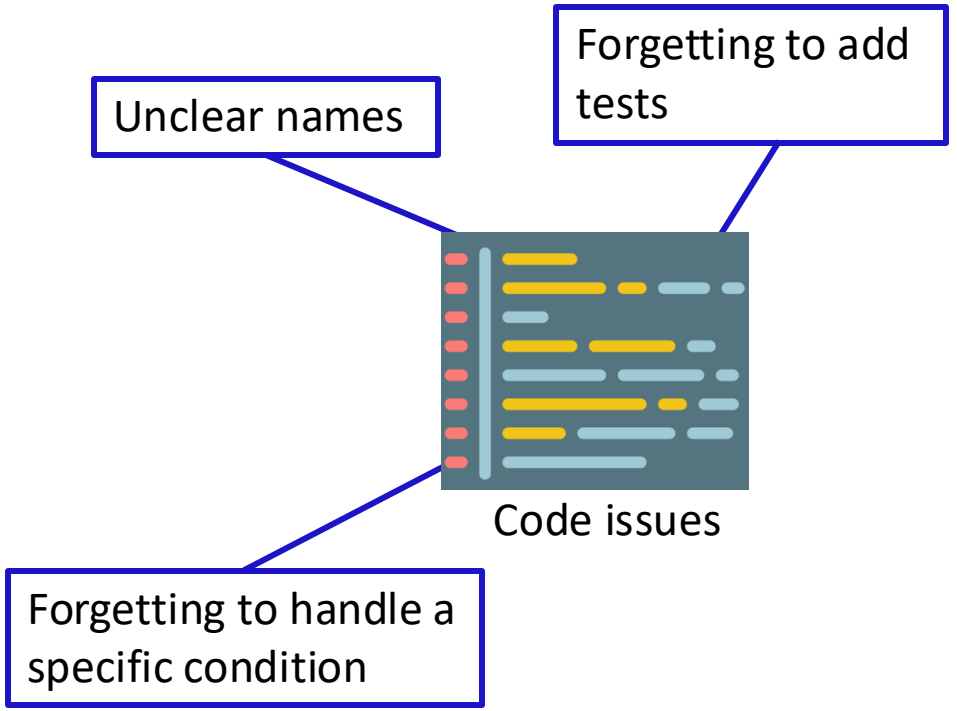
- At least one of the persons must not be the code's author.
- The persons performing the checking, excluding the **author**, are called "**reviewers**".

- Documentation
- Configurations
- Code
- Design Diagrams

Why do we need to perform code reviews?



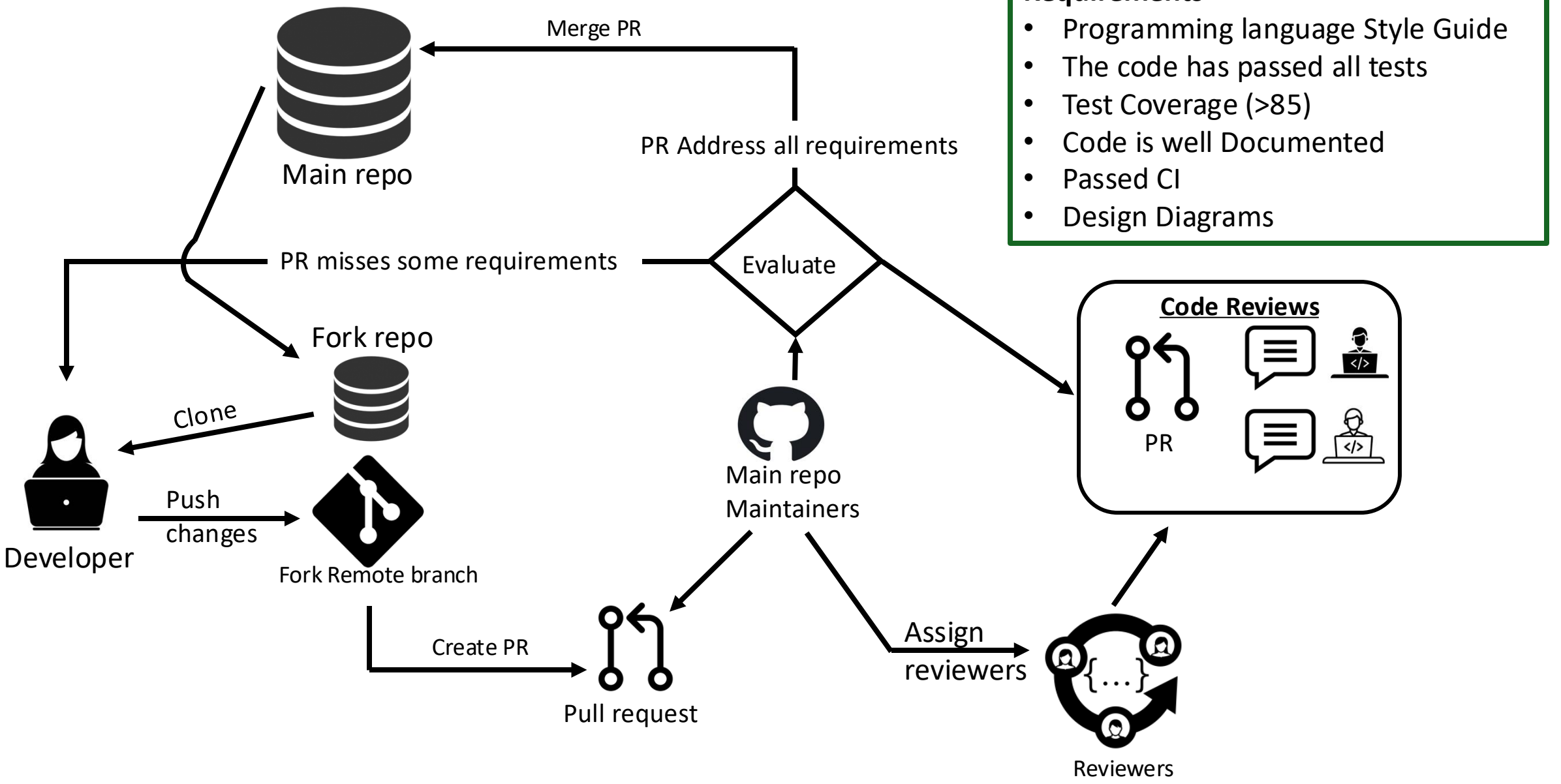
One thing to always remember, code reviews are not about us being good or bad coders, they're about making our code better.



Requirements

- Programming language Style Guide
- The code has passed all tests
- Test Coverage (>90)
- Code is well Documented
- Passed CI
- Design Diagrams

A simple Code Review Workflow



A simple Code Review Workflow

Before Jim-mister adds his colleagues as reviewers to his PR, he needs to **write a detailed Pull Request description** that addresses the following:

- **What has changed** between his feature branch and the dev branch
- Checklist of **prerequisites** (e.g. added documentation, added tests, ready to merge, etc.)
- **How to use the feature** (a GIF of the feature in action is extremely useful here)
- **How the design has changed** and how it compares to the design mockups (screenshots comparing the designs to the implementation)
- **Additional notes** the reviewers should be made aware of

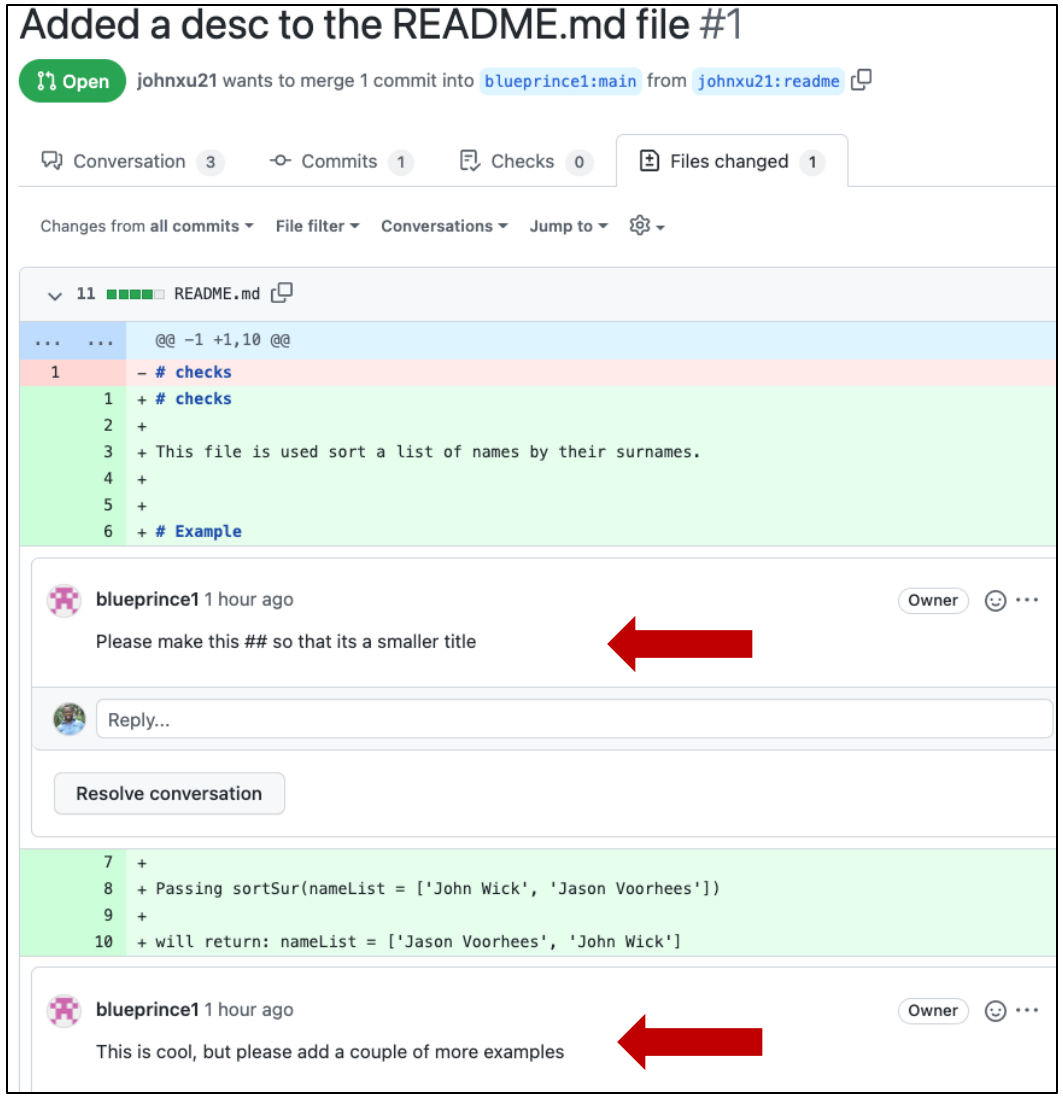
> *What goes into a pull request is usually highly project-dependant. Often times, you will see projects that have lots of pull requests make use of pull request templates to pre-populate a lot of the required information for the PR description.*

Once Jimmy is done submitting his detailed pull request, he waits...**but not too long!**

> *A good review process requires that pull requests get addressed as soon as possible in order to prevent the project from being impeded. Ideally, pull requests are reviewed within two hours of their submission.*

https://medium.com/osedea/the-perfect-code-review-process-845e6ba5c31

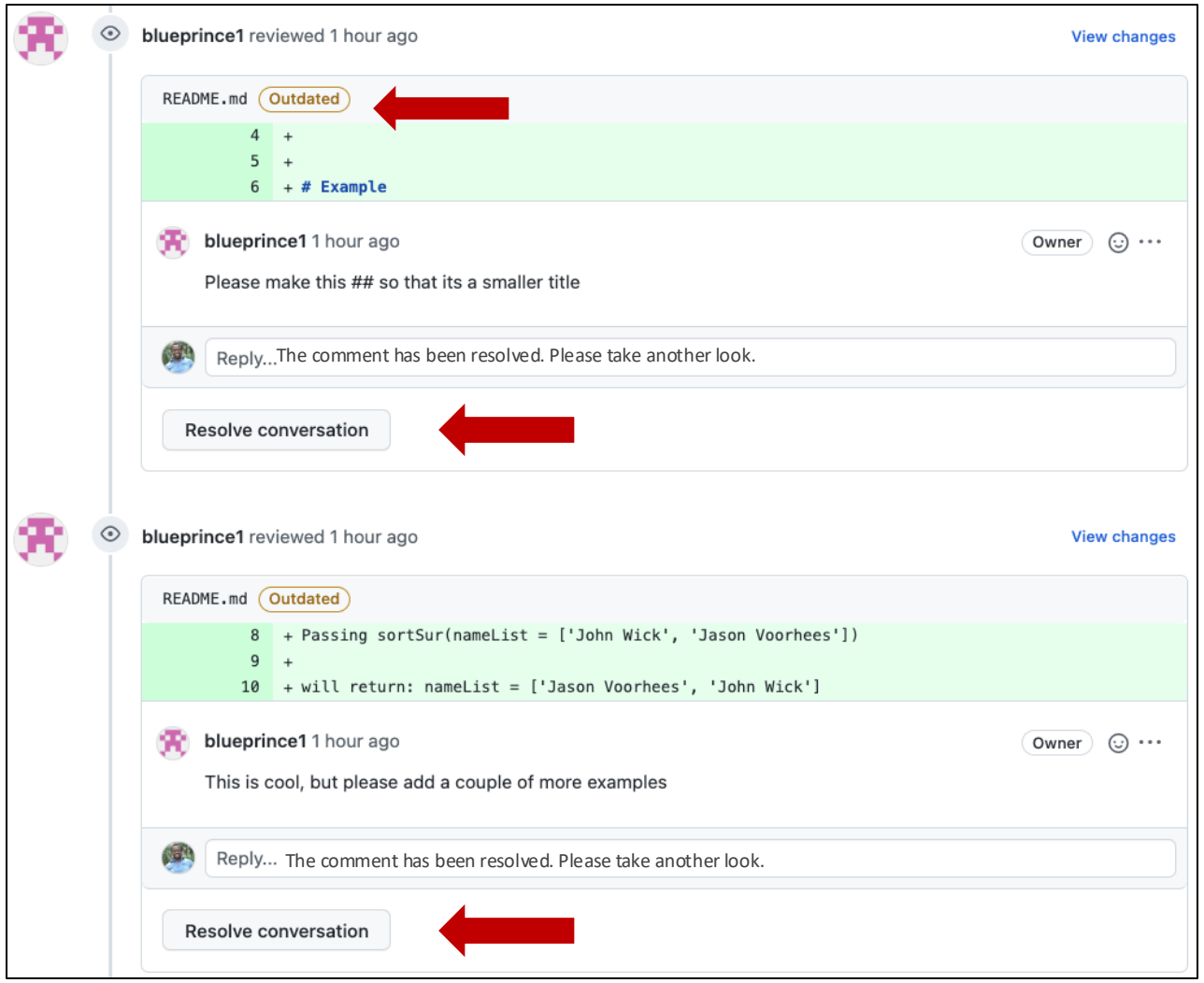
Code Reviews on GitHub

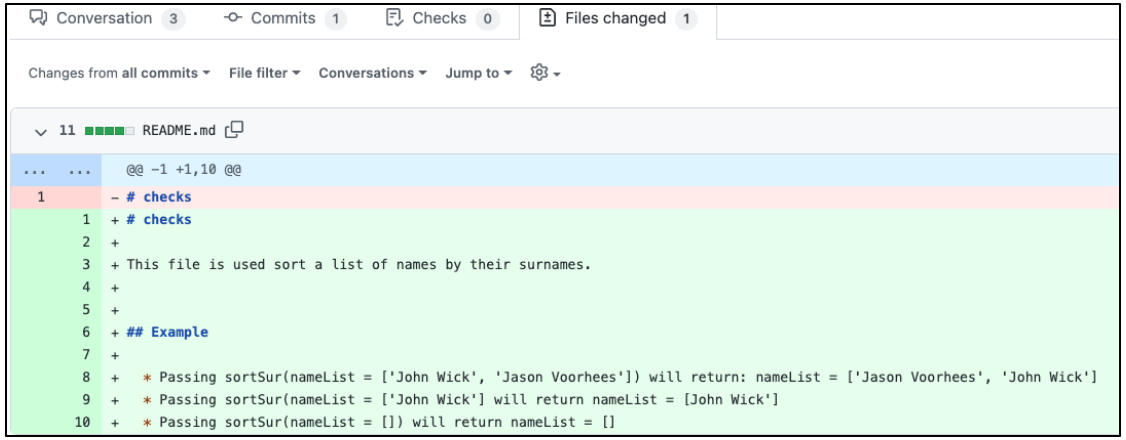



```
businge@Johns-MacBook-Pro-2 checks % atom README.md
businge@Johns-MacBook-Pro-2 checks % git status
On branch readme
Your branch is up to date with 'origin/readme'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
businge@Johns-MacBook-Pro-2 checks % git commit -a --amend
[readme 1abdaa6] Added a desc to the README.md file including an example use case
 Date: Tue Dec 6 13:16:42 2022 -0800
 1 file changed, 10 insertions(+), 1 deletion(-)
businge@Johns-MacBook-Pro-2 checks % git status
On branch readme
Your branch and 'origin/readme' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

nothing to commit, working tree clean
businge@Johns-MacBook-Pro-2 checks % git push -f
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 512 bytes | 512.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/johnxu21/checks.git
 + 78bc0e1...1abdaa6 readme -> readme (forced update)
```

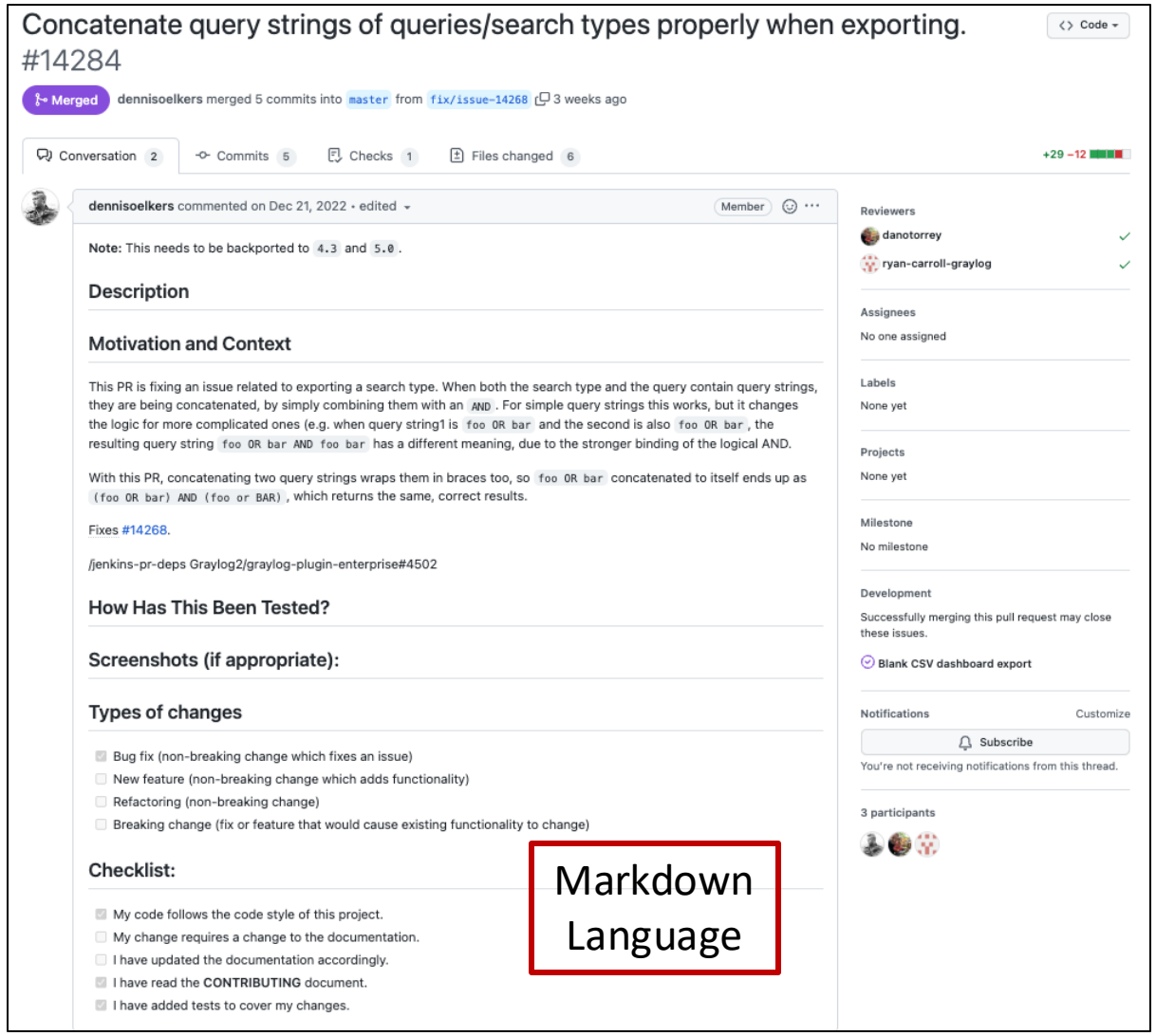
```
businge@Johns-MacBook-Pro-2 checks % git status
On branch readme
Your branch is up to date with 'origin/readme'.

nothing to commit, working tree clean
```

Code Reviews on GitHub





Code Reviews on GitHub



Concatenate query strings of queries/search types properly when exporting. #14284

Code

Merged dennisoelkers merged 5 commits into master from fix/issue-14268 3 weeks ago

Conversation 2 Commits 5 Checks 1 Files changed 6 +29 −12

dennisoelkers commented on Dec 21, 2022 • edited Member

Note: This needs to be backported to `4.3` and `5.0`.

Description

Motivation and Context

This PR is fixing an issue related to exporting a search type. When both the search type and the query contain query strings, they are being concatenated, by simply combining them with an `AND`. For simple query strings this works, but it changes the logic for more complicated ones (e.g. when query string1 is `foo OR bar` and the second is also `foo OR bar`, the resulting query string `foo OR bar AND foo bar` has a different meaning, due to the stronger binding of the logical AND.

With this PR, concatenating two query strings wraps them in braces too, so `foo OR bar` concatenated to itself ends up as `(foo OR bar) AND (foo or BAR)`, which returns the same, correct results.

Fixes #14268.

/jenkins-pr-deps Graylog2/graylog-plugin-enterprise#4502

How Has This Been Tested?

Screenshots (if appropriate):

Types of changes

- [ ] Bug fix (non-breaking change which fixes an issue)
- [ ] New feature (non-breaking change which adds functionality)
- [ ] Refactoring (non-breaking change)
- [ ] Breaking change (fix or feature that would cause existing functionality to change)

Checklist:

- [x] My code follows the code style of this project.
- [ ] My change requires a change to the documentation.
- [ ] I have updated the documentation accordingly.
- [x] I have read the **CONTRIBUTING** document.
- [x] I have added tests to cover my changes.

Reviewers: danotorrey ✓, ryan-carroll-graylog ✓

Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. Blank CSV dashboard export

Notifications: Customize — Subscribe — You're not receiving notifications from this thread.

3 participants

Markdown Language

danotorrey approved these changes on Dec 21, 2022 View changes

danotorrey left a comment Contributor

LGTM and tests successfully with instance of problematic dashboard. Exports are now successful.

@dennisoelkers Looks like there are some test failures in `AggregationEventProcessorTest`. Perhaps those tests need to be updated with the new grouping logic.

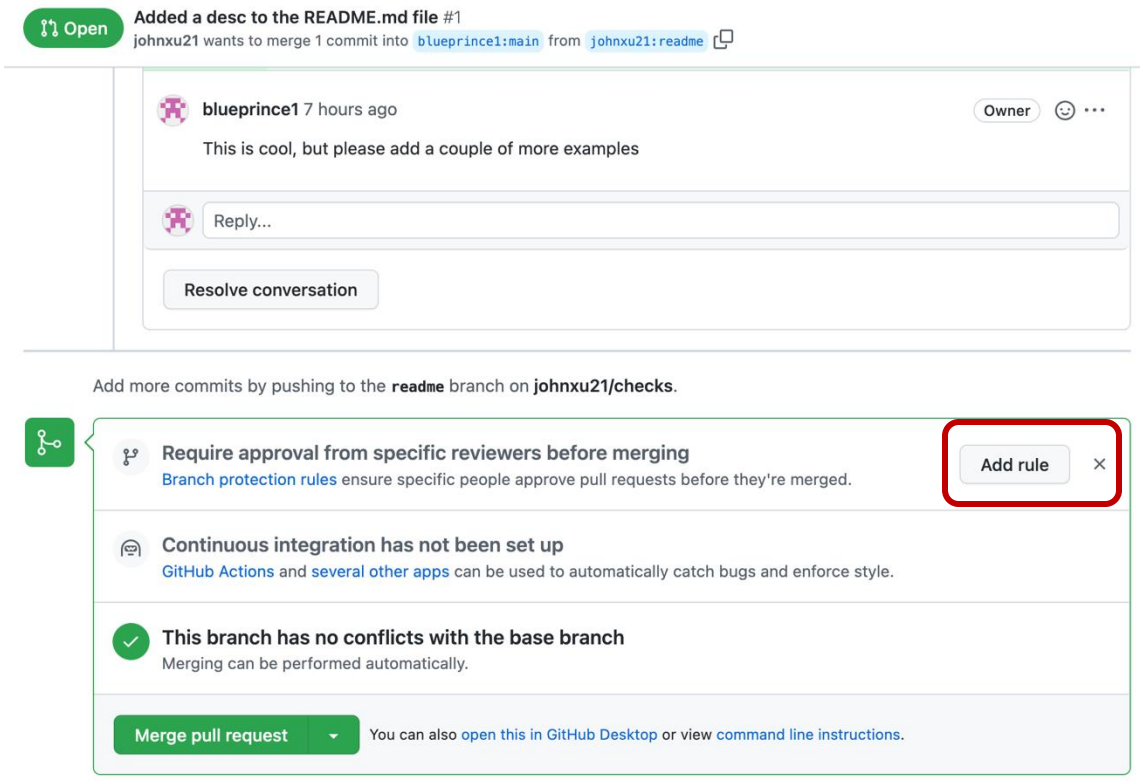
1

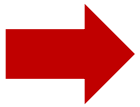
ryan-carroll-graylog approved these changes on Dec 21, 2022 View changes

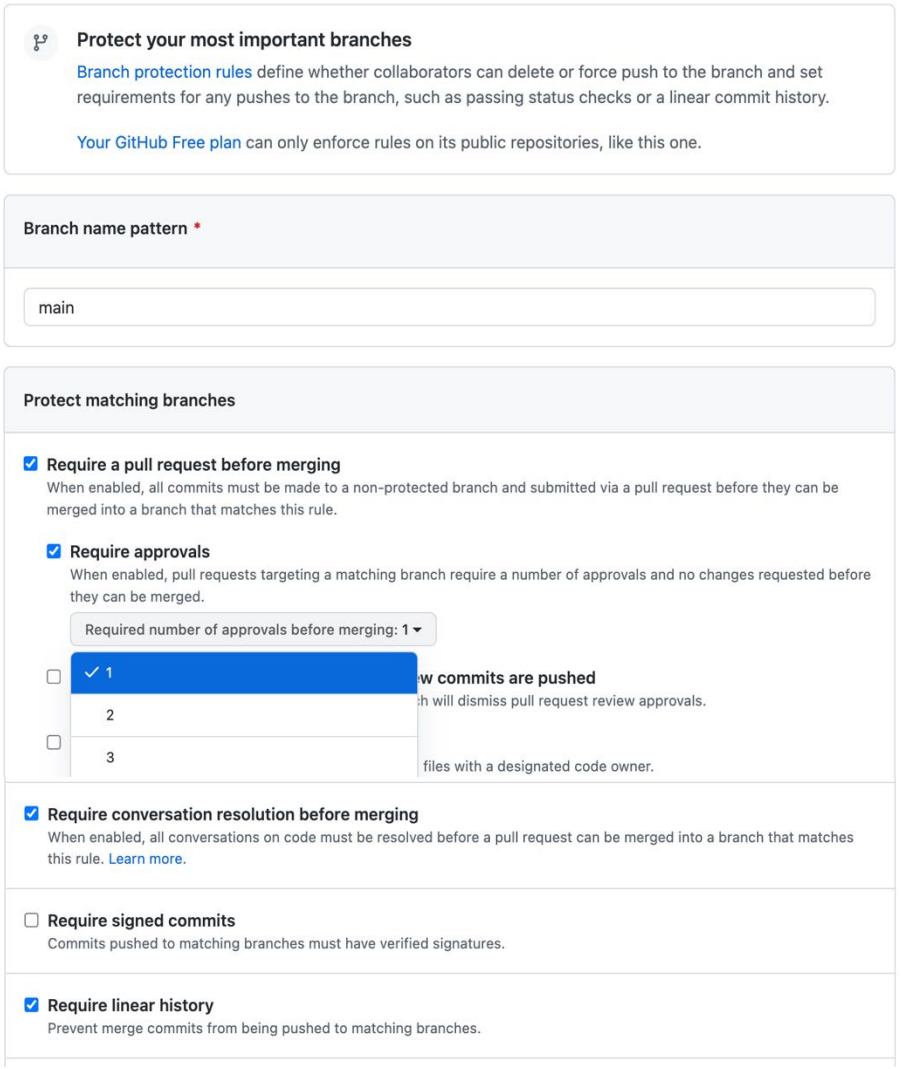
ryan-carroll-gra... left a comment Contributor

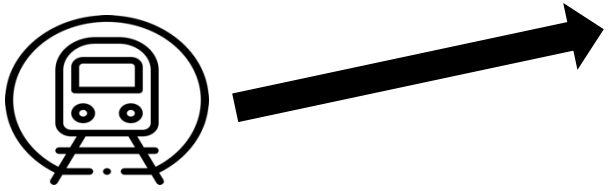
Looks good and tests successfully with Dashboard that was previously causing export issues.

Merging PRs – Branch Protection Rules



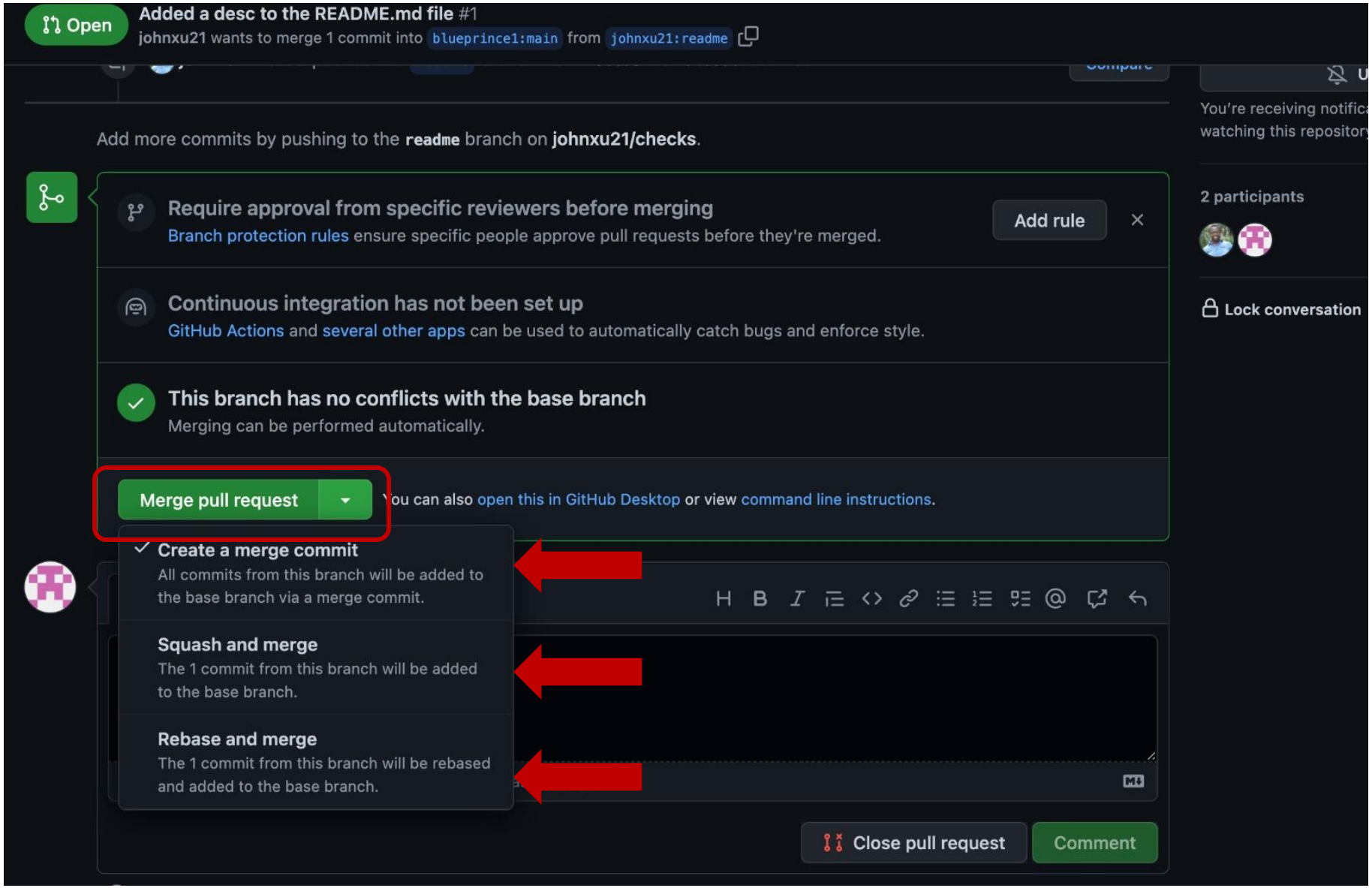






Project

Integrating PRs



Project

More Information on Code Reviews

Check out the following links for more information:

- http://google.github.io/styleguide/
- https://help.github.com/en/articles/about-pull-request-reviews
- https://medium.com/osedea/the-perfect-code-review-process-845e6ba5c31
- https://smartbear.com/learn/code-review/what-is-code-review/

Managing Team Projects

Good planning and communication key managing team projects

- Documenting what you do and why you do it becomes even more important; otherwise, you'll spend most of your time answering everybody else questions.
  - The basic form of this is writing clear code with good comments and documentation for those functions.
- If you are a project maintainer, you must promptly reply to pull requests and not let them stagnate.
- When coordinating who does what and when a common strategy for active software projects is to use an issue tracker.

Issue Template

name	about	title	labels	assignees
Feature request	Suggest an idea for Botonic		enhancement	

Is your feature request related to a problem? Please describe. A clear and concise description of what the problem is. Ex. I'm always frustrated when [...]

Describe the solution you'd like A clear and concise description of what you want to happen.

Describe alternatives you've considered A clear and concise description of any alternative solutions or features you've considered.

Additional context Add any other context or screenshots about the feature request here.

Feature request

name	about	title	labels	assignees
Bug report	Create a report to help Botonic improve		bug	

Describe the bug A clear and concise description of what the bug is.

To Reproduce Steps to reproduce the behaviour:

1. Go to '...'
2. Click on '....'
3. Scroll down to '....'
4. See error
5. Intuition about the error / Proposed solution (optional)

What is the expected behaviour? A clear and concise description of what you expected to happen.

Screenshots If applicable, add screenshots to help explain your problem.

About the environment Which versions of **botonic** packages, and which browser/OS or environment (development/production) are affected by this issue? Did this work in previous versions of **botonic** packages?

Additional context Add any other context about the problem here.

Bug report

https://github.com/hubtype/botonic/tree/master-lts/.github/ISSUE_TEMPLATE

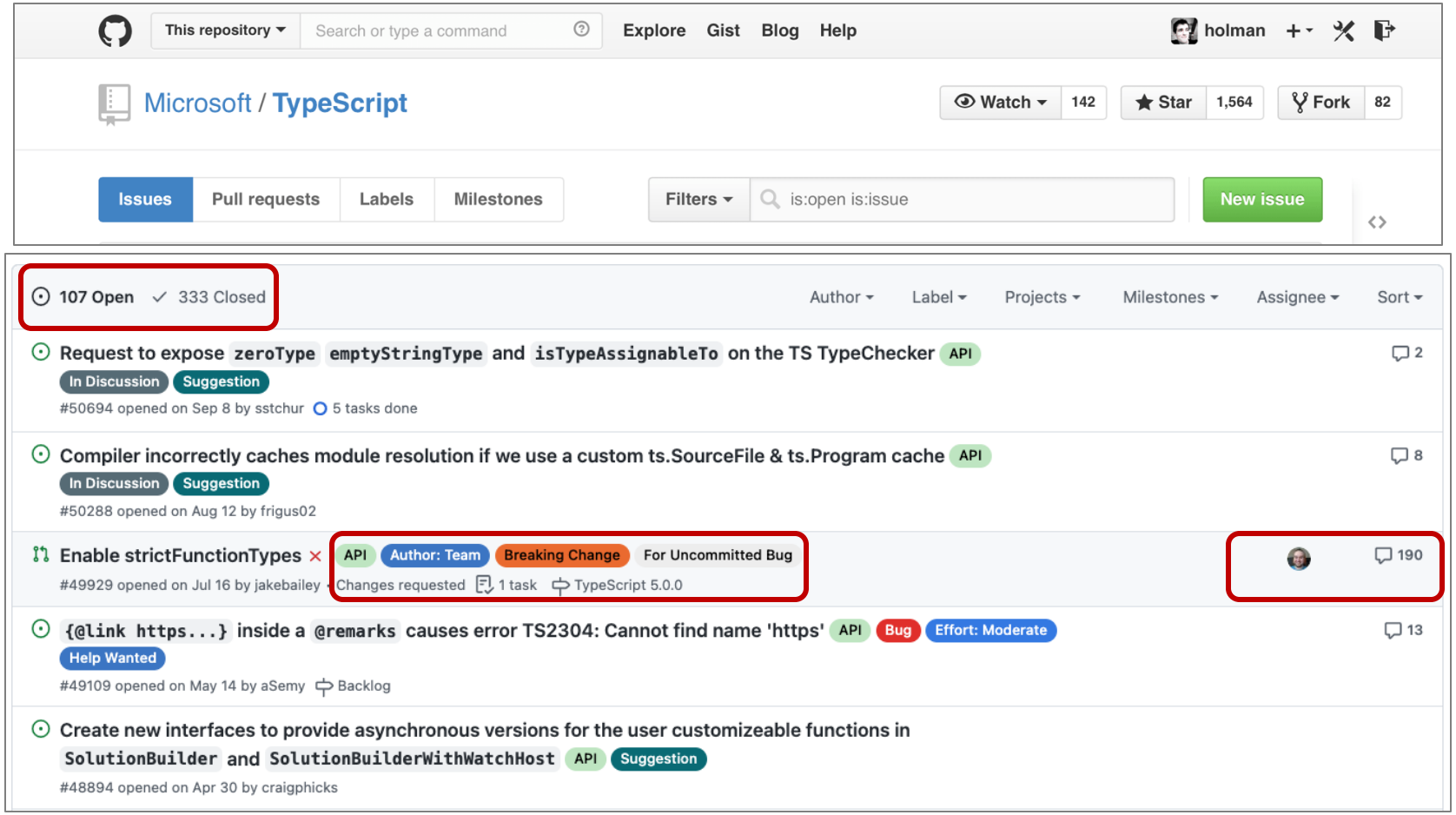
Issue Triage

- Issue triaging is the process by which maintainers of a project takes to review and organize new GitHub issues and pull requests.
- It involves categorizing issues and pull requests based on factors such as priority/urgency, ownership of the issue, and the issue/pull request label (bug, feature, enhancement, etc.).

Read about Issue Triaging

- https://github.com/microsoft/vscode/wiki/Issues-Triaging
- https://learn.wordpress.org/tutorial/how-to-do-triage-on-github/

Issue Tracker - GitHub



Issues

- Feature requests
- Bugs to fix
- New tests to add

Tracking Issues - GitHub

Issue

Question marks (JSDoc types) allowable in instantiation expression type arguments #51802

New issue

Closed jcalz opened this issue 11 hours ago · 9 comments · Fixed by #51804

Title

jcalz commented 11 hours ago Contributor

Bug Report

Search Terms

instantiation expressions, question mark, jsdoc nullable, any, Java wildcard

Version & Regression Information

- This is the behavior in every version since Instantiation expressions #47607 introduced instantiation expressions (TypeScript 4.7 and up)

Playground Link

Playground link with relevant code

Detailed description

Code

```
function foo<T>(x: T): T { return x }
const WhatFoo = foo<?> // (x: any) => any
const HuhFoo = foo<string?> // (x: string | null) => string | null
const NopeFoo = foo<?string> // (x: string | null) => string | null
const ComeOnFoo = foo<?string?> //  (x: string | null) => string | null
```

Actual behavior

No compiler errors. The question marks seem to be treated like JSDoc nullable types and a question mark by itself seems to be interpreted as `any` (not sure if that's true for JSDoc either).

Expected behavior

Compiler errors on every question mark, saying that it's not valid, perhaps with a "JSDoc types are not allowed here" message.

This Stack Overflow question is asking about how to use the Java wildcard syntax in TypeScript, and I'm like 😻. Apparently this was suggested by VSCode type hints. I don't know much about JSDoc but I assume some support for it has accidentally leaked into instantiation expression checking.

Assignees: ahejlsberg

Labels: Bug, Fix Available

Projects: None yet

Milestone: TypeScript 5.0.0

Development: Successfully merging a pull request may close this issue. Visit child nodes in checkExpressionWithTyp... microsoft/TypeScript

Notifications: Customize — Subscribe

More Description later – As comments

Pull Request

Visit child nodes in `checkExpressionWithTypeArguments` #51804

Merged ahejlsberg merged 3 commits into `main` from `fix51802` 8 hours ago

Conversation 0 · Commits 3 · Checks 17 · Files changed 9 · +355 −4

ahejlsberg commented 9 hours ago Member

Fixes #51802.

ahejlsberg added 3 commits 9 hours ago

- Visit child nodes in checkExpressionWithTypeArguments aa2b4d2
- Accept new baselines a097911
- Add tests 781b588

typescript-bot assigned ahejlsberg 9 hours ago

typescript-bot added Author: Team, For Milestone Bug labels 9 hours ago

ahejlsberg requested

DanielRosenwass

Linking a pull request to an issue using a keyword

You can link a pull request to an issue by using a supported keyword in the pull request's description or in a commit message. The pull request **must be** on the default branch.

- close
- closes
- closed
- fix
- fixes
- fixed
- resolve
- resolves
- resolved

Keyword #Issue_No

Reviewers: DanielRosenwasser, RyanCavanaugh, sandersn

Assignees: ahejlsberg

Labels: Author: Team, For Milestone Bug

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues.

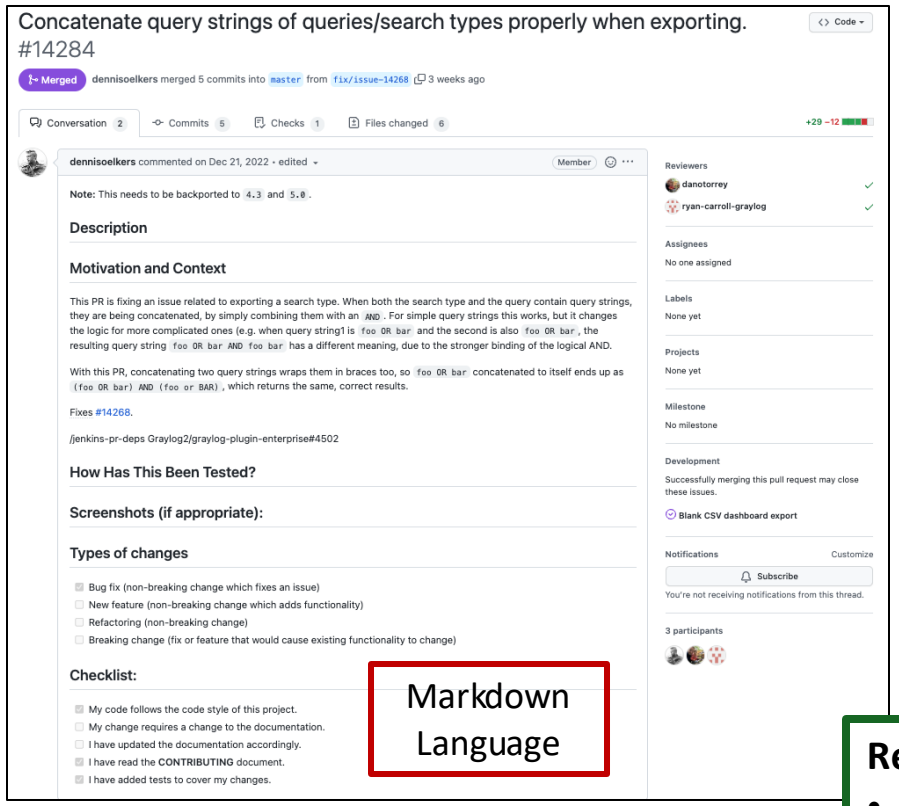
In the project, it is required that you link the Issues to the Pull requests

Managing Projects Additional Tools

Check out the following links for more information:

- https://arp242.net/diy.html
- https://help.github.com/en/articles/closing-issues-using-keywords
- https://help.github.com/en/articles/setting-guidelines-for-repository-contributors
- https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html
- https://stackify.com/what-is-cicd-whats-important-and-how-to-get-it-right/
- https://docs.github.com/en/actions/quickstart

Project – Pull requests and reviews requirements



Concatenate query strings of queries/search types properly when exporting. #14284

Merged dennisoelkers merged 5 commits into master from fix/issue-14268 3 weeks ago

Conversation 2 · Commits 5 · Checks 1 · Files changed 6 · +29 −12

dennisoelkers commented on Dec 21, 2022 · edited · Member

Note: This needs to be backported to `4.3` and `5.0`.

Description

Motivation and Context

This PR is fixing an issue related to exporting a search type. When both the search type and the query contain query strings, they are being concatenated, by simply combining them with an `AND`. For simple query strings this works, but it changes the logic for more complicated ones (e.g. when query string1 is `foo OR bar` and the second is also `foo OR bar`, the resulting query string `foo OR bar AND foo bar` has a different meaning, due to the stronger binding of the logical AND.

With this PR, concatenating two query strings wraps them in braces too, so `foo OR bar` concatenated to itself ends up as `(foo OR bar) AND (foo or BAR)`, which returns the same, correct results.

Fixes #14268.

/jenkins-pr-deps Graylog2/graylog-plugin-enterprise#4502

How Has This Been Tested?

Screenshots (if appropriate):

Types of changes

- [ ] Bug fix (non-breaking change which fixes an issue)
- [ ] New feature (non-breaking change which adds functionality)
- [ ] Refactoring (non-breaking change)
- [ ] Breaking change (fix or feature that would cause existing functionality to change)

Checklist:

- [x] My code follows the code style of this project.
- [ ] My change requires a change to the documentation.
- [ ] I have updated the documentation accordingly.
- [x] I have read the **CONTRIBUTING** document.
- [x] I have added tests to cover my changes.

Reviewers: danotorrey, ryan-carroll-graylog

Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone: No milestone

Development: Successfully merging this pull request may close these issues. Blank CSV dashboard export

Notifications: Customize · Subscribe · You're not receiving notifications from this thread.

3 participants

Markdown Language

Before Jim-mister adds his colleagues as reviewers to his PR, he needs to **write a detailed Pull Request description** that addresses the following:

- **What has changed** between his feature branch and the dev branch
- Checklist of **prerequisites** (e.g. added documentation, added tests, ready to merge, etc.)
- **How to use the feature** (a GIF of the feature in action is extremely useful here)
- **How the design has changed** and how it compares to the design mockups (screenshots comparing the designs to the implementation)
- **Additional notes** the reviewers should be made aware of

> *What goes into a pull request is usually highly project-dependant. Often times, you will see projects that have lots of pull requests make use of pull request templates to pre-populate a lot of the required information for the PR description.*

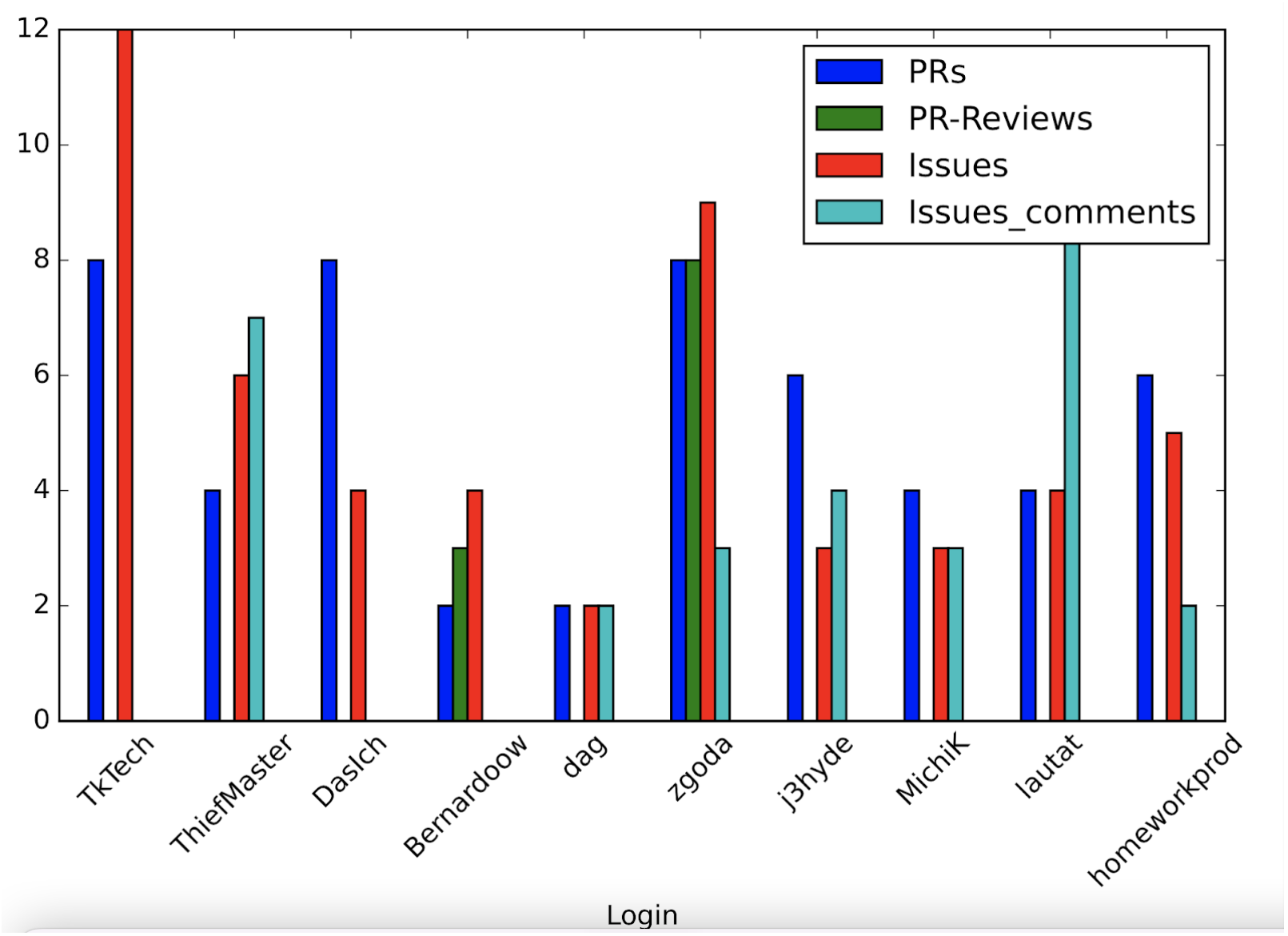
Once Jimmy is done submitting his detailed pull request, he waits…**but not too long**!

> *A good review process requires that pull requests get addressed as soon as possible in order to prevent the project from being impeded. Ideally, pull requests are reviewed within two hours of their submission.*

830 · 8 · •••

Requirements

- Programming language Style Guide
- The code has passed all tests
- Test Coverage (>90)
- Code is well Documented
- Passed CI
- Design Diagrams
- PR and issues are linked


12
10
8
6
4
2
0
PRs
PR-Reviews
Issues
Issues_comments
TkTech
ThiefMaster
DasIch
Bernardoow
dag
zgoda
j3hyde
MichiK
lautat
homeworkprod
Login

Project Evaluation

- Project Evaluation