Git

CSCI 4448/5448: Object-Oriented Analysis & Design

Learning Objectives

- Students will be able to…
  - Understand Git and related tools used in the class for source control
  - Create Markdown files for documentation
  - Use Git and GitHub for storage of code and markdown files for projects and collaboration with others

Tools for Review

- Git
- Markdown
- GitHub

https://xkcd.com/1597/

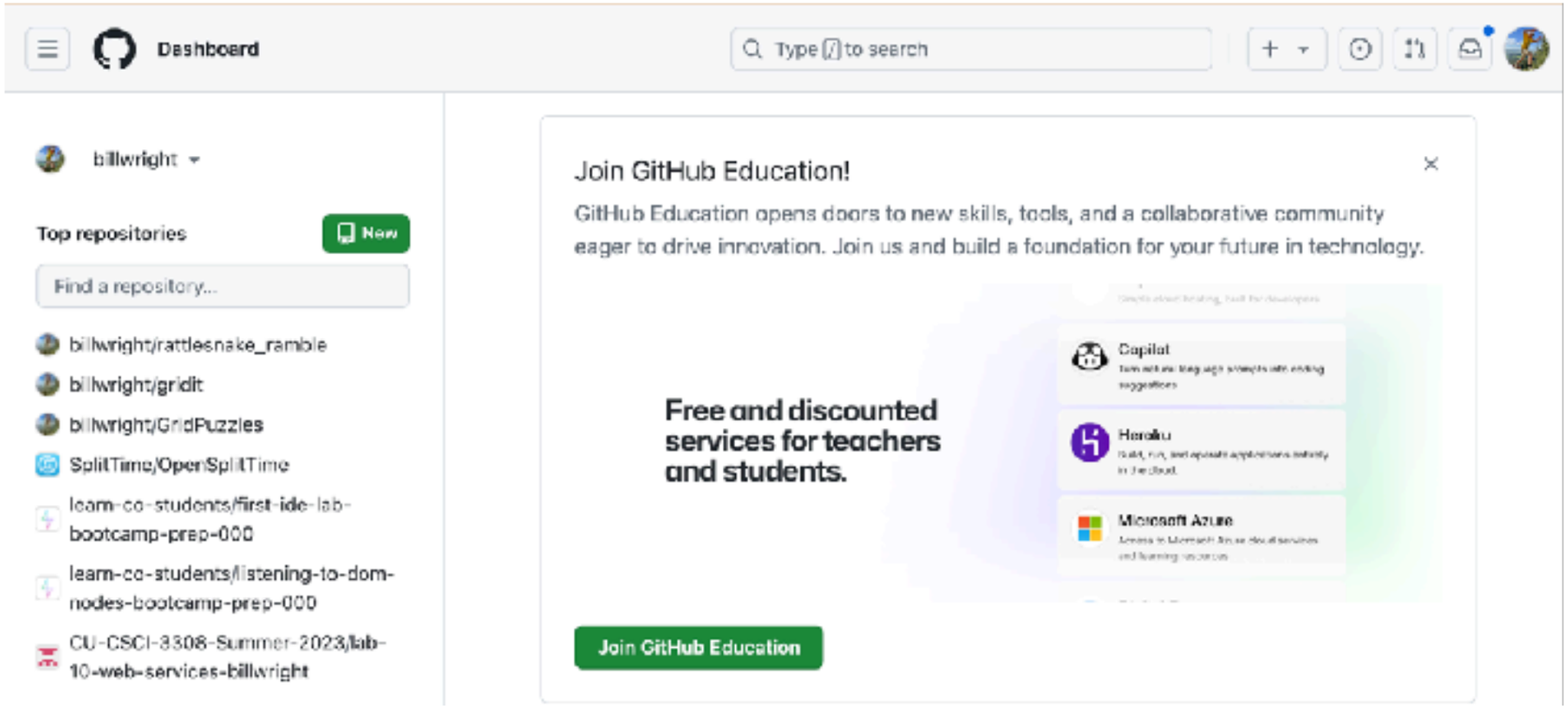
Git – A Version Control Tool

- Version control tools help us manage changes to code and documents, save copies of work, and collaborate with others, sharing/merging code
- Git is a standard and popular Version Control Tool
  - "Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency."
  - There are extensive training documents and videos [1]
  - Also there's an excellent online book, Pro Git [2]
- Git allows you to store code in local repositories (or repos) on your computer
- It's most often operated from a command line – in Git for Windows, this is called Git Bash – but you can find GUI tools for it as well, and it is often integrated into IDEs
  - Generally recommended to learn Git at the command line

Git – Setting Up

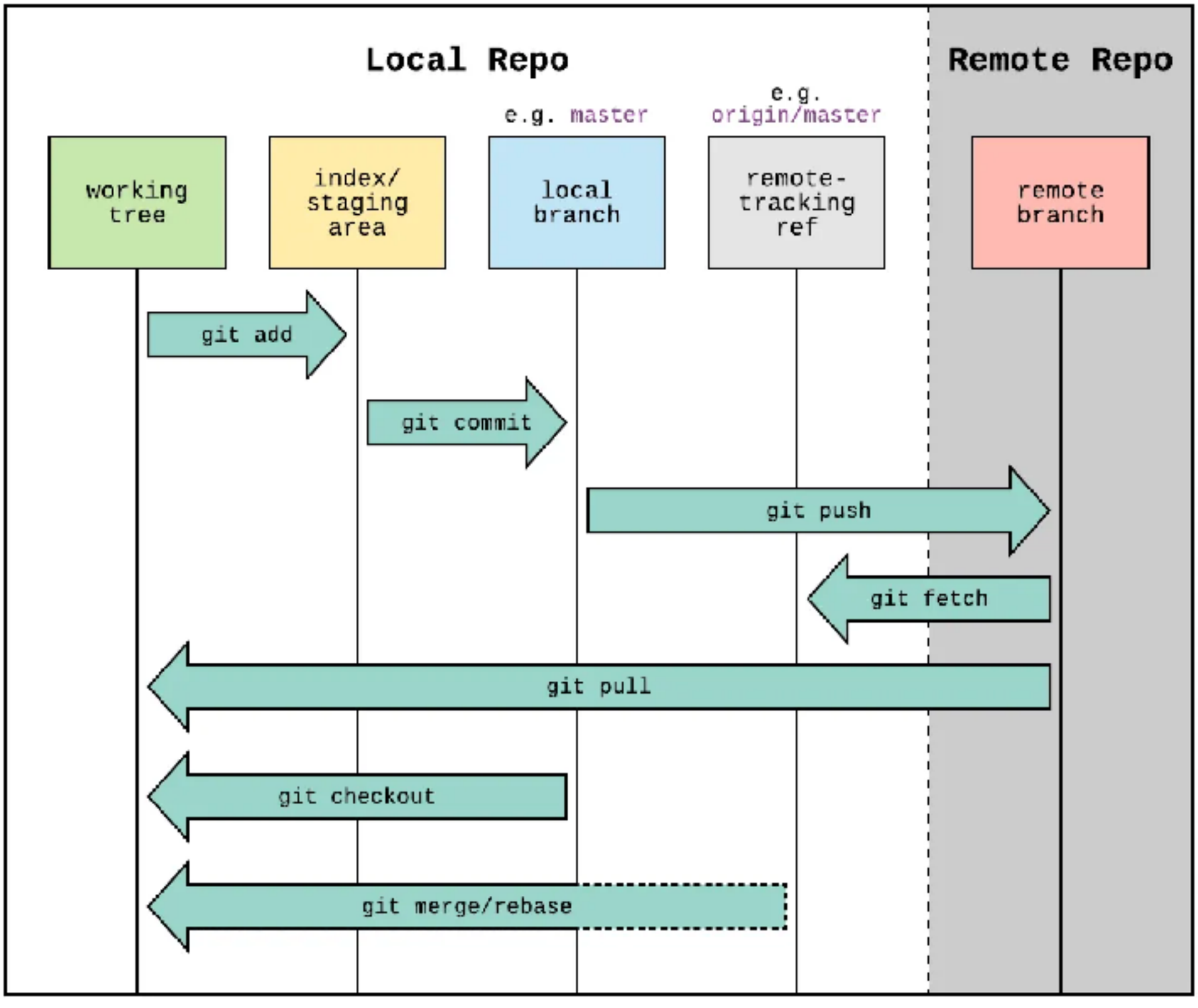
- You can install a version of Git for your environment (Mac, Windows, Linux), including GUI clients
- You will need to set your user name and e-mail in your Git instance
- Go to the Git Bash shell (or a Linux terminal prompt)
  - git config --global user.name “Your Name”
  - git config --global user.email “YourEmail@Example.com”
  - git config --list to see changes

Git with GitHub

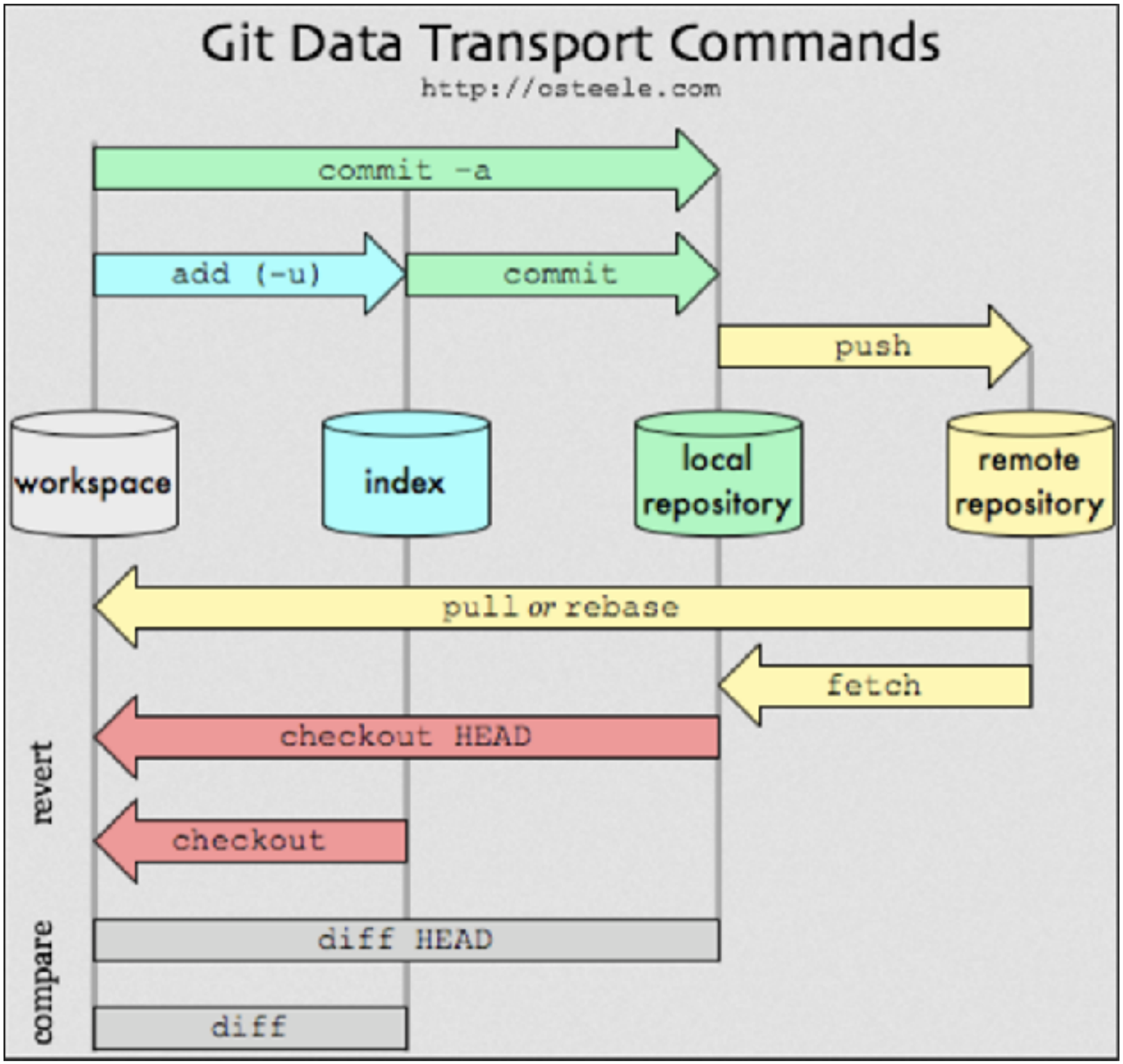


- GitHub is a web-based Git repository tool
  - Users can push/pull local repositories to web-based repositories
  - Provides a homepage for and backup of web-based repositories
- To set up a free GitHub account, go to GitHub site [5]
- Use the same e-mail used when setting up Git
- Explore the GitHub site, and edit your profile information

Common Git Commands



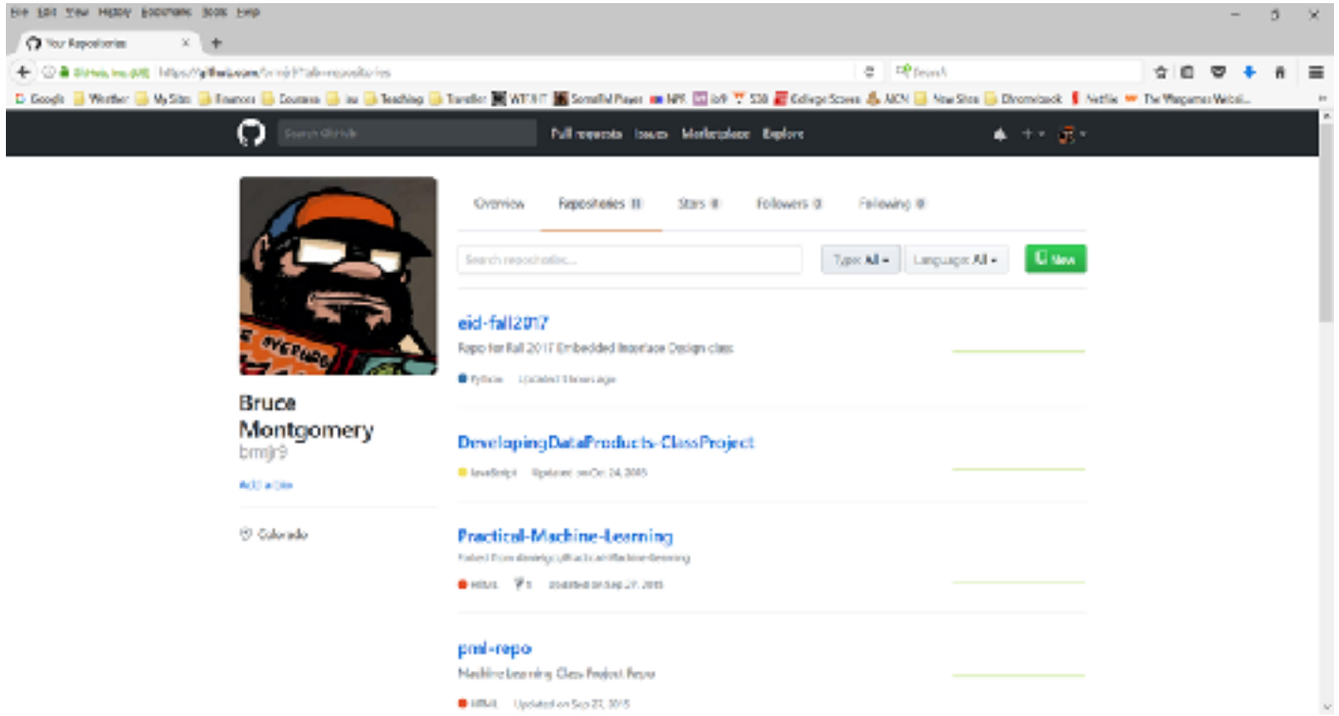
Another Command Summary



Basic Git/GitHub Change Flow

- git add
  - adds your working area or workspace files to your local index
- git commit
  - adds them to your local repo
- git push
  - pushes or sends your local repo to your remote repo (GitHub, GitLab, etc.)
- git pull
  - pulls or reads from your remote repo to your local repo for working on files

- See the Git documentation or the Pro Git book online for step by step examples



.gitignore

- A .gitignore file keeps you from worrying about files you should not commit
  - You only commit files necessary to build the application
  - You don’t save away files not relevant to the build
    - No class files ever!
- .gitignore files describe pattern used to add or exclude files from being tracked for the repo
- Intellij can create a .gitignore file for you
    - https://www.geeksforgeeks.org/what-is-git-ignore-and-how-to-use-it/
- A repository might have a single .gitignore file in its root directory which applies to the entire repo
- Also possible to have additional .gitignore files in subdirectories for local rules
    - https://stackoverflow.com/questions/5698148/where-does-the-gitignore-file-belong
- the purpose of gitignore files is to ensure that certain files not tracked by Git remain untracked
- To stop tracking a file that is currently tracked (already checked in), use git rm --cached
    - https://git-scm.com/docs/gitignore#:~:text=The%20purpose%20of%20gitignore%20files,use%20git%20rm%20%2D%2Dcached

Basic Git/GitHub Repo Creation

- Assume URL in Github is https://github.com/myname/my-repo.git for these examples
- Create a New Repo
  - GitHub: Create a New Repo
  - In your Git Bash or Linux Terminal
    - mkdir ~/my-repo
    - cd ~/my-repo
    - git init (to initialize the repo)
    - git remote add origin https://github.com/myname/my-repo.git (point your local repo to the web repo)
- Forking (copying) someone's repo
- Press Fork button in their repo on GitHub
  - In your Git Bash or Linux Terminal
    - git clone https://github.com/myname/my-repo.git

Git Extras

- New global configuration of .config/git/config and .config/git/ignore. This way the .gitignore file doesn’t need to be added to each new repo
- If you want to create the repo on GitHub yourself, make sure to do this:

  Create repo on GitHub, *not including* a License nor Git Ignore file (the repo cannot have any previous commit or there will be unrelated histories, which can’t be merged.

```
1.git remote add origin https://github.com/username/repo-name.git
2.git push origin main
```

Git Stash?

- If you have some work you…
  - Want to come back to later, but you don’t want to commit
  - Want to switch to another task and save work without a commit
  - Don’t want to change to another branch because of edit conflicts
- Stash lets you store work in progress without doing a commit
- Any work in progress that is not committed is saved and the staging area and working directory are cleared of changes

- git stash – saves changes from working directory in your stash
- git stash list – shows what’s in your stash
- git stash apply A – gets item A, leaves it in stash
- git stash drop A – deletes item A from stash
- git stash pop A – gets the item A, removes from stash
- git stash pop – gets last item you worked on
- git stash clear – clears out the stash

What is a Git Branch?

- A separate track of history, allowing you to work on different tasks/tickets/ideas simultaneously w/out overlap
- A branch is like a fresh copy of all your files
  - Experimentation
  - Stability
  - Collaborate with others
  - Diverging codebases or bucketing versions
  - Supports deployment workflows
  - They are cheap!
- Rule of thumb: If you're starting something new, do it in a branch

Git Branch, Merge, Diff, and Log

- git branch A – creates a branch called A
- git checkout A – moves you to the branch A (the HEAD pointer will point to A)
- git commit – will now commit to A
- git branch –a – lists the branches
- git log A..B – what commits are in B but not in A branches
- git diff A..B – the difference between A and B
- git diff A...B – changes that would be merged into master if you merged B
- git merge B – merges branch B into the branch you're on
- git log --oneline --decorate --graph --all – will show history as a graph with all commits and branches

Visualizing Git – http://pcottle.github.io/learnGitBranching
Great way to practice these operations

Collaborating on GitHub

- Many possible processes, here's a particularly good walkthrough [8]
  - Create a new repo on GitHub
  - Connect local git repos to remote GitHub repo
  - Push any initial files to GitHub repo
  - Add collaborators on GitHub
  - Collaborators will clone (not fork) the project to local working copies
  - Try to keep the master branch clean and deployable
  - Individuals work on code in their own branches, committing changes as needed
  - All collaborators push to Github and create pull requests
  - One person gets selected to merge (the "merger")
  - Working with the team, the merger reviews the pull requests and decides whether they are ready to merge
  - Once merged, branches can be deleted

Many GitHub features for development…

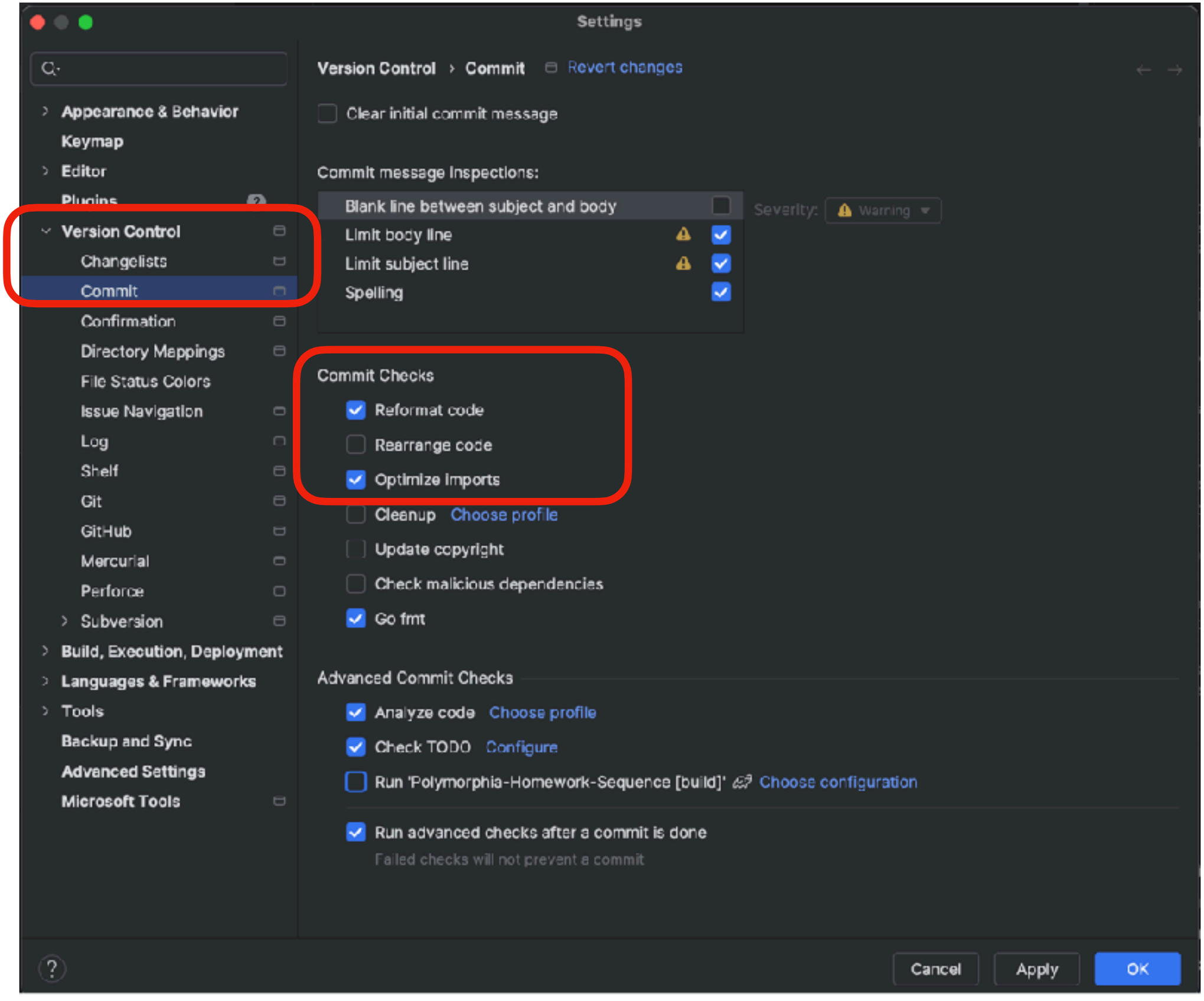
- Collaborative coding support – notifications, code reviews, etc.
- Continuous Integration, Deployment, and Automation – GitHub API, web pages, scripting, etc.
- Security – static analysis, secrets management, dependency graphs, etc.
- Various GitHub client apps
- Project management – issue/bug tracking, milestones, activity graphs, wikis, etc.
- Team administration and community management
- More details at: https://github.com/features

- Again, a good use of your time is to look at what GitHub provides for automating and improving the development process, and how it can help you and your team…

Git Thoughts

- Don’t check in .gradle directory into git! It can be gigantic and it’s binary
- Don’t check in any binary files (in general, there are exceptions) It’s not source
- NEVER do: “git add .” Why?
- Don’t commit broken code! Make sure all tests work first. How?
- Use good commit messages! “I hope this works” is not a good one…

> gradle test


Settings
Version Control › Commit
Revert changes
Clear initial commit message
Commit message Inspections:
Blank line between subject and body
Severity: Warning
Limit body line
Limit subject line
Spelling
Commit Checks
Reformat code
Rearrange code
Optimize Imports
Cleanup Choose profile
Update copyright
Check malicious dependencies
Go fmt
Advanced Commit Checks
Analyze code Choose profile
Check TODO Configure
Run 'Polymorphia-Homework-Sequence [build]' Choose configuration
Run advanced checks after a commit is done
Failed checks will not prevent a commit
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Changelists
Commit
Confirmation
Directory Mappings
File Status Colors
Issue Navigation
Log
Shelf
Git
GitHub
Mercurial
Perforce
Subversion
Build, Execution, Deployment
Languages & Frameworks
Tools
Backup and Sync
Advanced Settings
Microsoft Tools
Cancel
Apply
OK

Recommended Project Workflow

1. Create a branch
2. Write a failing test
3. Get the test to pass
4. Refactor
5. Commit
6. Done?
    1. If yes, push
    2. If no, go to 2

Markdown files

- Markdown files are text files with a simple set of formatting commands
- They are often used to create README or similar documentation files that accompany a set of code saved in version control
- There is a basic set of formatting syntax for adding headings, lists, bold/ italics, code blocks, links, and images
- There is extended syntax (used on GitHub and elsewhere), that adds tables, footnotes, strikethrough, and more

- Markdown was created in 2004 by John Gruber, with a goal that the syntax would not prevent reading a text file that had not been rendered [6]

Markdown example [7]

The quarterly results look great!

- Revenue was off the chart.
- Profits were higher than ever.

> *Everything* is going according to **plan**

The quarterly results look great!

- Revenue was off the chart.
- Profits were higher than ever.

> *Everything is going according to* ***plan***

Git and Related Resource Links

- Markdown Cheat Sheets/Editor
  - https://www.markdownguide.org/cheat-sheet/
  - https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf
  - Online Markdown Editor – https://stackedit.io/
- Git/GitHub Cheat Sheets
  - https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf
  - https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet
- Git/GitHub Materials
  - Pro Git 2 Book – https://git-scm.com/book/en/v2
  - Visualizing Git – http://pcottle.github.io/learnGitBranching
  - The Git Parable – http://bit.ly/1isB3K4
  - GitHub tutorial – https://docs.github.com/en/get-started/quickstart/hello-world
- External Merge Tools
  - https://www.sublimerge.com/ or https://www.scootersoftware.com/ (Beyond Compare)

References

[1] https://git-scm.com/doc

[2] https://git-scm.com/book/en/v2

[3] https://www.git-scm.com/downloads

[4] https://blog.osteele.com/2008/05/my-git-workflow/

[5] https://github.com/

[6] https://www.markdownguide.org/getting-started/

[7] https://www.markdownguide.org/basic-syntax

[8] https://medium.com/@jonathanmines/the-ultimate-github-collaboration-guide-df816e98fb67