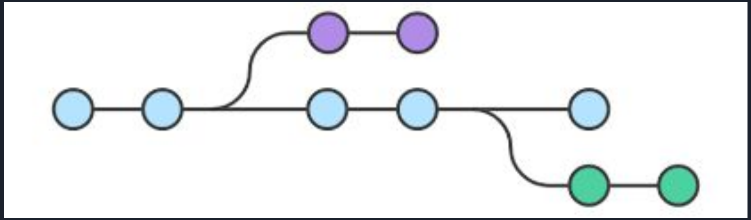
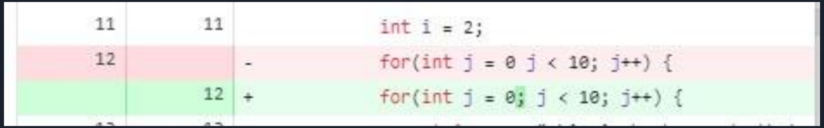
Continuous Integration

Git

- Version-control: Managing the history of changes in code
- Git is a version-control system for tracking changes in code for software development in the command-line
  - Command-line: A text-based user interface to the computer where you can enter text commands to perform tasks
- Git is widely used in the software development industry (Netflix, Apple, IBM, Amazon)
- Allows for:
  - Controlled collaborative work on software done through branching
  - Tracking who changes what code and what parts of the code were changed
  - Ensuring the code being worked on doesn't break the source code
  - Going back in time to work on older code if needed (e.g. code you are working on broke)





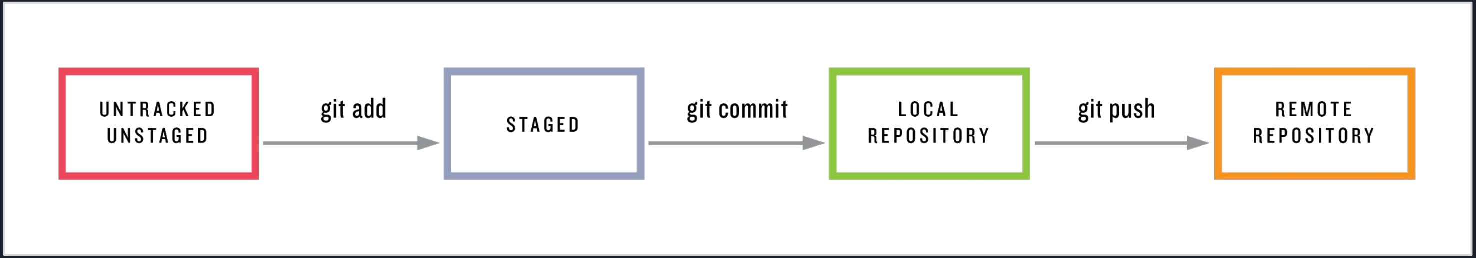
```
      int i = 2;
-     for(int j = 0 j < 10; j++) {
+     for(int j = 0; j < 10; j++) {
```

Git Terminology

- Working directory: The folder on your computer where you saved a copy of the source code
- Staging area: Changes to the code that will be added to the main source code
- Repository: The storage place that holds the source code
  - Local repository: the folder on your computer that holds your files
  - Remote repository: located at the service you've chosen to use (e.g. GitHub, GitLab, Kiln, etc)
- Commit: A snapshot of your code that can be revisited
- Branch: A pointer to a commit that indicates a divergence from the source code
  - main/master branch: a pointer to the commit that is the most recent up-to-date and working code
- Merge: Combines the changes in one or more branches so the history of commits is linear (no branching)
- HEAD: A pointer to the most current commit in your history

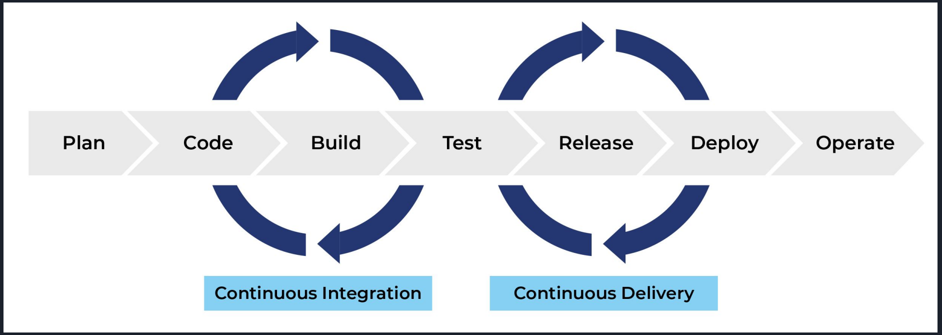
Git Commands

- `git add`: includes the files where the changes occured to the staging area
- `git commit -m " "`: creates a snapshot of your code with a message to describe it
- `git push -u origin main`: the snapshot changes are "committed" to the remote repository
  - `-u`: indicates the upstream (local) branch tracked to a specific remote branch
- `git fetch origin`: grabs any changes made to the source repository
- `git pull origin main`: grabs any changes and makes those changes on your local repository



CI/CD

- Continuous integration (CI): the automatic process of adding local code to a shared repository
  - Code is built, tested and awaits approval to be combined (integrated) to the shared repository
  - Build automation: Code is compiled, packaged, and run to ensure that it works (e.g. Gradle, Maven, Make, Ant, etc)
- Continuous delivery (CD): the process of producing software in short cycles and then automatically sending it for additional testing or to production



GitHub

- GitHub is a Git-based service hosted in the cloud that stores remote repositories
  - Others include: GitLab, Kiln, GitBucket
  - It provides a graphical view on your repositories
- GitHub Actions: Online CI/CD tool
  - Uses workflows to runs through the gradle build process and additional test cases
- GitHub Desktop: Work with GitHub on your desktop. It also makes the Git process more user-friendly by removing the need for the command-line
  - You are not allowed to work on the command-line

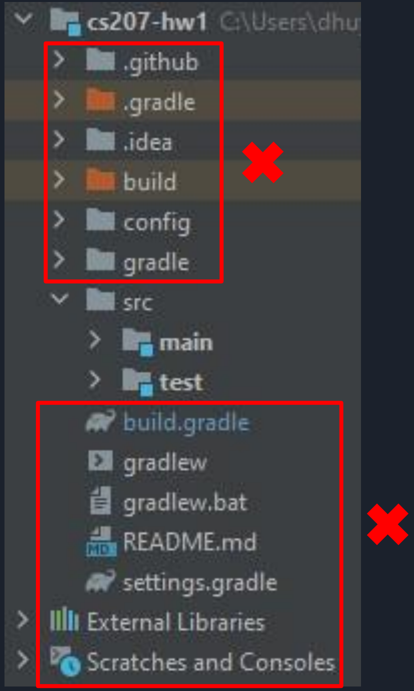
Git	GitHub
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the Git repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

Static Code Analyzers

- Static code analysis is the process of debugging code before it gets run
- The code is analyzed against a set of coding rules
- This ensures code is well-written and well-designed
- Checkstyle
  - Makes sure how code looks conforms to a specific standard
  - Ex. Naming conventions, code redundancy, unused code, etc.
- PMD
  - Makes sure how code is designed follows industry best practices
  - Ex. Closing resources, catching correct exceptions, equals method written correctly, etc.
- The rules chosen are specific to Programming II, there are actually more rules that code needs to follow

Restrictions

- You cannot work in the command-line
  - Why? Git has many more commands available through the command-line and this creates an exponential increase in potential issues that can arise
- You cannot alter or delete any files outside of what you are instructed to work with
  - Why? The issues range from breaking the GitHub Actions build process to passing the build process test cases when you actually aren't passing them



Getting Started

- Download and install Git
  - https://git-scm.com/downloads
- Create a GitHub account
  - https://github.com/
- Download and install GitHub Desktop
  - https://desktop.github.com/