Crash Course in Git

Fundamentals of Git

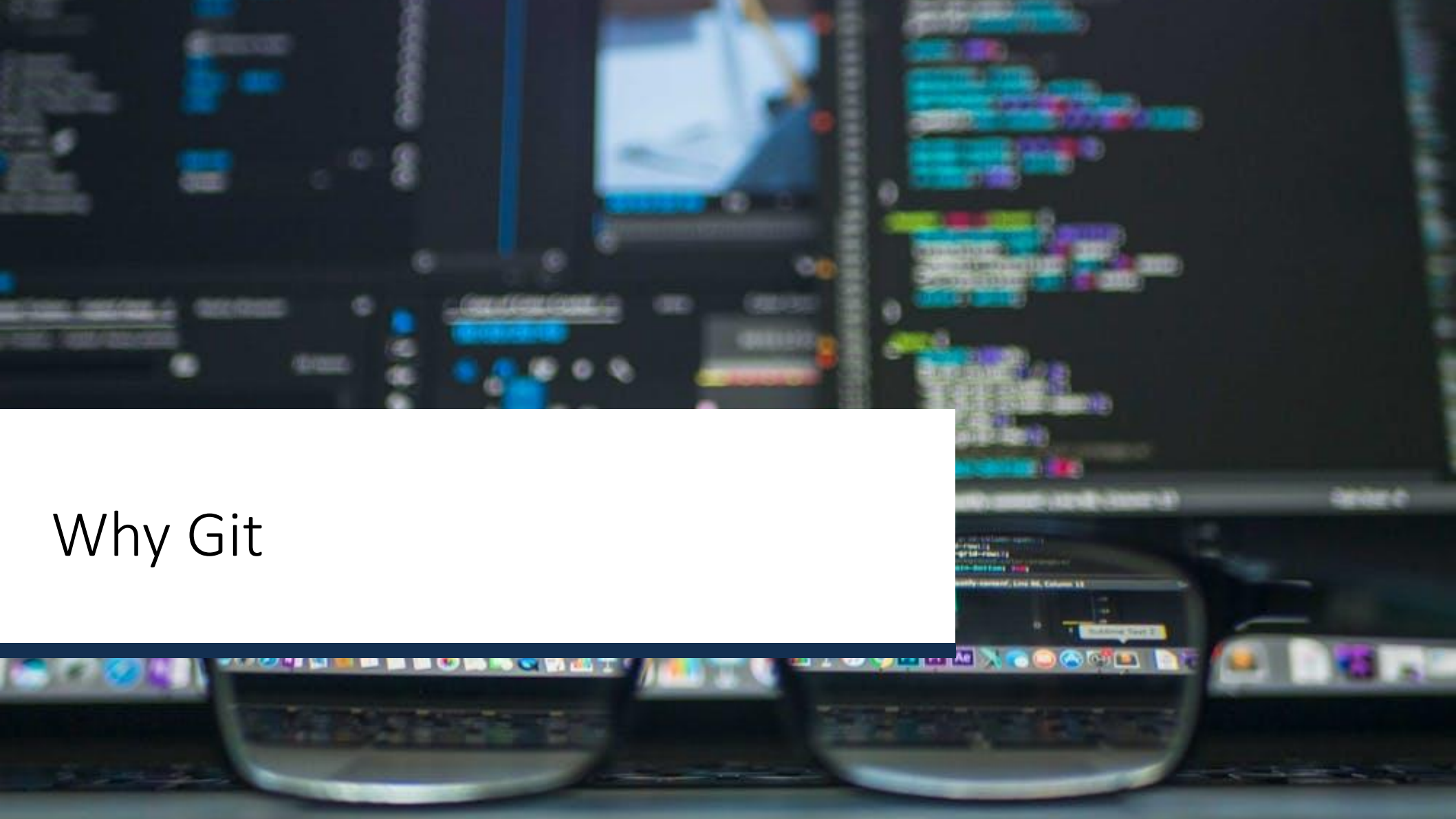
David Gundry

Learning Outcomes

- Explain the motivation behind using version control
- Describe key terminology and concepts in the use of git
- Perform basic Git tasks on the command line
- Perform basic tasks on GitHub

Overview

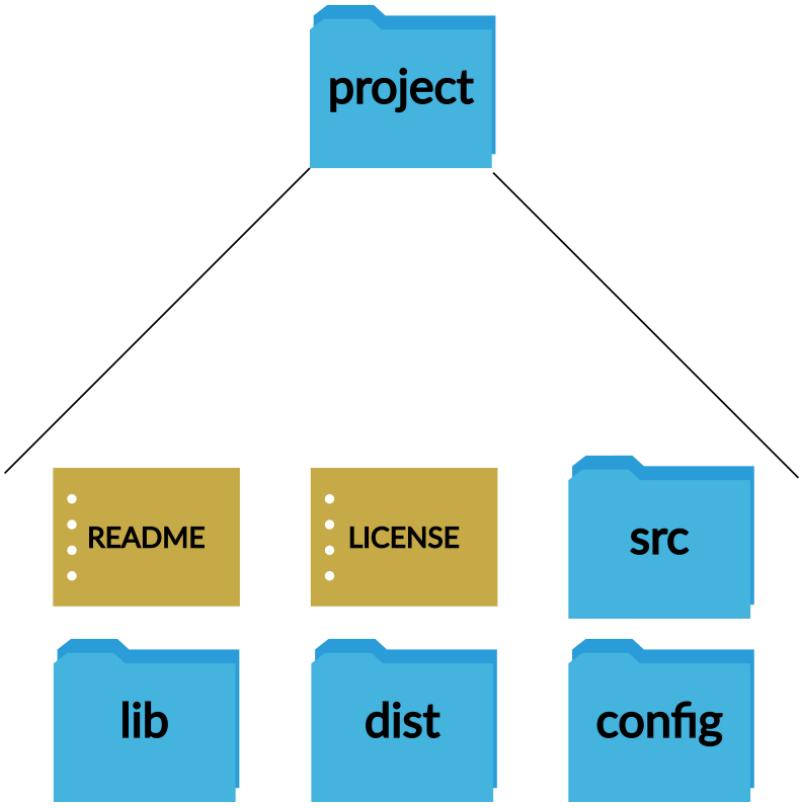
- Overview
  - Why Git
  - Git Concepts
    - Commits and Branches
    - Working with remotes



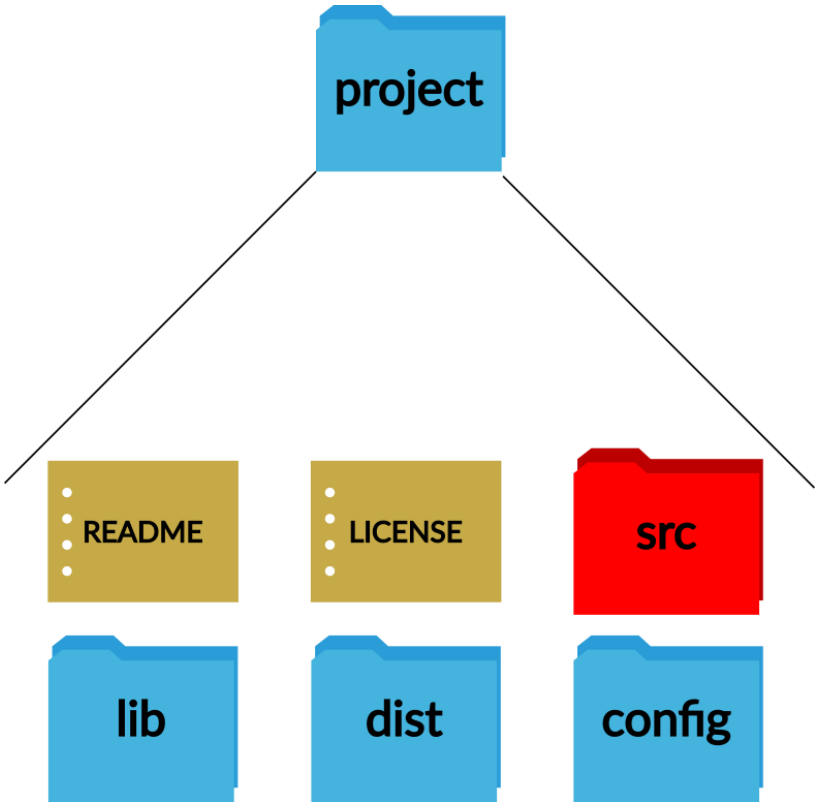
Why Git

- You have a project folder on your computer

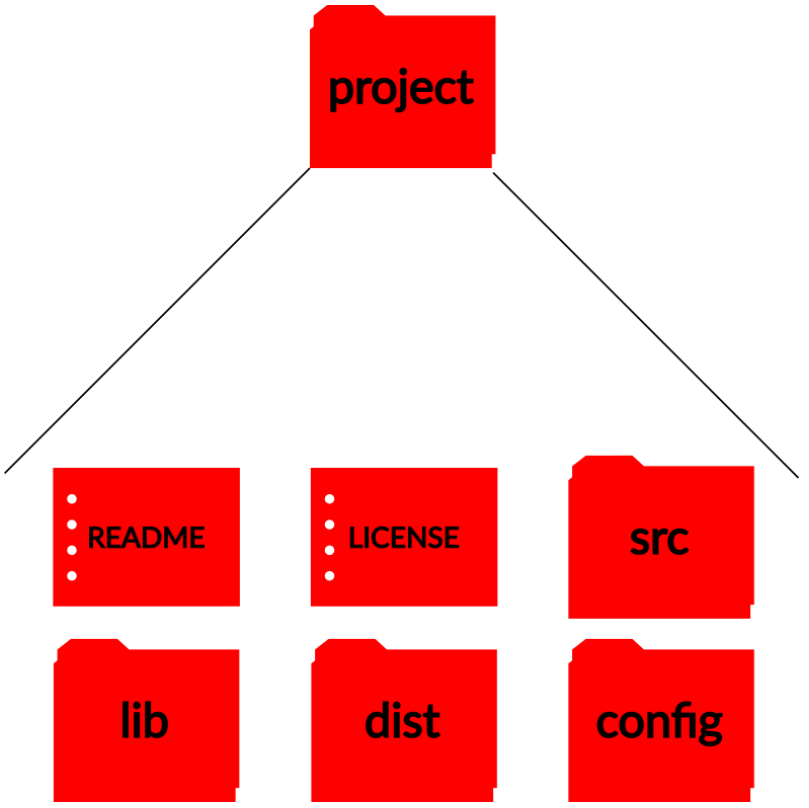
- Inside are the files for your project
  - Source code
  - Libraries
  - Built code
  - Configuration files
  - Assets



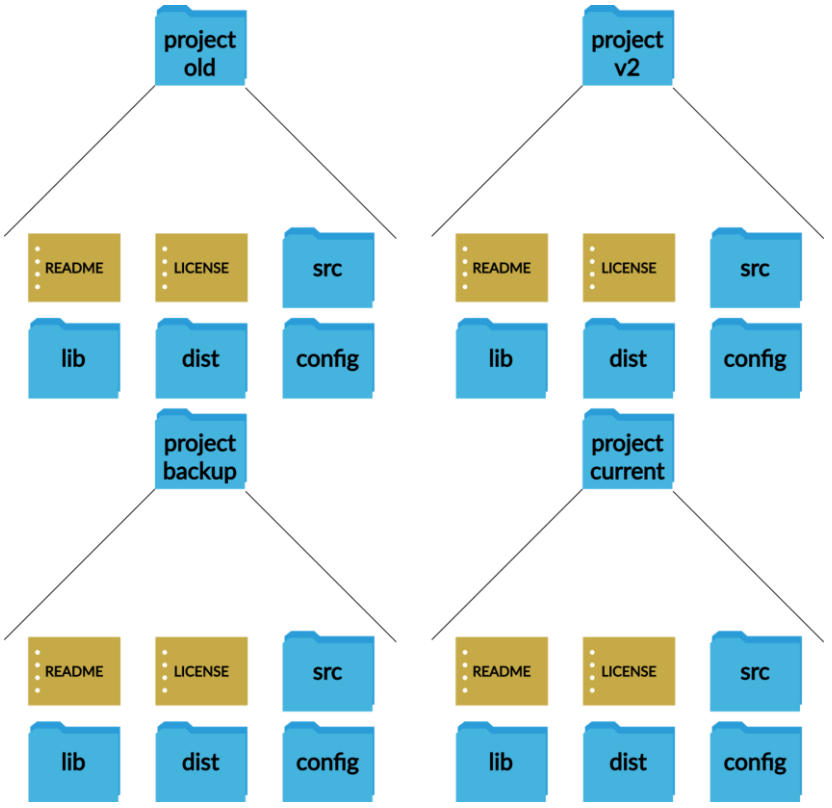
- What happens if
  - You delete a file?



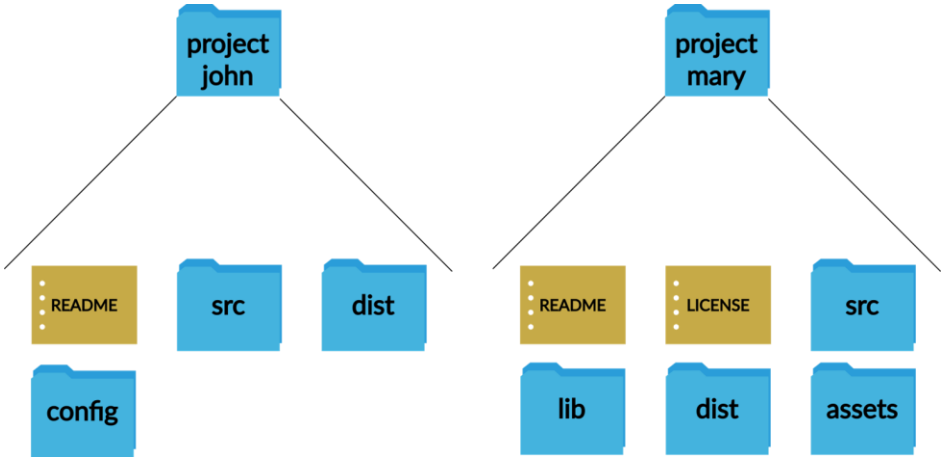
- What happens if
  - You delete a file?
  - You delete the whole folder?



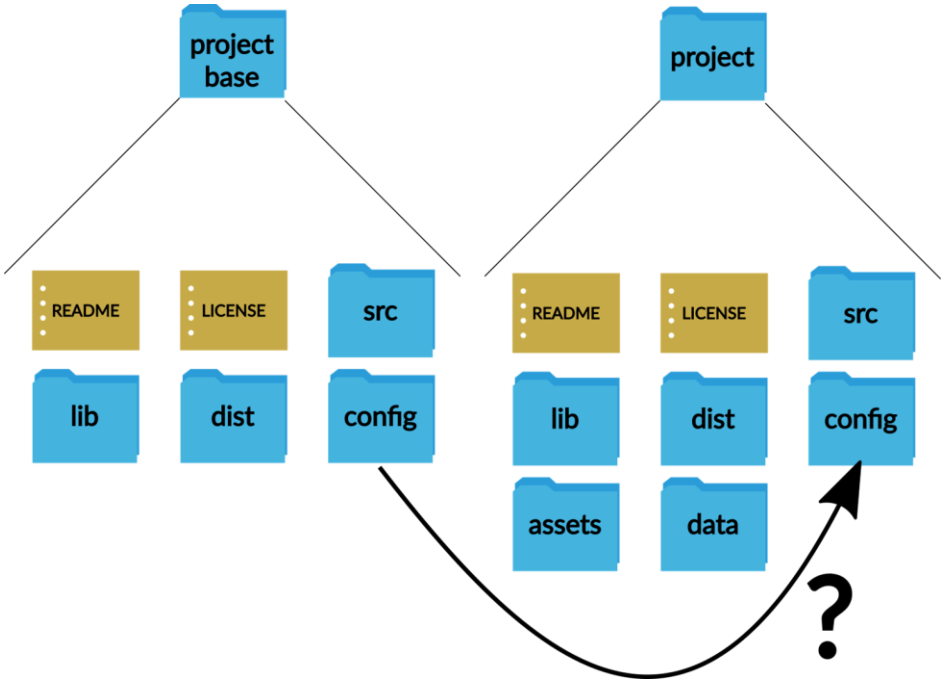
- What happens if
  - You delete a file?
  - You delete the whole folder?
  - You backup the folder?



- What happens if
  - You delete a file?
  - You delete the whole folder?
  - You backup the folder?
  - A team works on the same project?



- What happens if
  - You delete a file?
  - You delete the whole folder?
  - You backup the folder?
  - A team works on the same project?
  - You adapt a template project but want to apply updates?



- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

new version of your product every time the main version is updated

And do all this with freely available tools and services

- What happens if
  - You want to **backup a team project** that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

- You use git to
  - create a remote repository which each team member clones

- What happens if
  - You want to backup a team project that **adapts an existing project,** and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

- You use git to
  - create a remote repository which each team member clones
  - which forks another repository

- What happens if
  - You want to backup a team project that adapts an existing project, and then **incorporate updates to the base project** while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

- You use git to
  - create a remote repository which each team member clones
  - which forks another repository
  - that you keep as a remote and merge changes from

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while **simultaneously developing new features**
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

- You use git to
  - create a remote repository which each team member clones
  - which forks another repository
  - that you keep as a remote and merge changes from
  - Your repository has multiple branches

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a **central server that regularly runs tests on everyone's contributions**
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

- You use git to
  - create a remote repository which each team member clones
  - which forks another repository
  - that you keep as a remote and merge changes from
  - Your repository has multiple branches
  - You add a Continuous Integration tool to your git server that automatically builds and tests the code on push

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and **deploy a new version of your product** every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and automatically running tests and automatically publishing them

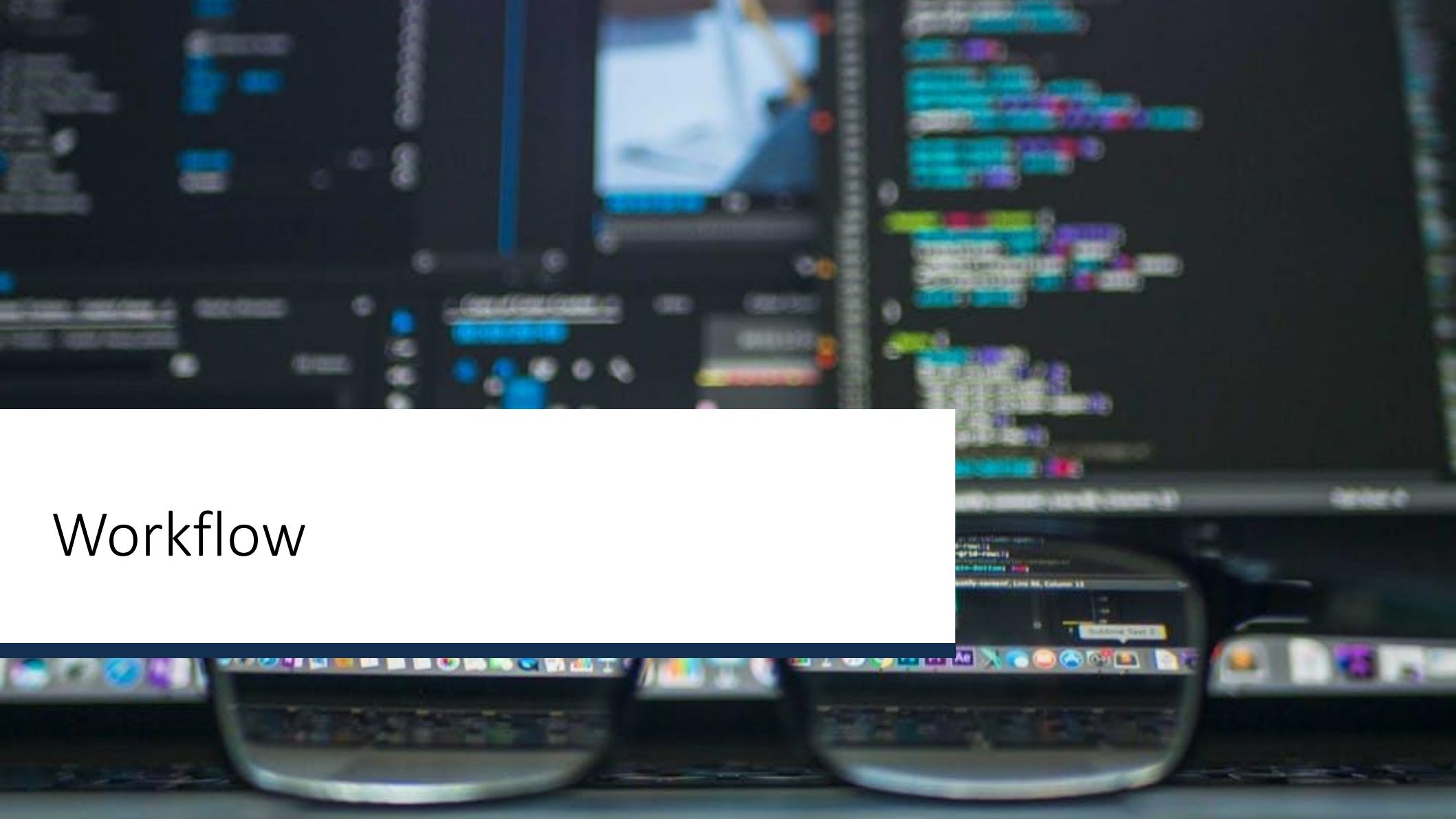
- You use git to
  - create a remote repository which each team member clones
  - which forks another repository
  - that you keep as a remote and merge changes from
  - Your repository has multiple branches
  - You add a Continuous Integration tool to your git server that automatically builds and tests the code on push
  - You get your CI tool fire a webhook to a service that builds and deploys your code

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then **split your project into multiple libraries that can be reused across multiple projects while keeping them updated** and automatically running tests and automatically publishing them

- You use git to
  - create a remote repository which each team member clones
  - which forks another repository
  - that you keep as a remote and merge changes from
  - Your repository has multiple branches
  - You add a Continuous Integration tool to your git server that automatically builds and tests the code on push
  - You get your CI tool fire a webhook to a service that builds and deploys your code
  - You turn your libraries into submodules

- What happens if
  - You want to backup a team project that adapts an existing project, and then incorporate updates to the base project while simultaneously developing new features
  - **AND** have a central server that regularly runs tests on everyone's contributions
  - **AND** automatically build and deploy a new version of your product every time the main version is updated
  - **AND** then split your project into multiple libraries that can be reused across multiple projects while keeping them updated and **automatically running tests and automatically publishing them**

- You use git to
  - create a remote repository which each team member clones
  - which forks another repository
  - that you keep as a remote and merge changes from
  - Your repository has multiple branches
  - You add a Continuous Integration tool to your git server that automatically builds and tests the code on push
  - You get your CI tool fire a webhook to a service that builds and deploys your code
  - You turn your libraries into submodules
  - That integrate with CI tools and automated publishing workflows

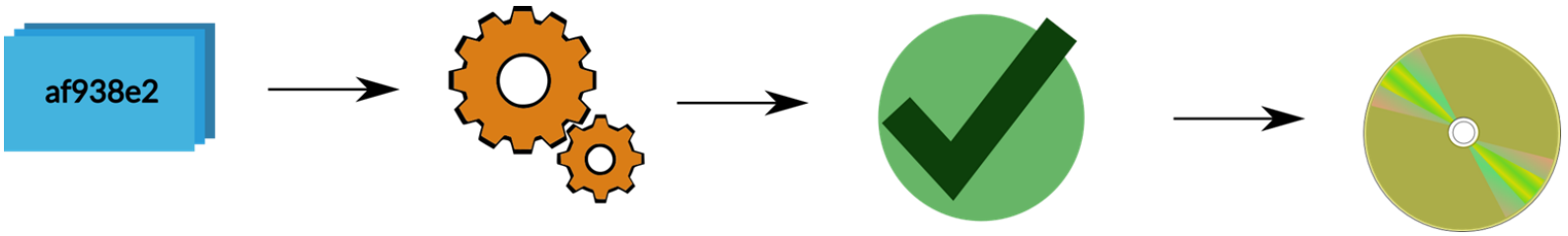


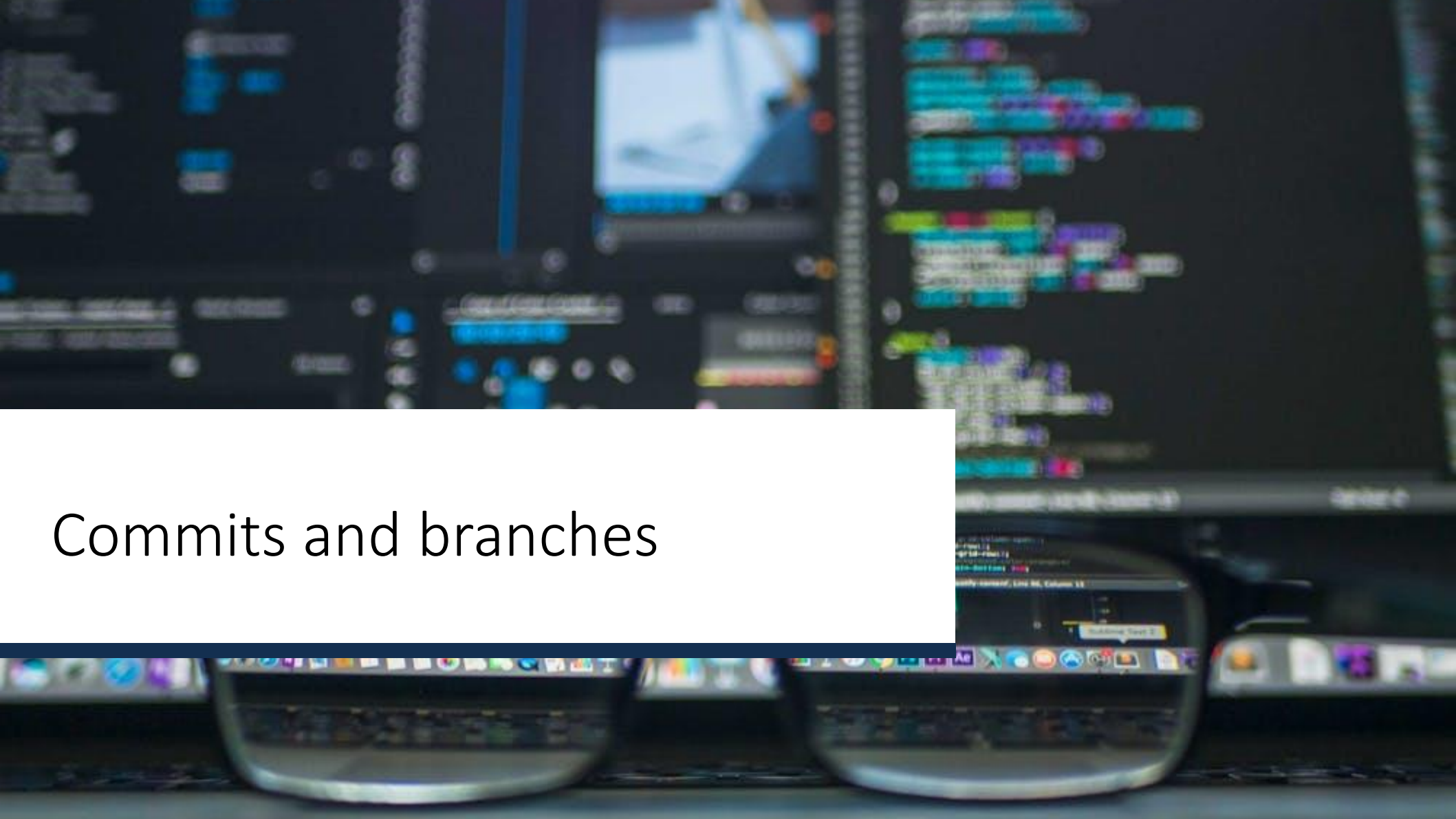
Workflow

Workflow

- Good workflows make development easier
- Bad workflow leads to bad projects
  - Poor code quality
  - Cutting corners
  - Slow updates
  - Integration hell

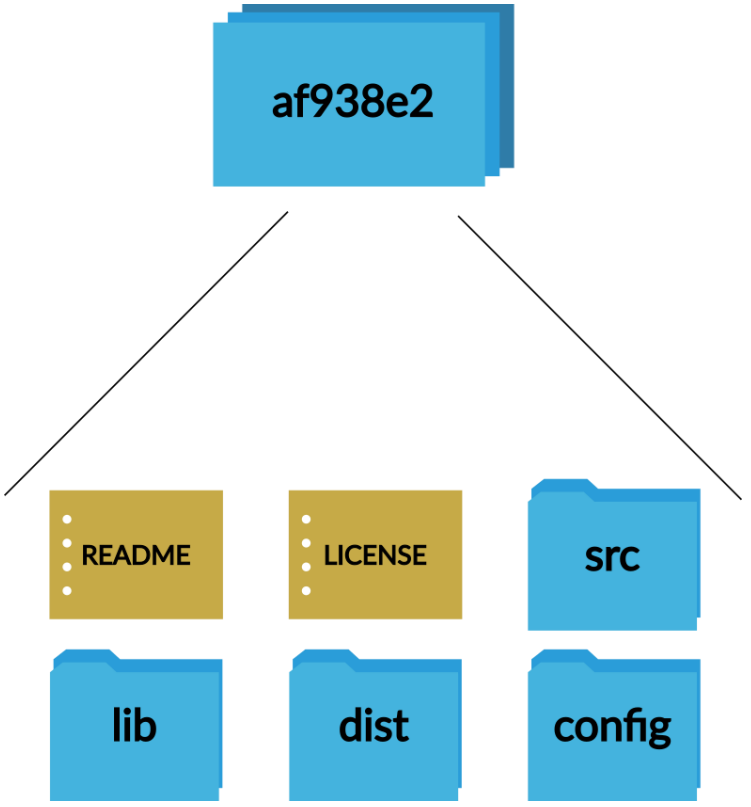
Continuous Delivery



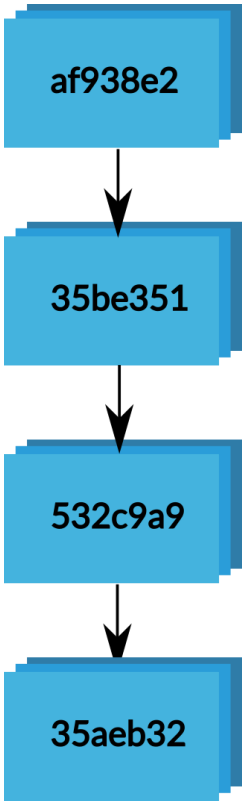


Commits and branches

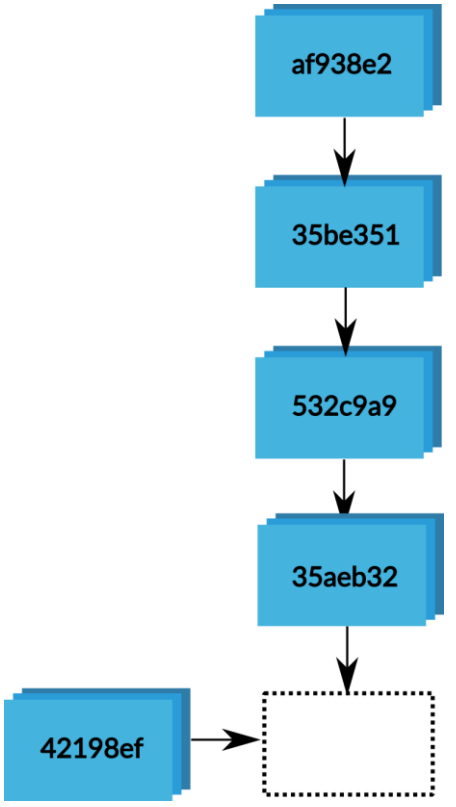
- Commit (noun)
  - A snapshot of your project



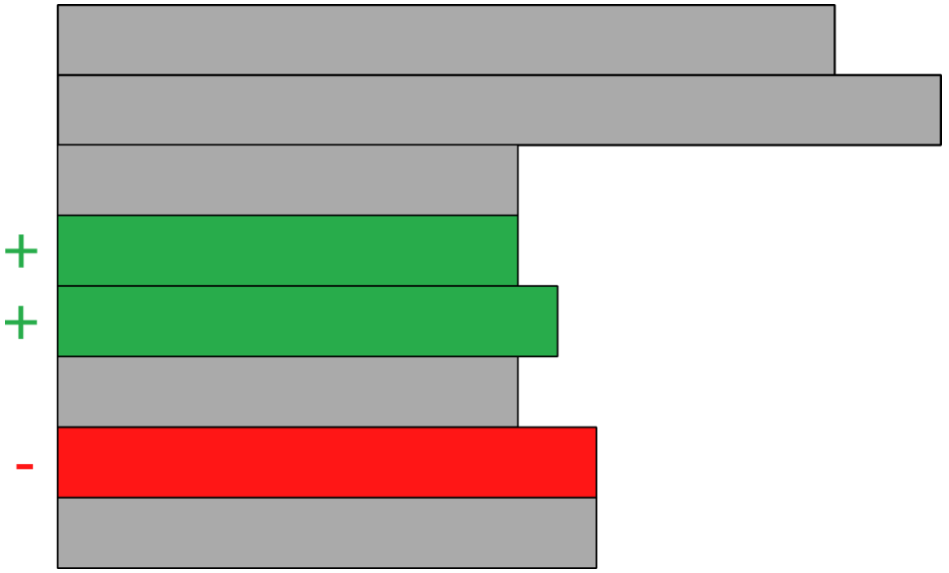
- Branch (noun)
  - A sequence of commits
  - Shows how your project has changed over time



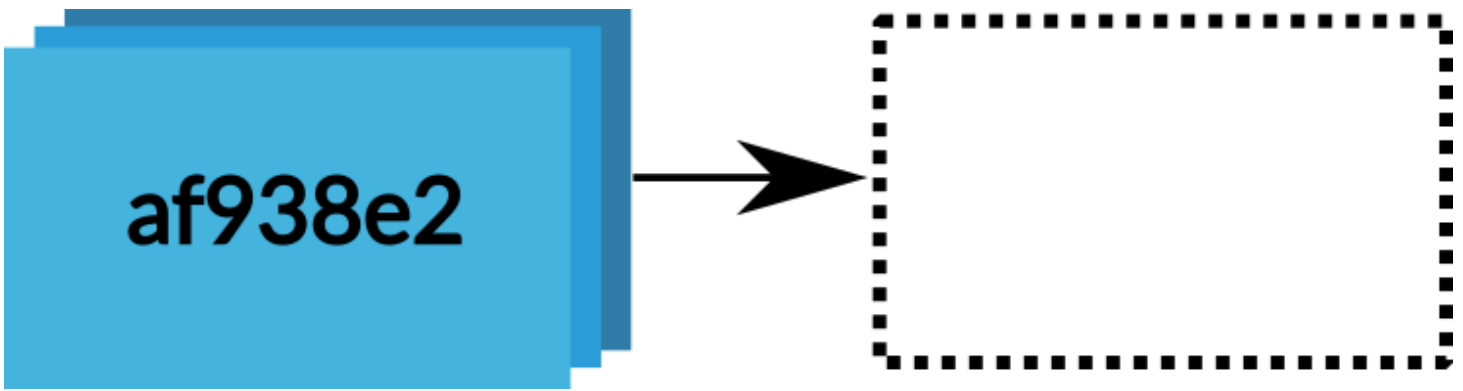
- Commit (verb)
  - Take a snapshot of your project and add it onto the end of the current branch



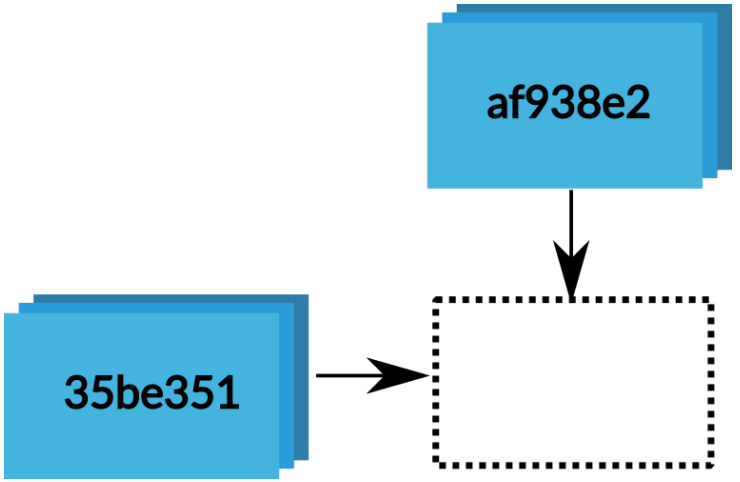
- Commits
  - Commits are calculated line-by-line
  - A commit stores just the changes that have been made since the last commit



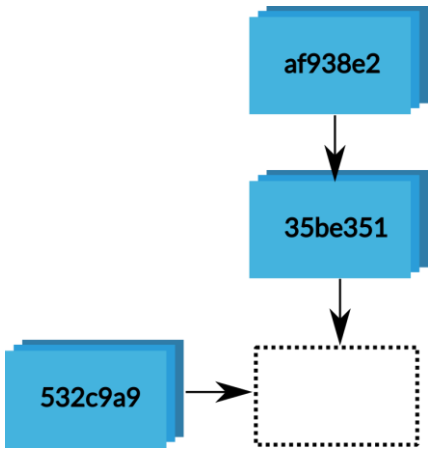
- Simple workflow example
  - Create a new repository and make an initial commit



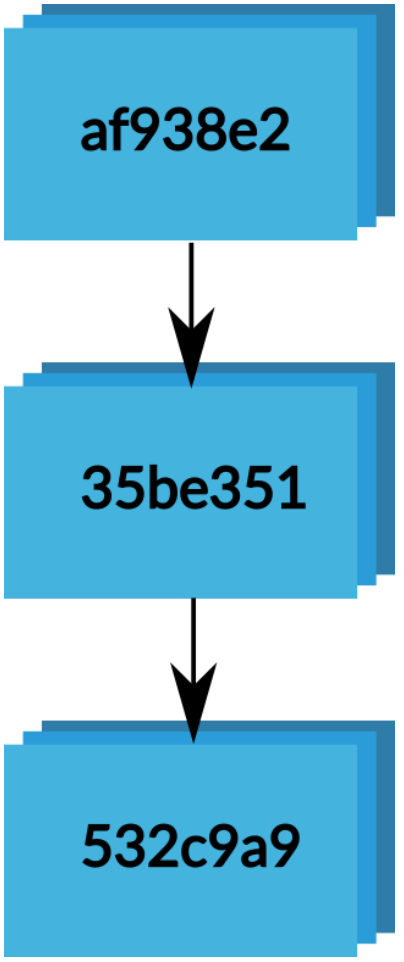
- Simple workflow example
  - Create a new repository and make an initial commit
  - Do some work, then commit it



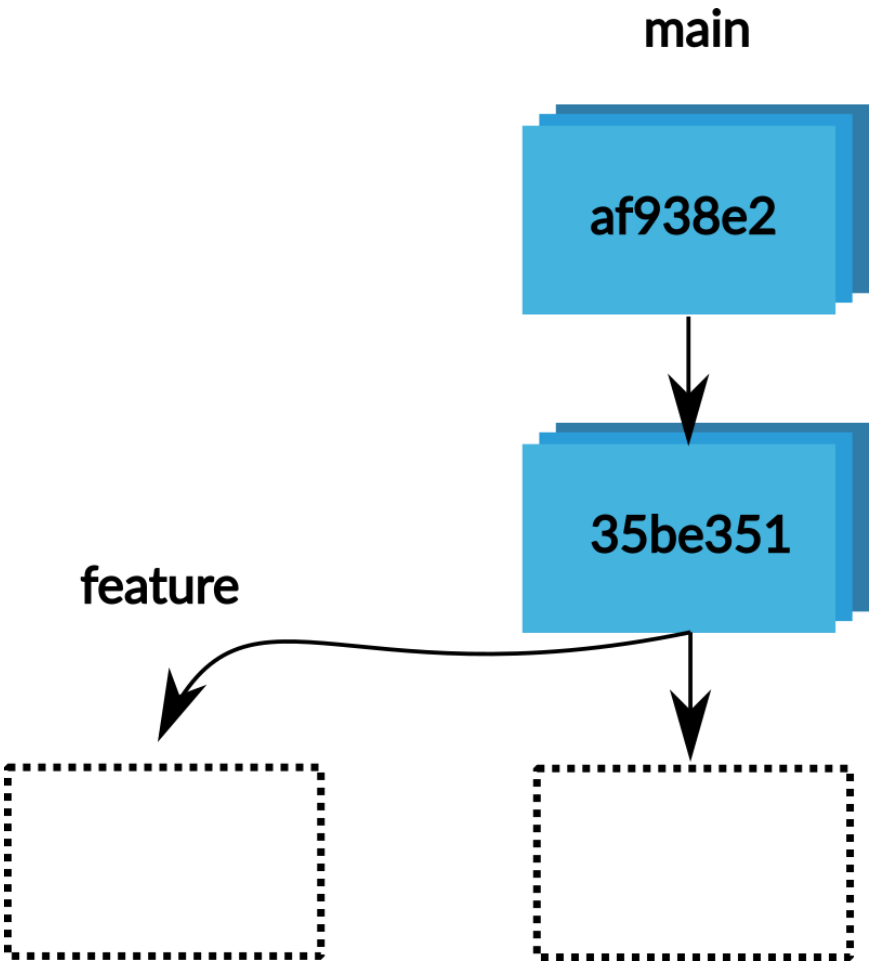
- Simple workflow example
  - Create a new repository and make an initial commit
  - Do some work, then commit it
  - Do some more work, then commit it



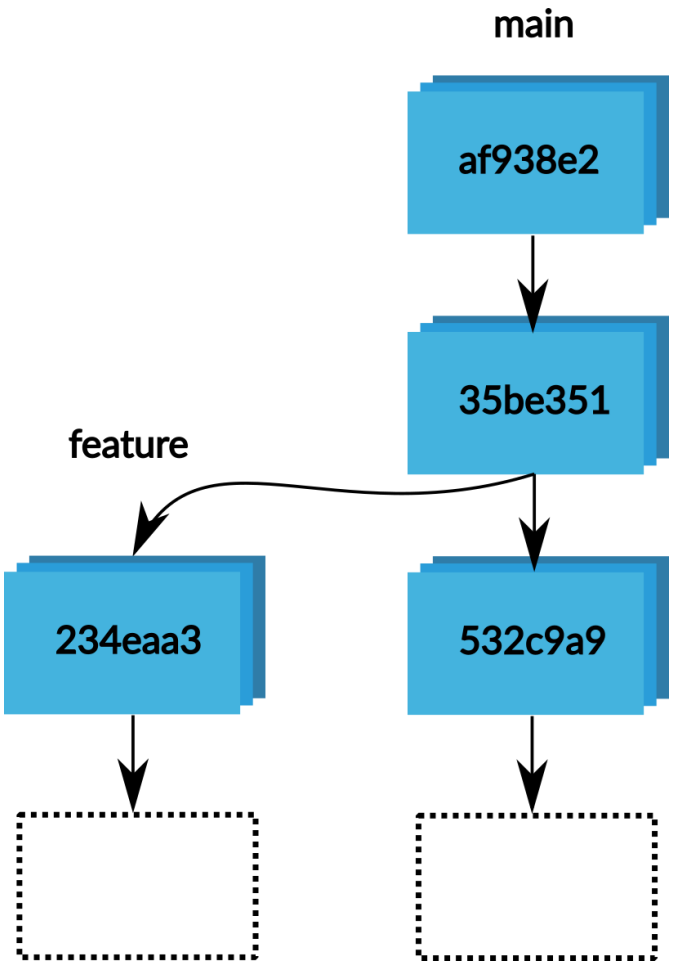
- Simple workflow example
  - Now you've got a branch with
    - The current state of your project
    - All changes made to your project



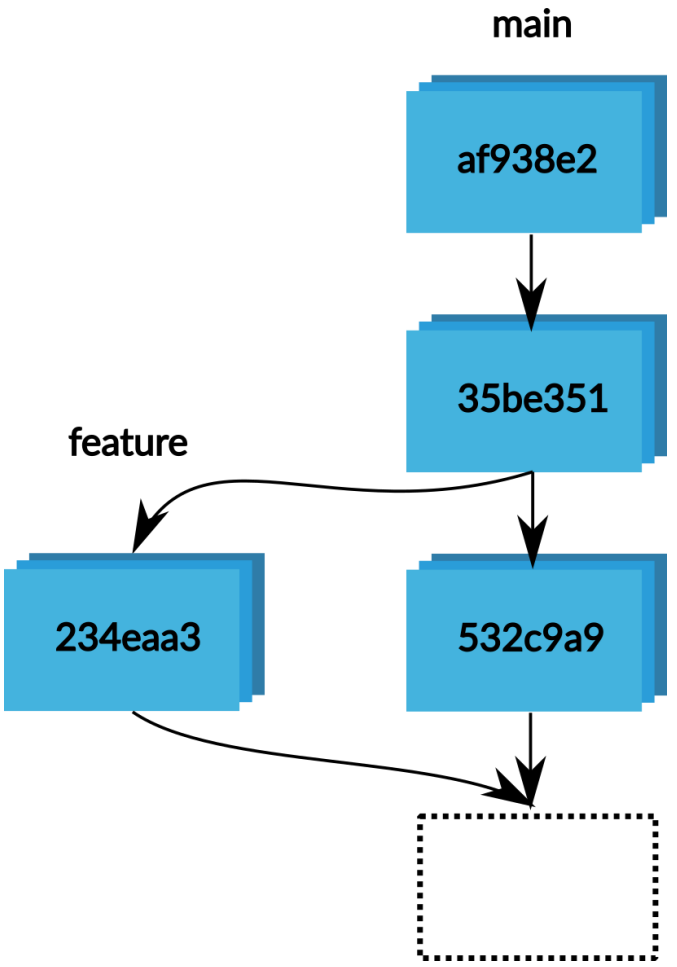
- Branch (verb)
  - Take a branch and split a new one off it
  - Now you've got two branches



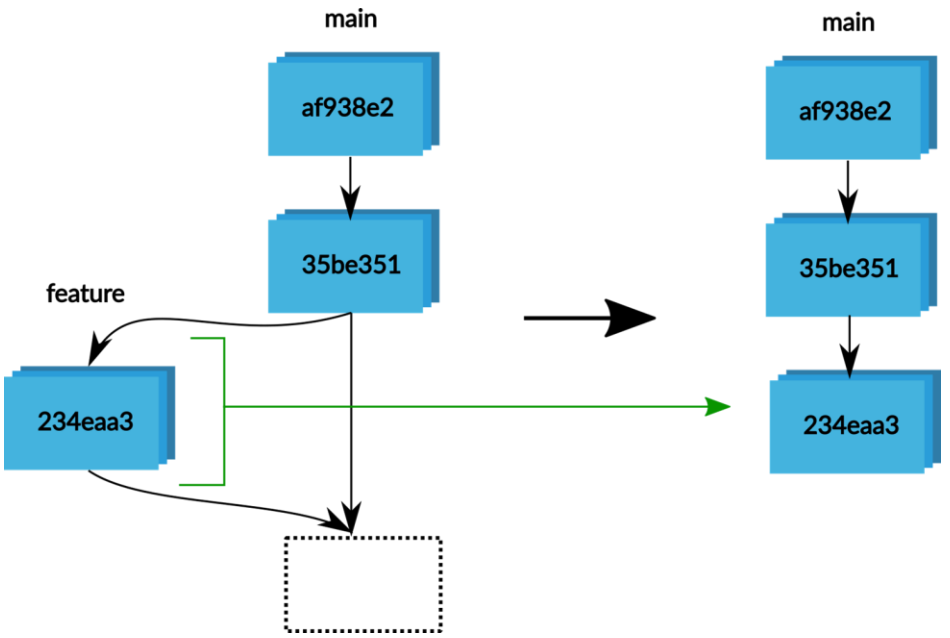
- Multiple branches
  - You can add commits to each branch
  - This gives you two different versions of your project



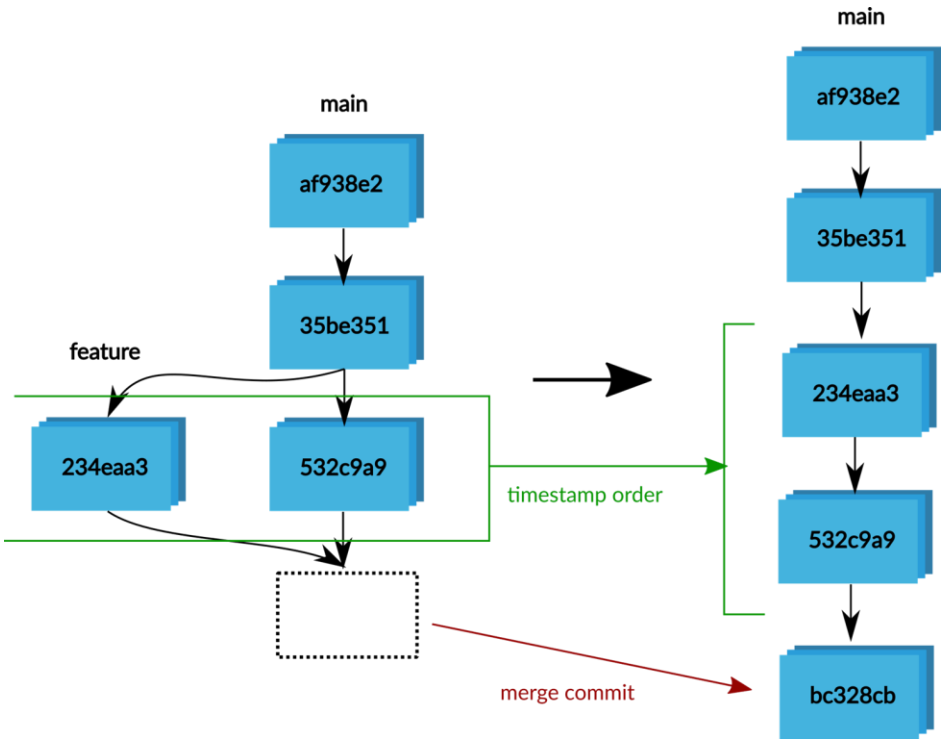
- Merge
  - Combine the changes made in two branches



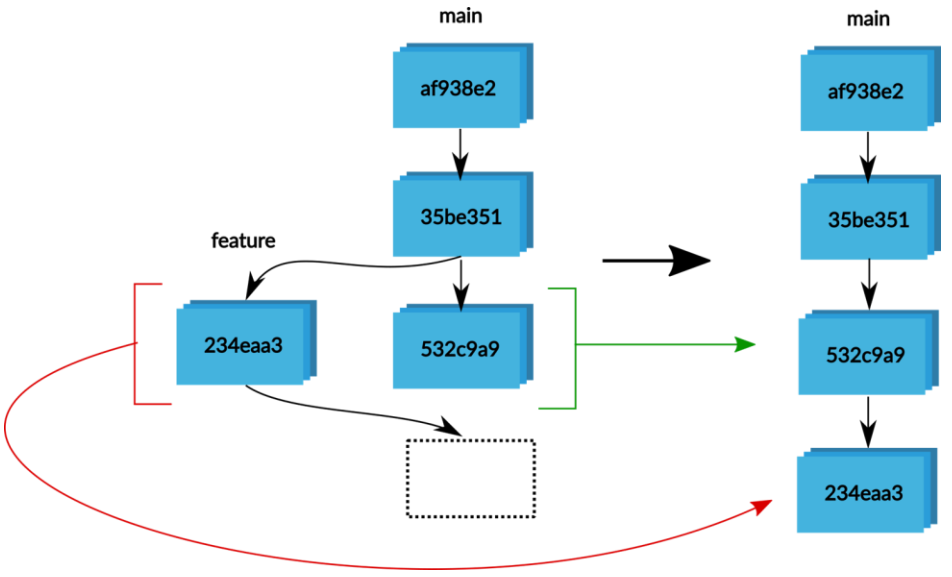
- Merge (Fast Forward)
  - There are only commits on the branch to merge in
  - These commits can be added on to the end of the main branch



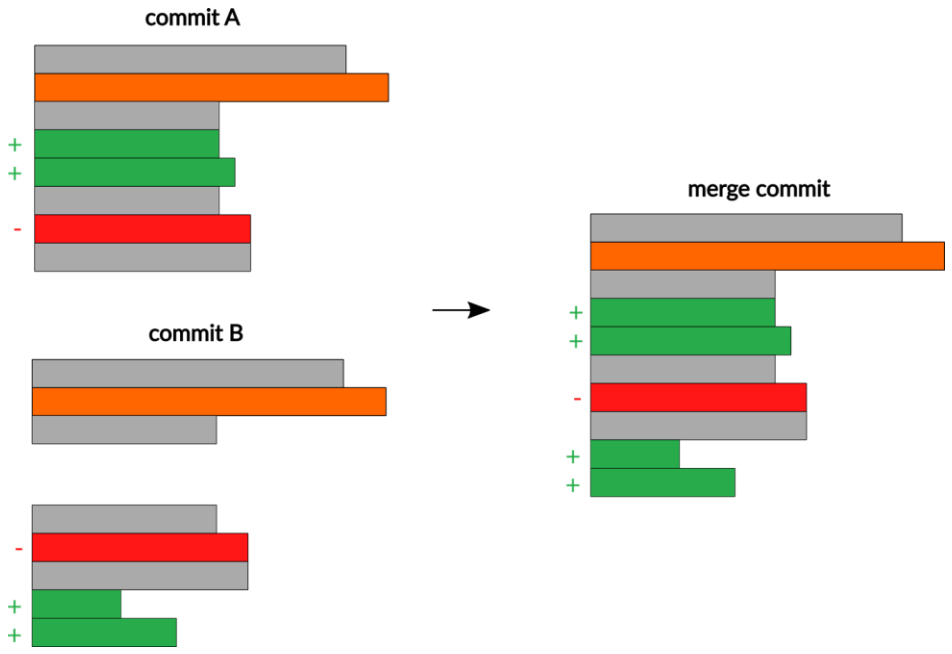
- Merge (Standard Merge)
  - Adds commits in timestamp order
  - Create a new commit for the merge



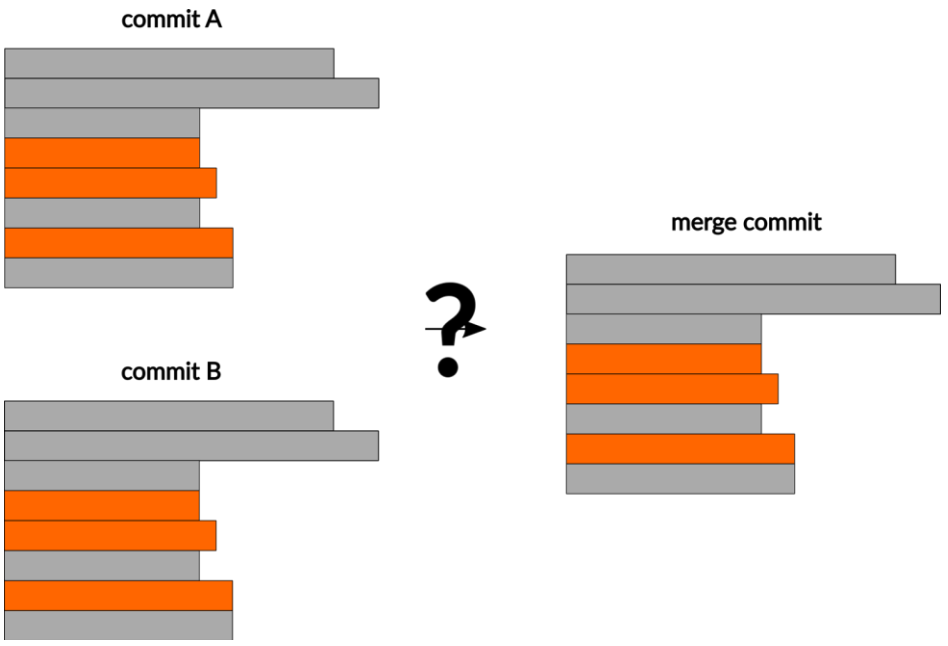
- Merge (Rebase)
  - Replay the feature branch on top of the new commits on the main branch



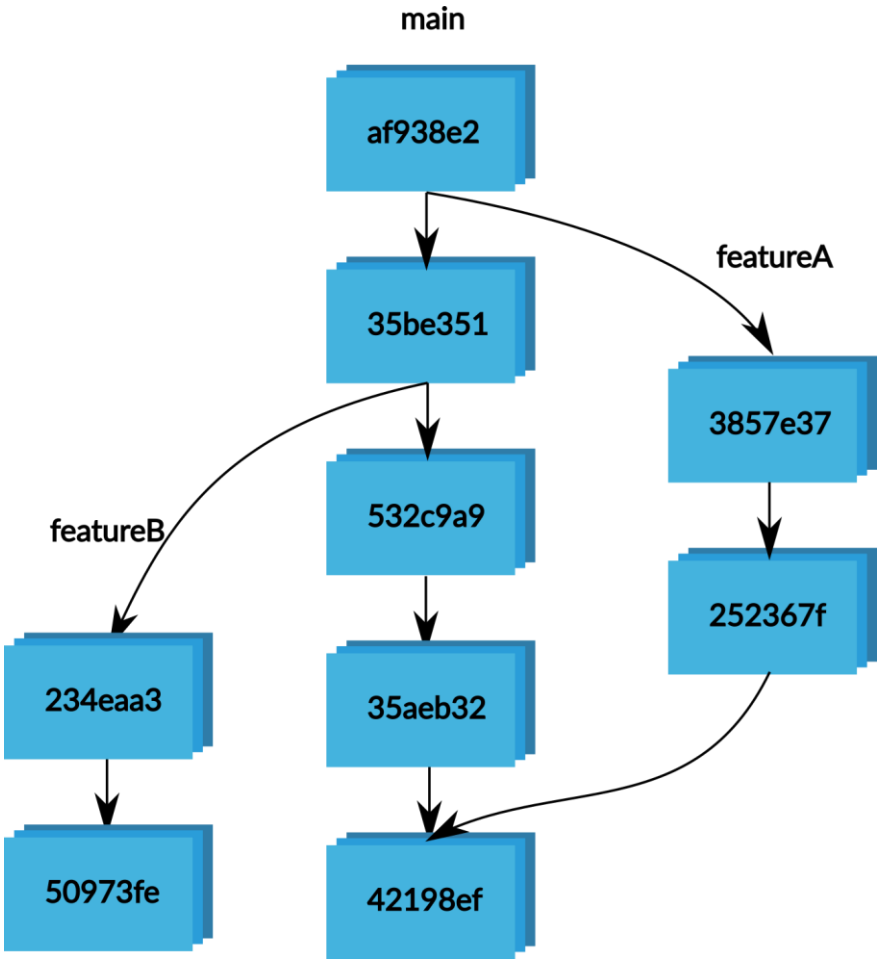
- Resolving a merge
  - Git merges line-by-line
  - If the branches change different lines everything resolves automatically



- Resolving a merge
  - If both branches change the same line, you need to resolve the merge manually
  - Git will edit the files with the options. You delete the appropriate section and commit the merge

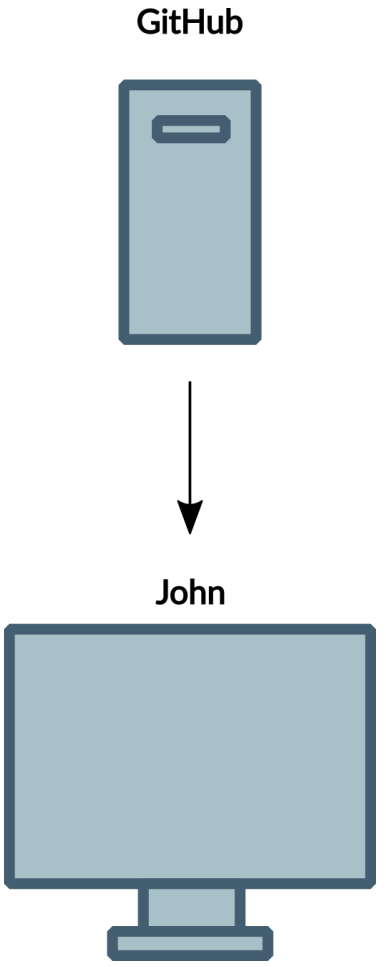


- Repository (noun)
  - Everything git keeps track of for your project
  - A collection of branches

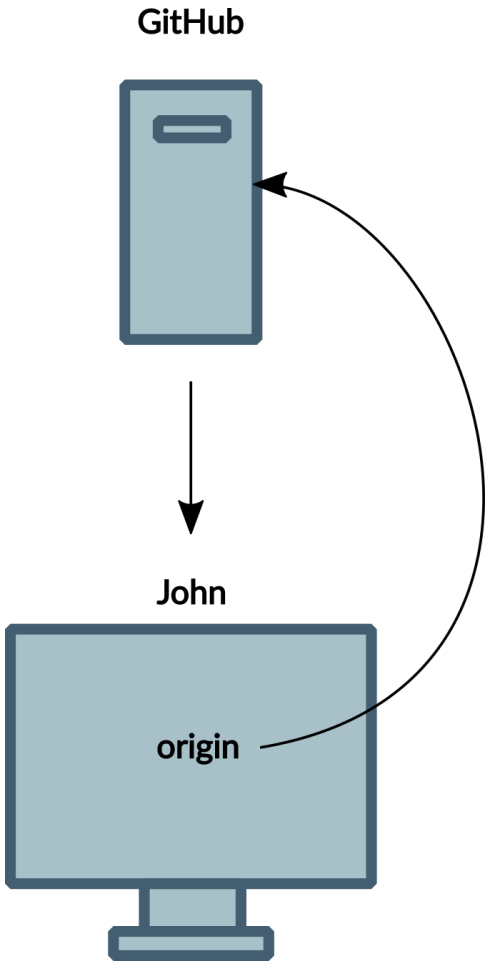


Working with Remotes

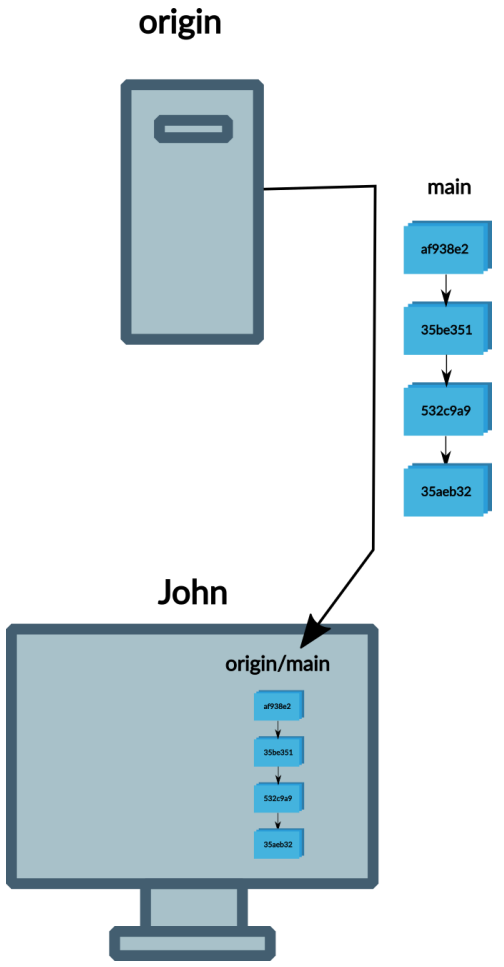
- Clone (verb)
  - Make a copy of a repository held on a git server



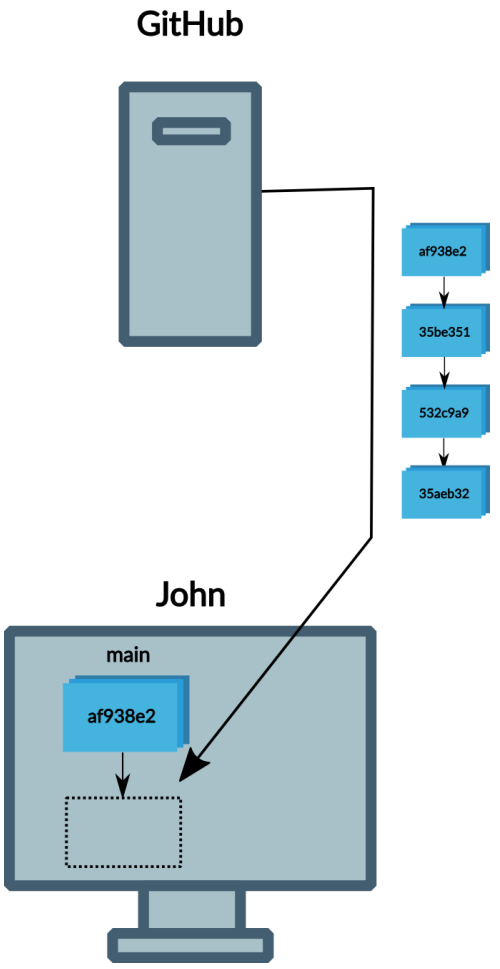
- Remote (noun)
  - A reference in your git repo to a repository elsewhere
  - When you clone a repo, it creates a remote called origin



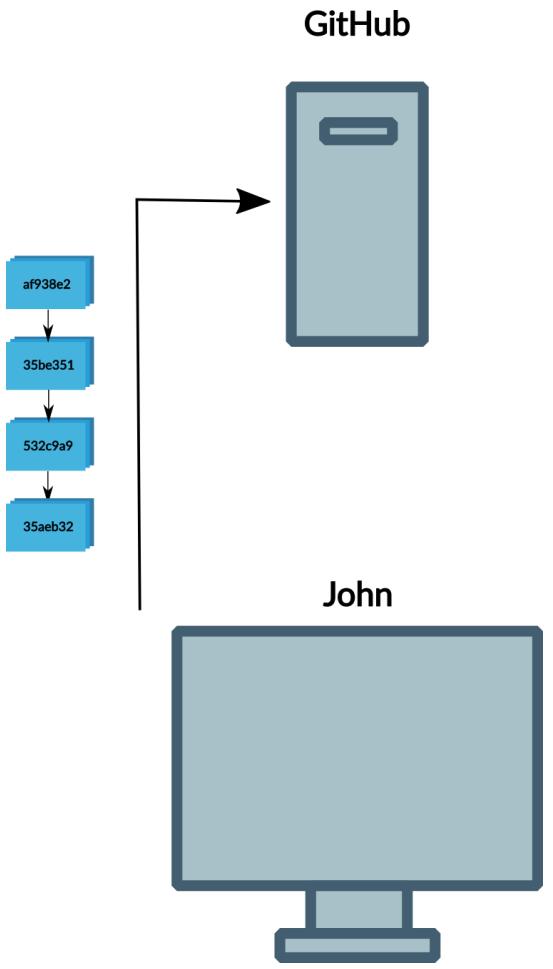
- Fetch (verb)
  - Download commits made on a remote branch
  - Fetching branch main from origin would put result in branch origin/main



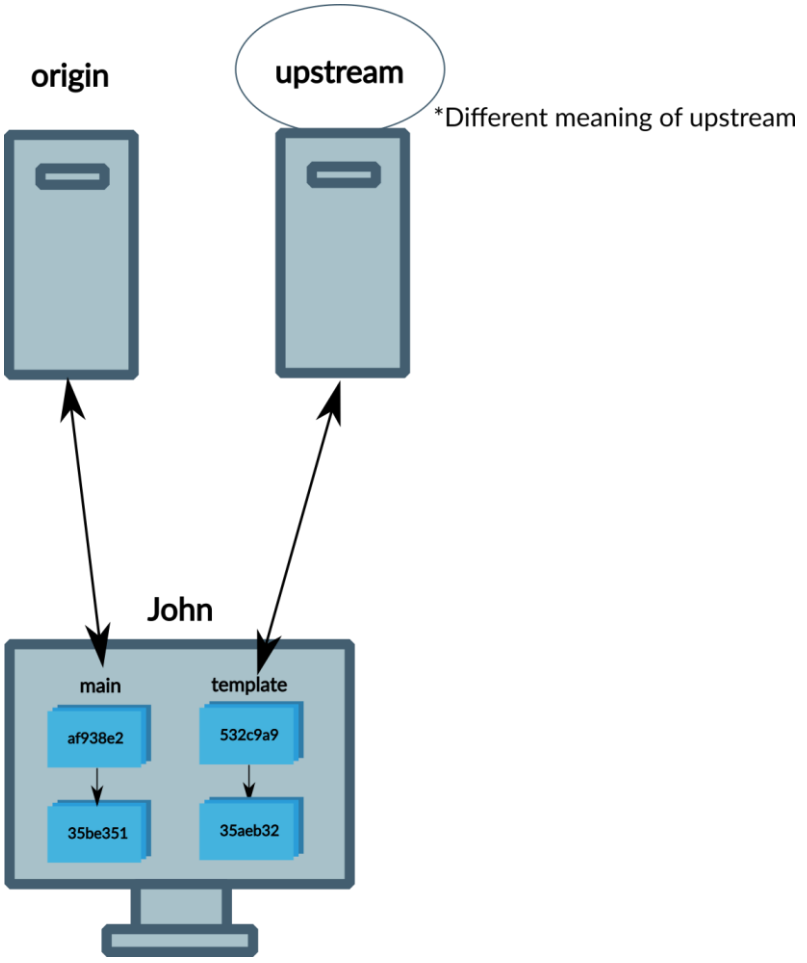
- Pull (verb)
  - Fetch **and merge** commits made on a remote branch



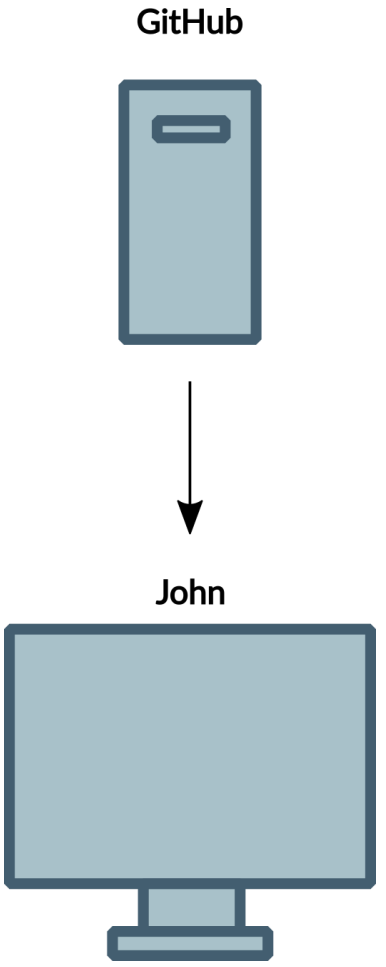
- Push (verb)
  - Upload new commits in the local repository to a remote branch



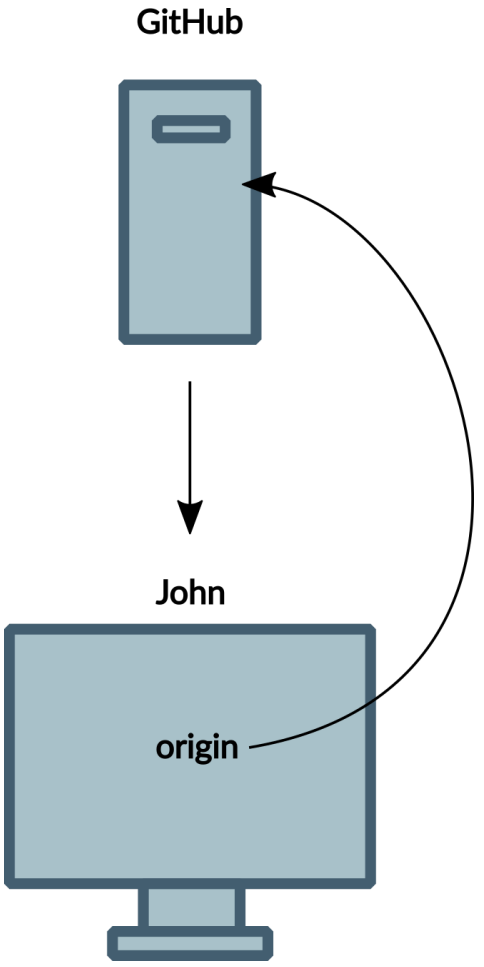
- Upstream (noun)
  - Which remote branch our branch should fetch/pull from and push to
  - Different branches may have upstreams on different remotes



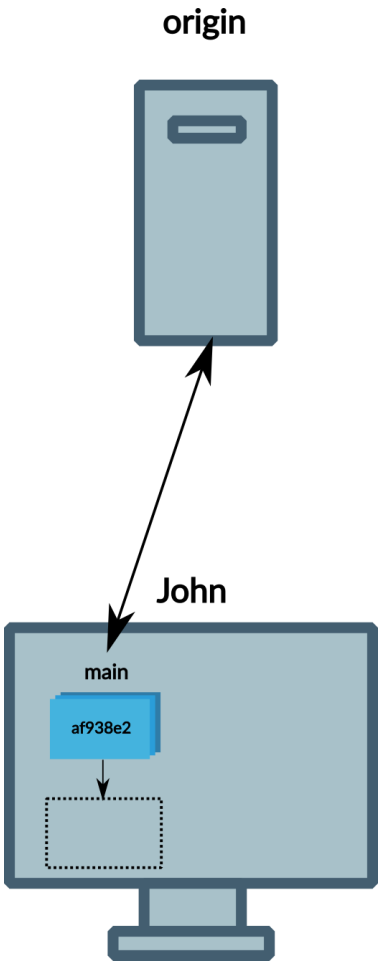
- Example workflow
  - Clone a repository



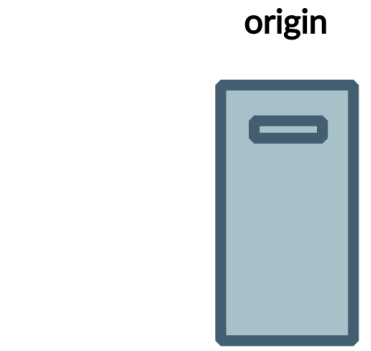
- Example workflow
  - Clone a repository
    - Remote **origin** is set automatically

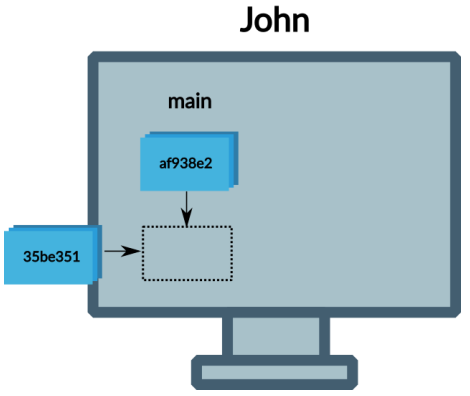


- Example workflow
  - Clone a repository
    - Remote **origin** is set automatically
  - Set upstream for branch **main**

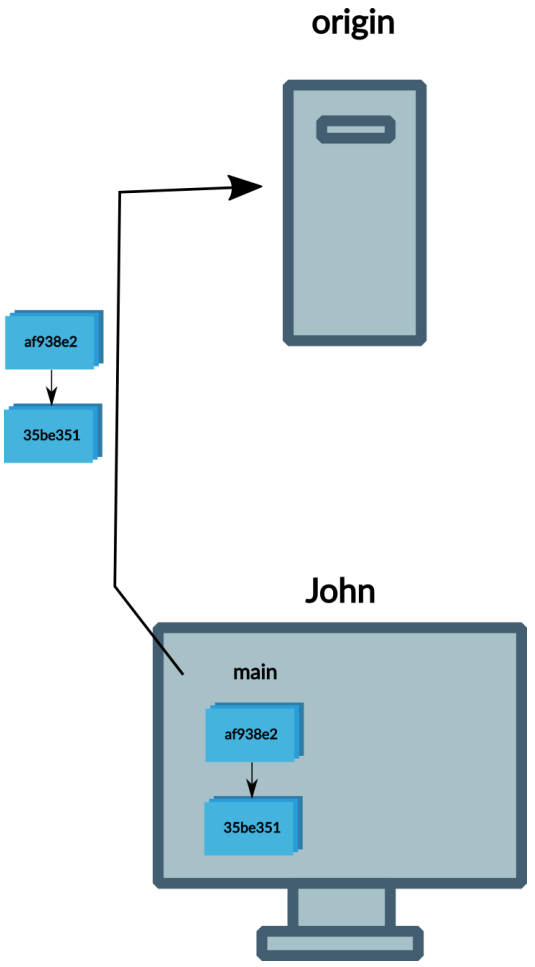


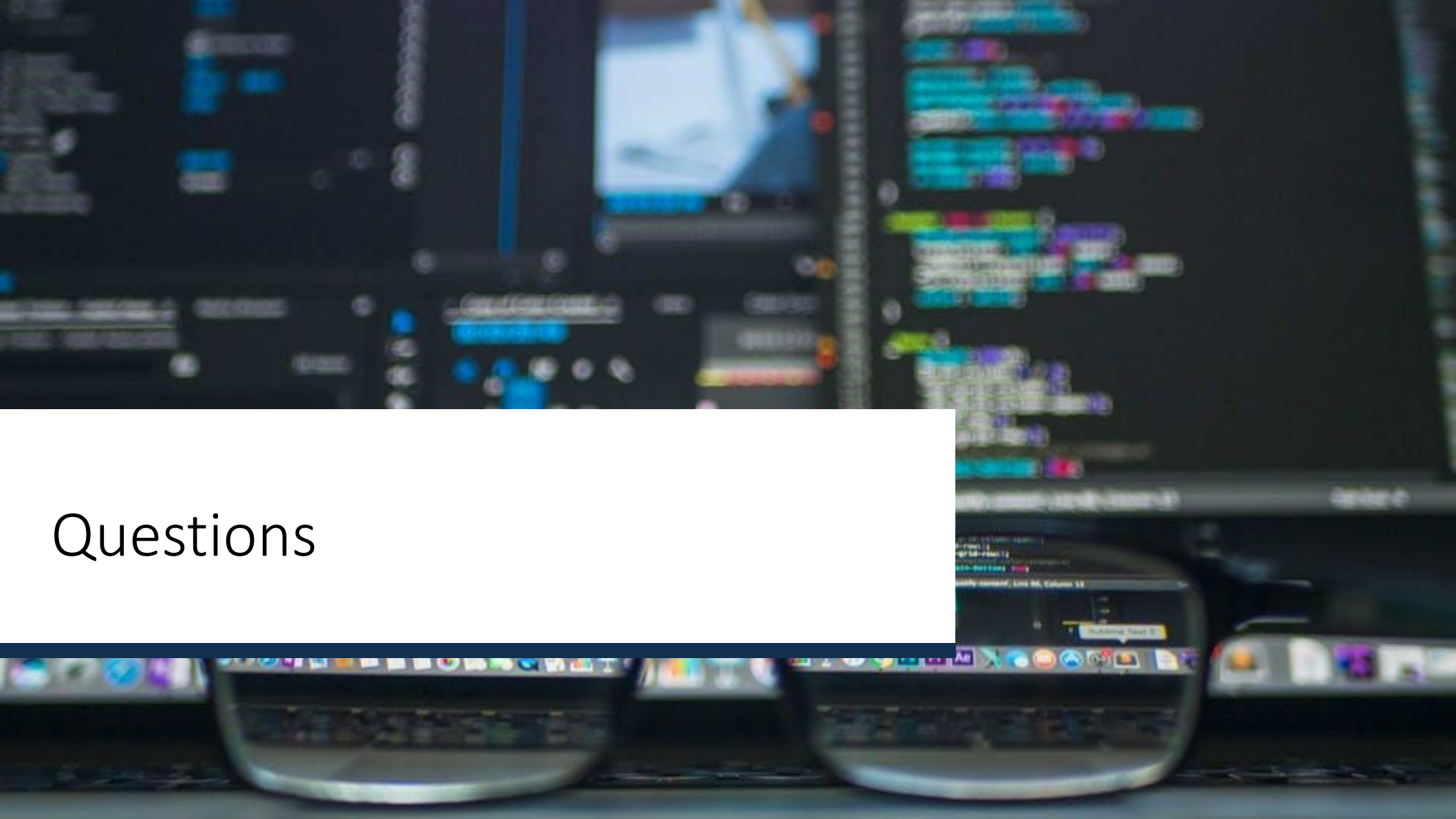
- Example workflow
  - Clone a repository
    - Remote **origin** is set automatically
  - Set upstream for branch **main**
  - Commit changes on branch **main**





- Example workflow
  - Clone a repository
    - Remote **origin** is set automatically
  - Set upstream for branch **main**
  - Commit changes on branch **main**
  - Push commits to remote branch





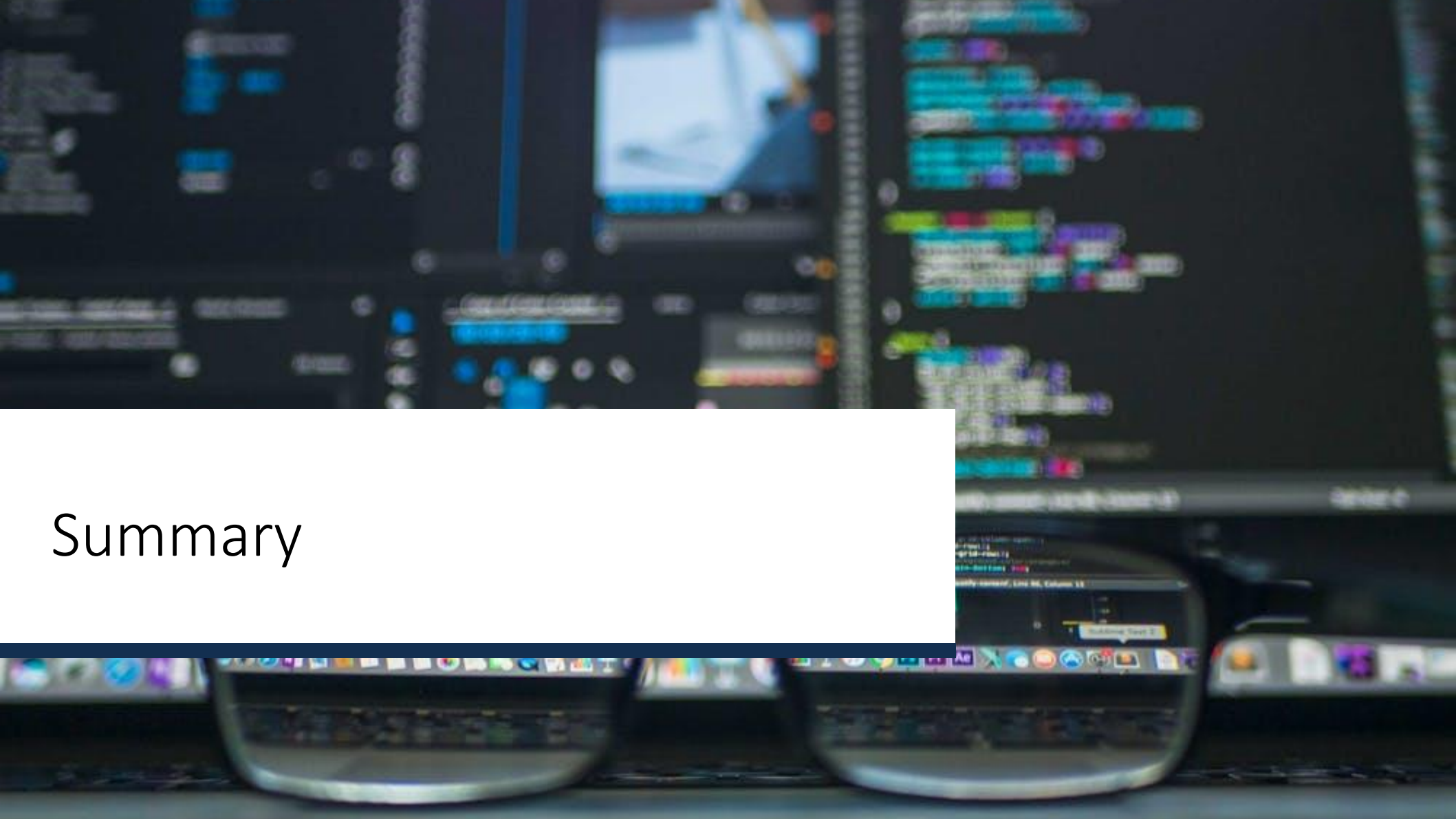
Questions

- **Question:** What would be the steps involved in the following:
    1. Create a repository locally
    2. Add content to it
    3. Create an empty repository on GitHub
    4. Push your content to the empty repository

- **Answer:** What would be the steps involved in the following:
  - Create a repository locally
    - git init
  - Add content to it
    - git commit
  - Create an empty repository on GitHub
    - git remote add
    - git push --set-upstream
  - Push your content to the empty repository
    - git push

- **Question:** What would be the steps involved in the following:
    1. Copy an existing repo (upstream)
    2. Make a new GitHub repo for it
    3. Work on a new feature
    4. Update with changes to upstream
    5. Incorporate new feature into main branch
    6. Push to GitHub repo

- **Answer:** What would be the steps involved in the following:
  - Copy an existing repo (upstream)
    - git clone
  - Make a new GitHub repo for it
    - git remote add
  - Work on a new feature
    - git branch
    - git commit
  - Update with changes to upstream
    - git pull
  - Incorporate new feature into main branch
    - git merge
  - Push to GitHub repo
    - git push --set-upstream



Summary

Summary

- Git Concepts
  - Repository
  - Commit
  - Branch
  - Merge
- Git Commands
  - git init
  - git commit
  - git pull
  - git push
  - git remote