Dev_Together

GitHub & Source Control Basics

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Workshop Overview

3 Rounds Of:

Presentation > Questions > Exercise

Workshop Agenda - Round 1

- Source Control – What, Why, Benefits for Individuals and Teams
- What is Git? What is a commit? How to commit? Best practices when doing commits.
- Basic Git and Source Control Terms
- Basic Git Commands
- Basic Git Workflow
- Questions > Exercise

Let's Talk Source Control

@devtogethermad | devtogether.co | devtogethermad@gmail.com

What is Source Control?

Source = Your Code

Control = What + Who + When + Why Changed

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Why Source Control?

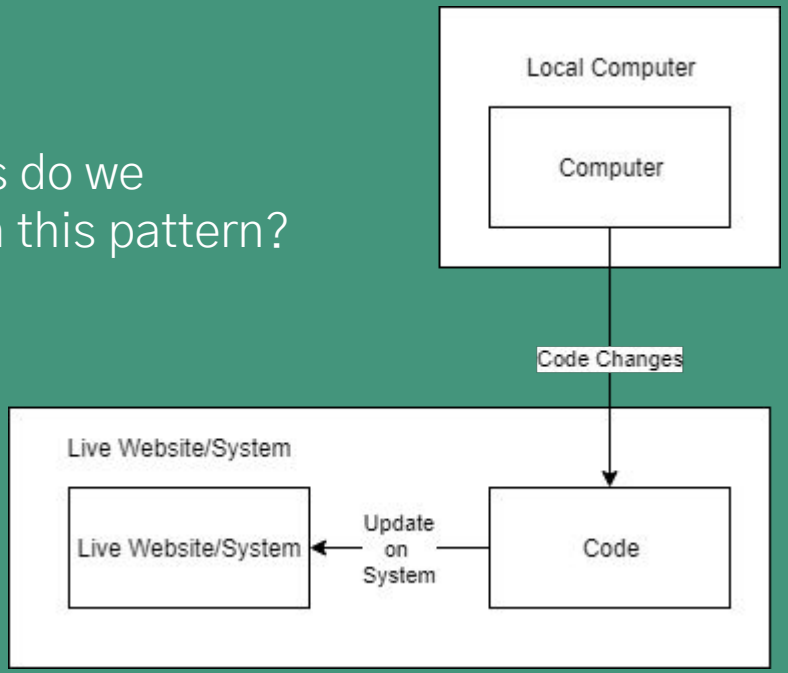
For Individuals and Teams

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Development Workflow - Individual

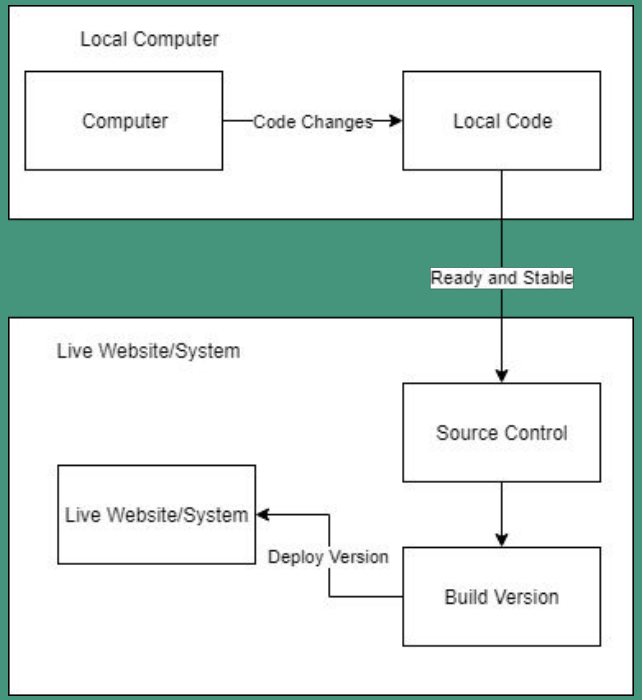
No Source Control

What problems do we encounter with this pattern?



Development Workflow - Individual

With Source Control



@devtogethermad | devtogether.co | devtogethermad@gmail.com

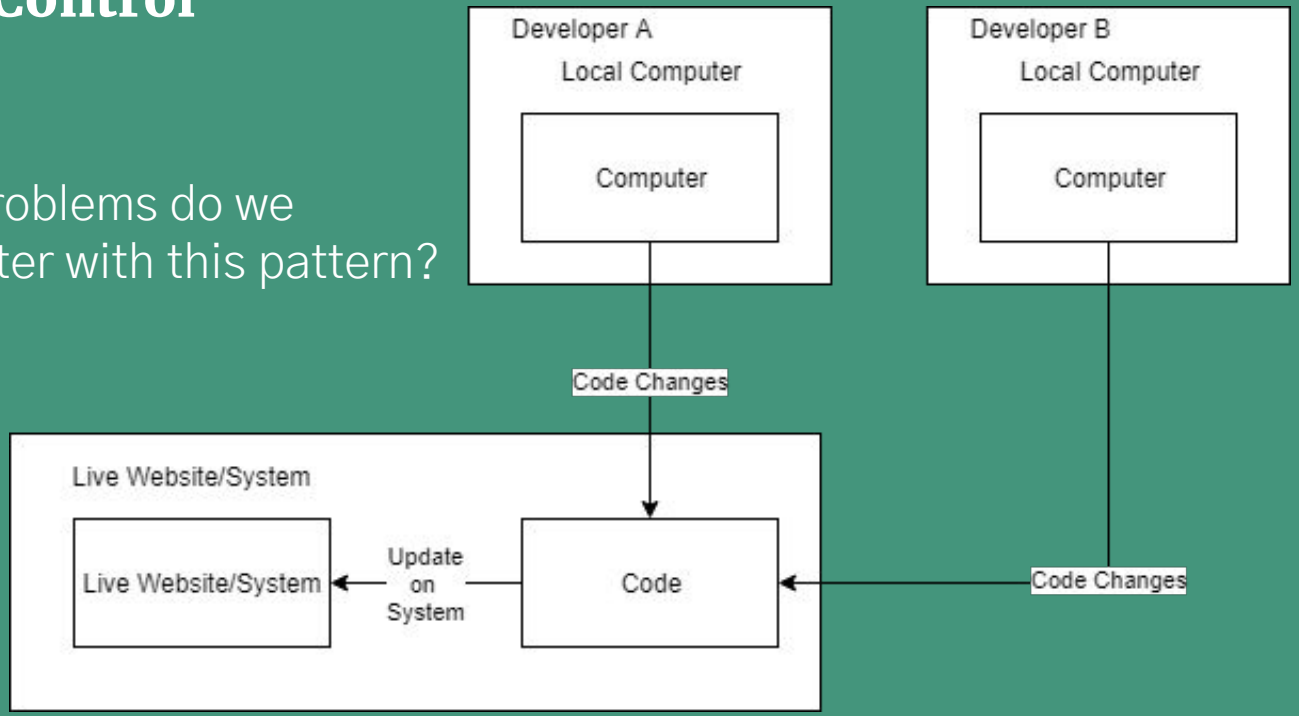
Why Source Control? For Individuals.

- Backups.
- Understand code history --
  - What, Why, When?
- Compare changes.
- Allows code stability via “branching” (more later)

Development Workflow - Team

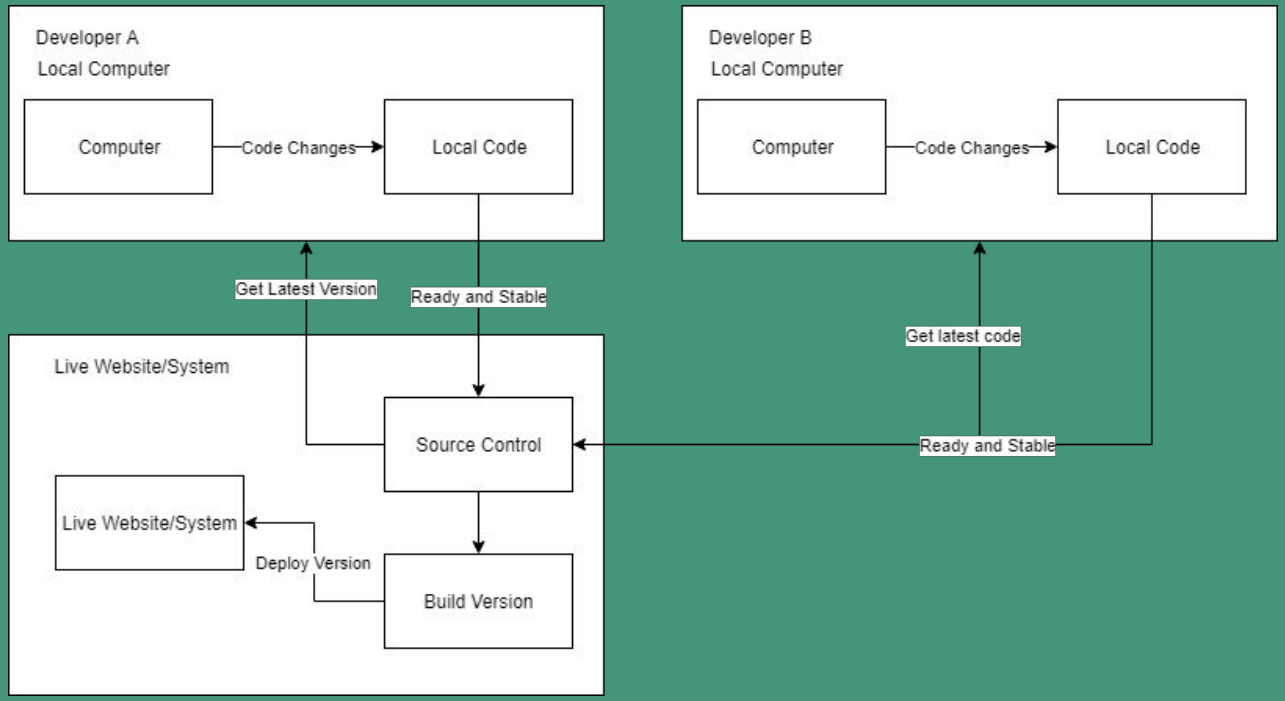
No Source Control

What problems do we encounter with this pattern?



Development Workflow - Team

With Source Control



@devtogethermad | devtogether.co | devtogethermad@gmail.com

Why Source Control? For Teams.

- Each dev can make changes without impacting other devs. (via branching)
- Versioning Benefits
  - Share changes.
  - Conflict detection.
  - Outdated version detection.

Why Source Control? For Teams.

- Understand code history --
  - What, Why, **Who**, When?
- Code versioning and deployment to various environments. (live vs non-live)
  - Deploy previous version if any issues with current version.

What is Git?

- Version Control System
  - Others – Subversion, TFVC, etc
- Code Changes = “Commit”
- Commit = Snapshot of Code

How To Commit

git commit –m “Commit message”

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Best Practices for Commits

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Best Practices for Commits

- Commit Related Changes
- Commit Often
- Be Descriptive in Commit Message
  - Answers “why” in future.

Let’s Talk GitHub

@devtogethermad | devtogether.co | devtogethermad@gmail.com

What is GitHub?

- A “hub” for Git
- Store Projects as “Repositories”
- View / Contribute to Repositories
- Public vs Private Repositories

Basic Terms

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Basic Terms

Repository (aka Repo)

- A storage for all your files/code
- Local vs Remote
  - Local – on your machine
  - Remote – on a server (e.g. GitHub)

Basic Terms

- Clone
  - Make a copy of a remote repo on your computer => Local Repo
- Index
  - “Staging” area. These will hold changes you are ready to commit.

Basic Terms

- Workspace
  - Where your changes are made.
- Push
  - Push changes from your local repo to the remote repo.

Basic Git Commands

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Basic Git Commands

- git add <file>
  - Add changed/new file to the index. AKA stage file.
- git status
  - Shows files you have changed/added/deleted.

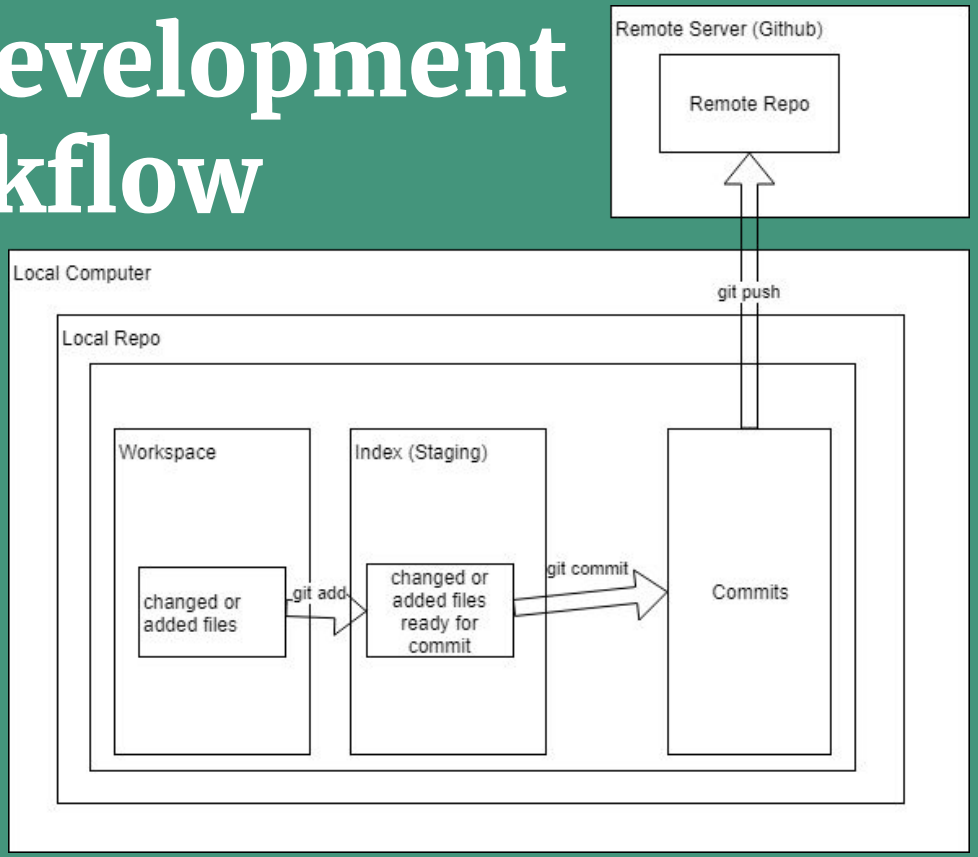
Basic Git Commands

- git push
  - Pushes local repo commits to remote repo.

Basic Git Workflow

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Basic Git Development Workflow



@devtogethermad | devtogether.co | devtogethermad@gmail.com

Exercise 1

Basic Commits to Remote Repo

See Handout

@devtogethermad | devtogether.co | devtogethermad@gmail.com

RECAP
Commit Best Practices

@devtogethermad | devtogether.co | devtogethermad@gmail.com

RECAP: Commit Best Practices

- Commit Related Changes
- Commit Often
- Be Descriptive in Commit Message
  - Answers “why” in future.

Workshop Agenda - Round 2

- What is branching?
- Basic Branching Terms
- Basic Git Branching Commands
- Basic Branching Workflow
- Questions > Exercise

What is branching?

What is branching?

- Copy of code
- Make changes without changing “stable” branch
- “Pull Request” changes when ready
- Changes merged = PR complete

Basic Branching Terms

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Basic Branching Terms

- Branch
  - A copy of the repo. Changes are isolated.
- Master
  - The “stable” branch.
- HEAD
  - Latest commit in current branch.

Basic Branching Terms

- Merge
  - Take changes from a branch and add them to another branch.
- Pull Request
  - Request to merge changes from a branch to another branch (usually master).

Basic Branching Terms

- Fetch
  - See changes from remote repo. (think preview)
- Pull
  - See changes AND apply changes from remote repo to your local repo.

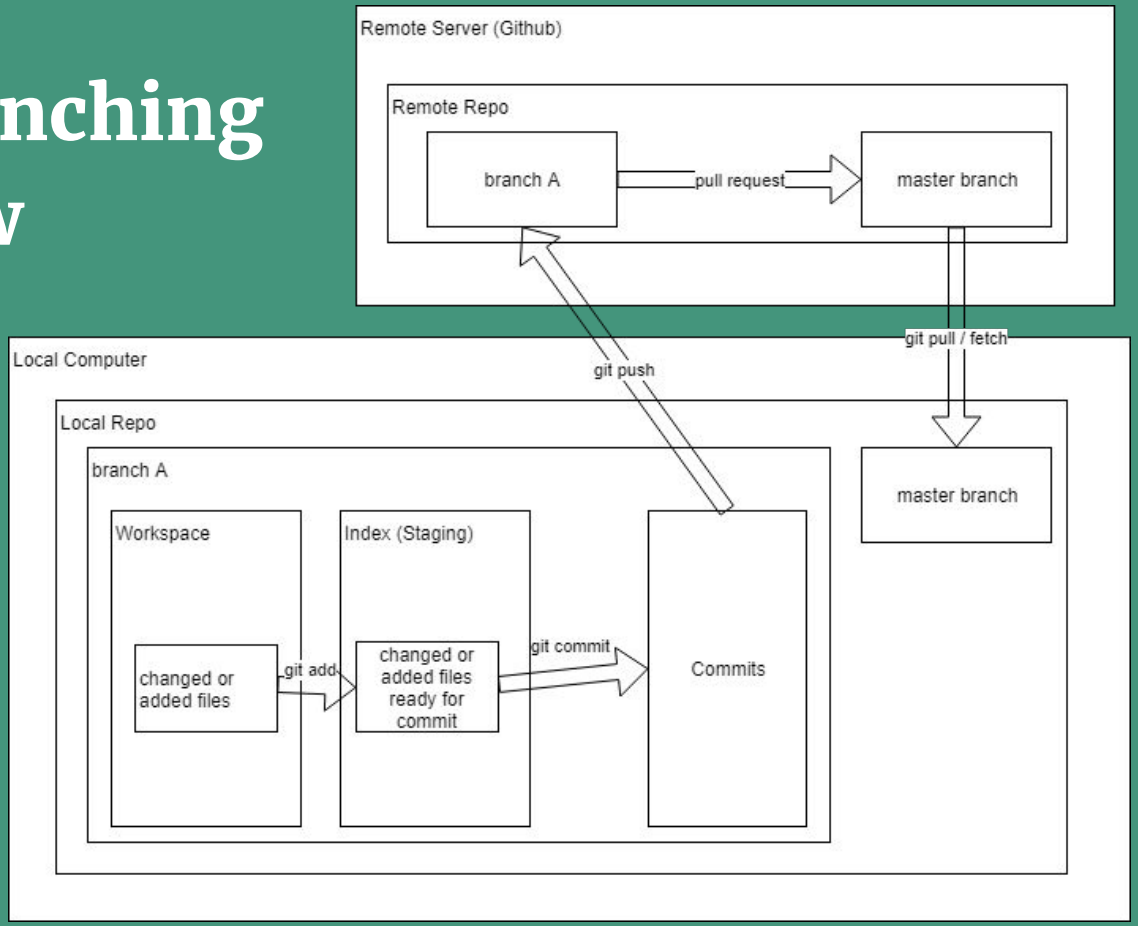
Basic Git Branching Commands

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Basic Git Branching Commands

- git checkout <branch>
  - Switch to <branch>
- git checkout -b <branch>
  - Create new <branch>
- git fetch
- git pull

Basic Branching Workflow



@devtogethermad | devtogether.co | devtogethermad@gmail.com

Exercise 2

Branching

See Handout

Workshop Agenda - Round 3

- Why Branch?
- Branching Strategies
- What is Forking? When to Fork?
- Questions > DEMO > Exercise

Why Branch?

- Keep master stable. Isolate changes.
  - Prevent untested/non-ready code to get into live app.
  - Prevent potential bugs to get into live app.
  - Allows small commit changes. Allows “commit often” workflow.
- Experiment.
- Prevent bugs/untested/non-ready code to impact other developers’ work.

Why Branch?

- “Capture” a snapshot of what was released at a certain time. And be able to make changes/hotfixes to that code and re-release.
- “Share” a development task with another developer while keeping code isolated.

Branching Strategies

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Branching Strategies

- Many different
- Feature-Based Strategy
  - feature/feature-name
  - bug/bug-name

Forking a Repo

@devtogethermad | devtogether.co | devtogethermad@gmail.com

Forking a Repo

- Similar to “branching”, but with repos
- Your own copy of a repo
- Your own branches

When to Fork a Repo?

- No contributory access to the main repo.
- Play around with repo code and make changes/make your own version.
- Pull Request from your fork –> HEAD Fork

Exercise 3

Forking

See Handout

Thank You!

@devtogethermad | devtogether.co | devtogethermad@gmail.com