

COSC 499: Capstone Software Engineering Project

git
+
open
AI

```
$ git commit -m "bug fix"
```

```
REJECTED: Please, you can do better than this.
```

```
$ git commit -m "validation bug fix"
```

```
REJECTED: I can play this game all day long...
```

```
$ git commit -m "including timestamp validation \  
> in the customer backend service"
```

```
REJECTED: Do you really believe your validation  
is working??? Run the tests before committing, dude!
```

```
$ yum remove git && yum install svn
```

```
REJECTED: No, no, no! fix that validation bug first.
```

Git

vs.

GitHub



Git is installed and maintained on your local system (rather than in the cloud)



Git is a high quality version control system

First developed in 2005



One thing that really sets Git apart is its branching model

GitHub is designed as a Git repository hosting service



You can share your code with others, giving them the power to make revisions or edits

GitHub is a cloud-based hosting service



GitHub is exclusively cloud-based



Git and GitHub

- Git is a modern **version control system** for software developers
- GitHub is a public platform that supports Git in the cloud
 - Do not push files with secrets (use `.gitignore`)
- Using GitHub
 - GitHub CLI <https://cli.github.com/>
 - GitHub Desktop <https://docs.github.com/en/desktop>
 - Some IDEs also have Git integration
 - This course: GitHub classroom

Examples?



Development Process in Git Commands

- Common steps:

```
git clone
```

```
git pull origin master
```

```
git checkout -b new_branch_name
```

```
[create files, write tests, write code]
```

```
git add file_name
```

```
git status
```

```
git commit -m "string message describing change"
```

```
git push new_branch_name
```

```
[create pull request, wait for review, make fixes as needed, merge to master]
```

repeat

Development Process in Git Commands

- Common steps:

```
git clone
```

So important!

Why?

```
git pull origin master
```

```
git checkout -b new_branch_name
```

[create files, write tests, write code]

```
git add file_name
```

```
git status
```

```
git commit -m "string message describing change"
```

```
git push new_branch_name
```

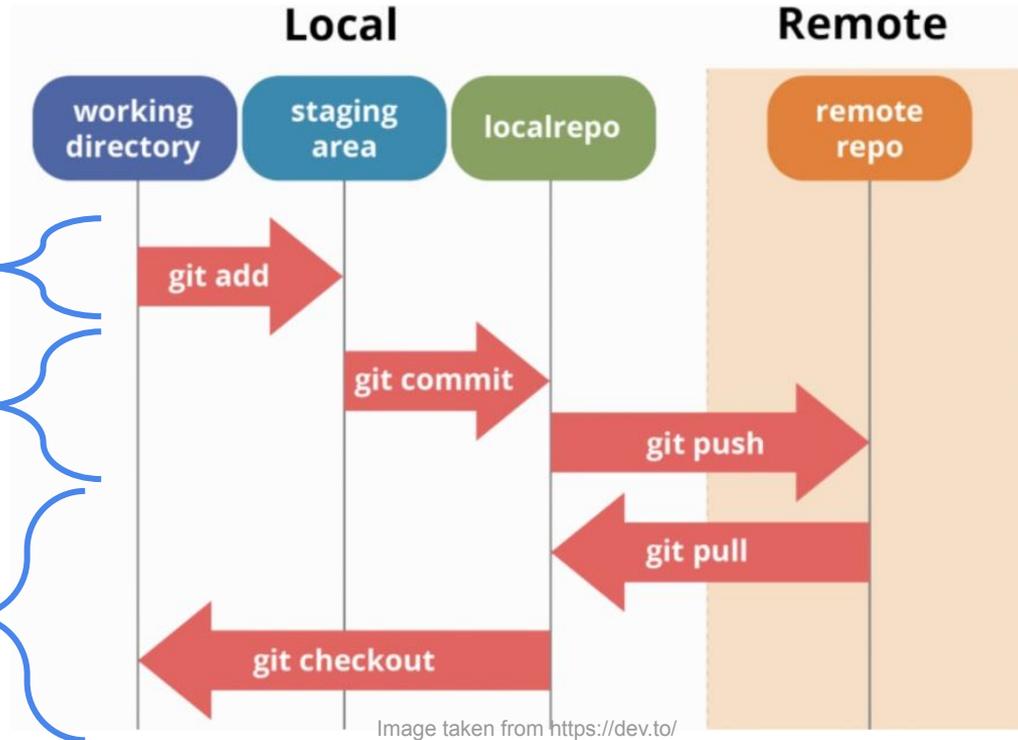
[create pull request, wait for review, make fixes as needed, merge to master]

repeat

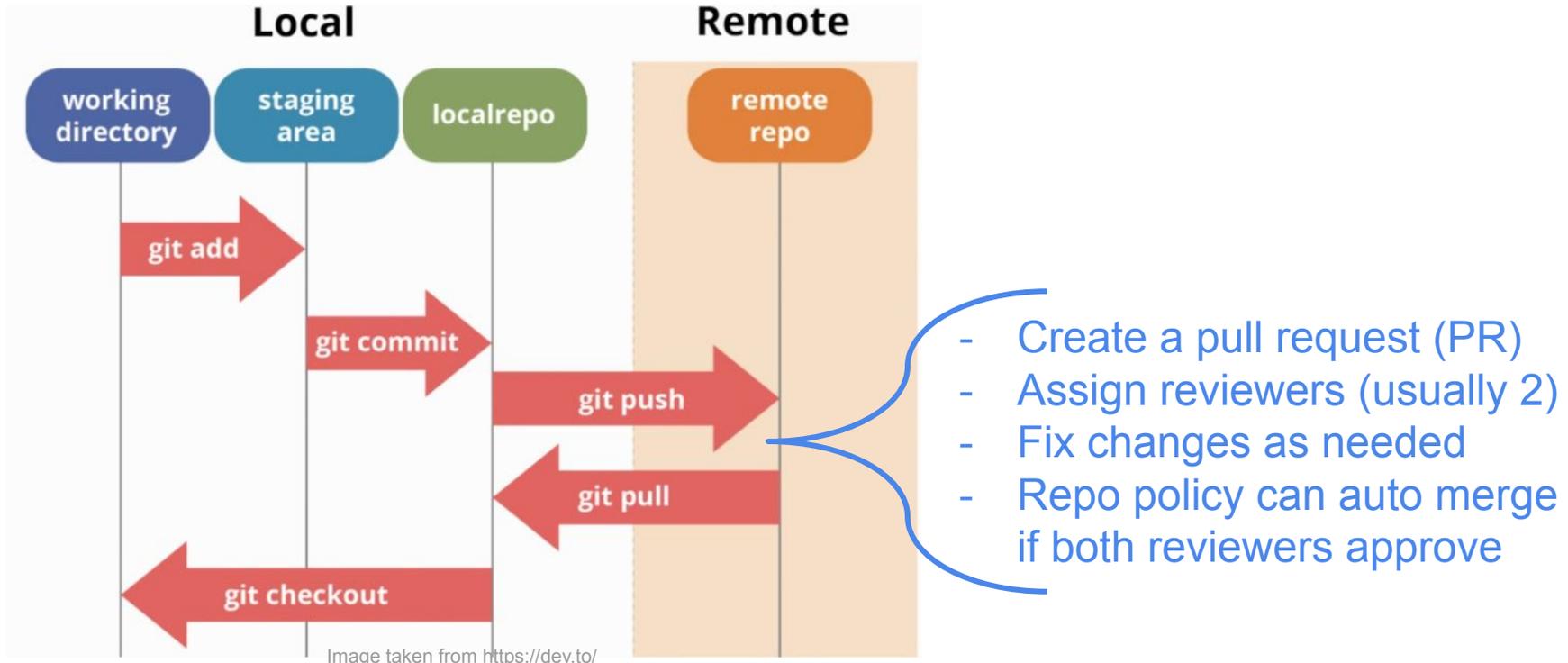
Basic Development Process: Alone

-

- Write new code
- Attempt to synch new code
- Merges branch to master



Basic Development Process: In a Team



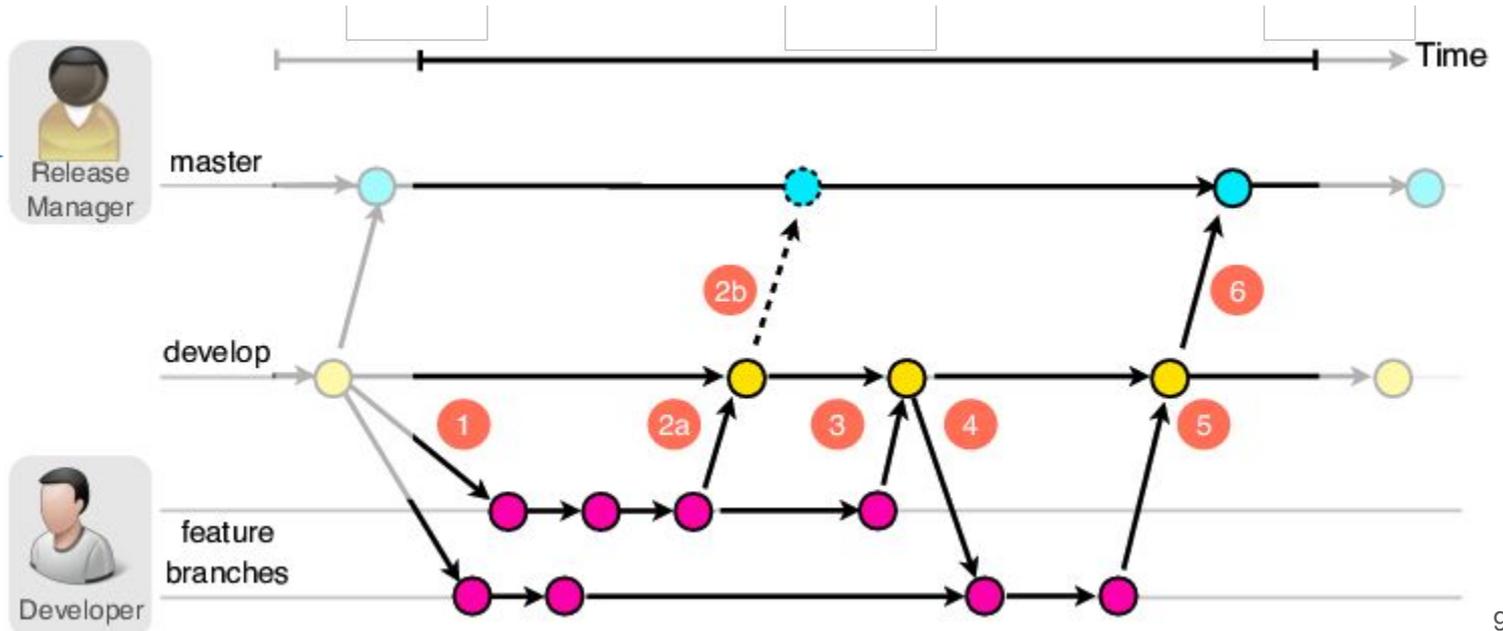
Branching and Merging

- What is a **branch**?
 - An independent line of development within a project
- Why?
 - Your edits don't immediately effect the original source code
 - Teams can work on parallel builds
- When?
 - Branch when you have a new feature
 - Name with a short representative description of the feature
 - **Do NOT** name it after yourself
- **Merging** your branches
 - Process of joining your branch with original source code (main/master) after fully tested and peer reviewed
 - **Do NOT** delete branch history

Visualizing Git Flow

- Master branch - stable code, ready to stage/deploy
- Develop branch - where features merge to

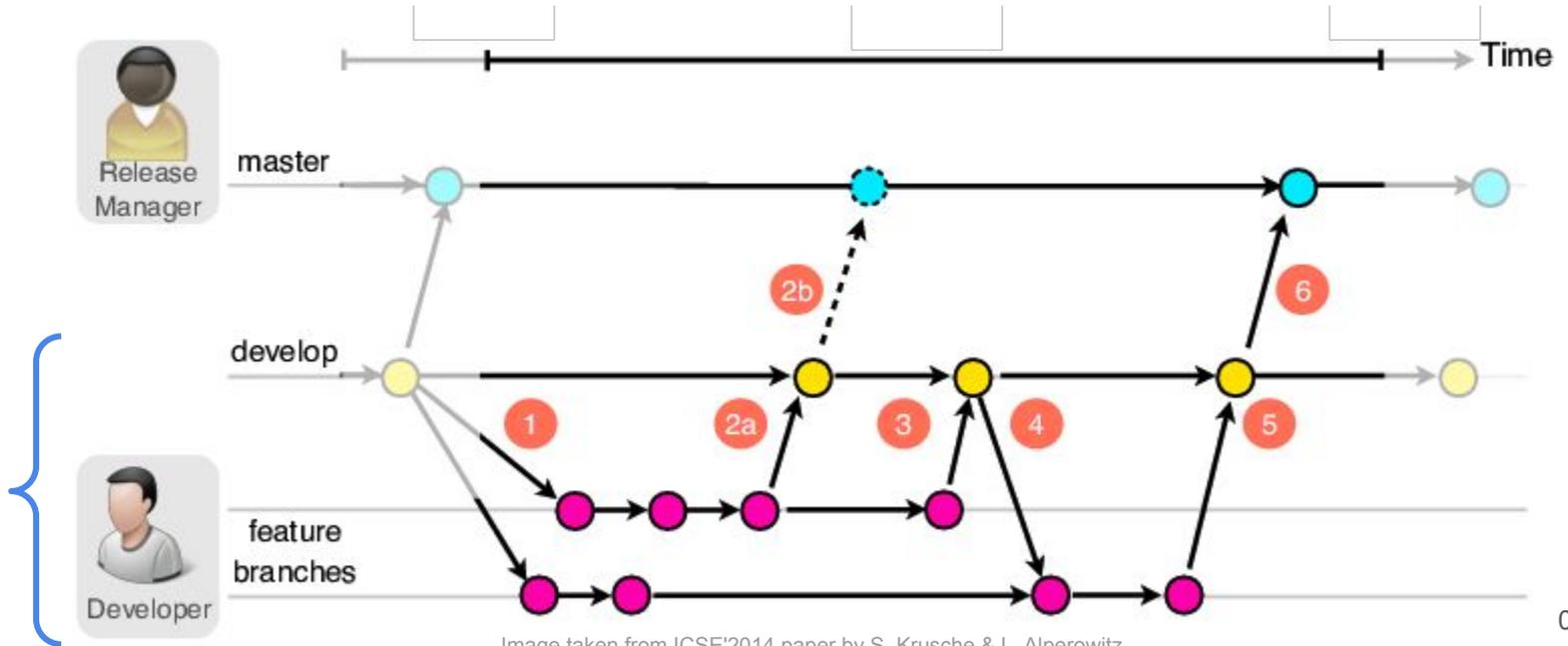
In charge of integration →



Visualizing Git Flow

- Master branch - stable code, ready to stage/deploy
- Develop branch - where features merge to

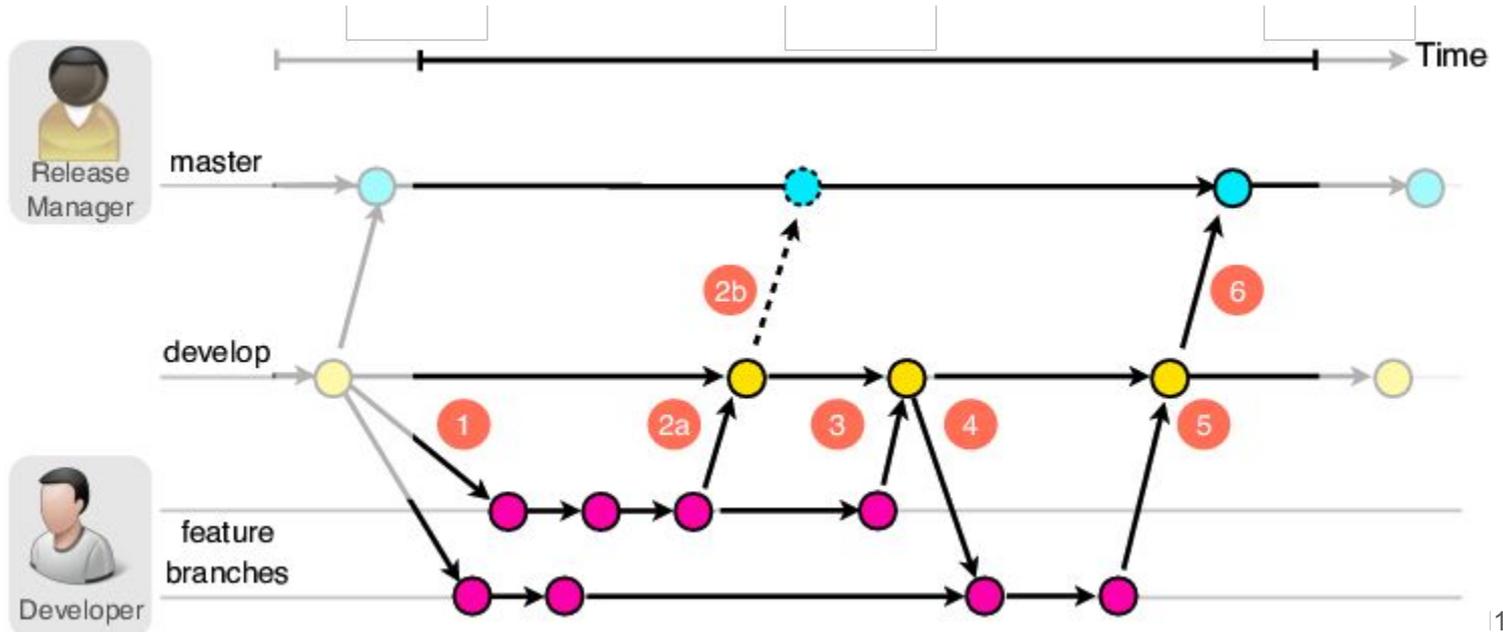
Features
branch off of
and merge
into develop



Visualizing Git Flow

- Master branch - stable code, ready to stage/deploy
- Develop branch - where features merge to

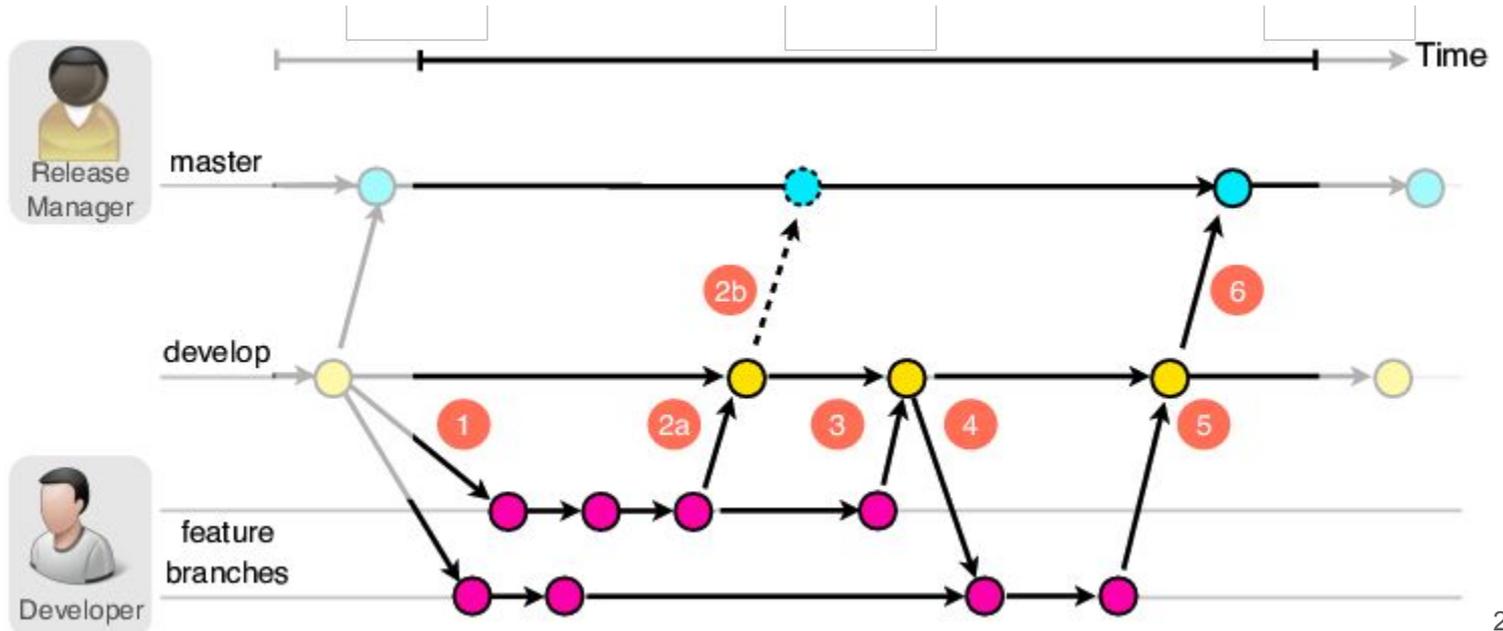
How many features were merged to develop?



Visualizing Git Flow

- Master branch - stable code, ready to stage/deploy
- Develop branch - where features merge to

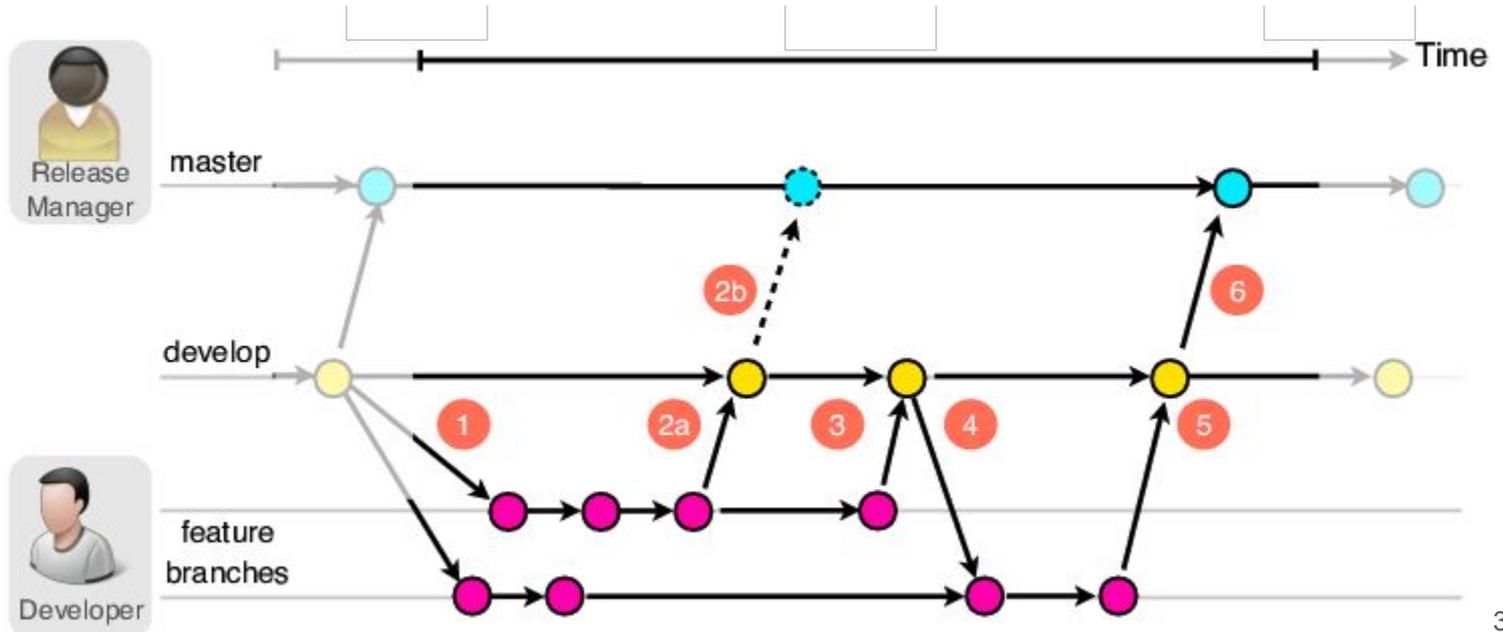
After the merge in 2a, why was there continued development that led to 3?



Visualizing Git Flow

- Master branch - stable code, ready to stage/deploy
- Develop branch - where features merge to

Why is there another pull from develop at 4?



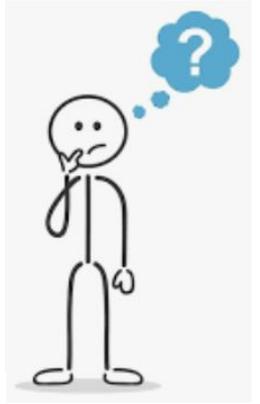
Branching Conventions for this Course

- Branches for your repo
 - master for deployment / deployment-ready
 - develop for active development
 - doc for repo documentation only
 - log for class logs only; irrelevant to client
 - **one per feature**
- In log branch:
 - student A indiv logs
 - student B indiv logs
 - ...
 - team logs



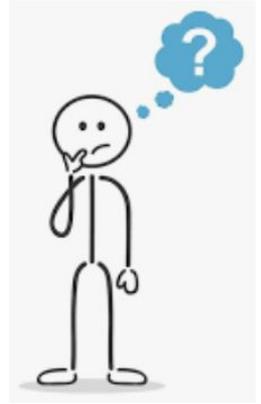
"Can a Branch have more than One PR?"

- Like asking: "Can a feature have more than one PR?"
- "Should it?"



"Can a Branch have more than One PR?"

- Like asking: "Can a feature have more than one PR?"
- "Should it?"
- Process for reusing a branch for multiple PRs:
 - Open a PR from branch
 - When done, merge PR into main
 - Continue working on the same branch feature and push more commits
 - Open another PR from that same branch to main
- Each PR follows after the previous one is closed/merged

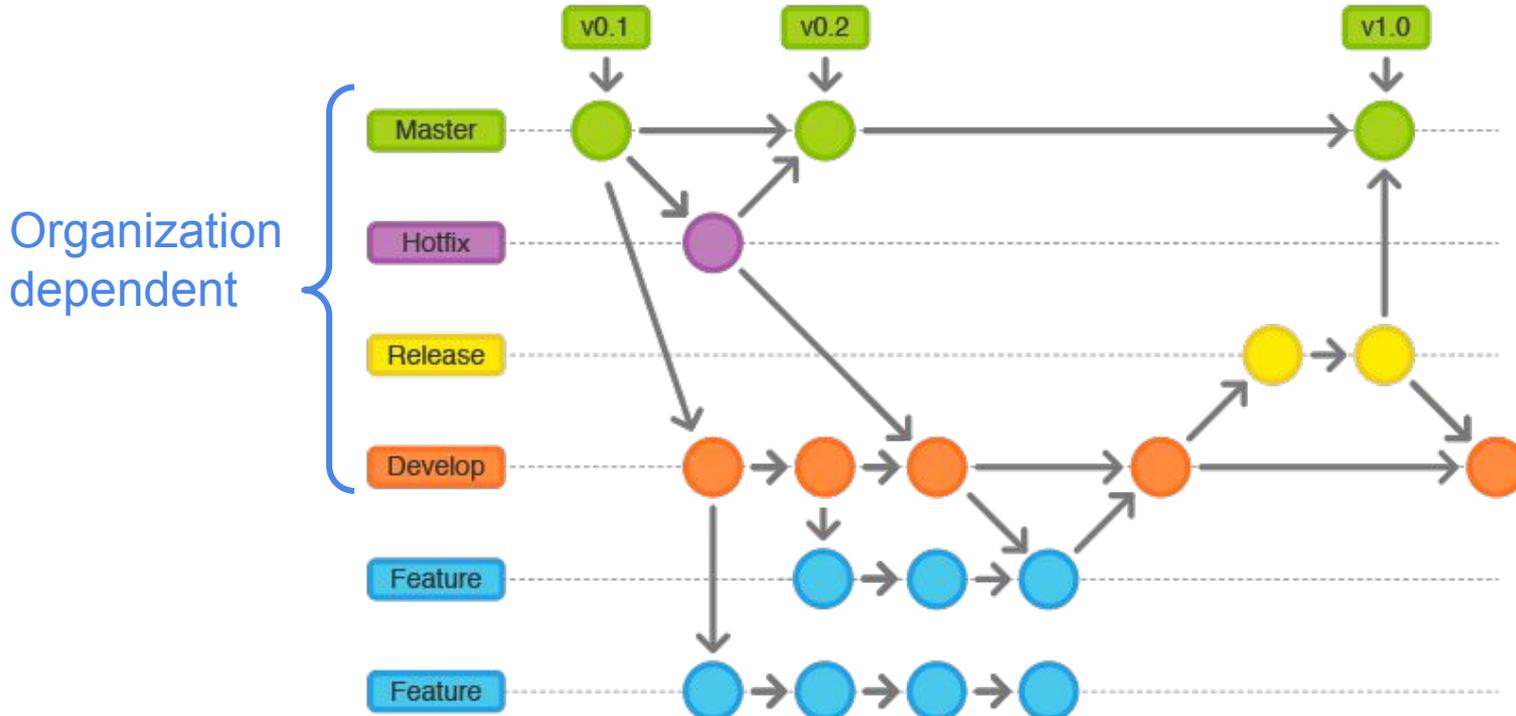


Branching Conventions for this Course

- Industry practice: **integration lead** is the only one who can merge develop to master
 - Promotes ownership across team members
 - You may want 2+ people assigned to integration
 - You may want individuals to merge their own approved code
 - Be careful: Must still test what you merge!
- **Every PR requires 2 reviewers**
 - Branch setting
 - Applies to all code and tests
 - Logs require 1 reviewer

Another Example

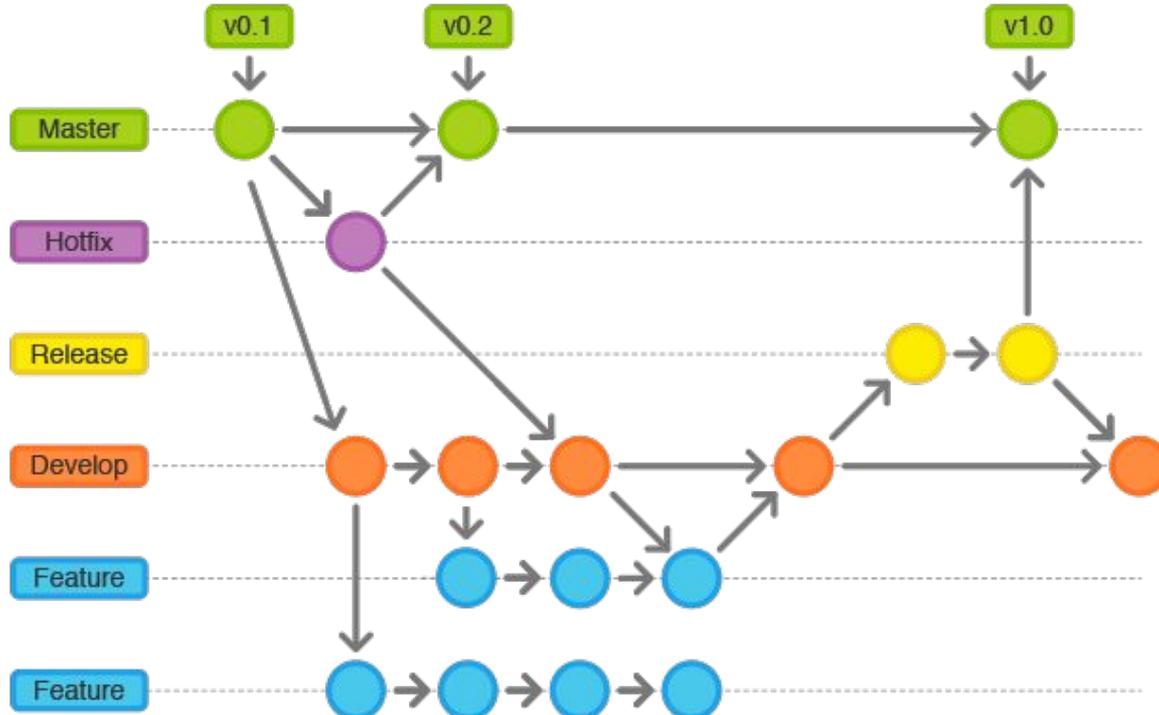
- More complex branching strategy



Another Example

- More complex branching strategy

Hotfix
requires a
synch with
both master
and develop



Another Example

- More complex branching strategy

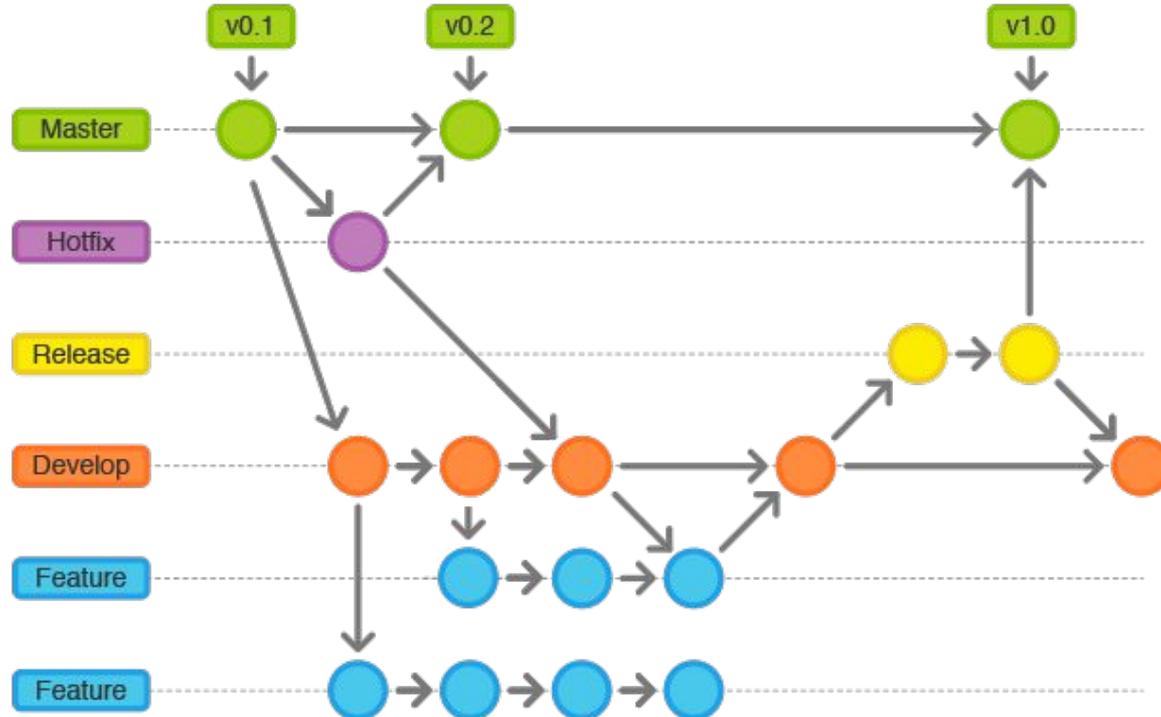


Image taken from <https://lrtecnologia.wordpress.com/>

Developer Maxim: Merge Early and Often

- Merging is analogous to a "synch" action
- What happens if you develop your feature for a long time while everyone else is developing and merging their work?

Developer Maxim: Merge Early and Often

- Merging is analogous to a "synch" action
- What happens if you develop your feature for a long time while everyone else is developing and merging their work?
 - You get stuck handling all the merge conflicts
- Therefore: Pull changes and integrate frequently
 - Keep your PRs small
 - Avoid massive merge conflicts



Other Goodies: GitHub Student Developer Pack

- Available to students and offers a range of free services (optional)

<https://education.github.com/pack>

Intro to Web Dev offers (Total 8) ×

	Bootstrap Studio	Design	Developer tools	
	DigitalOcean	Cloud		
	JetBrains	Developer tools		
	Microsoft Azure	Cloud	Virtual Events	
	Educative	Learn		
	Polypane	Design	Developer tools	
	Microsoft Visual Studio Dev Essentials	Cloud	Developer tools	Learn
	GitHub Pages	Developer tools		

Explore the experience

Mobile App Development offers (Total 8) ×

	Microsoft Azure	Cloud	Virtual Events	
	FrontendMasters	Learn		
	LambdaTest	Developer tools		
	Lingohub	Developer tools	Infrastructure & APIs	Productivity
	Kodika	Design	Developer tools	Mobile
	Bump.sh	Infrastructure & APIs	Developer tools	
	Replit	Developer tools	Learn	
	GitHub Codespaces			

Explore the experience

Developer Operations offers (Total 7) ×

	Travis CI	Developer tools	Infrastructure & APIs
	GitHub	Developer tools	
	Sentry	Infrastructure & APIs	Developer tools
	BrowserStack	Developer tools	
	DevCycle	Developer tools	Security & analytics
	CodeScene	Security & analytics	Developer tools
	New Relic	Developer tools	Cloud

Explore the experience