Git Workflows and Repo Etiquette

MAXWELL DAUM, SPENCER BYERS, TYLER NEWSOME

What is your opinion on git?

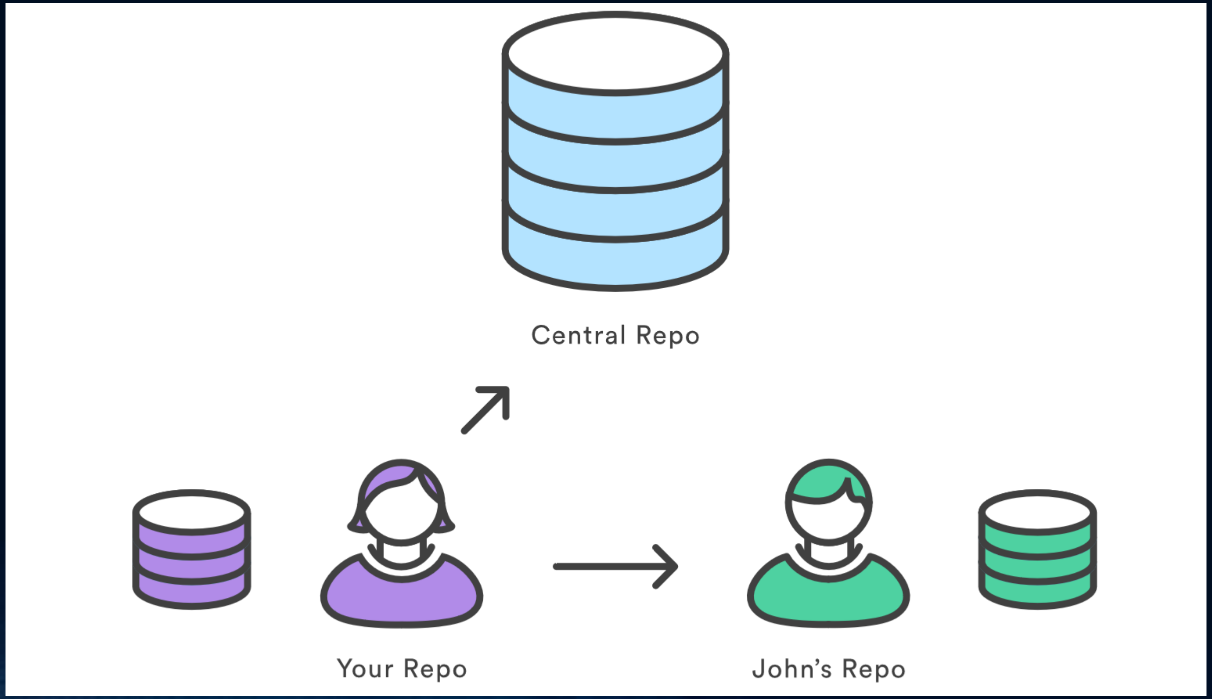
- Familiarity?
- How comfortable with commands other than push, commit and pull?
- Merge conflict issues?
- Does it feel like it “gits” in the way? ☺

What is Git?

- Example of a version-control system
- Tool to manage your code as you change and update it
- Point is to be able to “undo” large code changes, manage conflicting versions, have a centralized online location to send code while still being able to work online

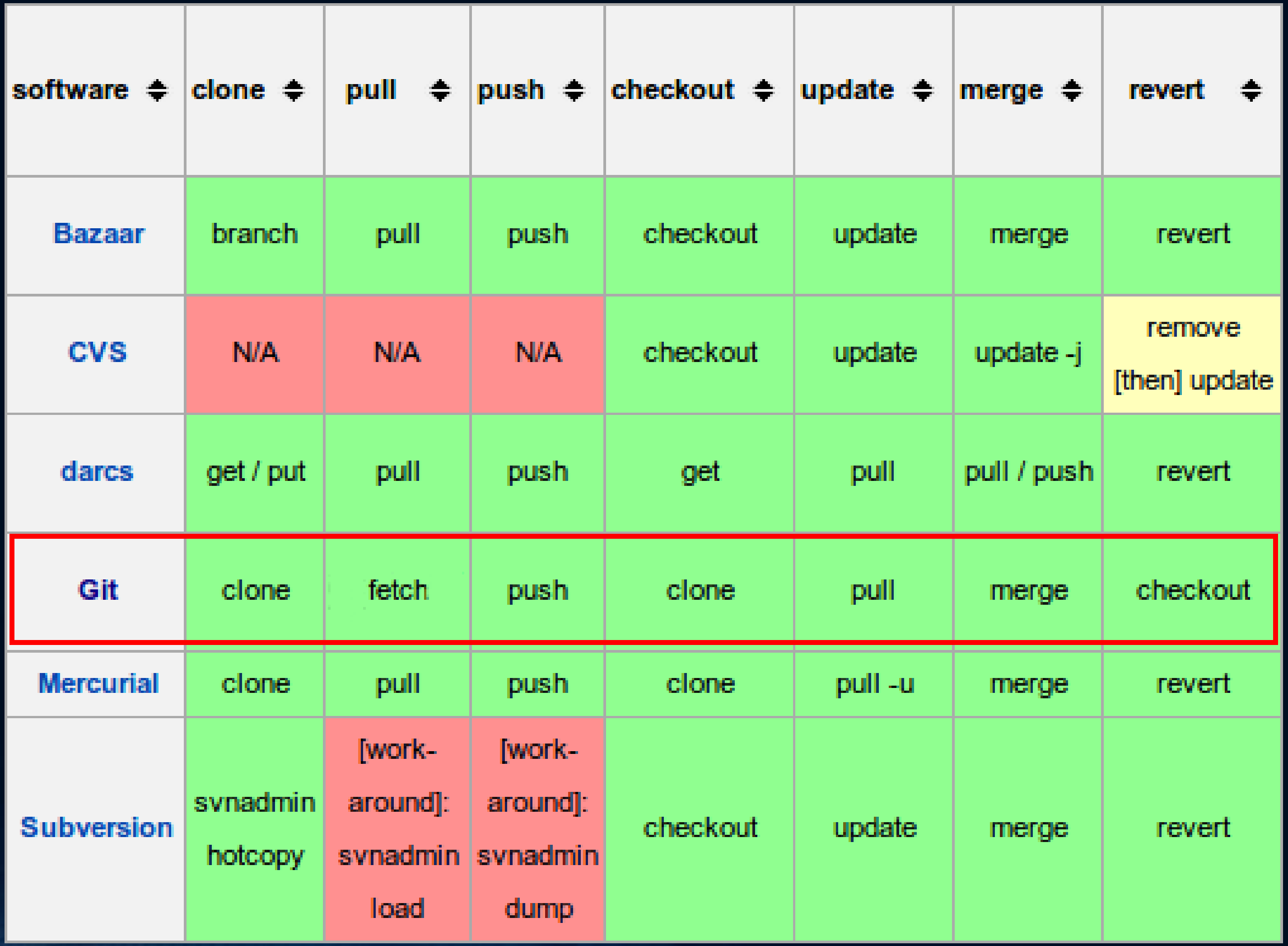
Local versus Remote

- *distributed* version control system
- local repository sits on your machine
- Remote is the shared code base on server
- Advantages?



Basic Git Commands

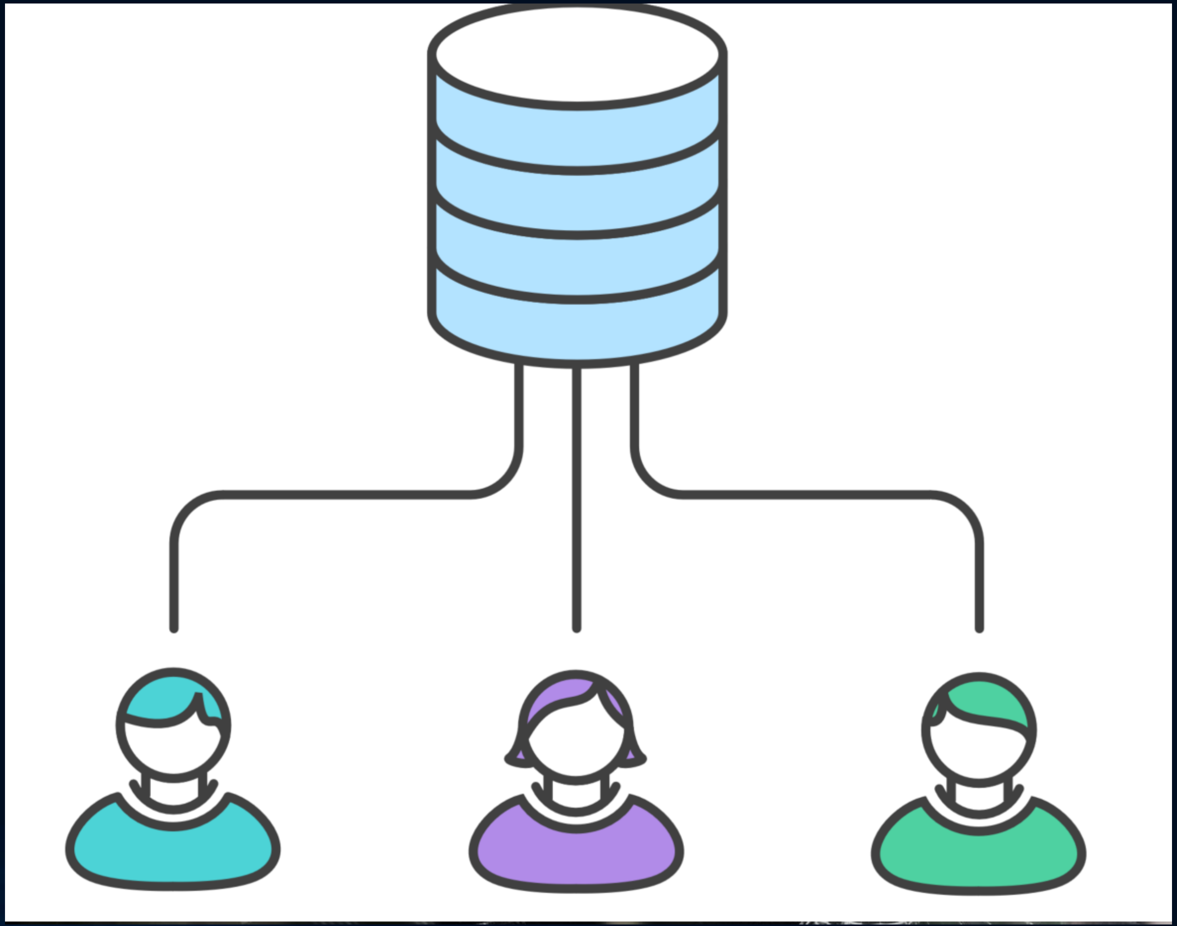
- Push
- Pull
- Add
- Commit
- Checkout
- Will speak on merge later



software	clone	pull	push	checkout	update	merge	revert
Bazaar	branch	pull	push	checkout	update	merge	revert
CVS	N/A	N/A	N/A	checkout	update	update -j	remove [then] update
darcs	get / put	pull	push	get	pull	pull / push	revert
Git	clone	fetch	push	clone	pull	merge	checkout
Mercurial	clone	pull	push	clone	pull -u	merge	revert
Subversion	svnadmin hotcopy	[work-around]: svnadmin load	[work-around]: svnadmin dump	checkout	update	merge	revert

Small Example of Vanilla use of Git

- Bob, Joe, and Lucy
- Bob initializes the repo
- Joe and Lucy clone



Small Example of Vanilla use of Git

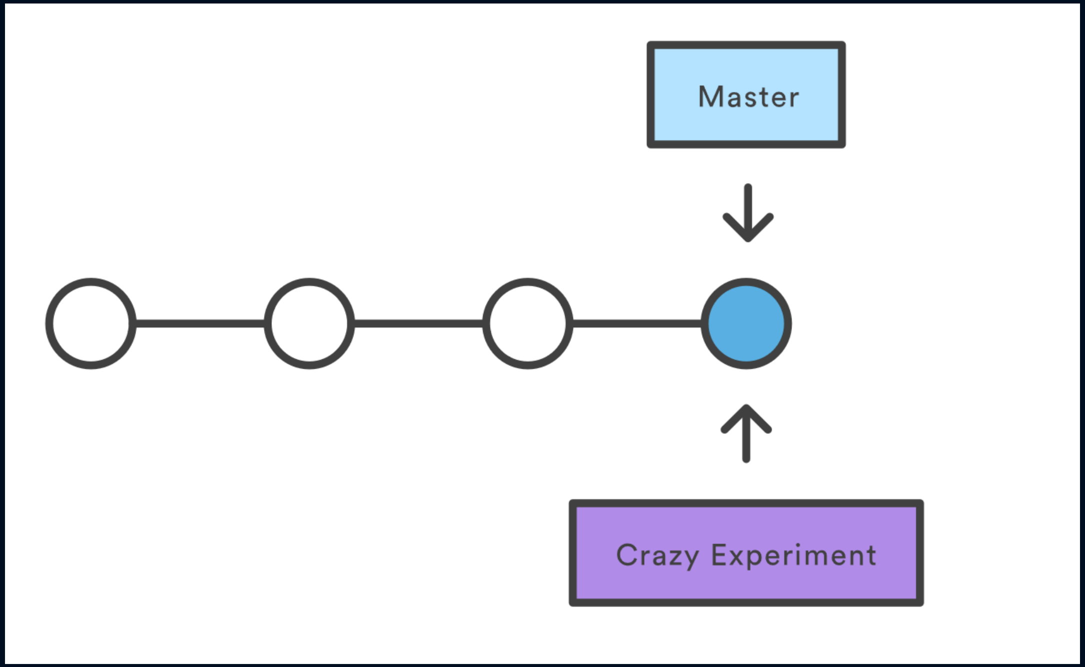
- Bob works on some functionality with file x
- Lucy works on some functionality with file y
- Joe works on some functionality with file x
- Bob pushes first, then Lucy (after pulling).
- What will happen when Joe wants to push his code?
- When will this happen?

Merge Conflicts (just don't)

- Bane of a developer's existence.
- Local changes to be overwritten by repo changes upon pull
- How to resolve?
- How to avoid?
- Repo Etiquette
- Pull requests

Branches

- independent line of development
- exact copy of remote repo
- Great for dividing work/experimenting
- Redundancy reduces likelihood of revert
- Can merge branches back
- Can be leveraged for modularized dev



Golden Rules of Git

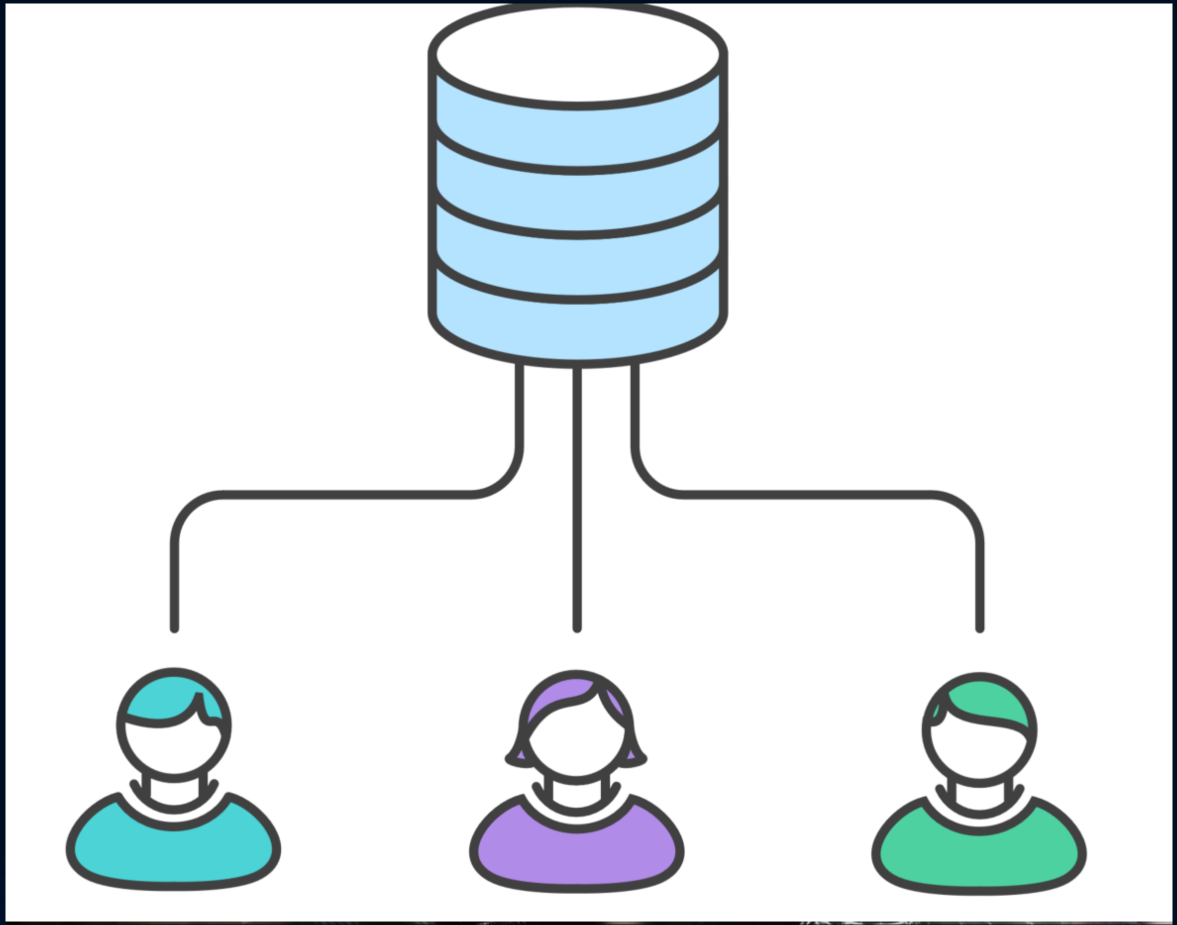
- Always write a detailed commit message! (I should know exactly what has changed in functionality just by viewing the message)
- Always specify what files you are going to work with during a coding session.
- Make sure you only add the files you want to add, especially if you are not using a git ignore file.
- It is ok to compromise your local branch, but never the remote!

Workflows

- What are they?
- Advantages?
- Centralized
- Feature Branch
- Gitflow

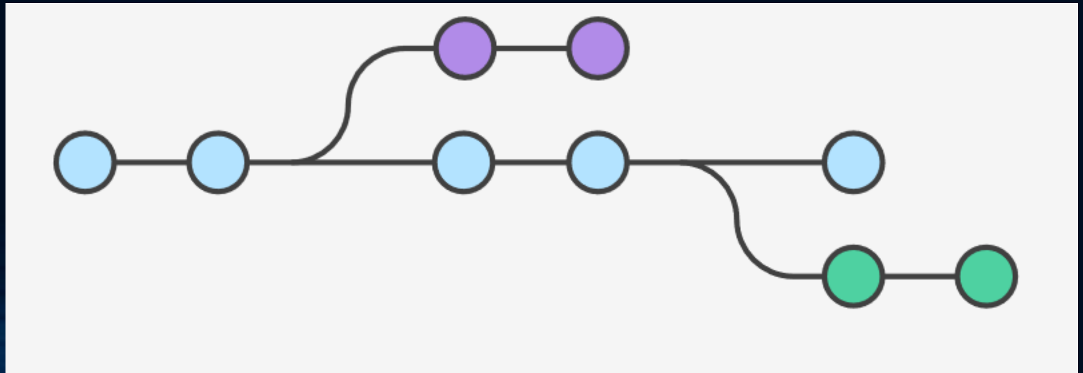
Centralized Workflow

- Vanilla use of Git
- One Master Branch
- Ok for small teams
- Why is this a problem for big team?
- Are you really utilizing git?



Feature Branching Workflow

- Every time you implement functionality, pull off a branch
- Modularized development
- Leverages pull requests
- Master is official project
- Just delete feature after merging into master
- Fits wonderfully into Agile development environment

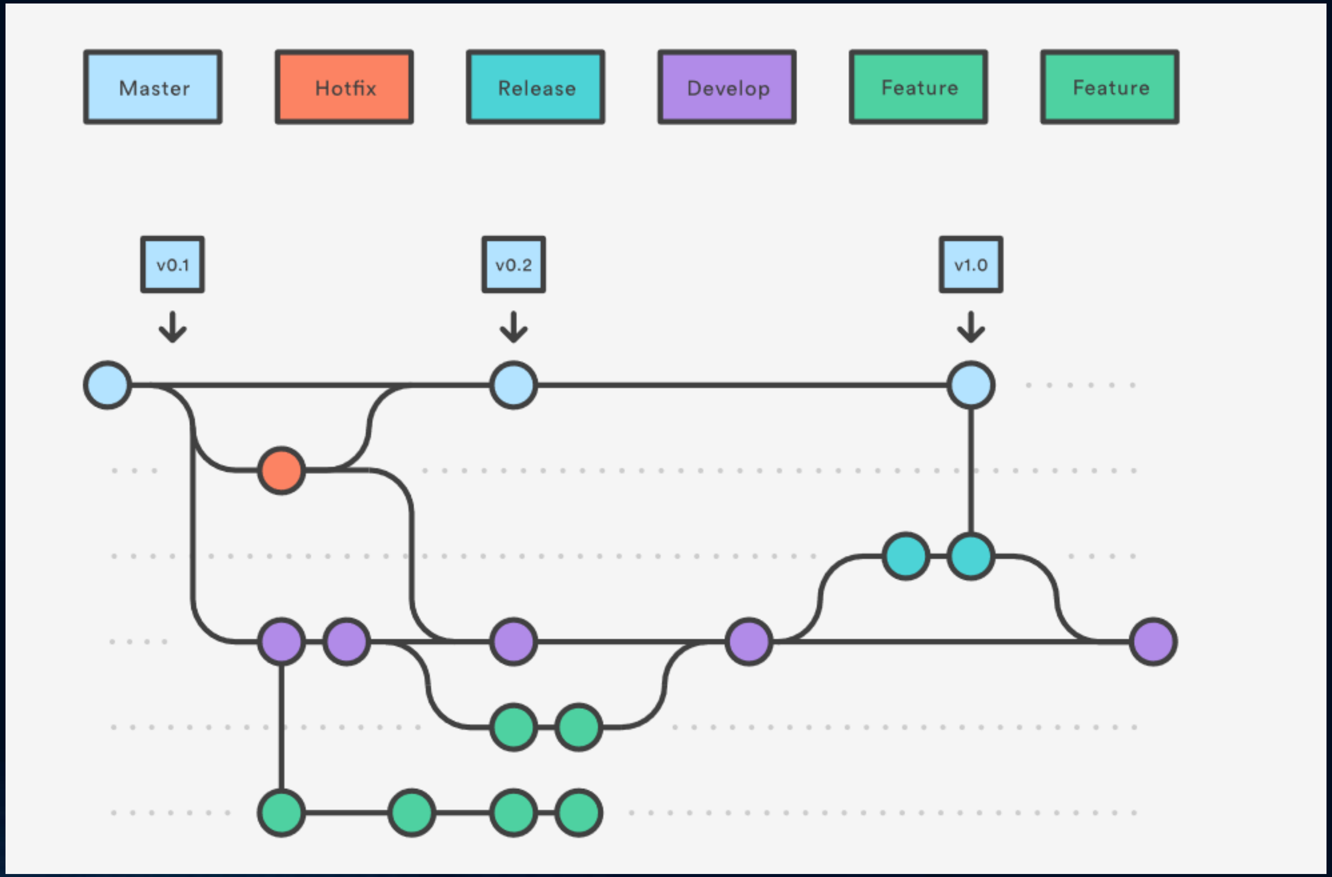


Example

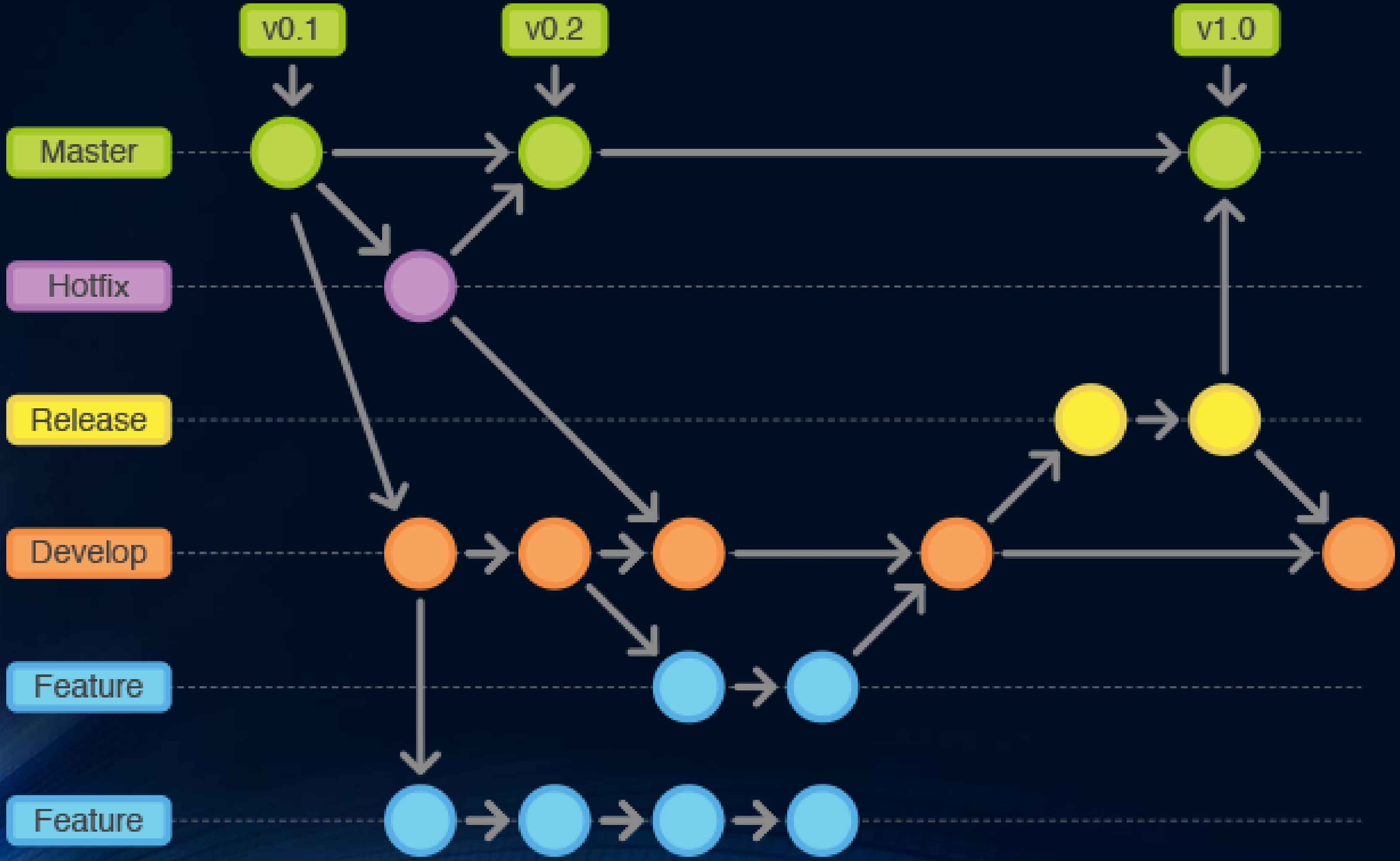
- Bob, Joe, and Lucy
- Joe begins a new feature (creates feature branch)
- Joe works a bit on feature
- Joe goes off to lunch and pushes his code to the feature branch
- After lunch, he finishes his feature and sends out the pull request
- Bob and Lucy review the changes and discuss, them
- Final version is merged to master
- Feature branch deleted

GitFlow

- Essentially expanded Feature Branching w/ redundancy and a couple auxiliary branches
- Master
- Development
- Features
- Hotfix
- Release



Example



Questions?

- Many pictures courtesy of Atlassian