

INTERMEDIATE GIT

Orlando Code Camp 2017

GIT WORDS

add	commit	fsck	mergetool	rev-list	tag
am	commit-tree	gc	mv	rev-parse	update-index
apply	config	gitk	pop	rm	update-ref
archive	count-objects	grep	prune	send-email	update-server-info
bisect	daemon	hash-object	pull	shortlog	verify-pack
blame	describe	help	push	show	write-tree
branch	diff	init	read-tree	show-ref	
bundle	diff-index	instaweb	rebase	stage	
cat-file	fast-import	log	reflog	stash	
checkout	fetch	ls-files	remote	status	
cherry-pick	filter-branch	ls-tree	request-pull	submodule	
clean	for-each-ref	merge	reset	svn	
clone	format-patch	merge-base	revert	symbolic-ref	

GIT WORDS - INTRO

add	commit	fsck	mergetool	rev-list	tag
am	commit-tree	gc	mv	rev-parse	update-index
apply	config	gitk	pop	rm	update-ref
archive	count-objects	grep	prune	send-email	update-server-info
bisect	daemon	hash-object	pull	shortlog	verify-pack
blame	describe	help	push	show	write-tree
branch	diff	init	read-tree	show-ref	
bundle	diff-index	instaweb	rebase	stage	
cat-file	fast-import	log	reflog	stash	
checkout	fetch	ls-files	remote	status	
cherry-pick	filter-branch	ls-tree	request-pull	submodule	
clean	for-each-ref	merge	reset	svn	
clone	format-patch	merge-base	revert	symbolic-ref	

GIT WORDS - INTERMEDIATE

add	commit	fsck	mergetool	rev-list	tag
am	commit-tree	gc	mv	rev-parse	update-index
apply	config	gitk	pop	rm	update-ref
archive	count-objects	grep	prune	send-email	update-server-info
bisect	daemon	hash-object	pull	shortlog	verify-pack
blame	describe	help	push	show	write-tree
branch	diff	init	read-tree	show-ref	
bundle	diff-index	instaweb	rebase	stage	
cat-file	fast-import	log	reflog	stash	
checkout	fetch	ls-files	remote	status	
cherry-pick	filter-branch	ls-tree	request-pull	submodule	
clean	for-each-ref	merge	reset	svn	
clone	format-patch	merge-base	revert	symbolic-ref	

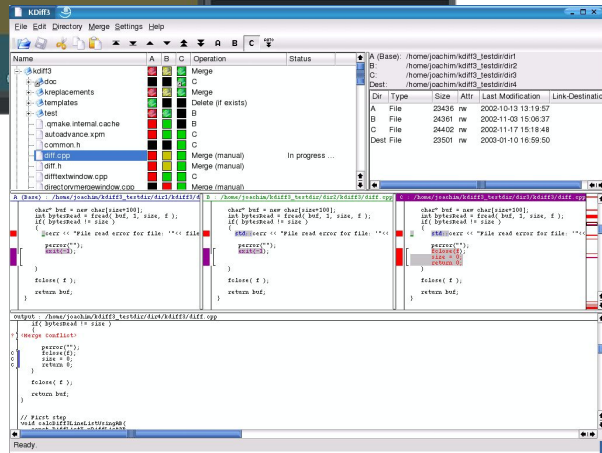
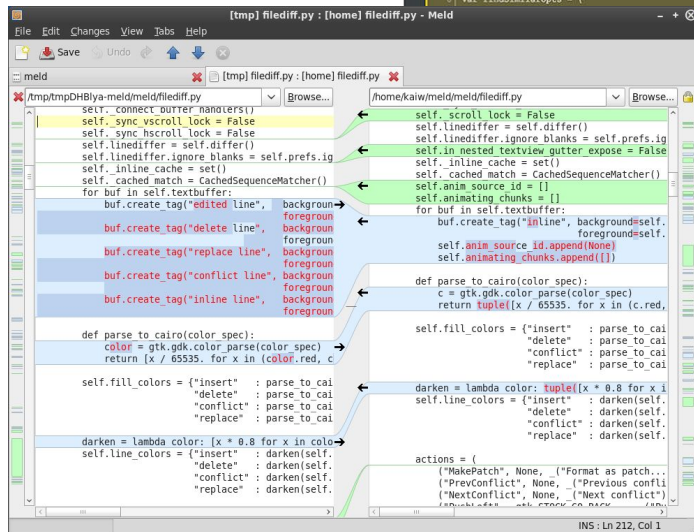
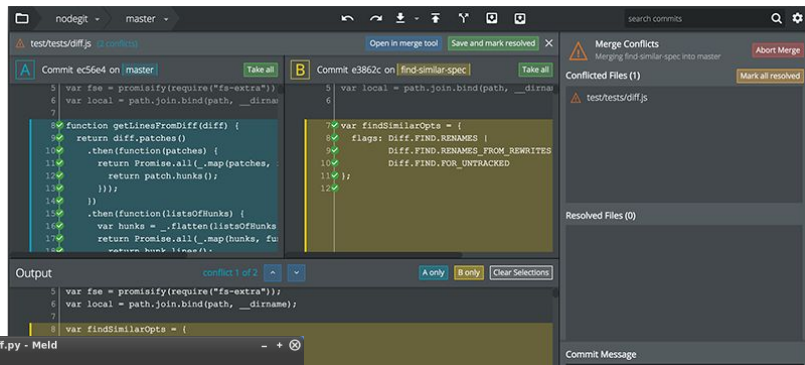
UI-COMMANDS NOT COVERED IN INTRO SESSION

MERGETOOL - CONFLICTS

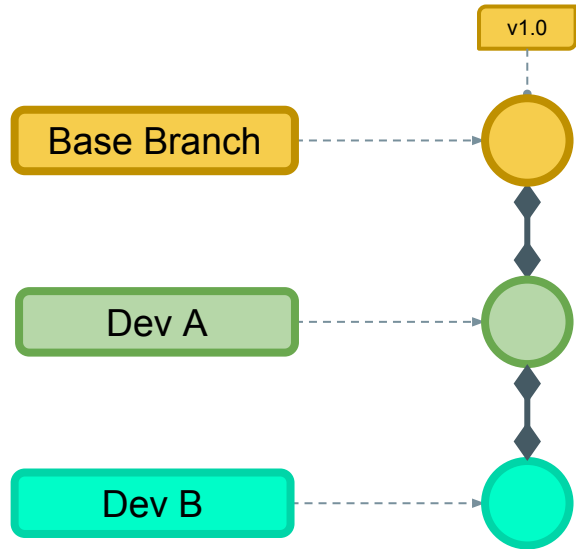
Simply your tool of choice to deal with the eventual code conflicts

A few are:

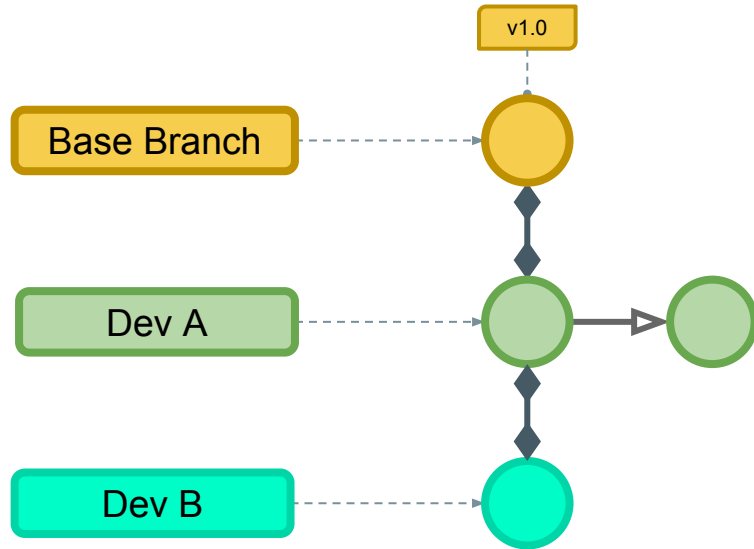
- meld
- kdiff3
- gitKraken



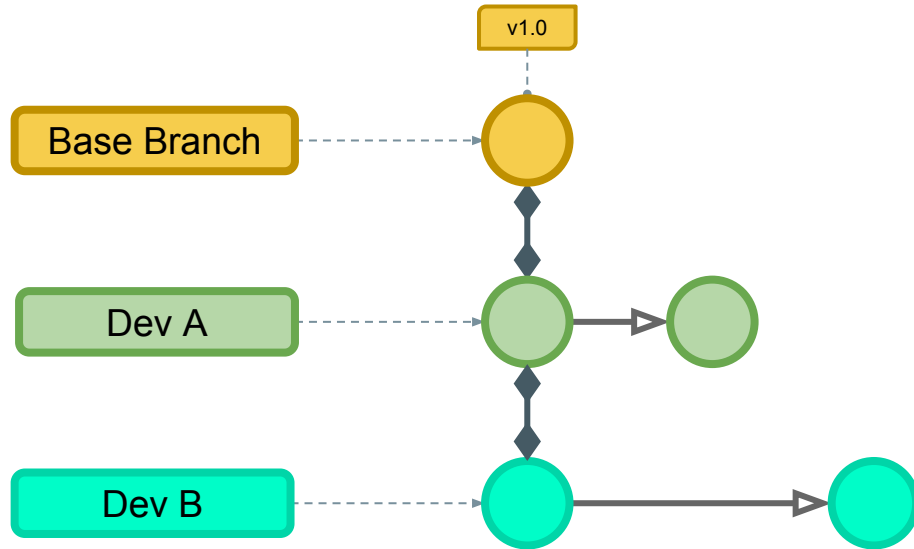
MERGE - CONFLICTS



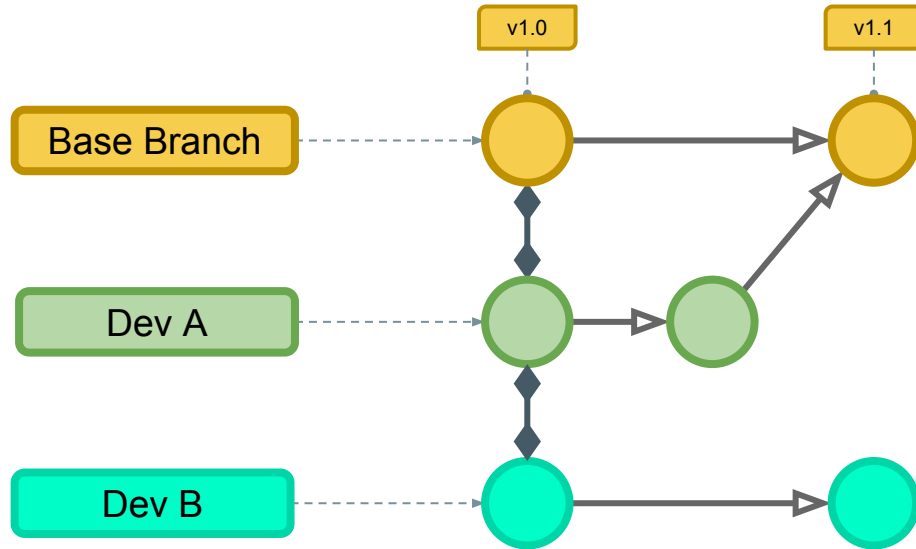
MERGE - CONFLICTS



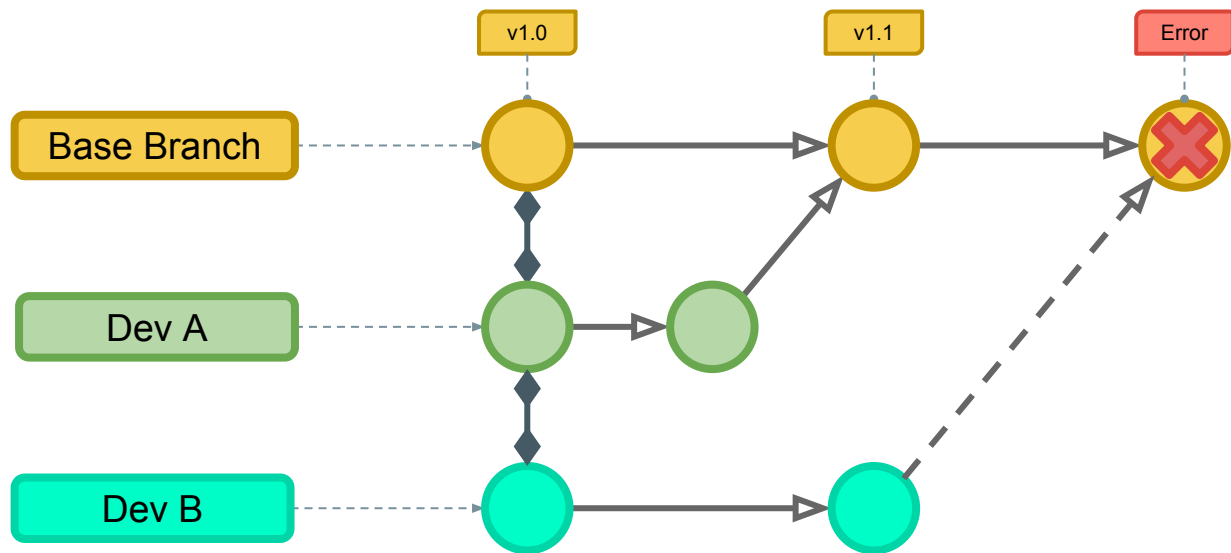
MERGE - CONFLICTS



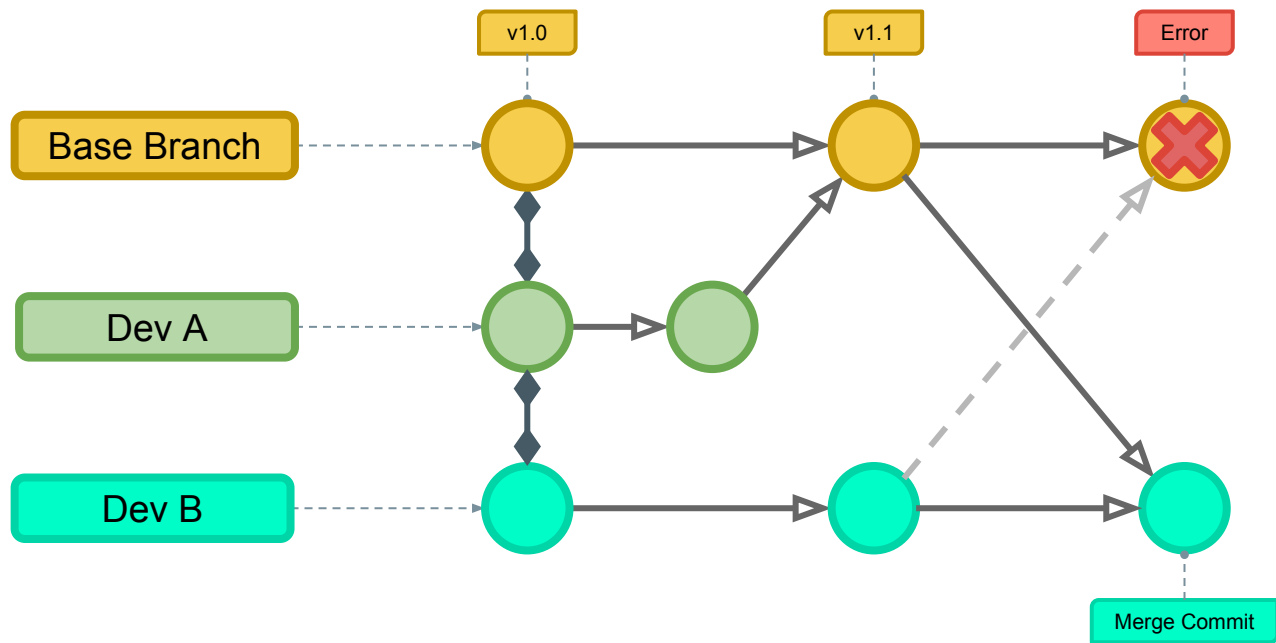
MERGE - CONFLICTS



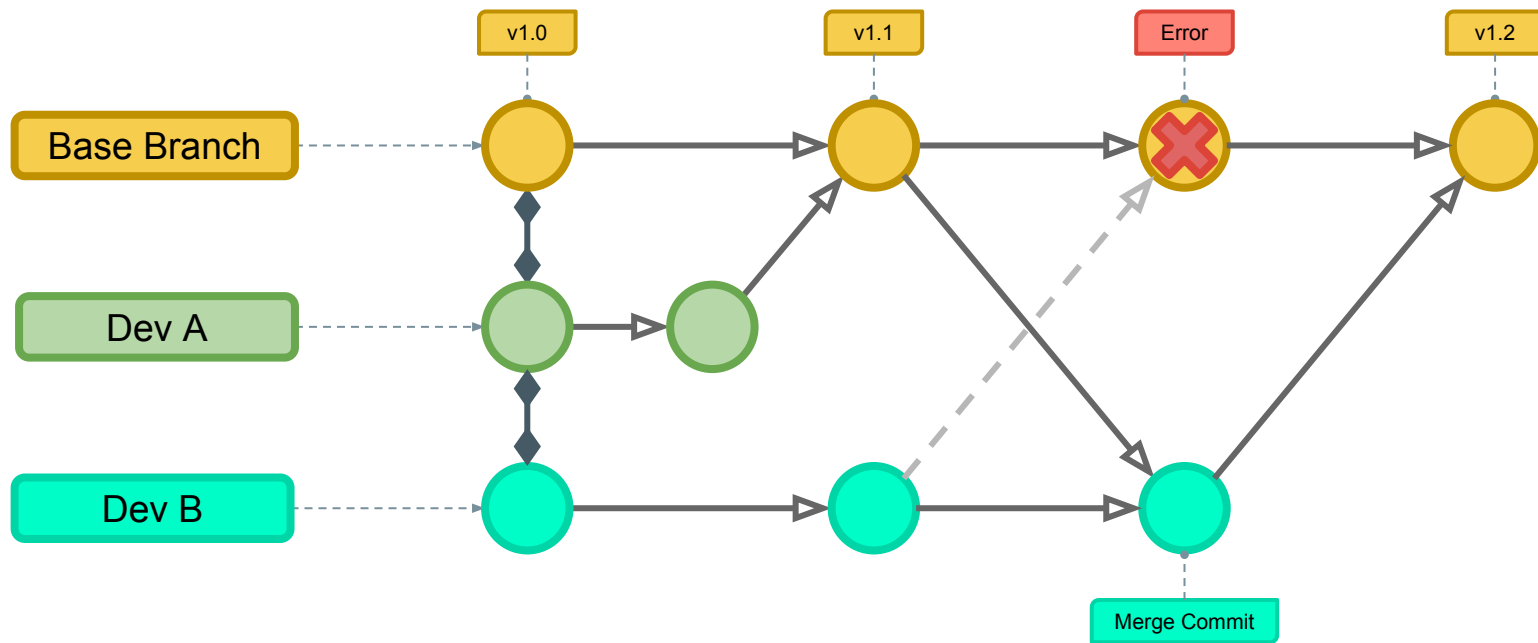
MERGE - CONFLICTS



MERGE - CONFLICTS



MERGE - CONFLICTS



BLAME

- Horrible command name!
- Wrong usage - blame someone for code
- Right usage - tracking down code changes
- Built in and on web interfaces

GitHub



Newer  Older

Raw

Normal view

History

 Bitbucket

 a803b1b 5 hours ago Full commit

Blame

Raw

Edit

Daniel Shrader b172b8e 2017-03-21

Daniel Shrader a803b1b 2017-03-22

Daniel Shrader b172b8e 2017-03-21

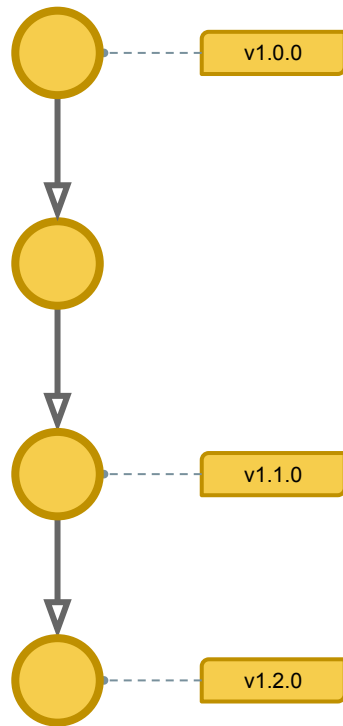
```
1 define([], function() {
2
3   var funFactory = function(Application) {
4     var radio = Application.radio;
5     var debug = false;
6     if(Application.options.debug){
7       debug = Application.options.debug;
8       console.log(Application.options.appName + ': Loading the funFactory')
9     }
10
11    // Handles the user login
12    radio.on('login:SignInTry', function(credential) {
13      Application.FBApp.auth().signInWithEmailAndPassword(credential.email, credential.password)
```

TAG

Markers on the repo

Typically used for versioning

Semantic Versioning would be good here if you don't already use one <http://semver.org/>



STASH AND POP

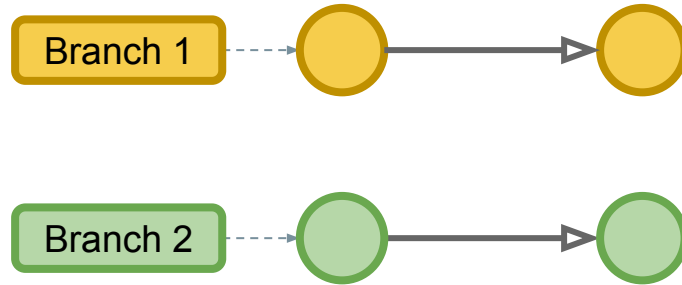
A way to store your code temporarily without committing it.

Pro: You can see if the current change broke your code.

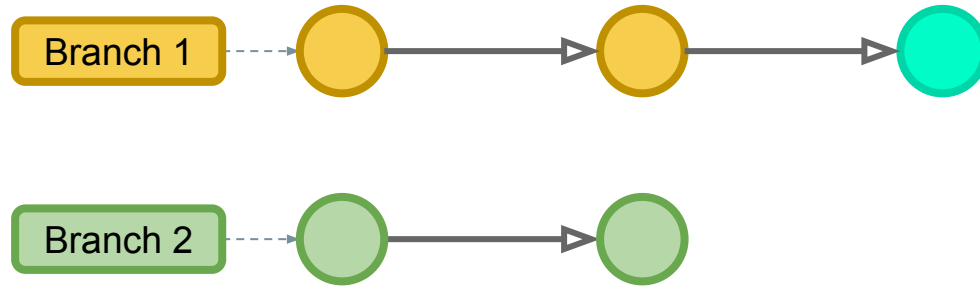
Lets you experiment quickly and revert if it fails.

Con: This is not pushed to your remote repo - aka no backup.

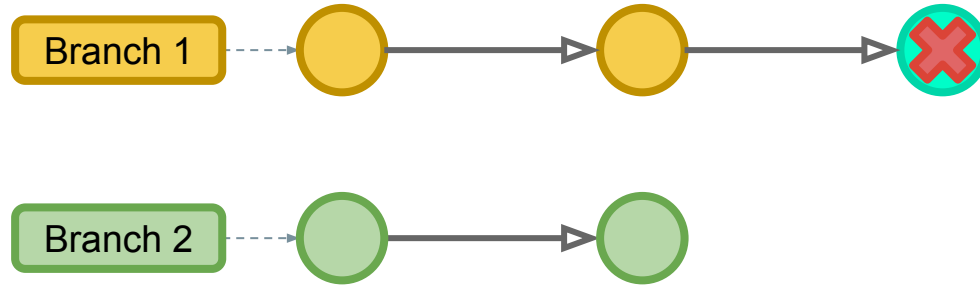
STASH AND POP-ALT USAGE



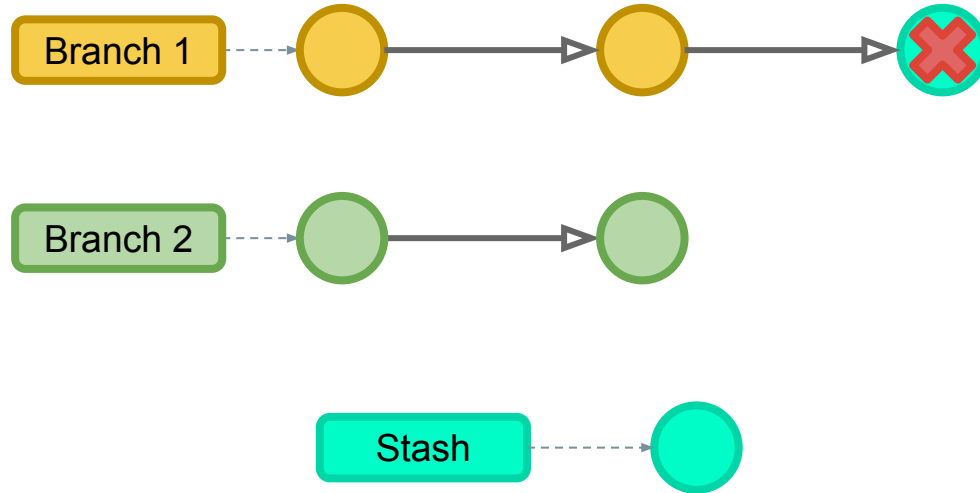
STASH AND POP-ALT USAGE



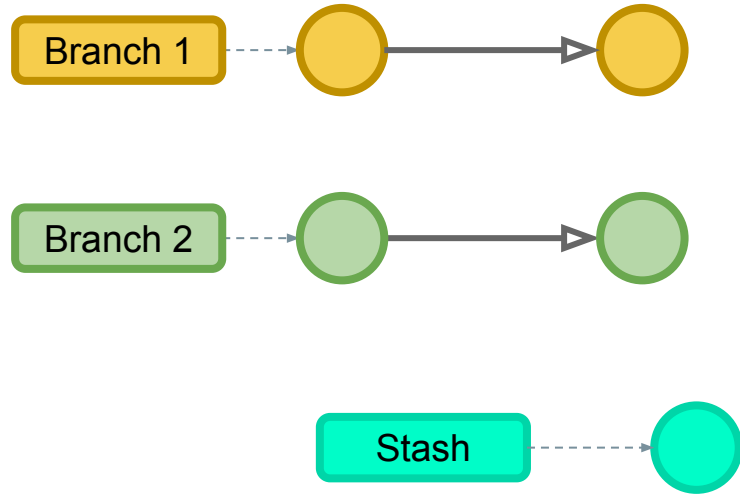
STASH AND POP-ALT USAGE



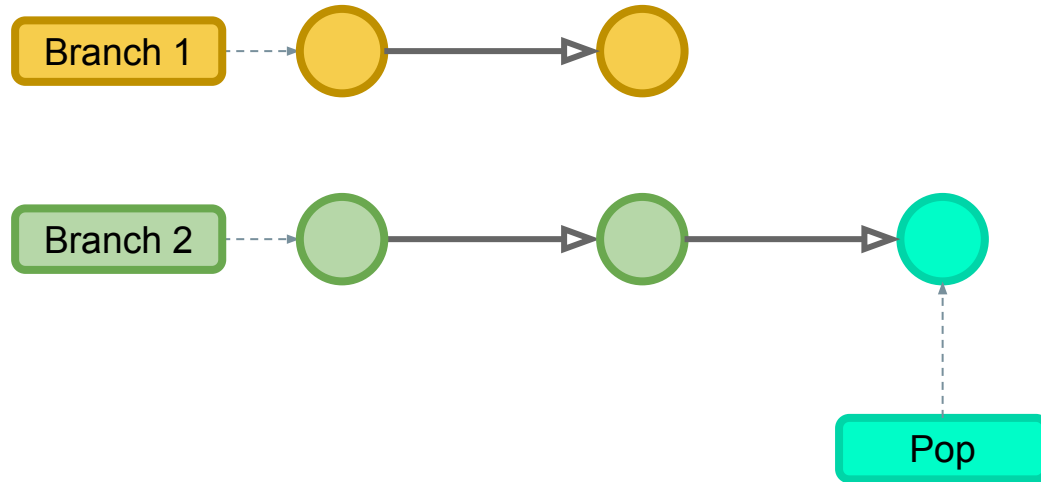
STASH AND POP-ALT USAGE



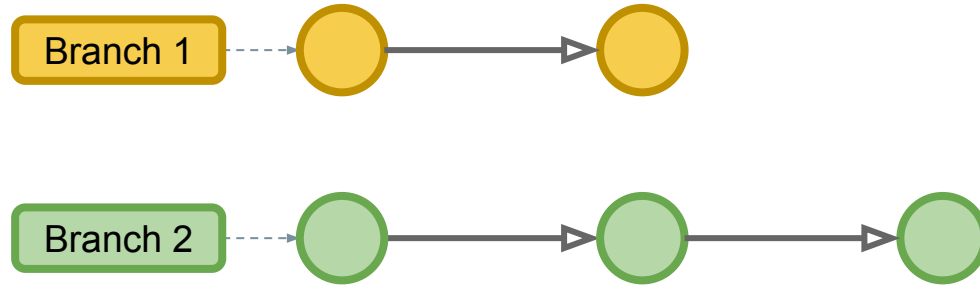
STASH AND POP-ALT USAGE



STASH AND POP-ALT USAGE



STASH AND POP-ALT USAGE



SUBMODULES

A simple way to have another 'global' project within your project.

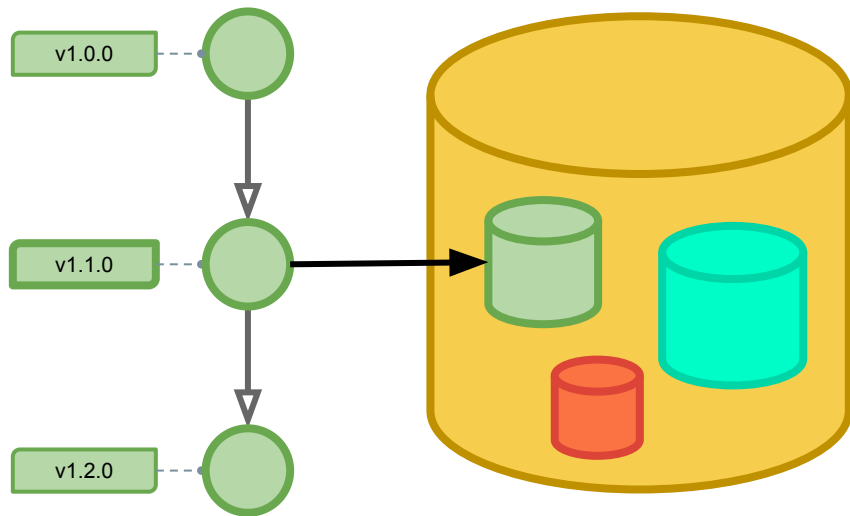
Reduces the need to 'copy' code to multiple projects.



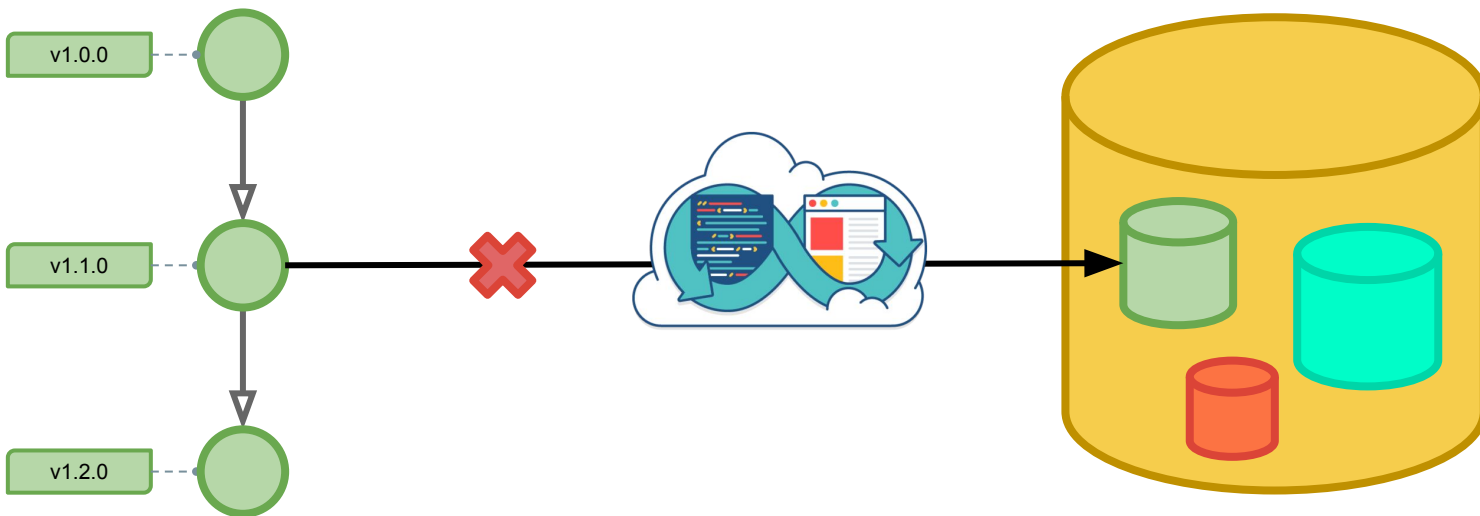
SUBMODULES

They only reference a specific commit on the submodule and do not defaultly update.

This is a pro and a con as your code does not break because of updates, but you do need to update manually.



SUBMODULES - BUILD SERVICES



Keep in mind permissions on build services!

PULL REQUESTS

PULL REQUESTS

- Command line or UI when used with services
- Code discussions / reviews
- Collaboration

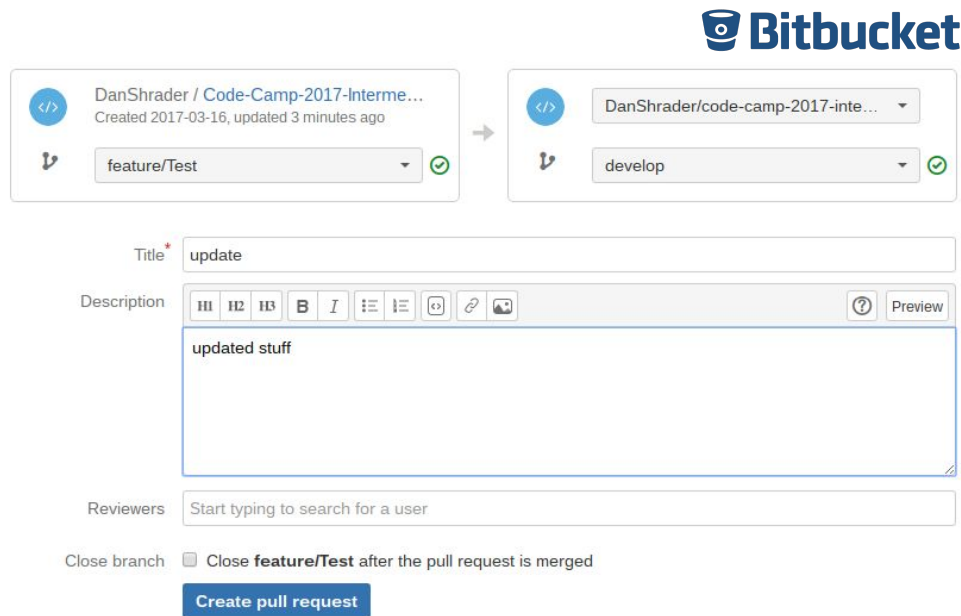
Keep in mind the following:

- Only contain commits related to the request
- Keep discussions respectful
- Add as much detail as needed if you were explaining it to someone without any knowledge of the issue.

PULL REQUESTS - BAD EXAMPLE

Vague requests will typically get denied.

From the example, there is no information on what happened.



The screenshot illustrates a Bitbucket pull request interface. At the top right is the Bitbucket logo. Below it, two panels show the source and target branches. The left panel shows the source branch as 'feature/Test' with a green checkmark. The right panel shows the target branch as 'develop' with a green checkmark. An arrow points from the source to the target. Below these panels is the pull request form. The 'Title' field contains the word 'update'. The 'Description' field contains the text 'updated stuff'. The 'Reviewers' field is empty with a placeholder text 'Start typing to search for a user'. At the bottom, there is a checkbox labeled 'Close branch' with the text 'Close feature/Test after the pull request is merged' and a blue button labeled 'Create pull request'.

Bitbucket

DanShrader / Code-Camp-2017-Interme...
Created 2017-03-16, updated 3 minutes ago

feature/Test

DanShrader/code-camp-2017-inte...
develop

Title * update

Description

updated stuff

Reviewers

Start typing to search for a user

Close branch ☐ Close feature/Test after the pull request is merged

Create pull request

PULL REQUESTS - BETTER EXAMPLE

Here, there are details on what is going on.

Keep in mind the reviewer(s) might not know what you were trying to resolve.

If there is an issue number reference it!



Source: DanShrader / Code-Camp-2017-Interme...
Created 2017-03-16, updated 2 minutes ago
feature/Test

Target: DanShrader/code-camp-2017-inte...
develop

Title* A descriptive title would go here

Description

A general idea of the the changes would go here along with the reasons why you made those decisions if needed.

Bullet or number lists work great for this

Reviewers Start typing to search for a user

Close branch ☐ Close **feature/Test** after the pull request is merged

Create pull request

WORKFLOWS

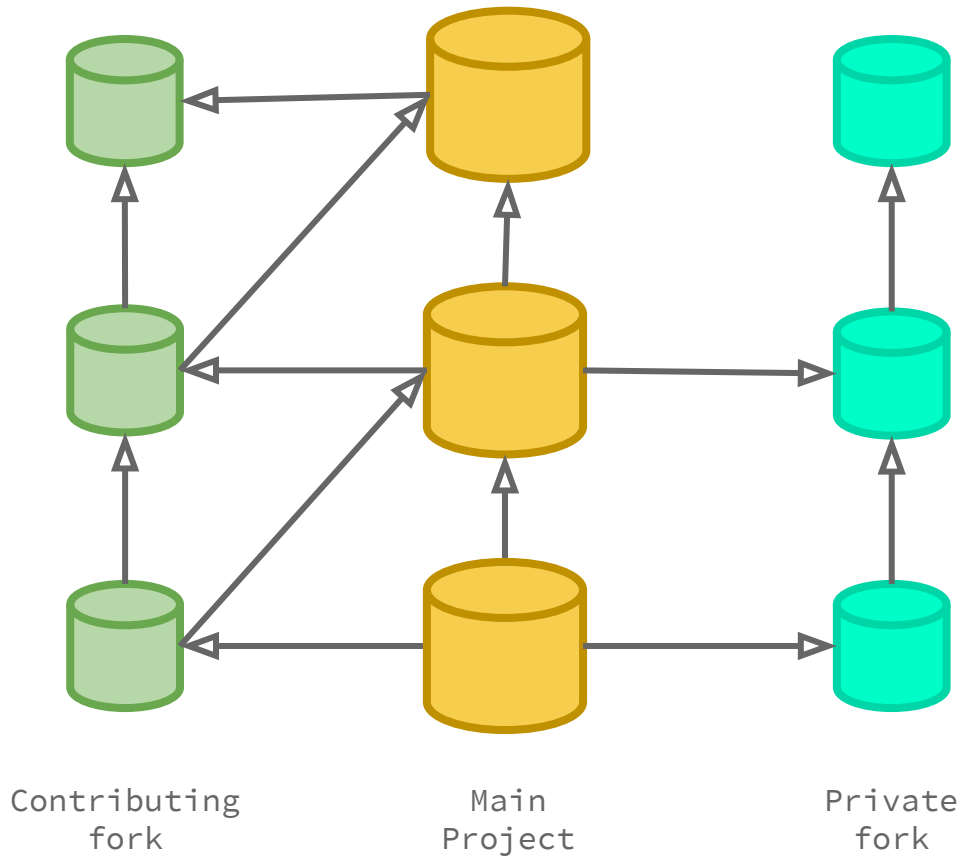
FORKING WORKFLOW

Used for public projects

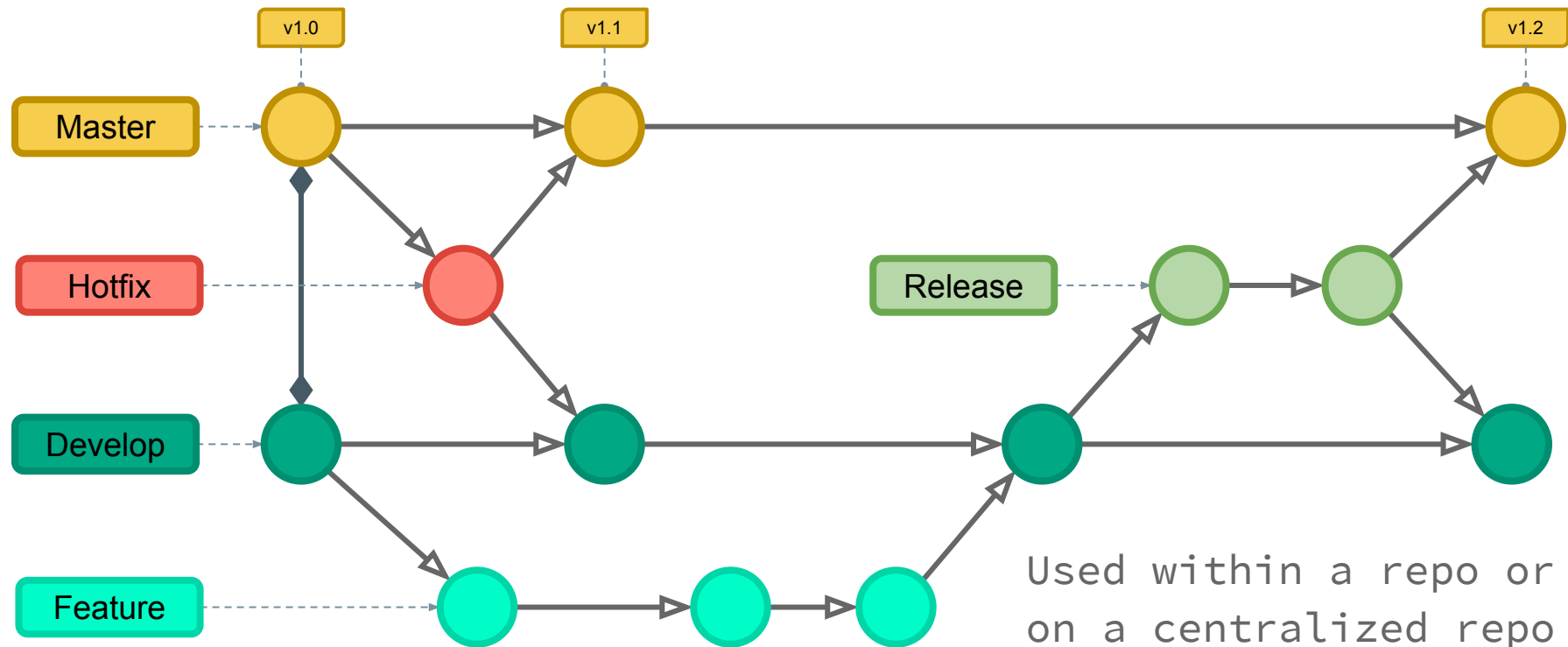
Permissions limited on main

Changes merged thru pull requests

Good for where you need an 'altered' version of a project



GIT FLOW / BRANCHING WORKFLOW



HOOKS

HOOKS - WHAT?

Scripts that run automatically when you interact with git

List-o-hooks

applypatch-msg

post-commit

update

pre-applypatch

pre-rebase

post-receive

post-applypatch

post-checkout

post-update

pre-commit

post-merge

pre-auto-gc

prepare-commit-msg

pre-push

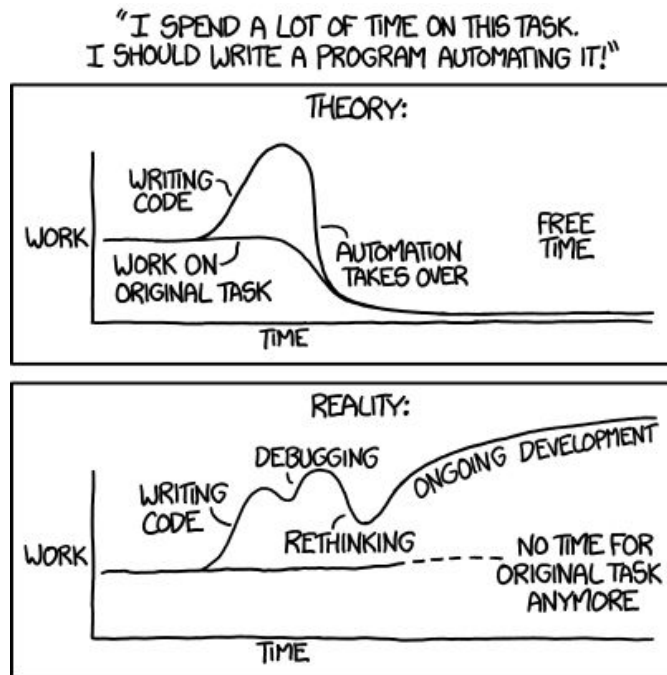
post-rewrite

commit-msg

pre-receive

HOOKS - TWO TYPES, BUT THEN THERE IS C...

- Client Side
- Server Side
 - Continuous delivery
 - Continuous integration
 - Continuous _insert Buzz word_



HOOKS - CLIENT SIDE

- Local only - they are NOT saved in git
 - This means that they are only on the local machine
- Not automatically shared
- Can deploy code - in certain cases this is right
- Great for checking your stuff
 - Spelling
 - Syntax formatting
 - Commit message structure

HOOKS - SERVER SIDE

- Work great for enforcing things
 - Can prevent force pushes
 - Commit messages
- Apply to everybody
- They can kick off deployments
 - Allows developer machines to be 'dumb' boxes
 - Can reduce your licensing fees for some software
 - Will reduce configuration times

HOOKS - ON BRANCHES

Example of a local post-merge hook that I use to do deployments from my computer.

Simply name the file:
post-merge.sh

```
#!/bin/bash
branch_name=$(git branch | grep "*" | sed "s/\* //" )
reflog_message=$(git reflog -1)
merged_branch_name=$(echo $reflog_message | cut -d" " -f 4 | sed "s/://")

# echo $branch_name
# echo $reflog_message
# echo $merged_branch_name

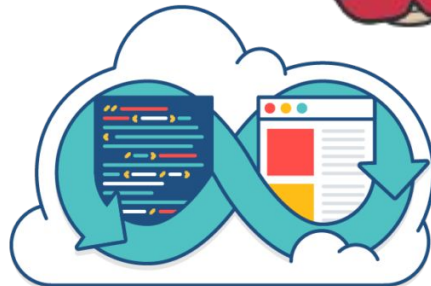
if [[ $branch_name == "develop" ]]; then
    echo "Merged to Develop branch - running script!"
    Your_Dev_Branch_Script_Here.sh
fi

if [[ $branch_name == "awesome" ]]; then
    echo "Merged to awesome branch - running script!"
    Your_Awesome_Branch_Script_Here.sh
fi
```

HOOKS - SERVER SIDE WITH CONTINUOUS...

Lots of options limited to your needs and creativity

- Testing
- Deployments / Builds
 - Different environments for different stages
- Announcements




 **Bamboo**



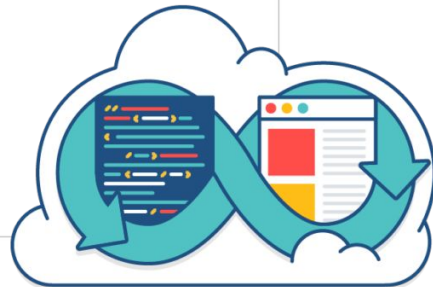
HOOKS - SERVER SIDE WITH PIPELINES

Almost the world's simplest example

 0104dab 2017-03-16 ▾ [Full commit](#)

[Blame](#) [Raw](#) [Edit](#) ▾

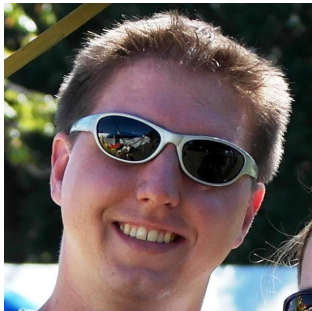
```
1 image: node:4.6.0
2
3 pipelines:
4   default:
5     - step:
6       script:
7         - echo "This is for items that don't match the pattern"
8   branches:
9     master:
10      - step:
11        script:
12          - echo "This runs on the master branch."
13     feature/*:
14      - step:
15        script:
16          - echo "This runs on any feature/* branch."
17     release/*:
18      - step:
19        script:
20          - echo "This runs on any release/* branch."
```



APPENDIX

- <https://www.atlassian.com>
- <https://www.github.com>
- <https://git-scm.com>
- <https://gitlab.com>
- <https://www.digitalocean.com/community/tutorials/how-to-use-git-hooks-to-automate-development-and-deployment-tasks>
- <https://xkcd.com/1319/>
- <https://github.com/brianstorti/git-hooks/blob/master/hooks/post-merge>
- <https://blog.axosoft.com/>
- <http://meldmerge.org/>
- <http://kdiff3.sourceforge.net/>

ABOUT ME



I've been dabbling with code since the late 90's and currently work with Microsoft Business Intelligence products. When given the chance, I write single page applications in JavaScript. I'm also a stickler for automating as much of the day to day tasks as I can. When I'm not coding, I enjoy camping and hanging out with my wife and sons.

Email: Dan@ShraderLand.com