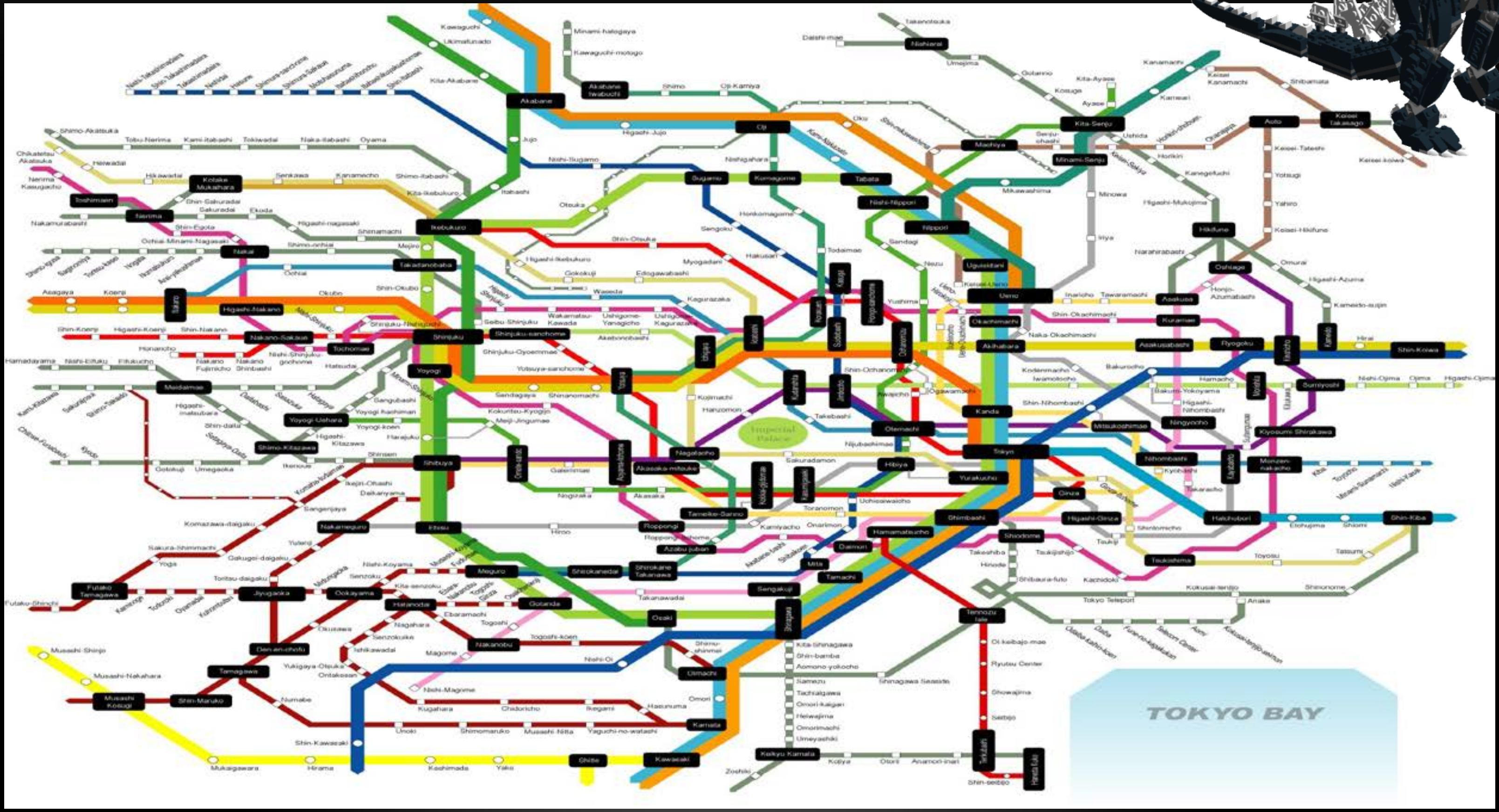
CHOOSING GIT WORKFLOWS THAT MAKE SENSE



WHO I AM? (SELF LOVE SLIDE)

- @TimCunningham71 twitter
- Vice President of Operations at IDMI
- Adobe Community Professional
- Codebass Radio
- Bolttalks.com
- CFmumboJumbo.com

WHAT IS GIT?

- Distributed Revision Control System
- Fast
- Scalable
- Powerful beyond belief

BUILDING BLOCKS

- All the tools you need to work how you wish
- Not a hammer – A hardware store
- Understand the principles of git rather than following things by rote.

OBJECT

- The unit of storage in git.
- It is uniquely identified by the SHA1 of its contents.
- Objects can not be changed

REF OR REFERENCE

- A 40-byte hex representation of SHA1
- A name the denotes a particular object

REPOSITORY

- The sum total of:
  - All the refs
  - All the objects
- Every repository can send and receive objects between other repositories

CLONE

- **Makes exact copy of another repository.**
- **Includes all branches and all the history.**
- **Creates remote-tracking branches for each branch in the source repository.**
- **Checks out an initial branch.**
- **Sets things up so that a “git fetch” will update all remote tracking branches**
- **Sets things up so that a “git pull” will merge the remote master branch into the current master branch**

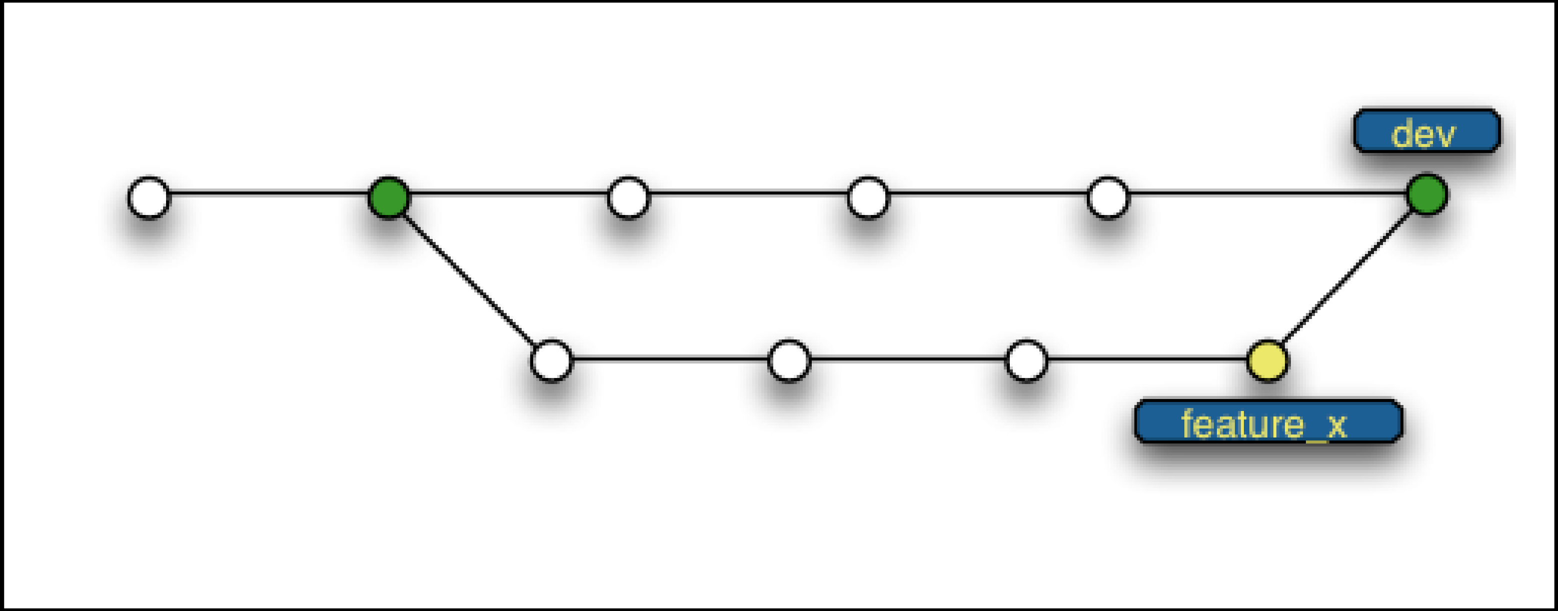
COMMITS

- A single point in the git history
- Entire history of project is represented as a set of interrelated commits.

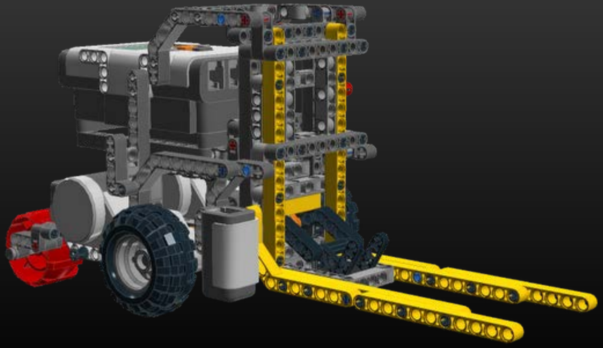
BRANCHES

- Branch is an active line of development.
- Can have unlimited number of branches.
- When you check out a branch it changes all your local files (Working Tree) to match the last work done on that branch (HEAD)

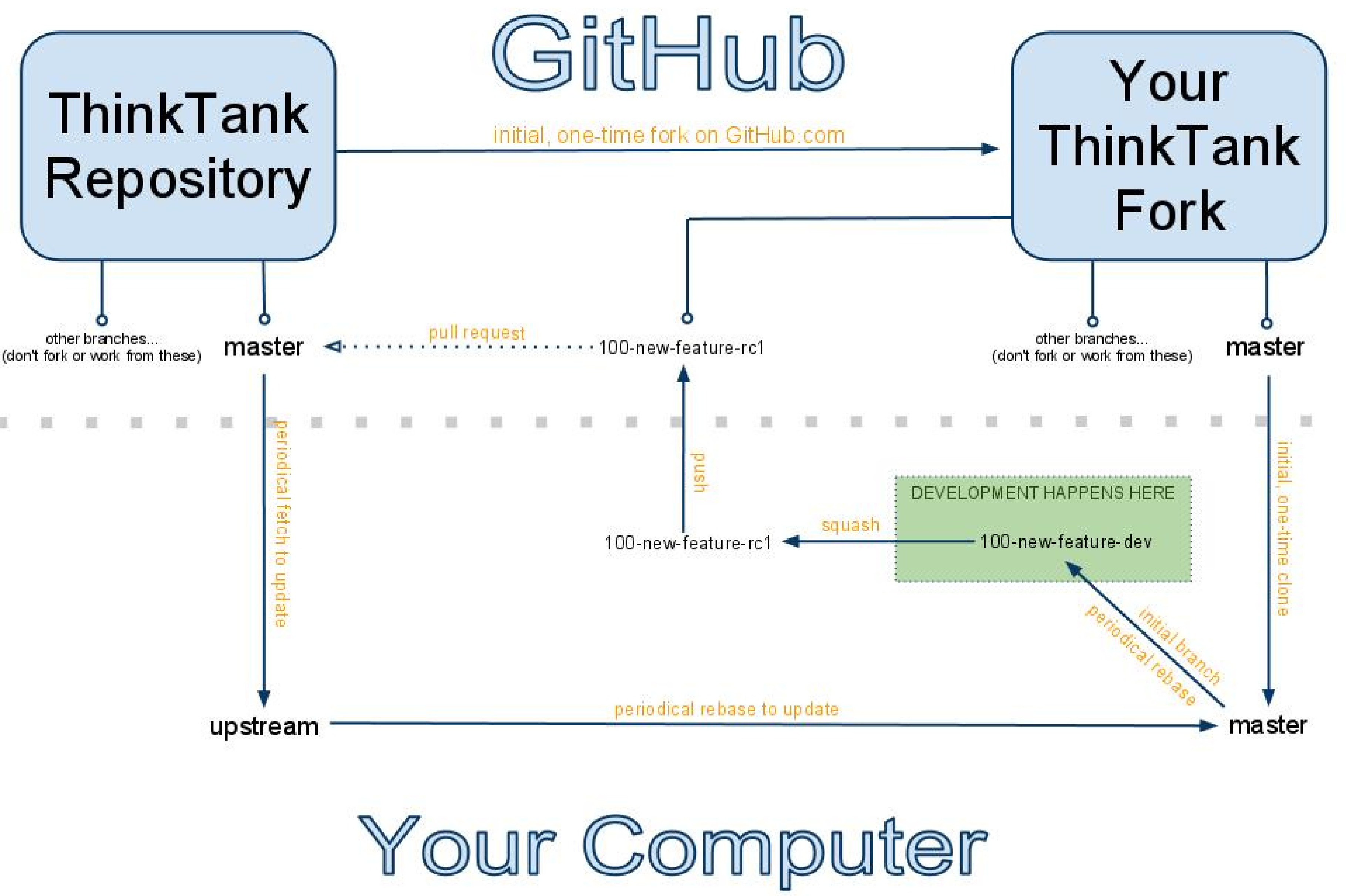
BRANCHING



FORK



- Similar to a clone
- The original repository is not considered the parent.
- Can not push to the original repository
- Considered a spin-off repo that you may do divergent work from the original repository.
- More common in Git Hub open source type projects



GitHub

ThinkTank Repository

initial, one-time fork on GitHub.com

Your ThinkTank Fork

other branches...
(don't fork or work from these)

master

pull request

100-new-feature-rc1

other branches...
(don't fork or work from these)

master

periodical fetch to update

push

DEVELOPMENT HAPPENS HERE

squash

100-new-feature-rc1

100-new-feature-dev

initial, one-time clone

initial branch

periodical rebase

upstream

periodical rebase to update

master

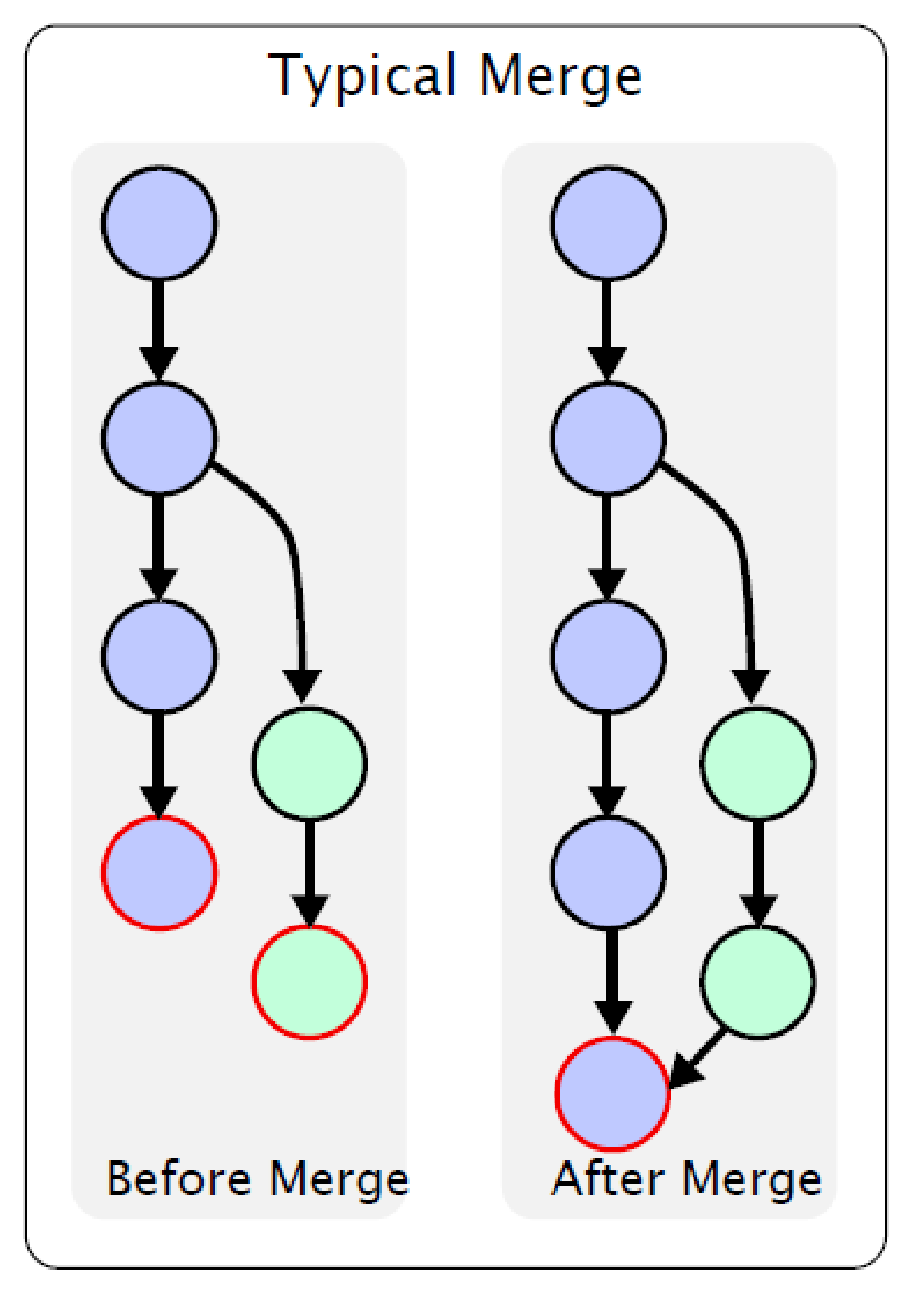
Your Computer

MERGE

- Bring the contents of another branch into the current branch
- Merge can be from another local branch or a remote branch
- Merge identifies changes made to both branches since they diverged and applies all those changes together.
- If changes conflict, manual intervention required to complete the merge.

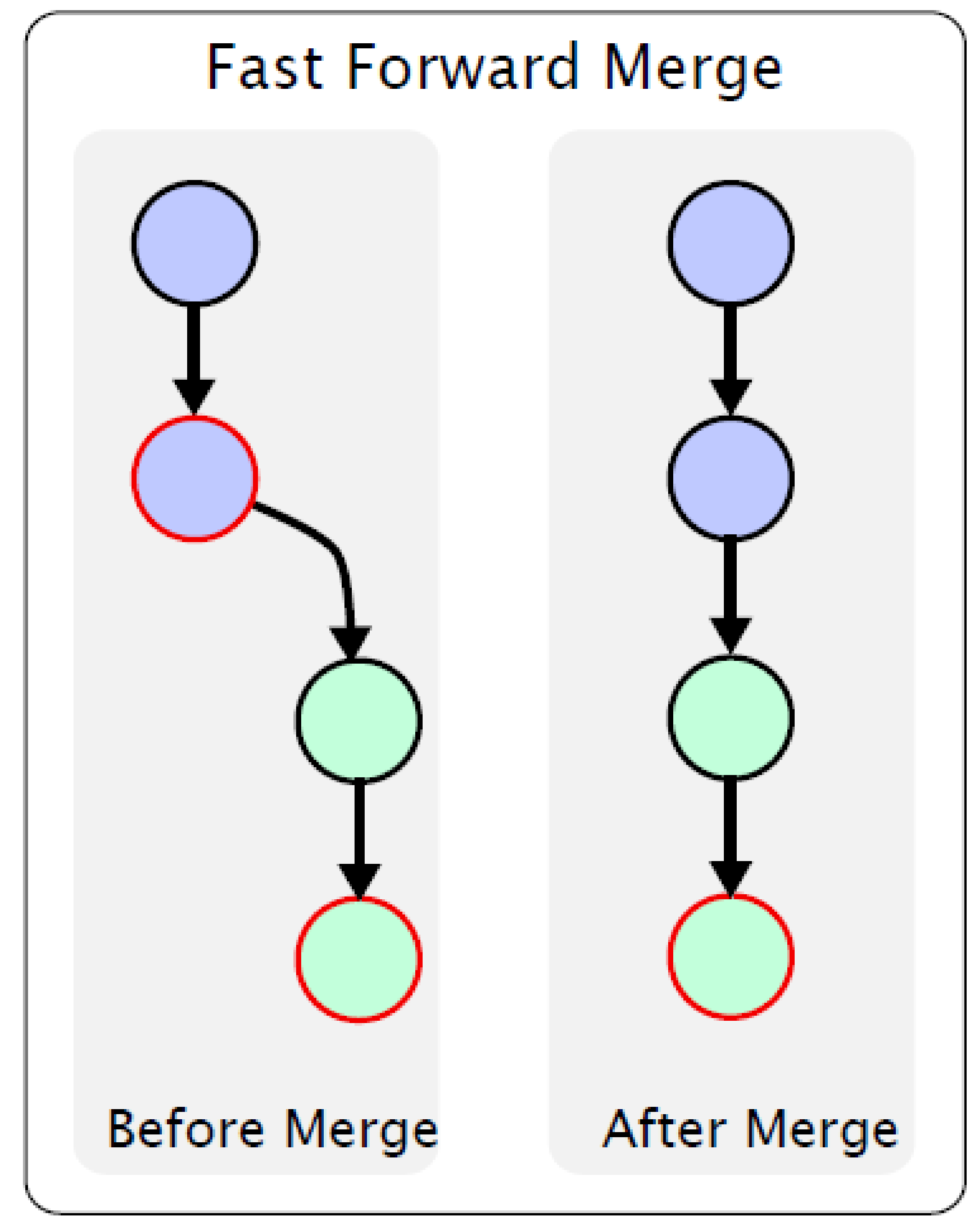
MERGE COMMIT

- **“Merge Commit” if the source branch has changed since you branched off of it a new commit is created representing the result of the merge. HEAD is moved to this new commit**


Typical Merge
Before Merge
After Merge

FAST FORWARD MERGE

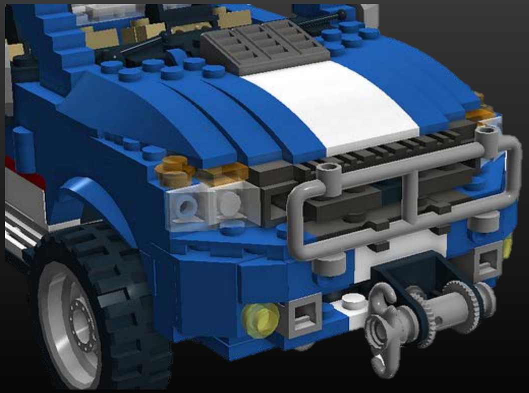
- **A “fast-forward” merge. If nothing has changed on the source branch since you branched off of it, when you attempt to merge your branch back to the source it will not create a new commit. HEAD will just point the source branch to the latest commit of the changed branch.**


Fast Forward Merge
Before Merge
After Merge

FETCH

- **Finds out which objects are on the remote repository that are missing on your local repository and download them.**
- **Doesn’t change anything in your work tree.**

PULL



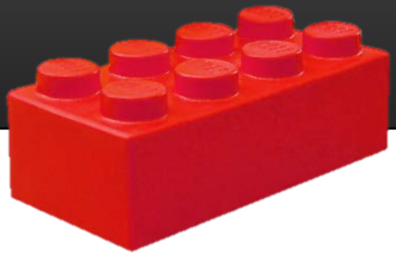
- Fetches a remote branch
- Automatically tries to merge the remote branch into the current branch

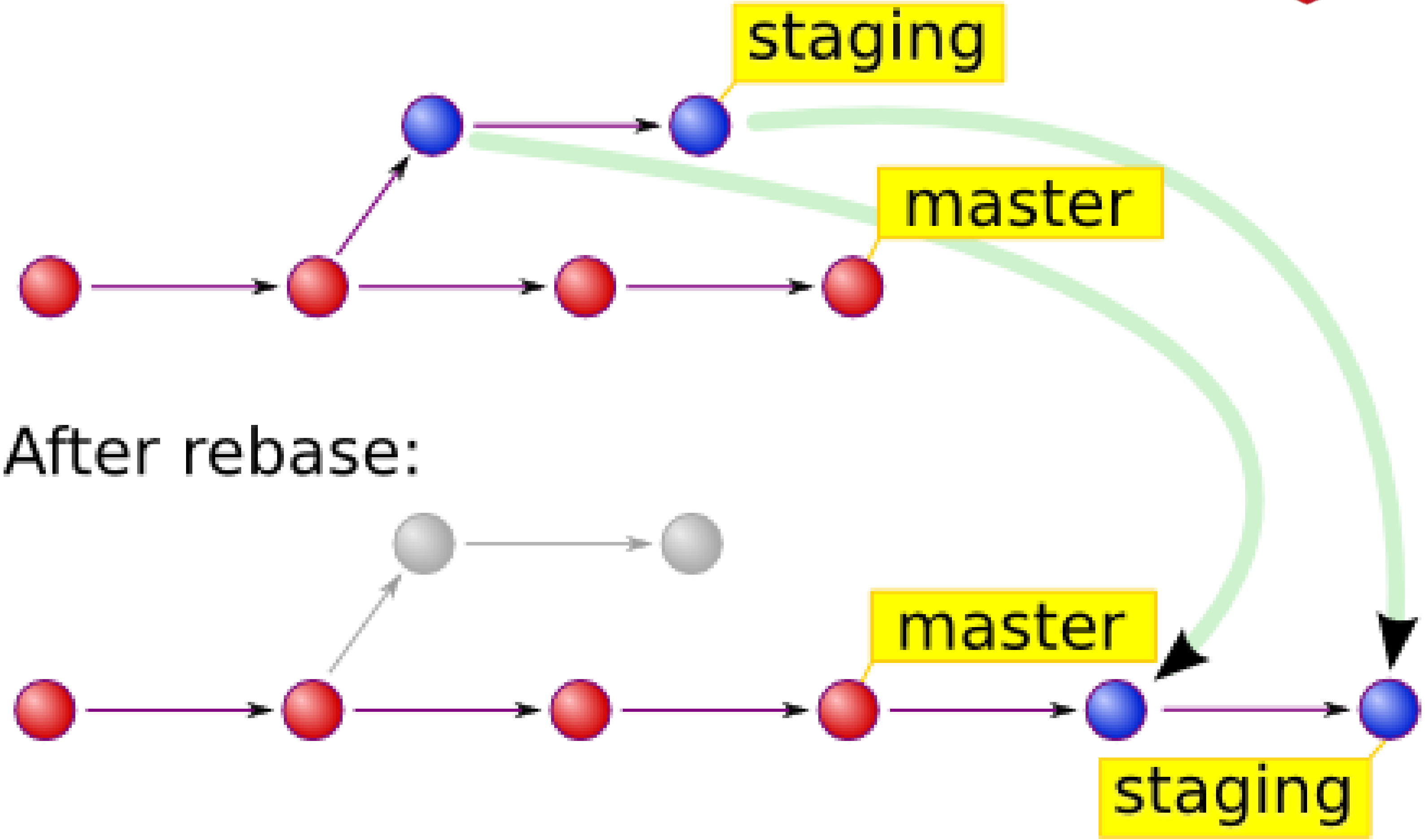
PUSH

- Updates the remote branch with the latest commits from your local branch
- Remote branch must be an ancestor of your local branch

REBASE

- Takes all your changes in your current branch that are not in the remote branch saves them to temporary area.
- Current branch is reset, or made identical to, the remote (upstream branch)
- HEAD is moved to last commit
- Your changes are taken from the temporary area and applied one by one to this new starting base
- HEAD is moved forward to last commit of your changes.





staging

master

After rebase:

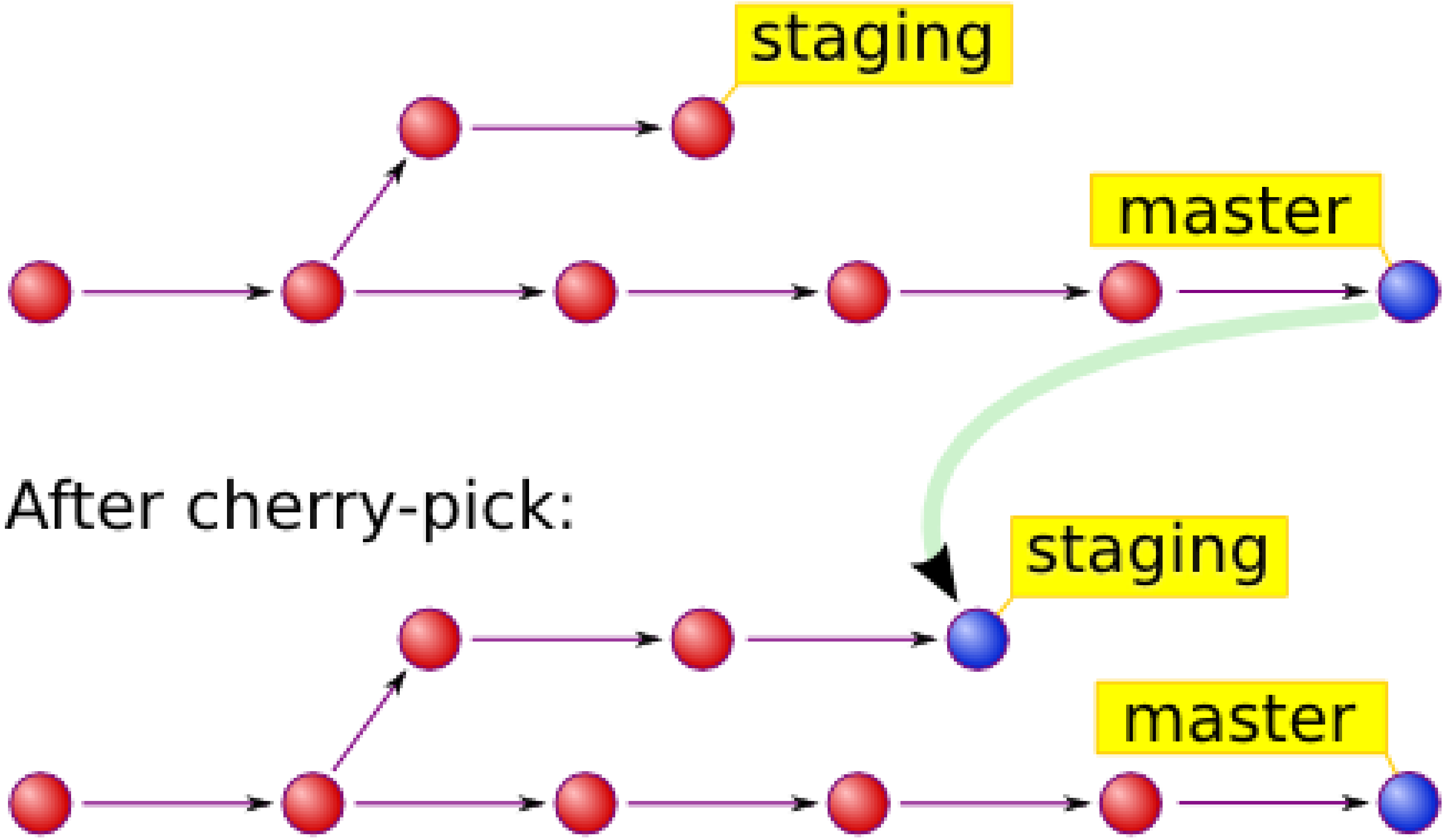
master

staging

CHERRY PICKING

- **Take a subset of commits and record them as a new series of changes on top of a different codebase.**

CHERRY PICK



TAGS

- **A reference pointing to a specific point in the code history.**
- **Generally used to mark release point 9.0.1 etc…**

HOW GIT COMMUNICATES

- Local Protocol
  - Talks to your local drives
  - Can talk to shared drives
  - Shared folder on VPN etc.

HOW GIT COMMUNICATES

- Git Protocol
  - Fast
  - Port 9418
  - *NO* authentication

HOW GIT COMMUNICATES

- HTTP / HTTPS
    - Easy to set up (on Linux)
    - Harder to set up on IIS (Bonobo, git-dot-aspx)
    - Slow

HOW GIT COMMUNICATES

- SSH
  - Most common set up
  - Secure
  - Simple to set up on Linux
  - Hard to setup on Windows

HOW WE COMMUNICATE W/ GIT

- Outsource that headache
- Git Hub
- Bitbucket

HOW I PROMOTE TO PRODUCTION

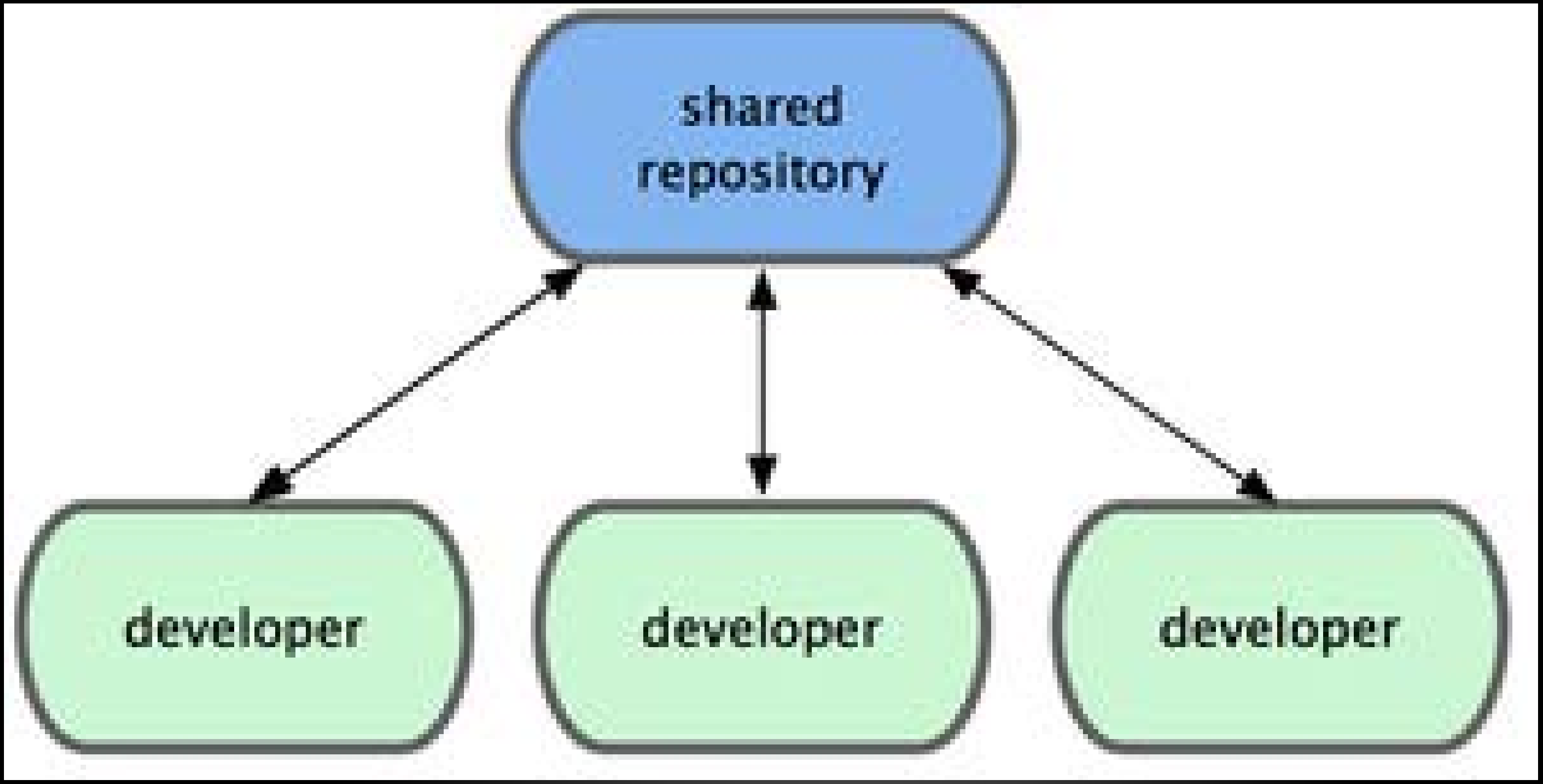
- Keep production ready code in a branch named master
- Have Git installed on production server
- C:\inetpub\wwwroot\myWebsiteName\.git
- Protect .git folder
- .cfm page that uses cfexecute to run .bat file
- .bat file changes to the website directory
- Resets the local repo match master branch on bitbucket
- Runs clean on repo
- Have another .cfm to do rollback to last commit

SHOW ME THE WORKFLOWS!

CENTRALIZED WORKFLOW

- Single Branch
- Pull from master
- Do work
- Commit
- Pull from master to merge any changes made while you were doing work
- Push to master

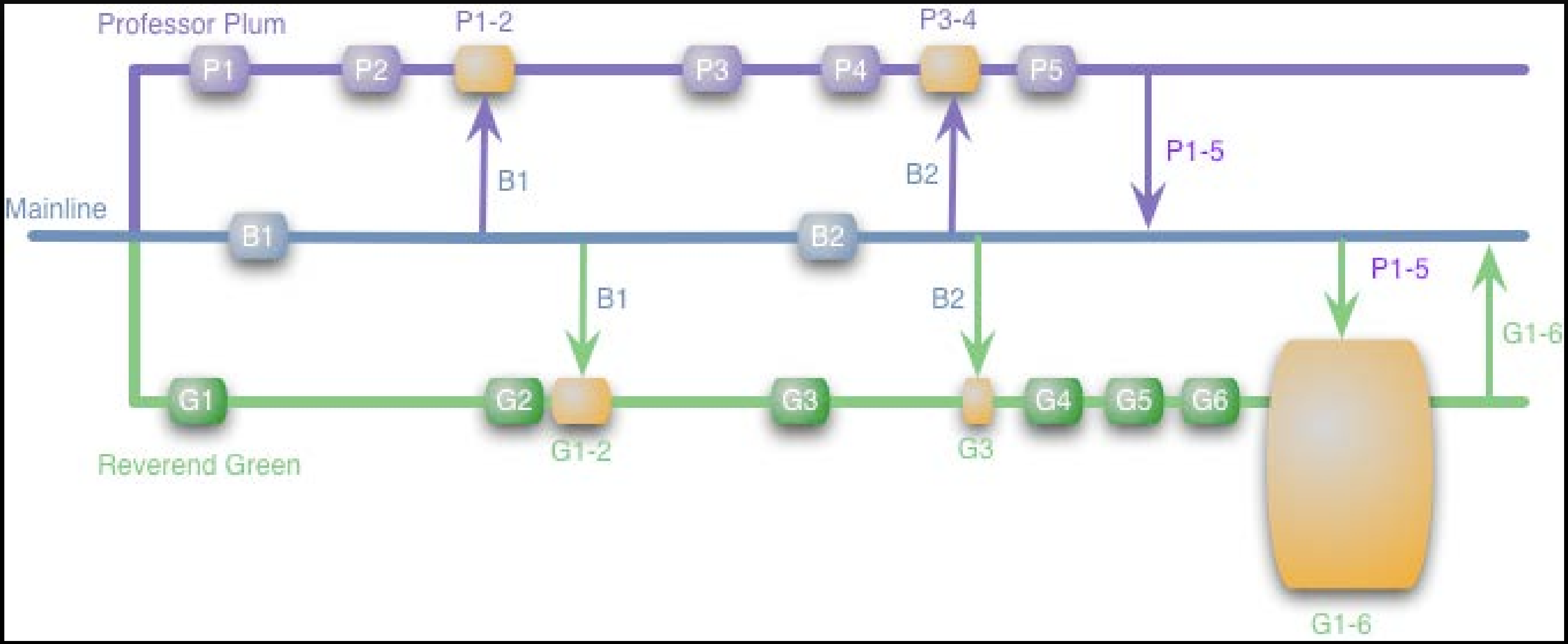
CENTRAL REPO



FEATURE BRANCH FLOW

- Master = production
- Create Feature branches off of Master
- Do work on Feature Branch
- Merge Features into master

FEATURE BRANCH FLOW

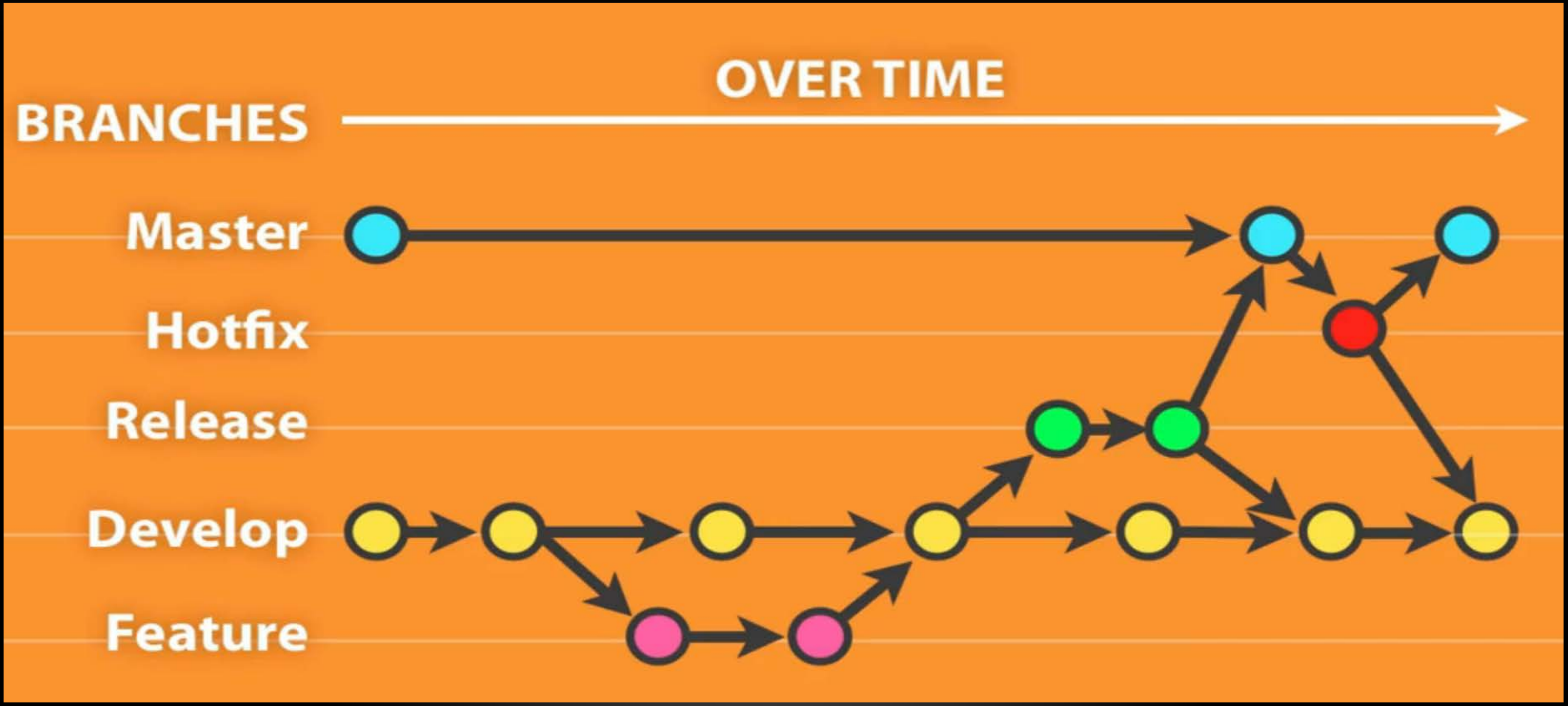


GIT FLOW

- Master Branch
- Develop Branch
- Feature Branches off of Develop
- Features Merged into Develop
- Develop branched into Release branch
- UAT done on release branch and any fixes need
- Release branch merged back to DEV & master

GIT FLOW

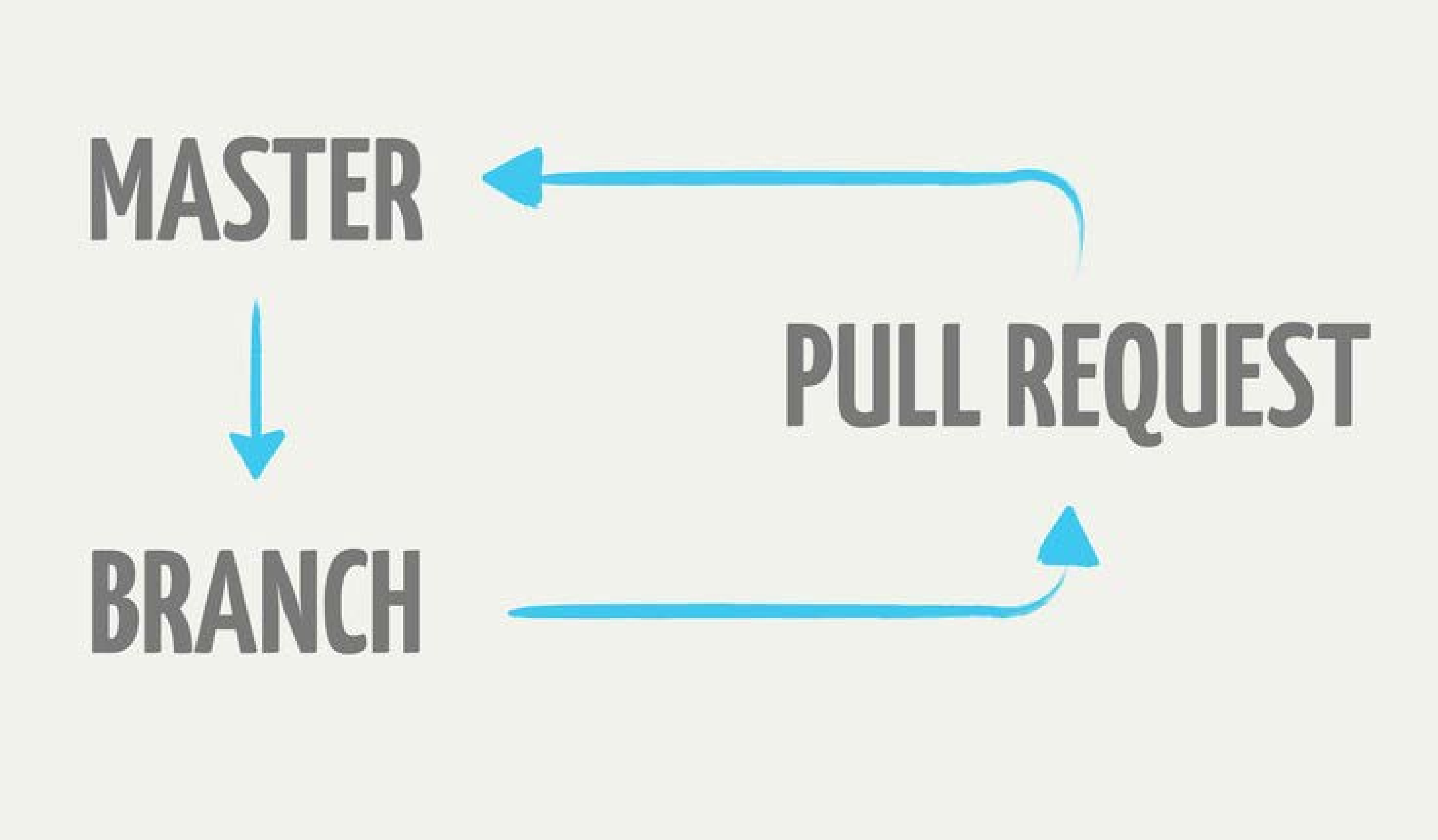
- http://nvie.com/posts/a-successful-git-branching-model/



GIT HUB FLOW

- Branch off master to create Topic Branches
- Anything in the master branch is deployable
- Push to named branches constantly
- Submit pull request to pull into master
- Code review
- Ship it!

GIT HUB FLOW

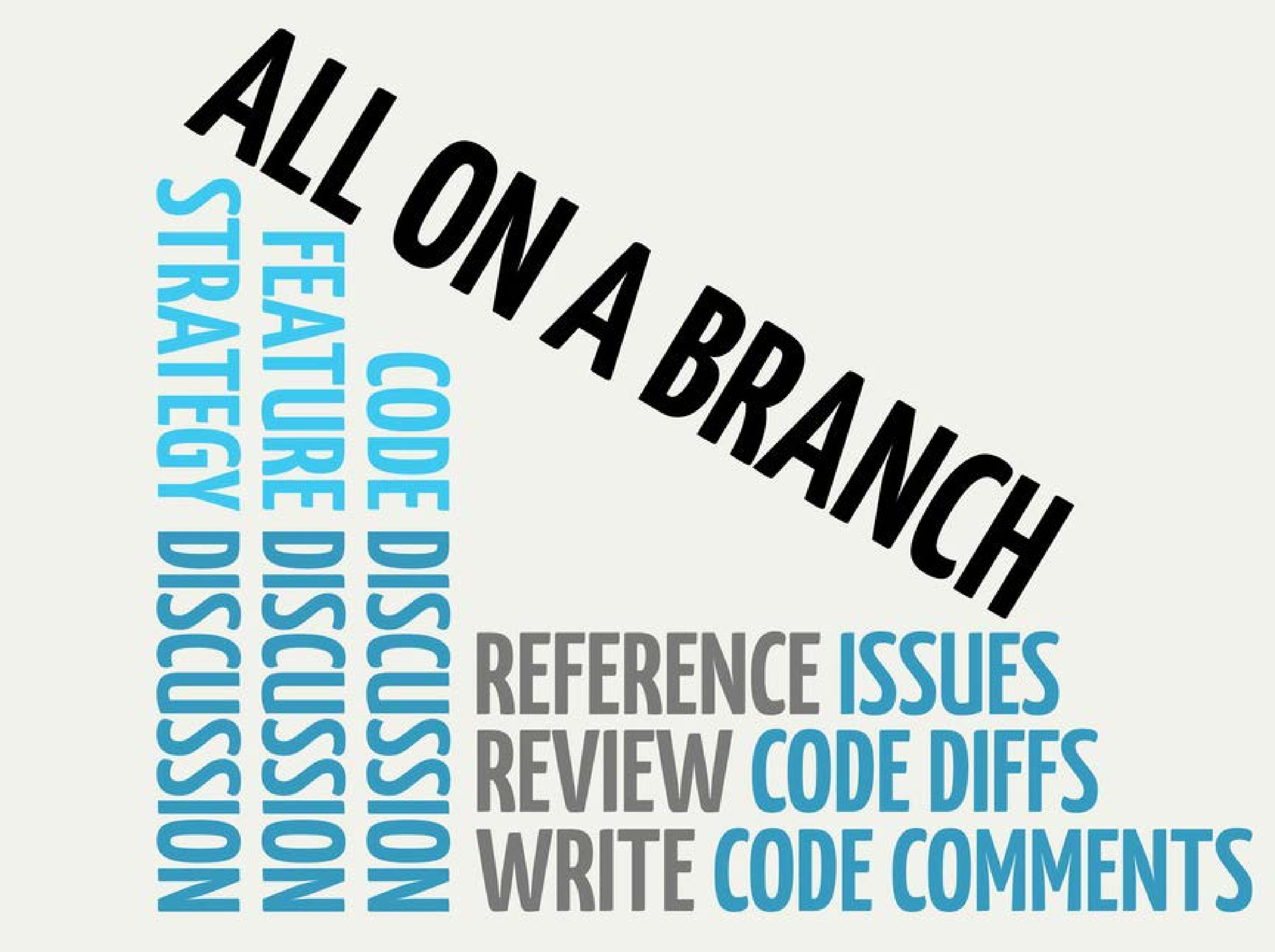


MASTER

BRANCH

PULL REQUEST

GIT HUB FLOW



ALL ON A BRANCH

STRATEGY DISCUSSION

FEATURE DISCUSSION

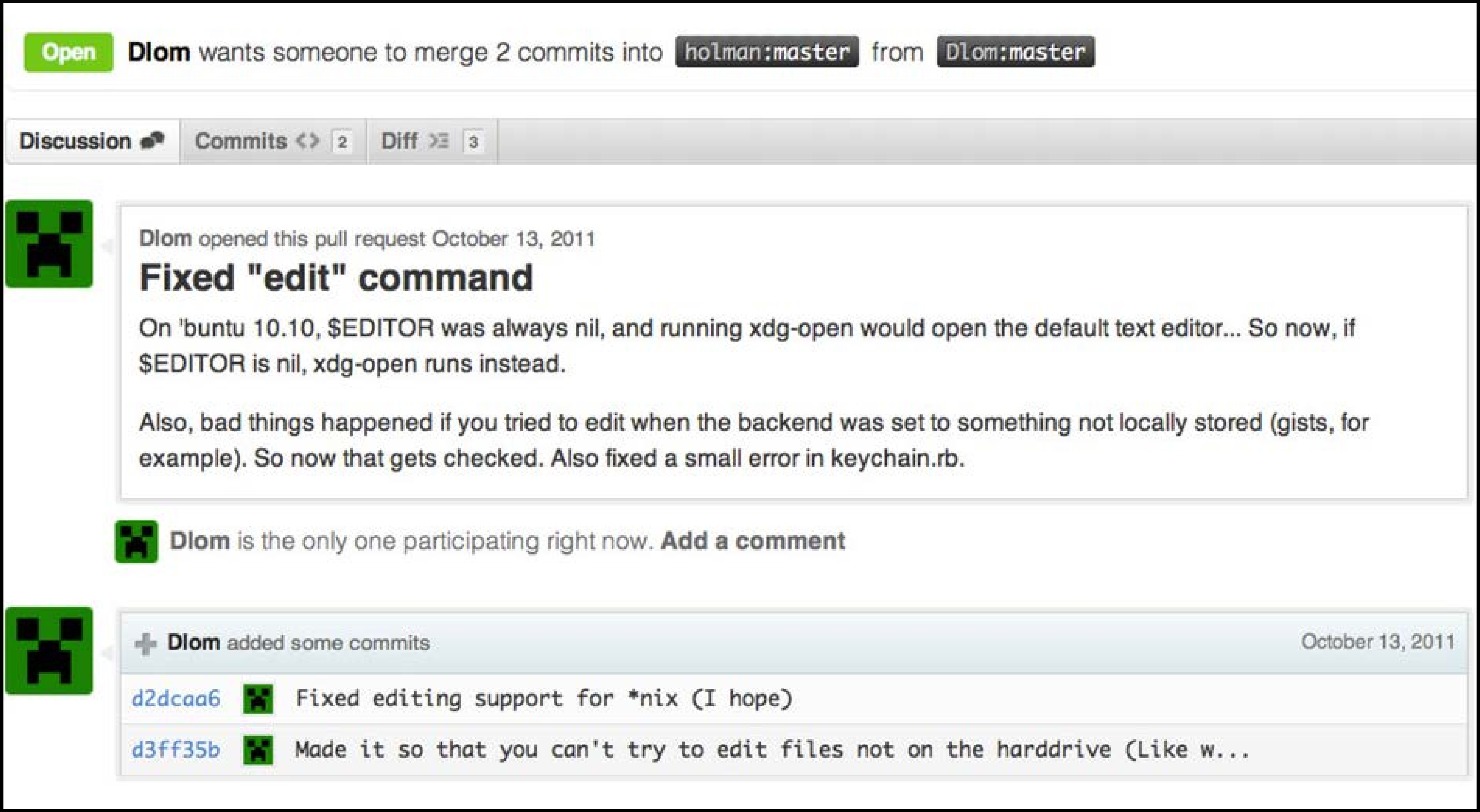
CODE DISCUSSION

REFERENCE ISSUES

REVIEW CODE DIFFS

WRITE CODE COMMENTS

GIT HUB FLOW



Dlom wants someone to merge 2 commits into `holman:master` from `Dlom:master`

Dlom opened this pull request October 13, 2011

Fixed "edit" command

On 'buntu 10.10, $EDITOR was always nil, and running xdg-open would open the default text editor... So now, if $EDITOR is nil, xdg-open runs instead.

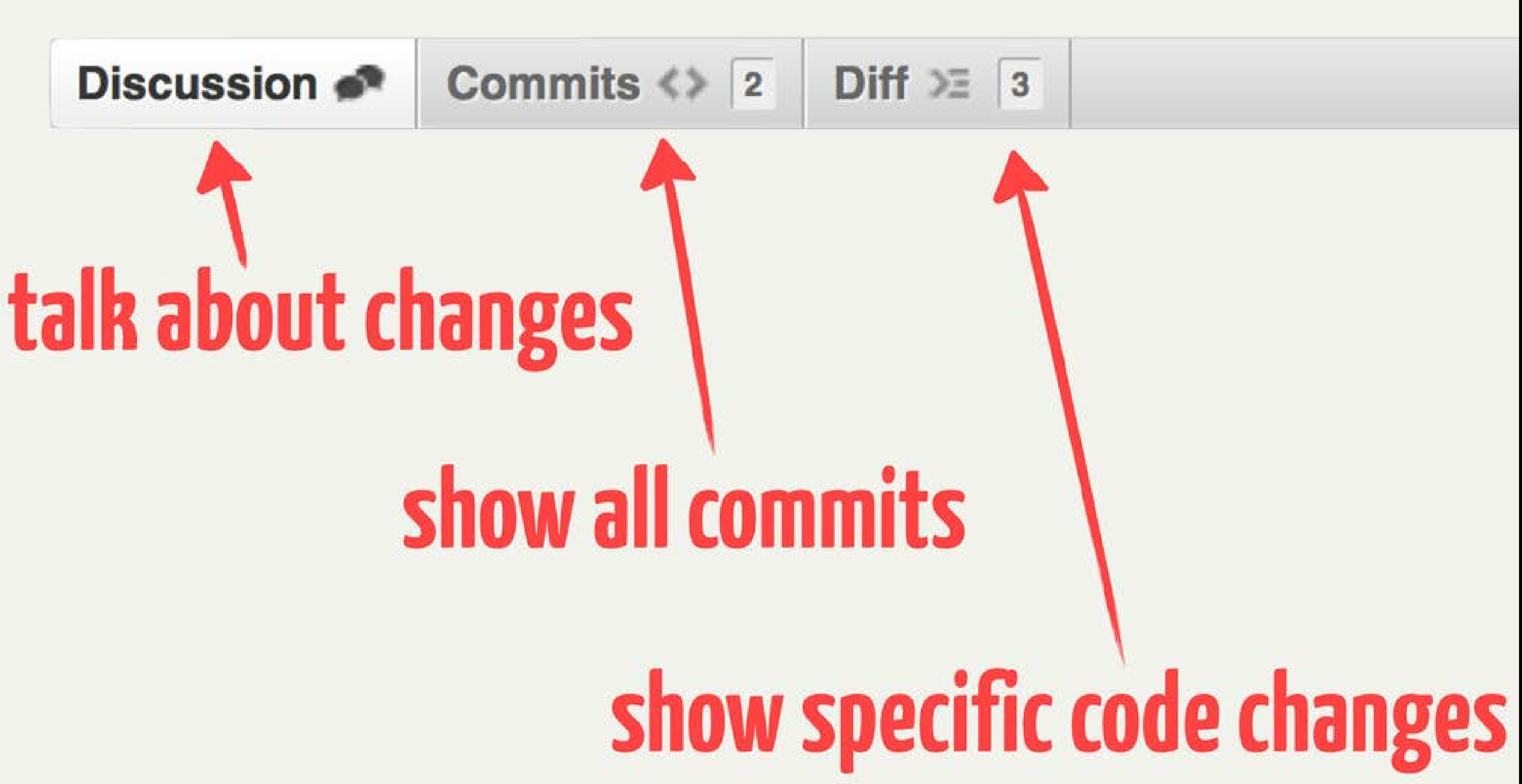
Also, bad things happened if you tried to edit when the backend was set to something not locally stored (gists, for example). So now that gets checked. Also fixed a small error in keychain.rb.

Dlom is the only one participating right now. **Add a comment**

Dlom added some commits October 13, 2011

```
d2dcaa6  Fixed editing support for *nix (I hope)
d3ff35b  Made it so that you can't try to edit files not on the harddrive (Like w...
```

GIT HUB FLOW

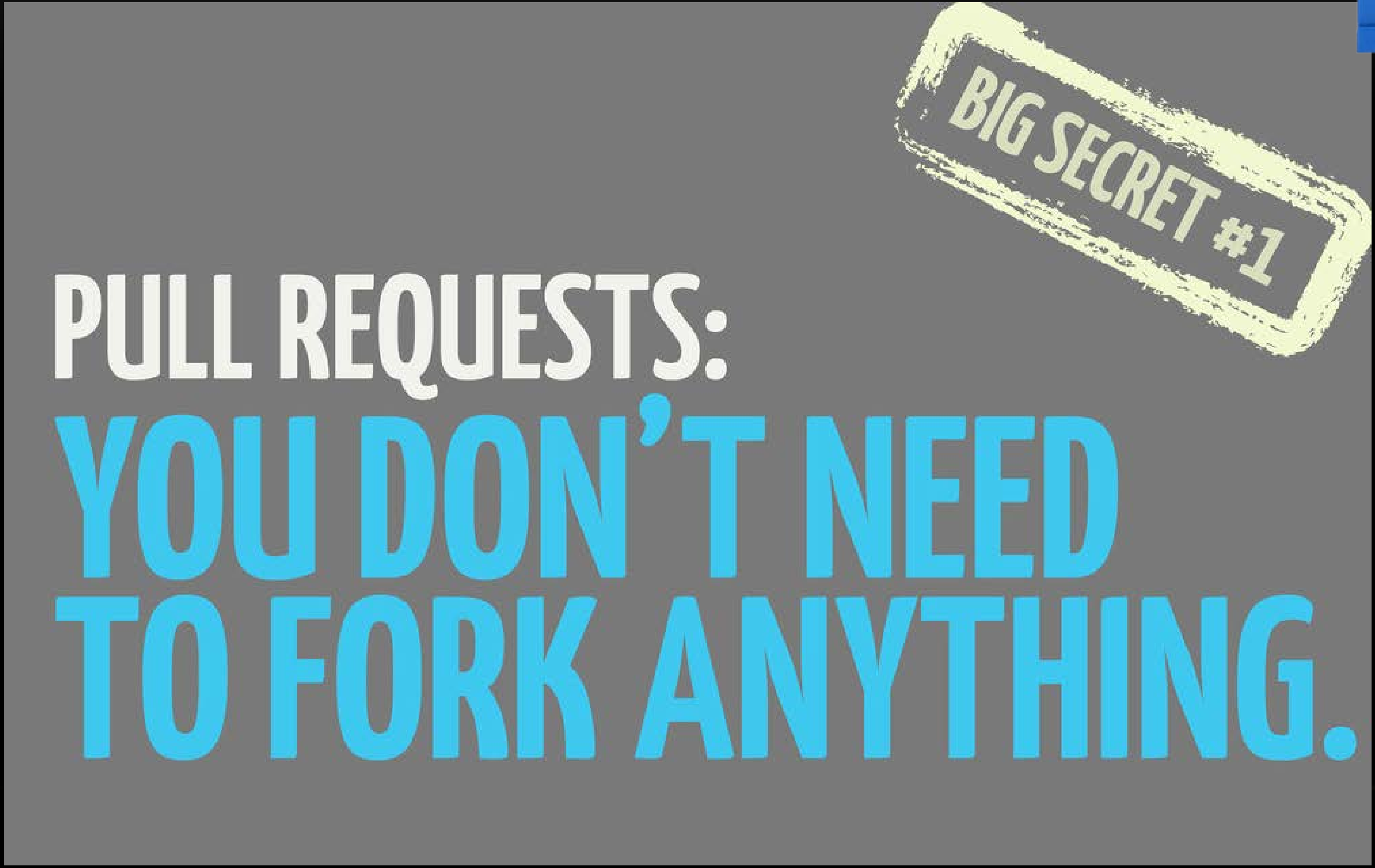


talk about changes

show all commits

show specific code changes

GIT HUB FLOW



BIG SECRET #1

PULL REQUESTS:

YOU DON'T NEED TO FORK ANYTHING.

CUSTOM FLOW

- Understand the principles of git
- Build *any* workflow you wish
- I do mean *any*
- You have the power to cut down any issue
- You also have the power to cut off your arm

NOTHING IS UNRECOVERABLE

LEGO FACEPALM

Because some things are just so retarded, even kids toys
will lose respect for you.

YOU ARE A TIME LORD!