Distributed Version Control with Git

Chris Dail - @chrisdail
http://chrisdail.com

Introduction

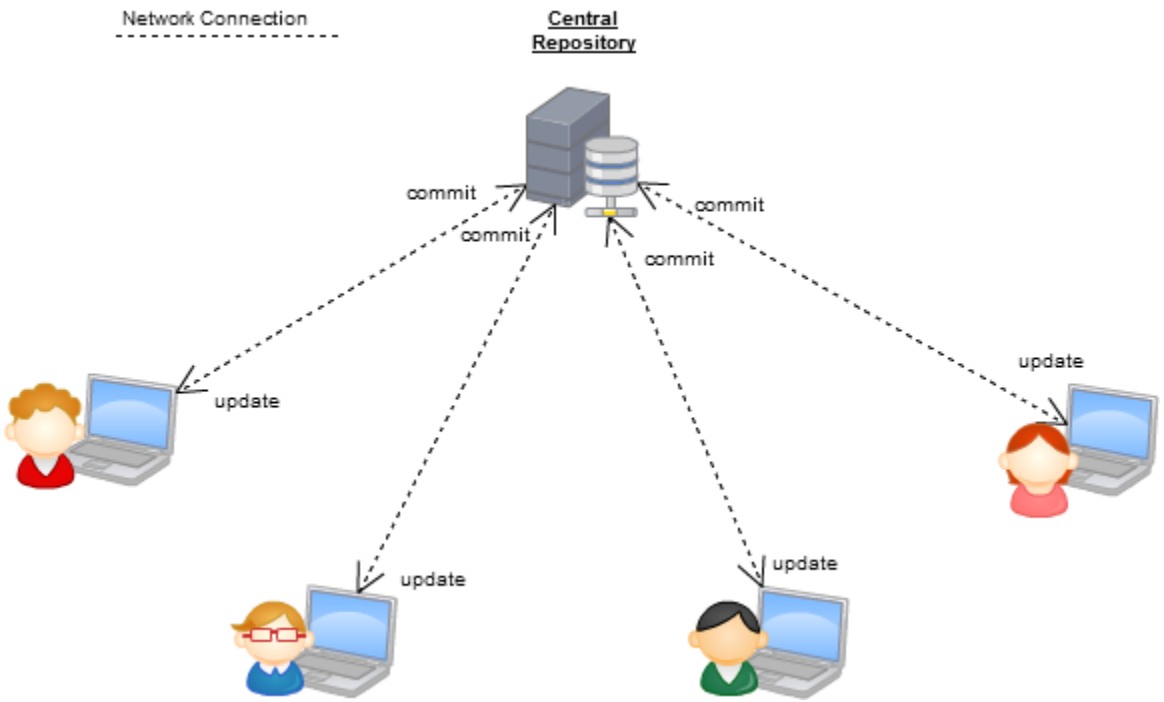
- About Me
  - Chief Software Architect at iWave Software LLC
  - Moncton, New Brunswick
  - Not a Git or version control "expert"
  - Version control "enthusiast"
  - Background with VSS, CVS, SVN and Git
- Poll for Version Control Software
  - CVS, Subversion
  - ClearCase, Perforce
  - VSS, Team Foundation Server
  - Git, Mercurial, Bazaar

- For the first 10 years of kernel maintenance, we literally used tarballs and patches, which is a much superior source control management system than CVS is, but I did end up using CVS for 7 years at a commercial company and I hate it with a passion. When I say I hate CVS with a passion, I have to also say that if there are any SVN (Subversion) users in the audience, you might want to leave. Because my hatred of CVS has meant that I see Subversion as being the most pointless project ever started. The slogan of Subversion for a while was "CVS done right", or something like that, and if you start with that kind of slogan, there's nowhere you can go. There is no way to do CVS right.
  - Linus Torvalds (http://www.youtube.com/watch?v=4XpnKHJAok8)

Introduction

- What is Version Control?
  - AKA Revision Control
  - Software developer's collaboration tool for tracking and controlling software changes
- Evolution
  - Initially Local Only - RCS
  - Client-Server
    - CVS, Subversion
    - ClearCase, VSS, Perforce, Team Foundation Server

Centralized Repository

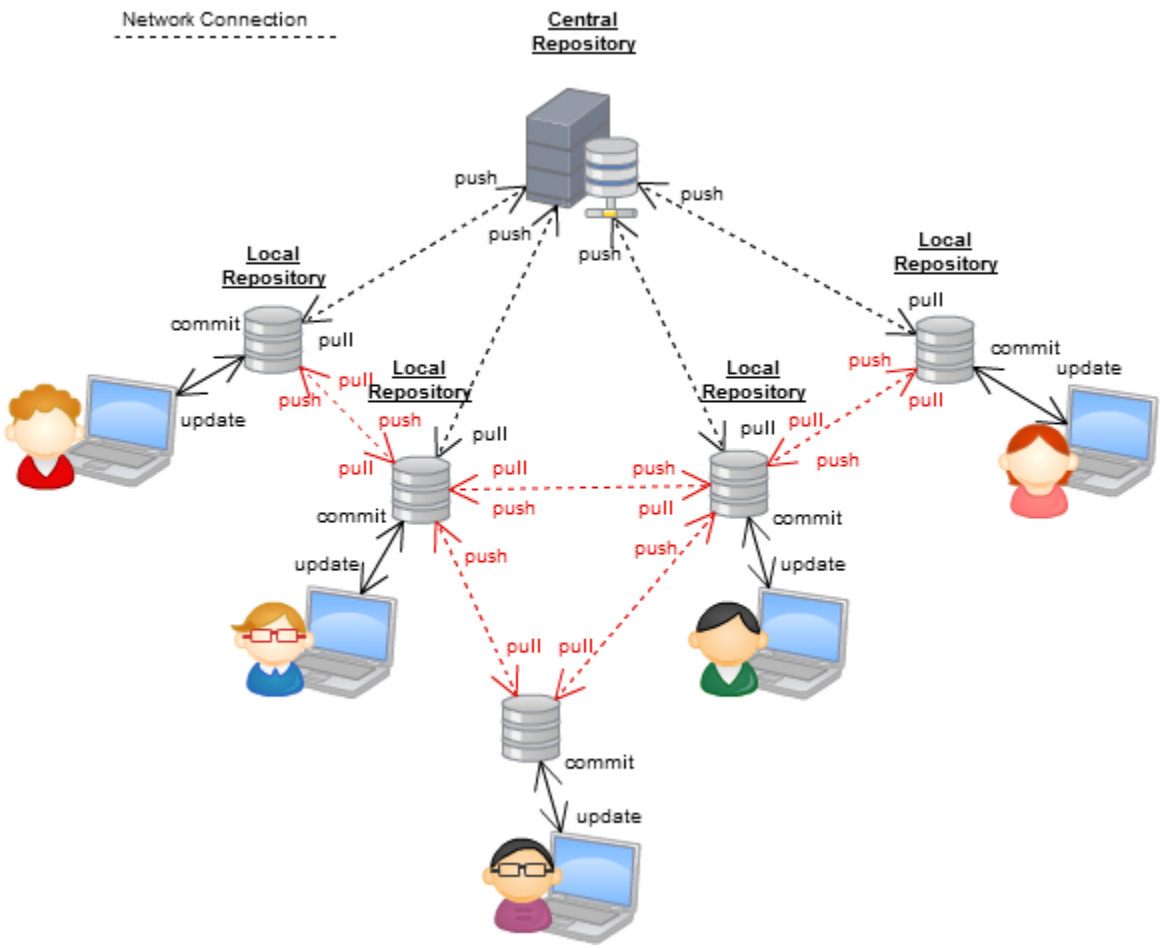


- Everyone works in a single repository
- Requires Network access for operations
- Obviously not conducive to decentralized teams

image src: http://docs.joomla.org

Introduction

- What is a DVCS?
  - Distributed (Decentralized) Version Control System
    - Git, Mercurial, Bazaar, Darcs and BitKeeper
  - No centralized server **required**
  - Peer to peer
  - Everyone has a complete local repository
    - Operations are **fast**
    - No network required
    - Encourages local version control



Decentralized Version Control

- Everyone has a repository
- Local work can be version controlled
- May have a central repository if desired
- **Merging is Critical**

image src: http://docs.joomla.org

- What is Git?
  - Distributed Version Control System
  - Created by Linus Torvalds in 2005
  - Created for Linux kernel development to replace BitKeeper
  - http://git-scm.com
- Why use Git?
  - Git is robust (SHA-1 identifier)
  - Git is Fast (Even for large trees)
  - Git encourages branching and merging

- Protocols
  - HTTP, SSH, Git, RSync, Local
- Efficient storage
  - Fast and Small
- Integration with CVS and Subversion
  - bi-directional access
- Repository Elements
  - Single .git directory containing the repository
  - Index - Staging area for commits

Git Tools

- Git - http://git-scm.com (Linux/Mac OS X)
- MsysGit - Git+Unix (Windows)
- TortoiseGit (Windows)
- IDE Plugins
  - Java: Eclipse (EGit/JGit), NetBeans (NBGit/JGit)
    - Add/Delete and Track versioned files
    - Comparison with previous versions
  - .NET: Visual Studio (GitExtensions)

Gitting Started

- Commands
  - git init - Create an empty git repository or reinitialize an existing one
  - git status - Show the working tree status
  - git add - Add file contents to the index
  - git rm - Remove files from the working tree and the index
  - git commit - Record changes to the repository
  - git log - Show commit logs
  - git diff - Show changes between commits, commit and working tree, etc
- Demo
  - Creating a repository

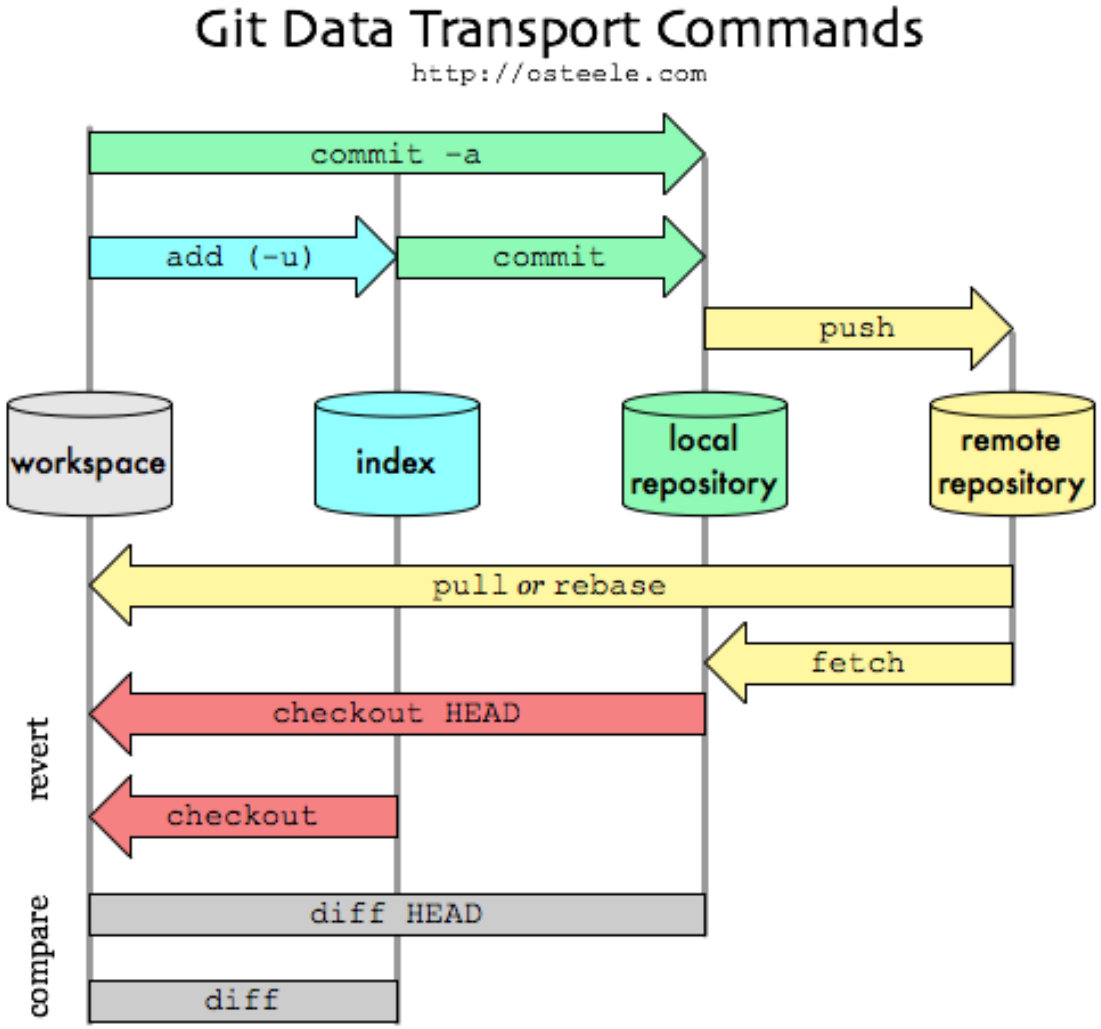
Branches

- Commands
  - git branch - List, create, or delete branches
  - git checkout - Checkout a branch or paths to the working tree
- Always commit to branches
  - No 'Trunk'
  - initial branch called master
- Demo
  - Creating a branch
  - Checking out a branch
  - git gui
  - gitk

Working with Remotes

- Commands
  - git clone - Clone a repository into a new directory
    - git clone <repo> <dir>
  - git fetch - Download objects and refs from another repository
  - git pull - Fetch from and merge with another repository or a local branch
  - git push - Update remote refs along with associated objects
- Demo
  - Cloning
  - Tour of .git

Git Data Transport



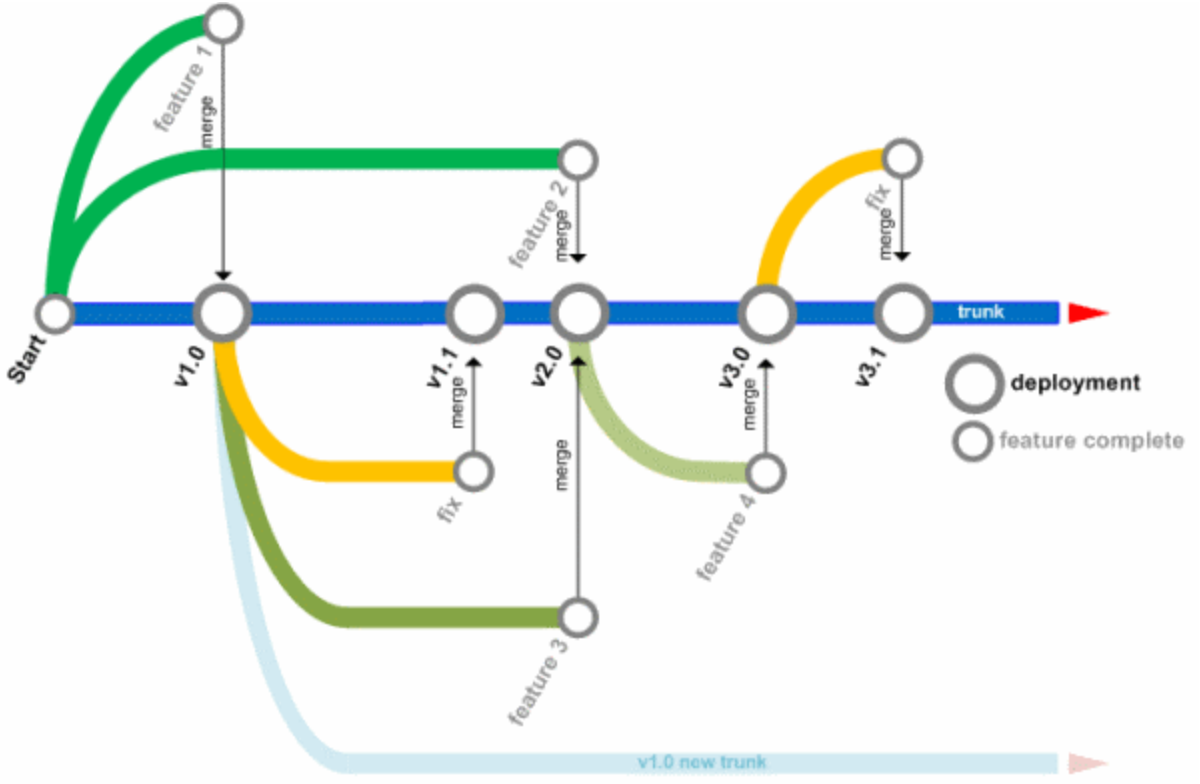
Branching in…

- CVS is possible
- Subversion is easy
- Git is easy

Merging in …

- CVS is hard
- Subversion is hard
- Git is **easy**

http://very-bored.com/pics/weirdtrees/weird-trees-8.jpg



Traditional Branch Management

- Always merging to the trunk
- Another style, always work on trunk and merge to branches
- Merges only happen once
- "feature complete"

image source:

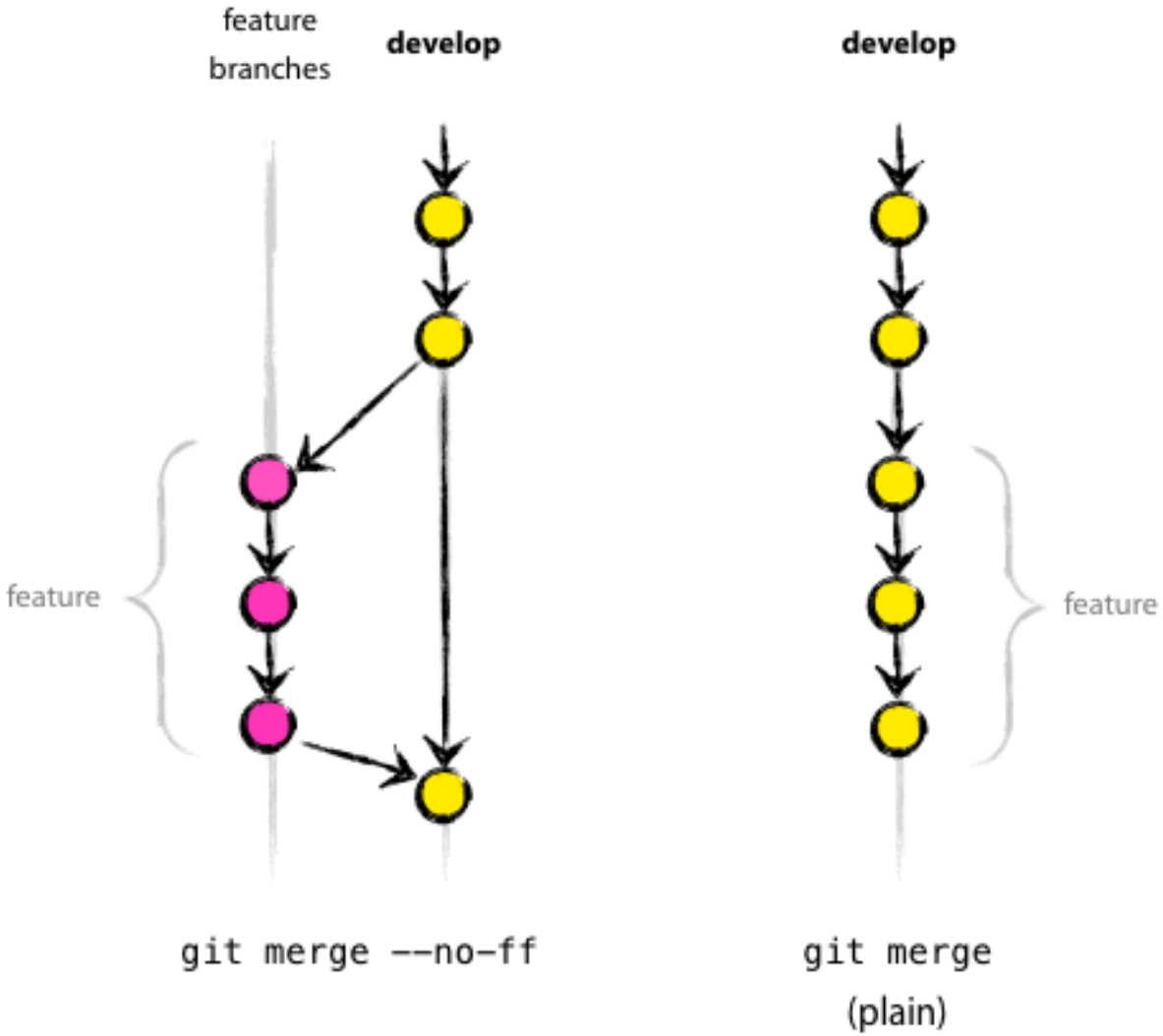
http://www.ronaldwidha.net

A new metaphor

- Tree is a bad analogy
- Source branches are more like **lanes** on a highway
- Commits are free to move between lanes seamlessly
- Commits are tracked through pointers as deltas

Merging

- Commands
  - git merge - Join two or more development histories together
  - git rebase - Forward-port local commits to the updated upstream head
  - git cherry-pick - Apply the changes introduced by some existing commits
- Merge
  - Keeps branch history through pointers
- Merge options
  - Squashing - Combine changes into a single commit
  - Fast Forward

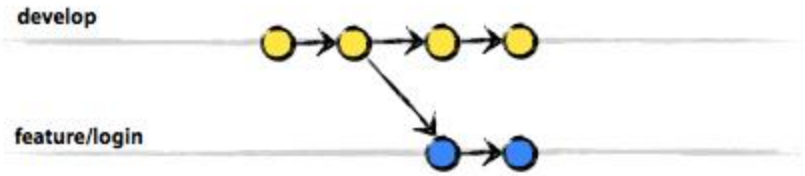


Fast Forwarding

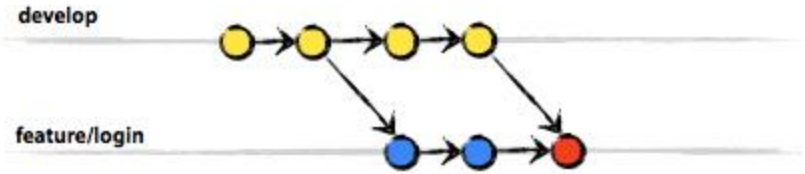
- Fast Forward is the default
- no-ff adds a new commit

image src: http://nvie.com

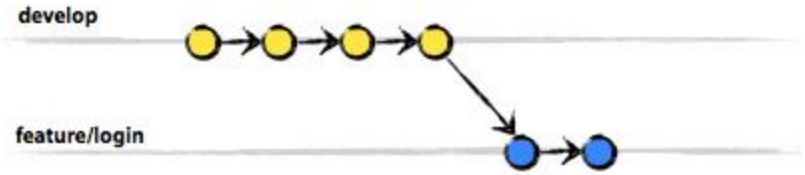
Rebasing



- Allows rewriting a branch's history
- Makes the history linear



- Standard Merge



- Rebase - Rewriting history

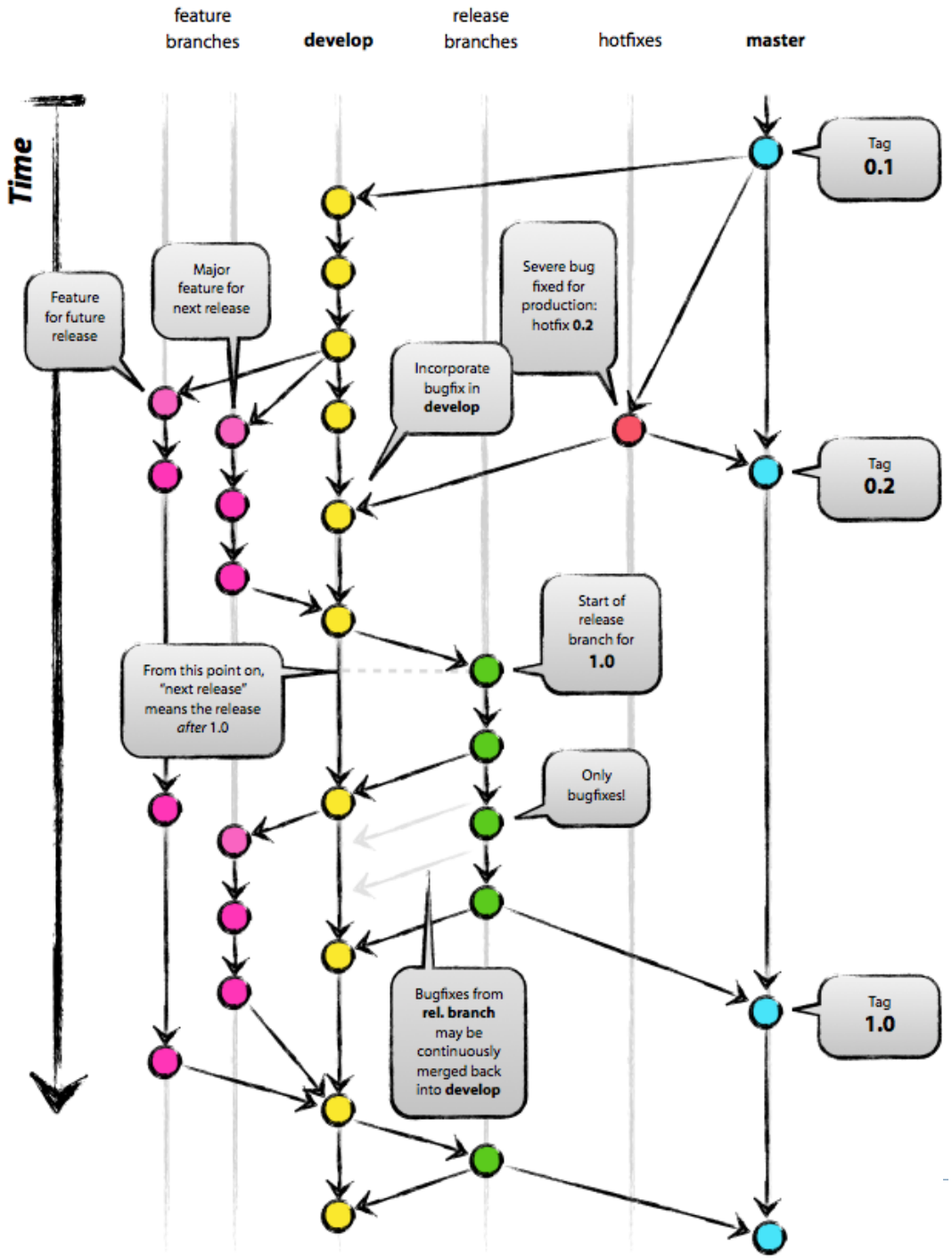
source: http://jeffkreeftmeijer.com

Oops

- Undoing local changes to particular file
  - git checkout <file>
- Undoing all changes since the last commit
  - git reset --hard
  - Like a Subversion "revert"
- Reverting last committed change
  - git revert HEAD
- "Amending" a commit
  - git commit -a --amend
- Get out of Jail free card
  - git reflog

Stashing

- Stores changes for future use
- Commands
  - git stash list - Shows the available stashes
  - git stash save - Saves current changes to the stash
  - git stash pop - Remove and apply the most recent stash
- Bug fixing Scenario
  - git stash save "work before bug fix"
  - git checkout -b bug1234
  - ... Fix the bug ... maybe push? maybe merge?
  - git checkout dev
  - git stash pop



A Git Branching Model

- Central repository with 2 main branches
  - master (stable)
  - dev
- Feature Branches
- Release Branches
- Hotfix Branches

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

Git+SVN

- Allows using git as a Subversion 'client'
- Gives some of the benefits of git
  - Local repository
  - Local branches
  - Complete history
- Collaboration Caveats
  - Recommended that each user do their own clone from SVN
  - Recommended that each user push and pull from SVN

Working with Subversion

- Checkout out a subversion project
  - git svn clone --stdlayout http://host/path/to/svn/repo
  - Warning: This will take a long time
- Checking in files to subversion
  - git commit -a ...
  - git svn dcommit
- Updating from subversion
  - git svn rebase