Using Git on OS-X

Munich Cocoaheads
2009-11-12
2009-12-10

Coming up

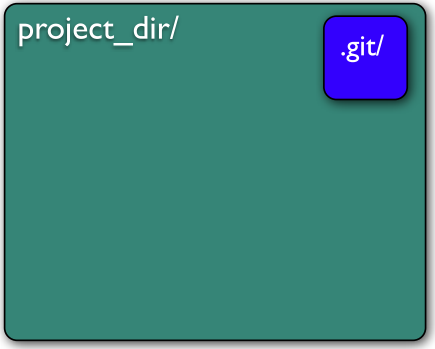
Basic Concepts

Daily Git

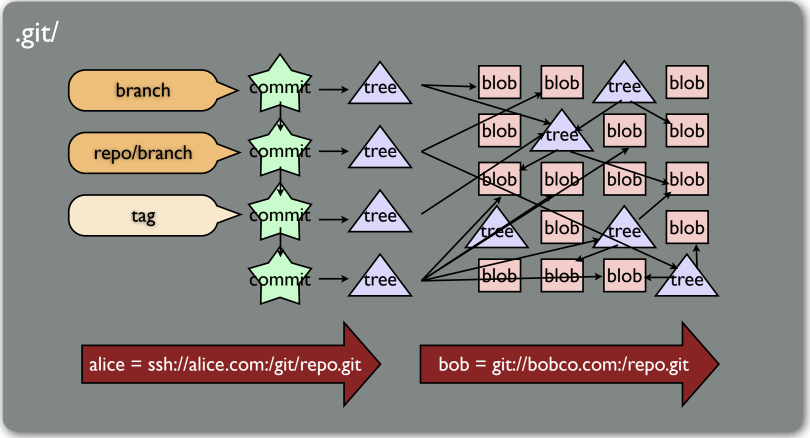
Git with XCode

Non-Obvious Git

What is git?



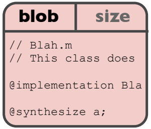
What is git?


.git/
branch
repo/branch
tag
commit
commit
commit
commit
tree
tree
tree
tree
blob
tree
alice = ssh://alice.com:/git/repo.git
bob = git://bobco.com:/repo.git

Blobs

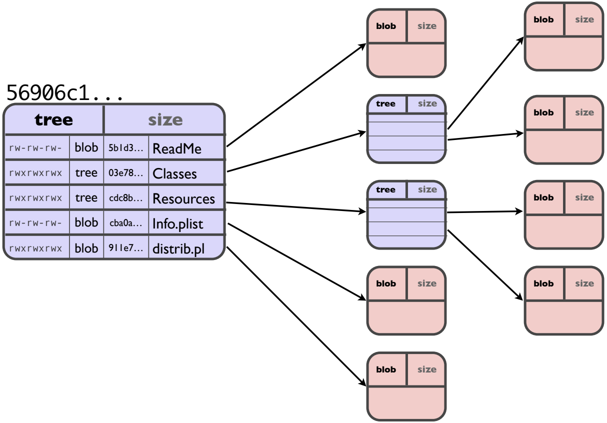
sha1...

5b906c1725109d441fe846300fd4e57063cc6d6b

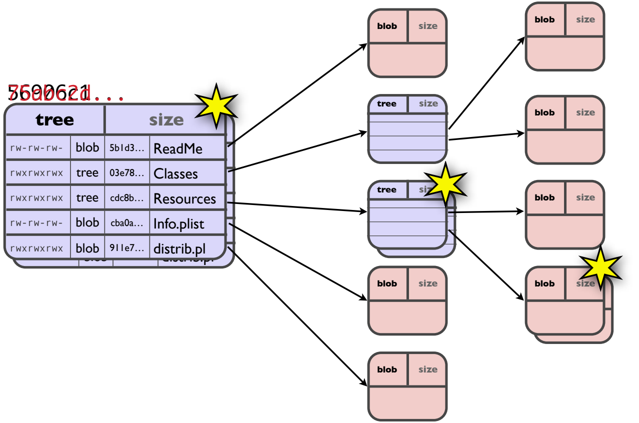


```
// Blah.m
// This class does

@implementation Bla

@synthesize a;
```

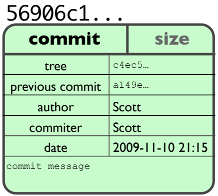
trees


56906c1...
tree
size
rw-rw-rw- blob 5b1d3... ReadMe
rwxrwxrwx tree 03e78... Classes
rwxrwxrwx tree cdc8b... Resources
rw-rw-rw- blob cba0a... Info.plist
rwxrwxrwx blob 911e7... distrib.pl
blob size
tree size
tree size
blob size
blob size
blob size
blob size
blob size
blob size
blob size

changes


5609621...
76ab62d...
tree
size
rw-rw-rw-
blob
5b1d3...
ReadMe
rwxrwxrwx
tree
03e78...
Classes
rwxrwxrwx
tree
cdc8b...
Resources
rw-rw-rw-
blob
cba0a...
Info.plist
rwxrwxrwx
blob
911e7...
distrib.pl
blob
size

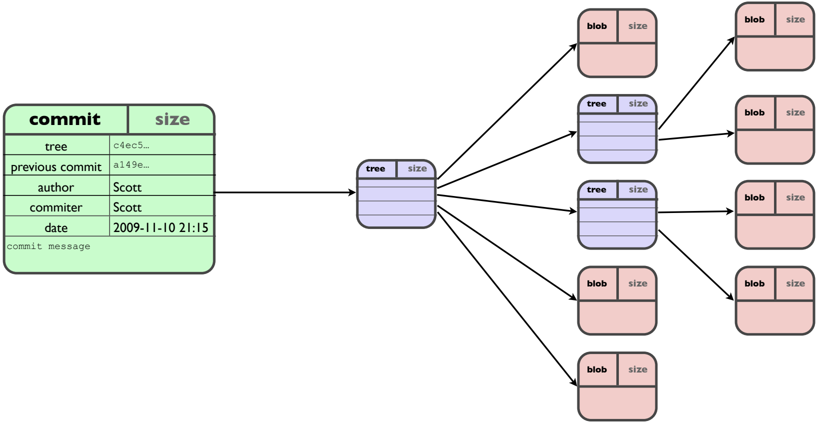
commits



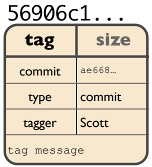
56906c1...

commit	size
tree	c4ec5…
previous commit	a149e…
author	Scott
commiter	Scott
date	2009-11-10 21:15
commit message	

commits


commit
size
tree
c4ec5...
previous commit
a149e...
author
Scott
commiter
Scott
date
2009-11-10 21:15
commit message
tree
size
blob
size

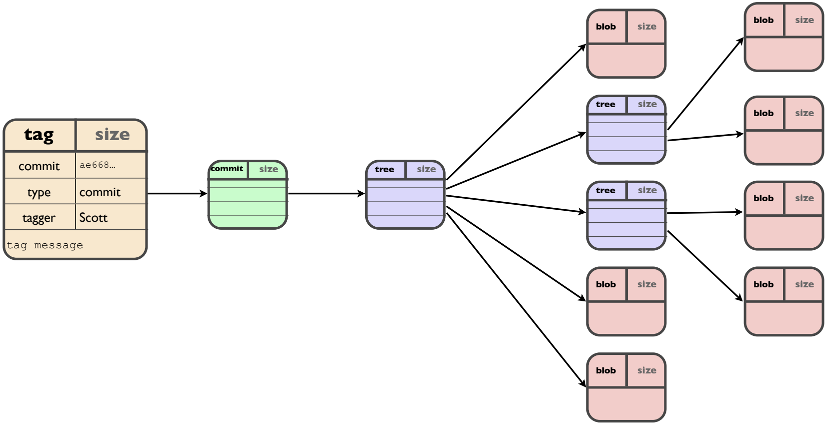
tags


56906c1...

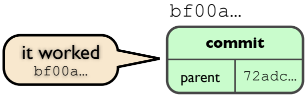
tag	size
commit	ae668…
type	commit
tagger	Scott

tag message

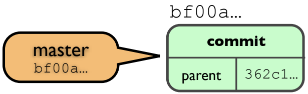
Tags


tag
size
commit
ae668…
type
commit
tagger
Scott
tag message
commit
size
tree
size
blob
size
tree
size
tree
size
blob
size
blob
size
blob
size
blob
size
blob
size
blob
size

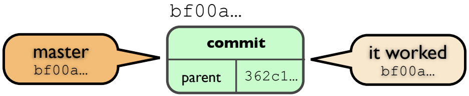
Light-weight tags



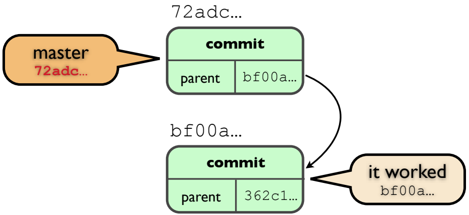
Branches



Branches ‘v’ Tags

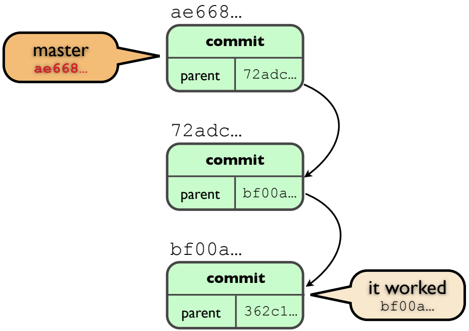


Branches ‘v’ Tags



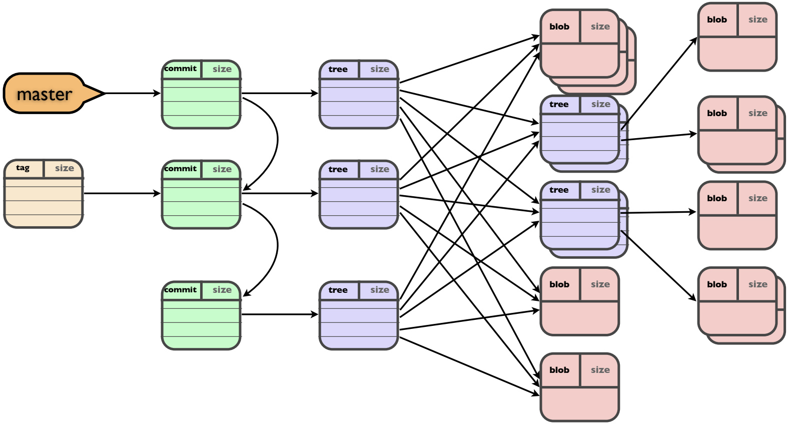
Branches ‘v’ Tags

HEAD and your current branch are automatically updated when you commit a change.

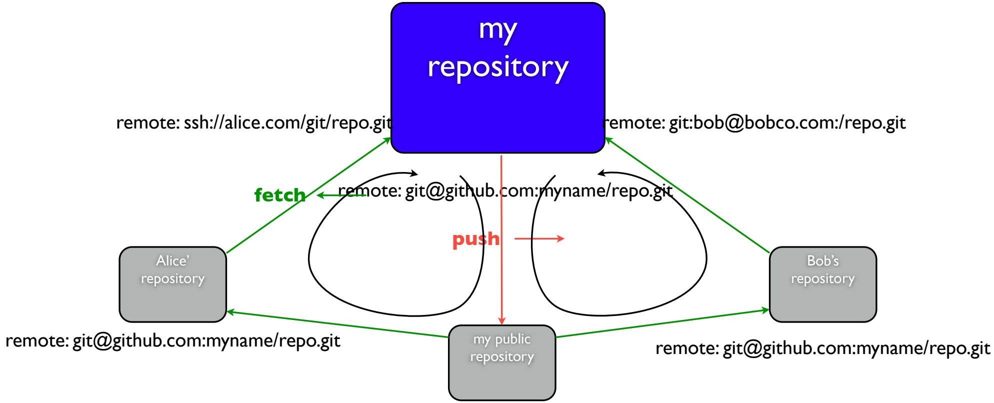


tags are immutable and reliably point to a known state of the entire repository

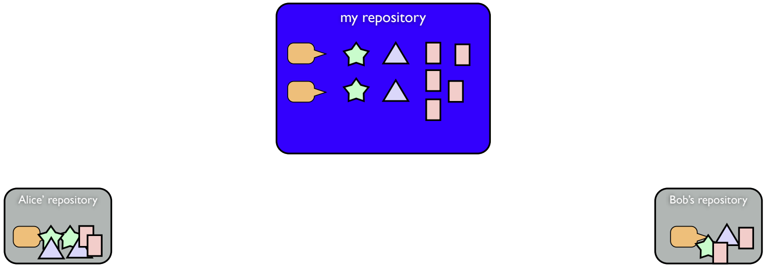
The whole lot



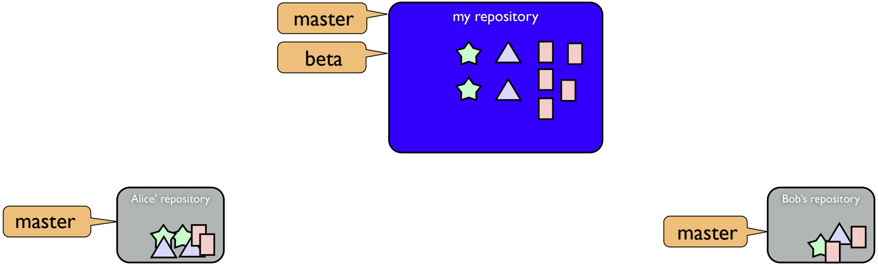
Sharing


my repository
remote: ssh://alice.com/git/repo.git
remote: git:bob@bobco.com:/repo.git
fetch
remote: git@github.com:myname/repo.git
push
Alice' repository
Bob's repository
my public repository
remote: git@github.com:myname/repo.git
remote: git@github.com:myname/repo.git

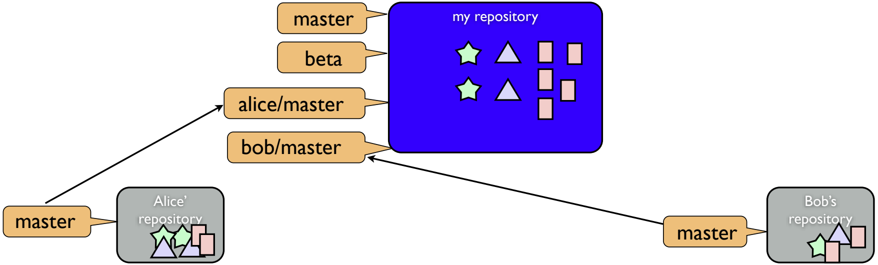
Namespaces



Namespaces

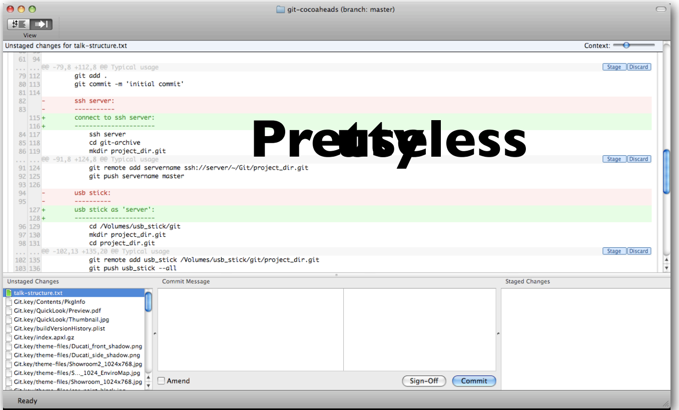

master
beta
my repository
master
Alice' repository
master
Bob's repository

Namespaces


master
beta
alice/master
bob/master
my repository
master
Alice' repository
master
Bob's repository

Git On OS-X

gitx



Pretty

Useless

git-cocoaheads (branch: master)

Unstaged changes for talk-structure.txt

Context:

```
@@ -79,8 +112,8 @@ Typical usage
         git add .
         git commit -m 'initial commit'

-        ssh server:
-        -----------
+        connect to ssh server:
+        ----------------------
             ssh server
             cd git-archive
             mkdir project_dir.git
@@ -91,8 +124,8 @@ Typical usage
             git remote add servername ssh://server/~/Git/project_dir.git
             git push servername master

-        usb stick:
-        ----------
+        usb stick as 'server':
+        ----------------------
             cd /Volumes/usb_stick/git
             mkdir project_dir.git
             cd project_dir.git
@@ -102,13 +135,20 @@ Typical usage
             git remote add usb_stick /Volumes/usb_stick/git/project_dir.git
             git push usb_stick --all
```

Unstaged Changes

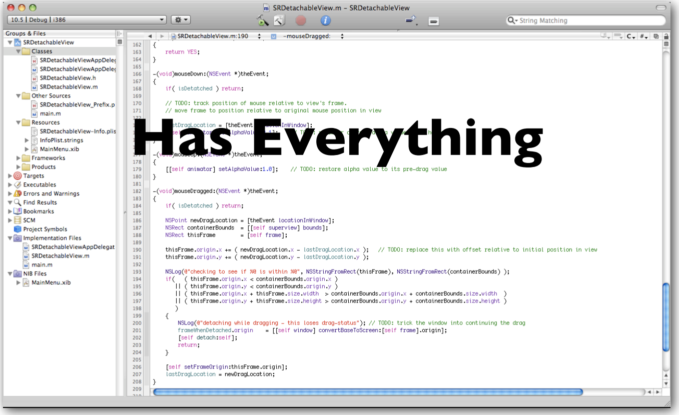
- talk-structure.txt
- Git.key/Contents/PkgInfo
- Git.key/QuickLook/Preview.pdf
- Git.key/QuickLook/Thumbnail.jpg
- Git.key/buildVersionHistory.plist
- Git.key/index.apxl.gz
- Git.key/theme-files/Ducati_front_shadow.png
- Git.key/theme-files/Ducati_side_shadow.png
- Git.key/theme-files/Showroom2_1024x768.jpg
- Git.key/theme-files/S..._1024_EnviroMap.jpg
- Git.key/theme-files/Showroom_1024x768.jpg

Commit Message

Staged Changes

Ready

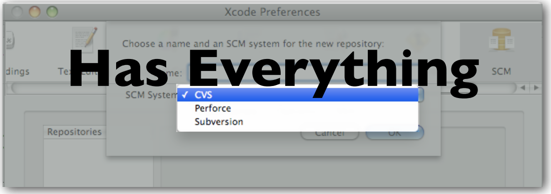
XCode



Has Everything

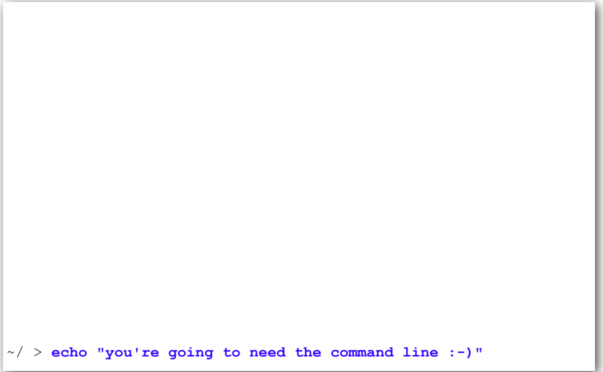
```
{
    return YES;
}

-(void)mouseDown:(NSEvent *)theEvent;
{
    if( isDetatched ) return;

    // TODO: track position of mouse relative to view's frame.
    // move frame to position relative to original mouse position in view

    [illegible]stDragLocation = [theEvent [illegible]InWindow];
    [illegible]
}

-(void)mouseUp:(NSEvent *)theEvent;
{
    [[self animator] setAlphaValue:1.0];    // TODO: restore alpha value to its pre-drag value
}

-(void)mouseDragged:(NSEvent *)theEvent;
{
    if( isDetatched ) return;

    NSPoint newDragLocation = [theEvent locationInWindow];
    NSRect containerBounds  = [[self superview] bounds];
    NSRect thisFrame        = [self frame];

    thisFrame.origin.x += ( newDragLocation.x - lastDragLocation.x );   // TODO: replace this with offset relative to initial position in view
    thisFrame.origin.y += ( newDragLocation.y - lastDragLocation.y );

    NSLog(@"checking to see if %@ is within %@", NSStringFromRect(thisFrame), NSStringFromRect(containerBounds) );
    if(   ( thisFrame.origin.x < containerBounds.origin.x )
       || ( thisFrame.origin.y < containerBounds.origin.y )
       || ( thisFrame.origin.x + thisFrame.size.width  > containerBounds.origin.x + containerBounds.size.width  )
       || ( thisFrame.origin.y + thisFrame.size.height > containerBounds.origin.y + containerBounds.size.height )
       )
    {
        NSLog(@"detaching while dragging - this loses drag-status"); // TODO: trick the window into continuing the drag
        frameWhenDetached.origin    = [[self window] convertBaseToScreen:[self frame].origin];
        [self detach:self];
        return;
    }

    [self setFrameOrigin:thisFrame.origin];
    lastDragLocation = newDragLocation;
}
```

XCode



Has Everything

Git your hands dirty



```
~/ > echo "you're going to need the command line :-)"
```

First Steps

global Configuration

```
~/ > $EDITOR ~/.gitconfig

[core]
   pager = more
   excludesfile = /Users/me/.gitignore
[user]
   name = My Full Public Identity
   email = h4x0r@example.com
[format]
   pretty = format:%h %ci [%aN] %s
```

pager stops long lists from flying past your nose uncontrollably

excludesfile specifies files should never be checked into a git repository

format:
- %h = hash id
- %ci = commit date, iso 8601 format
- %aN = author name
- %s = summary

more info: `git log --help`

global Configuration

```
~/ > $EDITOR ~/.gitignore

# apple typical files
.DS_Store
.Spotlight-V100
.com.apple.timemachine.supported
.fseventsdbuild

# XCode user state files
*.mode1v3
*.pbxuser
*.objc_sync

# other SCM systems
.svn

# editor temporary files
*~
*.swp

# files you generate while building
build/
version.txt
CHANGELOG
```

Where to look for Help

```
~/ > git help <cmd>


~/ > git <cmd> --help
```

make sure you install git's man-pages.

Use the `man` branch of the git repository

Where to look for Help

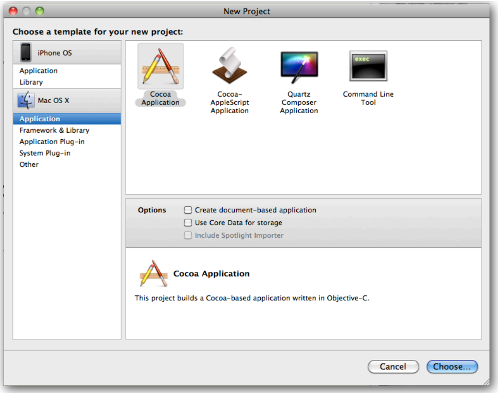
http://git-scm.com

http://github.com

http://gitready.com

http://google.com

A new Project



Choose a template for your new project:

Options
- [ ] Create document-based application
- [ ] Use Core Data for storage
- [ ] Include Spotlight Importer

Cocoa Application

This project builds a Cocoa-based application written in Objective-C.

New Project in Git

```
~/>             cd project


~/project/ >  git init
Initialized empty Git repository in project_dir/.git/


~/project/ >  git add .


~/project/ >  git commit -m 'initial commit'
[master (root-commit) 64fb323] initial commit
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 hello.txt
```

create your project in XCode
create a new git repository
add your files & directories to git
commit your changes

Git Configuration for Xcode

```
~/project/ >  $EDITOR .gitattributes


*.pbxproj -crlf -diff -merge
*.nib     -crlf -diff -merge
*.xib     -crlf -diff -merge
*.graffle -crlf -diff -merge


~/project/ >  git add .gitattributes


~/project/ >  git commit -m 'add .gitattributes - prevent accidental merging of special XCode files'
[master (root-commit) 64fb323] initial commit
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 .gitattributes
```

Tell git to treat these files as if they were binaries.

XML files are notorious for being text, but "unmergable".

`.gitattributes` is project-specific
  must be checked into each project seperately
  will automatically be used by all project members

Joining An Existing Project

```
~/ > git clone -o cloned_repo URL/project.git

~/ > cd project

~/project/ > git checkout -b my_stuff cloned_repo/master
```

cloning a repository automatically sets up a remote repository called `origin`.

You can specify a different name for the remote repository with `-o name`

Clone a local repository

cloning a repository automatically sets up a remote repository called `origin`.

You can specify a different name for the remote repository with `-o name`

```
~/ > git clone ~/old_project_dir ~/new_project_dir
```

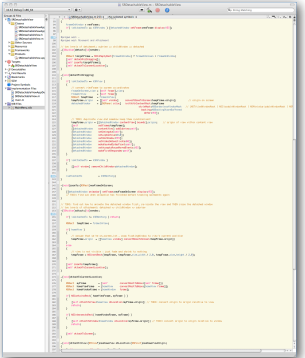
git with XCode

XCode & git

```
{
    frameOnWindow = newFrame;
    if( isAttachedTo == kSRWindow ) [detachedWindow setFrame:newFrame display:YES];
}

#pragma mark -
#pragma mark Movement and Attachment

// two levels of detchment: subview => childWindow => detached
-(IBAction)detach:(id)sender;
{
    NSRect targetFrame = NSIsEmptyRect(frameOnWindow) ? frameOnScreen : frameOnWindow;
    [self detachForDragging];
    [self zoomTo:targetFrame];
    [self attachToCurrentLocation];
}

-(void)detachForDragging;
{
    if( isAttachedTo == kSRView )
    {
        // convert viewFrame to screen co-ordinates
        frameOnScreen.size = [self frame].size;
        frameInView        = [self frame];
        NSRect tempFrame   = frameInView;
        tempFrame.origin   = [[self window]     convertBaseToScreen:tempFrame.origin];        // origin on sc
        detachedWindow     = [[NSPanel alloc]   initWithContentRect:tempFrame
                                                          styleMask:NSBorderlessWindowMask   // (NSTitledWindow
                                                            backing:NSBackingStoreBuffered
                                                              defer:YES];

        // TODO: duplicate view and somehow keep them synchronised!
        tempFrame.origin = [[detachedWindow contentView] bounds].origin;    // origin of view within content v
        [self               setFrame:tempFrame];
        [[detachedWindow    contentView] addSubview:self];
        [detachedWindow     setDelegate:self];
        [detachedWindow     setAlphaValue:0.8];
        [detachedWindow     setHasShadow:YES];
        [detachedWindow     setHidesOnDeactivate:NO];
        [detachedWindow     makeKeyAndOrderFront:self];
        [detachedWindow     setAcceptsMouseMovedEvents:YES];
        [detachedWindow     makeFirstResponder:self];
    }

    if( isAttachedTo == kSRWindow )
    {
        [[self window] removeChildWindow:detachedWindow];
    }

    isAttachedTo            = kSRNothing;
}

-(void)zoomTo:(NSRect)newFrameOnScreen;
{
    [[detachedWindow animator] setFrame:newFrameOnScreen display:YES];
    // TODO: find out when animation has finished before tracking movements again
}

// TODO: find out how to animate the detached window first, re-locate the view and THEN close the detached win
// two levels of attachment: detached => childWindow => subview
-(IBAction)attach:(id)sender;
{
    if( isAttachedTo != kSRNothing ) return;

    NSRect  tempFrame = frameInView;

    if( homeView )
    {
        // assume that we're on-screen-ish - zoom floatingWindow to view's current position
        tempFrame.origin  = [[homeView window] convertBaseToScreen:tempFrame.origin];
    }
    else
    {
        // view is not visible - just fade and shrink to nothing
        tempFrame = NSInsetRect(tempFrame, tempFrame.size.width / 2.0, tempFrame.size.height / 2.0);
    }

    [self zoomTo:tempFrame];
    [self attachToCurrentLocation];
}

-(void)attachToCurrentLocation;
{
    NSRect  myFrame         = [self         convertRectToBase:[self frame]];
    NSRect  homeViewFrame   = [homeView     convertRectToBase:[homeView frame]];
    NSRect  homeWindowFrame = [homeWindow   frame];

    if( NSContainsRect( homeViewFrame, myFrame ) )
    {
        [self attachToView:homeView atLocation:myFrame.origin]; // TODO: convert origin to origin relative to v
        return;
    }

    if( NSIntersectsRect( homeWindowFrame, myFrame) )
    {
        [self attachToWindow:homeWindow atLocation:myFrame.origin]; // TODO: convert origin to origin relative
        return;
    }

    [self attachToScreen];
}

-(void)attachToView:(NSView*)newHomeView atLocation:(NSPoint)newHomeViewOrigin;
{
```

```
~/project/ > open *.xcode*
```

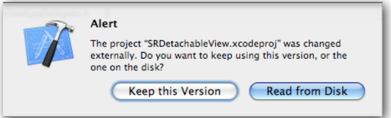
XCode & git



```
~/project/ > git status

~/project/ > git diff

~/project/ > git checkout -b fix

~/project/ > git commit -am '…'

~/project/ > git checkout master

~/project/ > git merge fix

~/project/ > git push public
```

if this happens...

```
git checkout
```



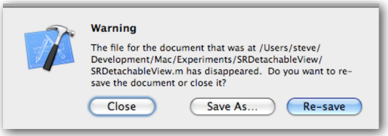
Alert

The project "SRDetachableView.xcodeproj" was changed externally. Do you want to keep using this version, or the one on the disk?

"**Read from Disk**" will bring XCode up-to-date with your git repository

everything OK, XCode just doesn't want to lose **your** work...

However…


Warning

The file for the document that was at /Users/steve/Development/Mac/Experiments/SRDetachableView/SRDetachableView.m has disappeared. Do you want to re-save the document or close it?

If you get this message, you should:

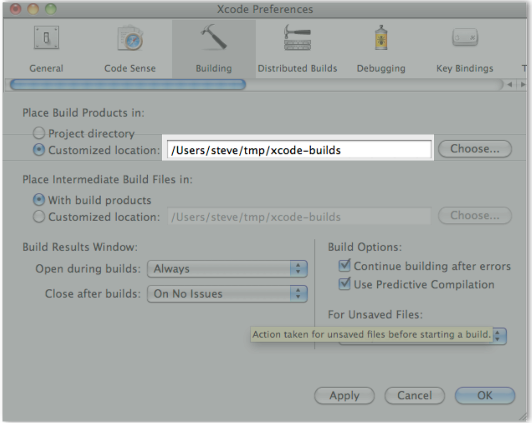
Save your work (possibly in a Temporary Directory)

Close XCode

Fix up your working directory

Open XCode again

XCode - Tips



Place Build Products in:

- ( ) Project directory
- (●) Customized location: /Users/steve/tmp/xcode-builds

Place Intermediate Build Files in:

- (●) With build products
- ( ) Customized location: /Users/steve/tmp/xcode-builds

Build Results Window:

- Open during builds: Always
- Close after builds: On No Issues

Build Options:

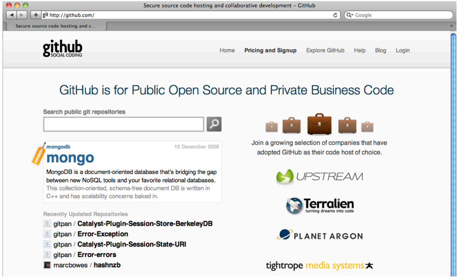
- ☑ Continue building after errors
- ☑ Use Predictive Compilation

For Unsaved Files:

Action taken for unsaved files before starting a build.

github

github



Secure source code hosting and collaborative development - GitHub

http://github.com/

Home · Pricing and Signup · Explore GitHub · Help · Blog · Login

GitHub is for Public Open Source and Private Business Code

Search public git repositories

mongodb — 10 December 2009

mongo

MongoDB is a document-oriented database that's bridging the gap between new NoSQL tools and your favorite relational databases. This collection-oriented, schema-free document DB is written in C++ and has scalability concerns baked in.

Recently Updated Repositories

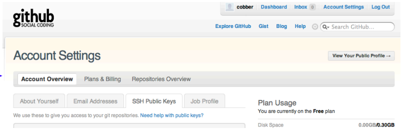
- gitpan / **Catalyst-Plugin-Session-Store-BerkeleyDB**
- gitpan / **Error-Exception**
- gitpan / **Catalyst-Plugin-Session-State-URI**
- gitpan / **Error-errors**
- marcbowes / **hashnzb**

Join a growing selection of companies that have adopted GitHub as their code host of choice.

UPSTREAM · Terralien · PLANET ARGON · tightrope media systems

github - ssh keys

```
~/.ssh/ > ssh-keygen -t rsa -f github

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): just hit return
Enter same passphrase again:

~/.ssh/ > ls

github
github.pub

~/.ssh/ > cat github.pub


ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEArkyf…
```



Account Settings

We use these to give you access to your git repositories. Need help with public keys?

Plan Usage

You are currently on the **Free** plan

Disk Space 0.00GB/**0.30GB**

Daily Work

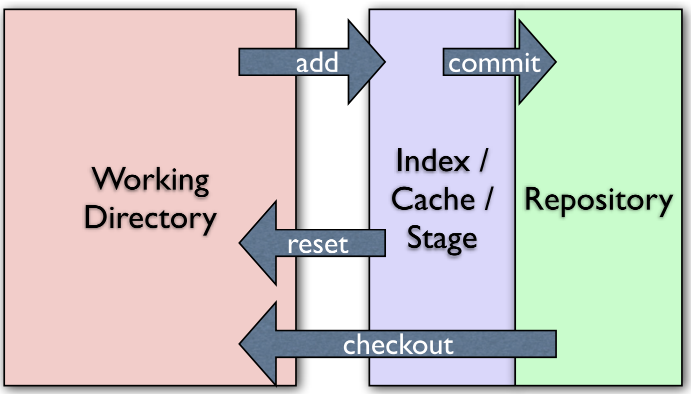
add & commit

work work work…

```
~/project/ > git status

~/project/ > git add file file file directory… or git add -A  or git add -u

~/project/ > git status

~/project/ > git commit -m 'what I just did'

~/project/ > git commit -a -m 'what I just did'
```

```
git add -A
   new files
   changed files
   removed files

git add -u
   changed files
   removed files

git commit -a
same as
git add -A; git commit
```

Why Add & Commit?



Branching

```
~/project/ > git checkout -b new_branch

~/project/ > git branch -a

~/project/ > git branch -d old_branch

~/project/ > git branch -D old_branch
```

TIP: you can create a new branch AFTER you have already made changes.

Just `checout -b new_branch` before you `git add`

Differences?

```
~/project/ > git diff

~/project/ > git diff --cached

~/project/ > git diff HEAD

~/project/ > git diff other_branch
```

Merging

git always merges into the working directory

merged files are added automatically

conflicts are not added - you need to resolve them first

```
~/project/ > git merge other_branch
```

fix conflicts…

```
~/project/ > git add -A

~/project/ > git commit -m 'merge changes from other_branch'
```

Throwing things away

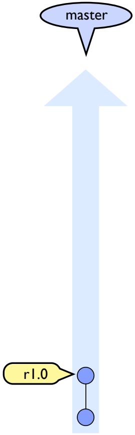
`git reset --hard` updates the cache and the working tree to match the named branch (by default `HEAD`)

```
~/project/ > git reset
```

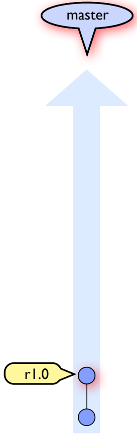
```
~/project/ > git reset --hard HEAD
```

Multiple Branches

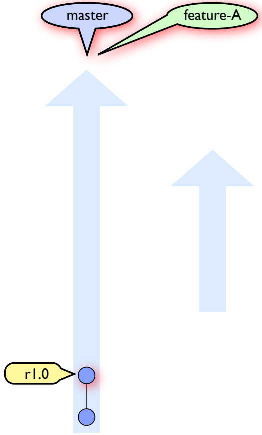
step by step…


master
r1.0

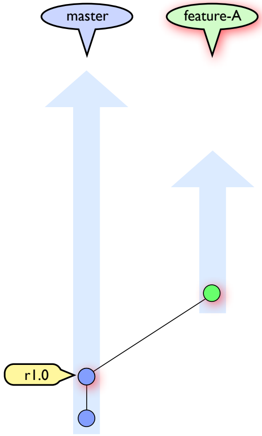
```
git checkout master
```



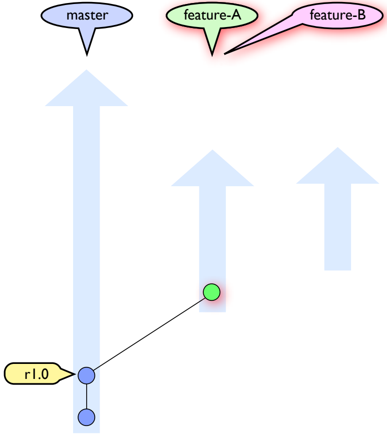
```
git checkout master
git checkout -b feature-A
```



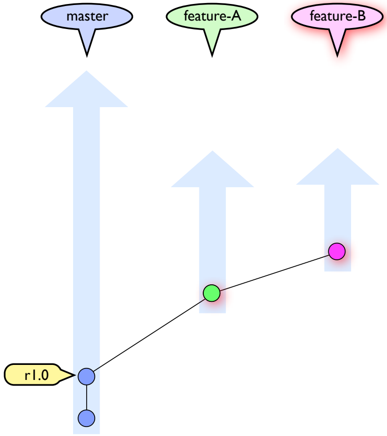
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
```



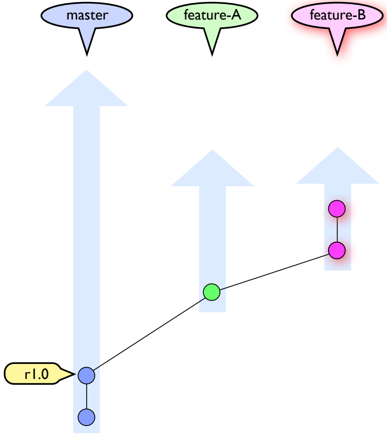
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
```



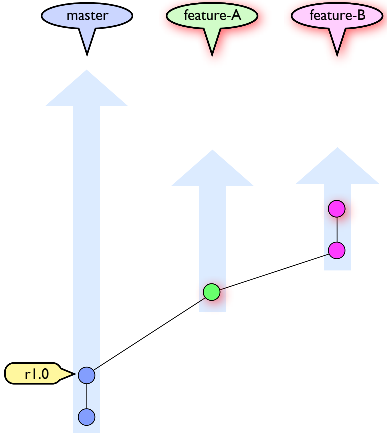
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
```
`git commit -a -m 'basic feature B structure'`



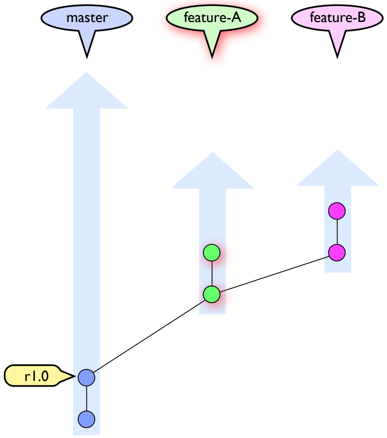
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
```



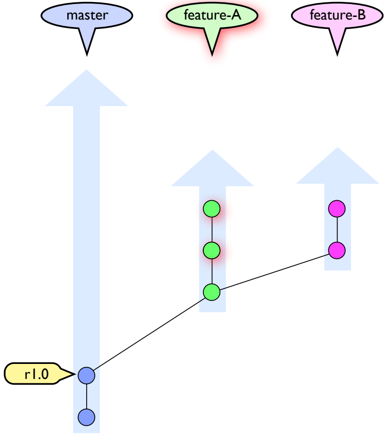
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
```



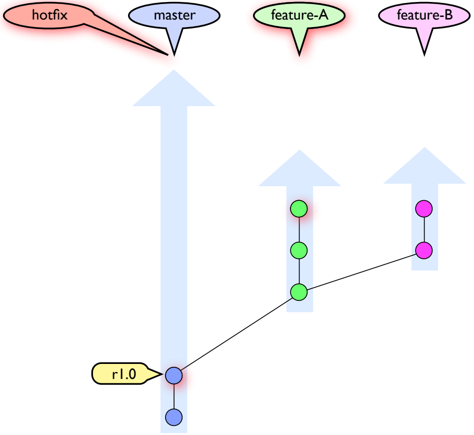
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
```



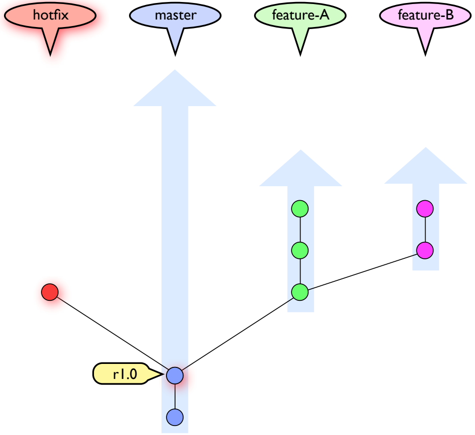
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
```
`git commit -a -m 'debug feature A'`



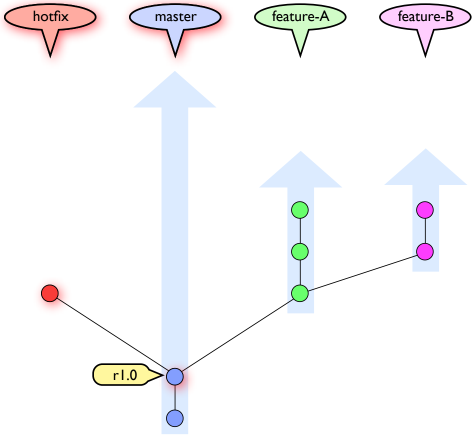
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
```



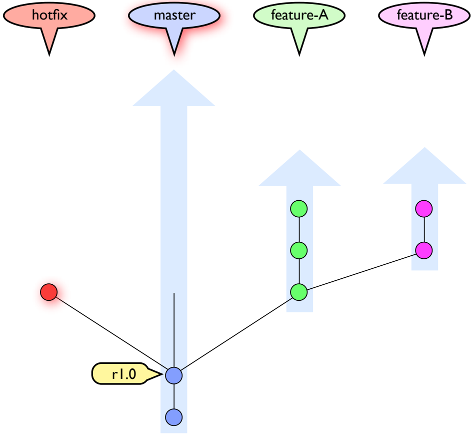
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
```



```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
```
`git checkout master`

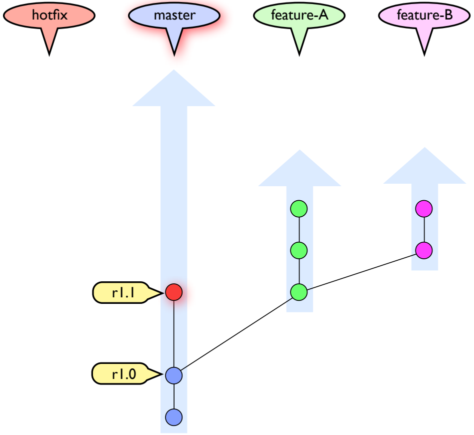


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
```

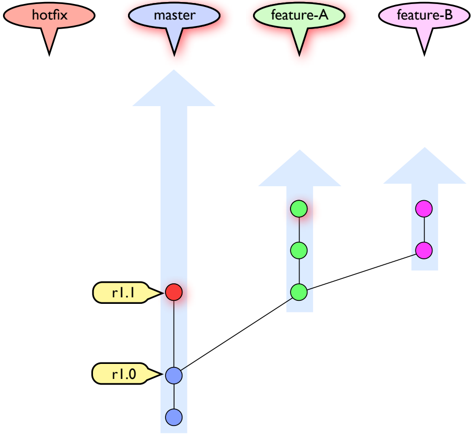


This is a "fast forward" merge.
No *merging* actually takes place.
Instead, the current branch is simply updated to the head of the branch being merged.

```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
```



```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
```

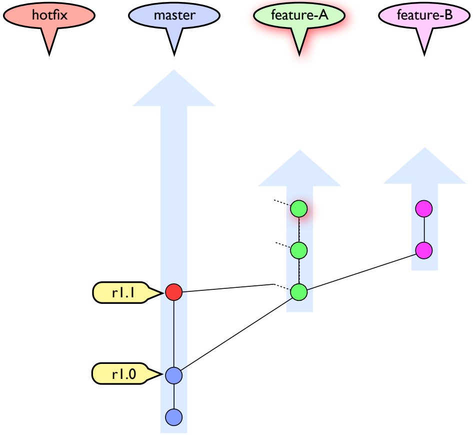


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
```

`git rebase master`

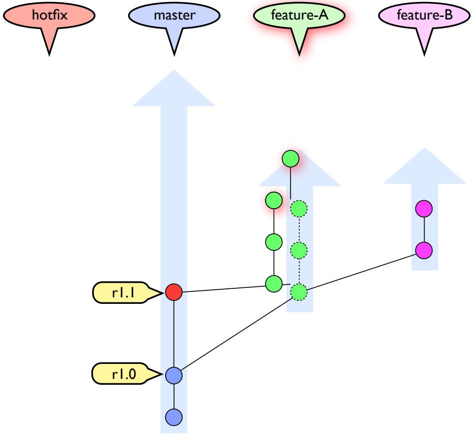
…resolve conflicts…

`git rebase --continue`



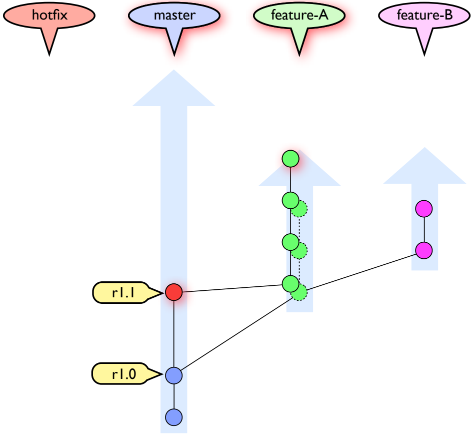
rebasing creates new copies of ***every*** commit between the base and the head!
DO use rebase if you are about to synchronise your work with a public repository.
DO NOT use rebase if the effected commits have already been published!

```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
…resolve conflicts…
git rebase --continue
git commit -a -m 'polish feature A'
```

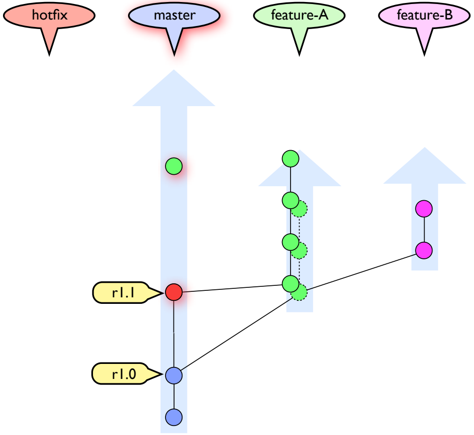


The original commits are no longer accessible via the branch.
Branches stemming from an original commit still reference it!

```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
…resolve conflicts…
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
```

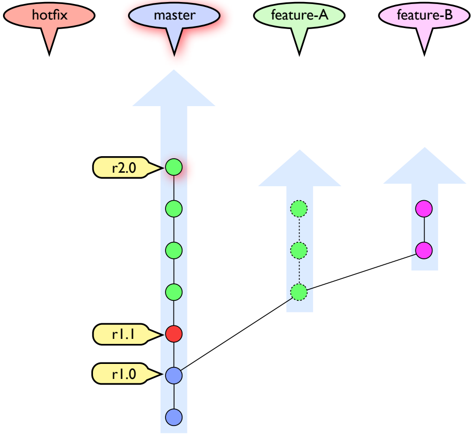


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
```
…resolve conflicts…
```
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
```
`git merge feature-A`

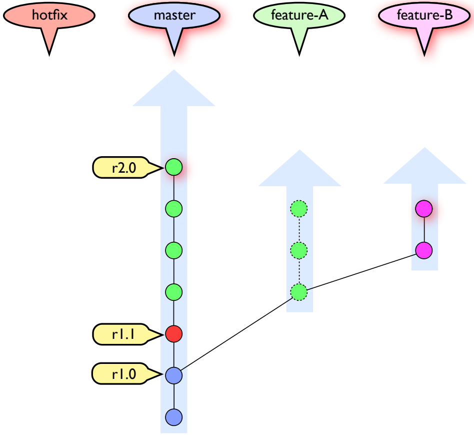


fast forward merge again

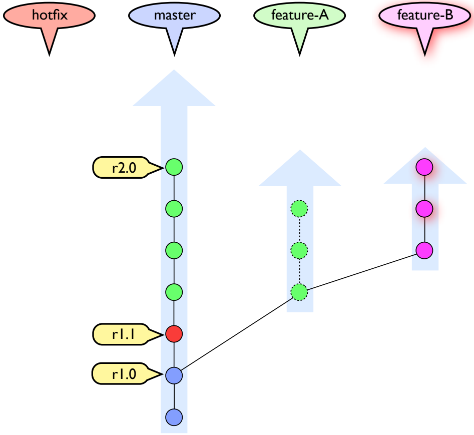
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
```
…resolve conflicts…
```
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
```



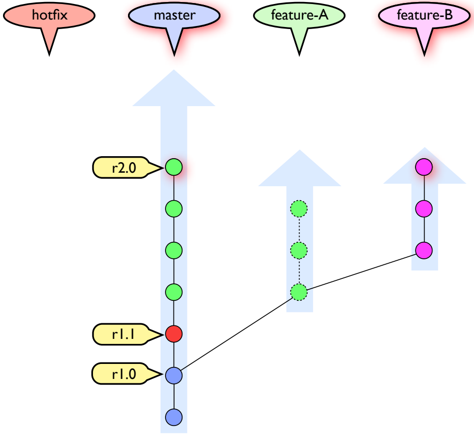
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
…resolve conflicts…
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
```



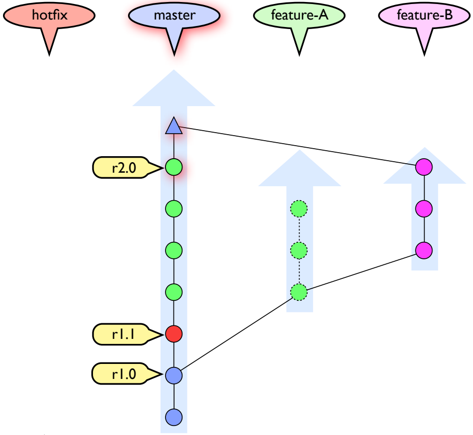
```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
…resolve conflicts…
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
```
`git commit -a -m 'polish feature-B'`



```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
```
…resolve conflicts…
```
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
```
`git checkout master`

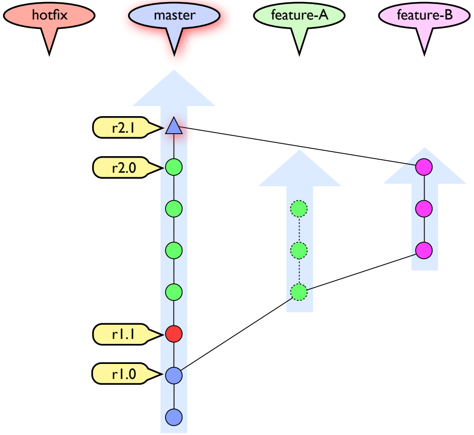


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
…resolve conflicts…
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
…resolve conflicts…
git commit -a -m 'merge feature-B into master'
```



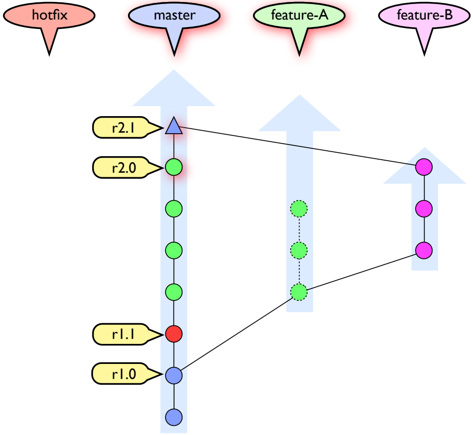
divergent branches require a new commit with 2 parents.
The new commit tracks conflict resolutions.
`git merge` automatically detects if a fast forward merge is possible or not.

```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
...resolve conflicts...
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
...resolve conflicts...
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
```

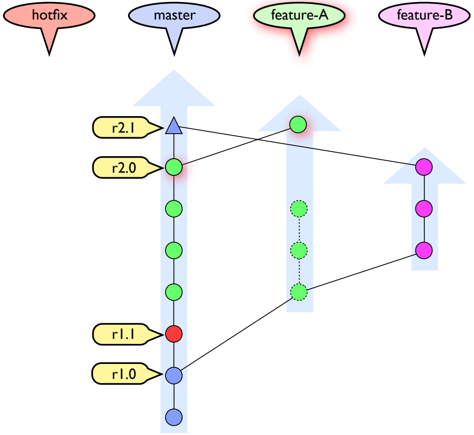


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
...resolve conflicts...
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
...resolve conflicts...
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
```

`git checkout feature-A`

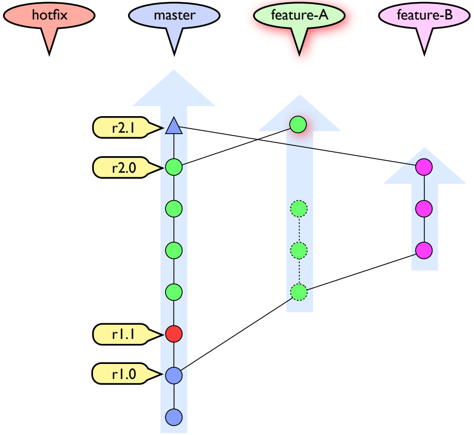


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
```
…resolve conflicts…
```
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
```
…resolve conflicts…
```
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
git checkout feature-A
```
`git commit -a -m 'feature-A extension'`

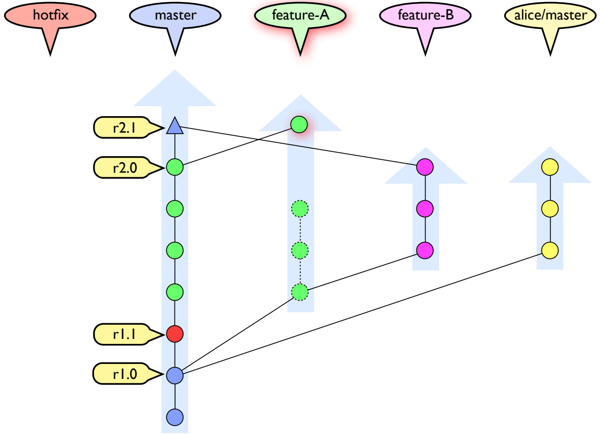


Multiple Repositories

```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
```
…resolve conflicts…
```
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
```
…resolve conflicts…
```
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
git checkout feature-A
git commit -a -m 'feature-A extension'
```

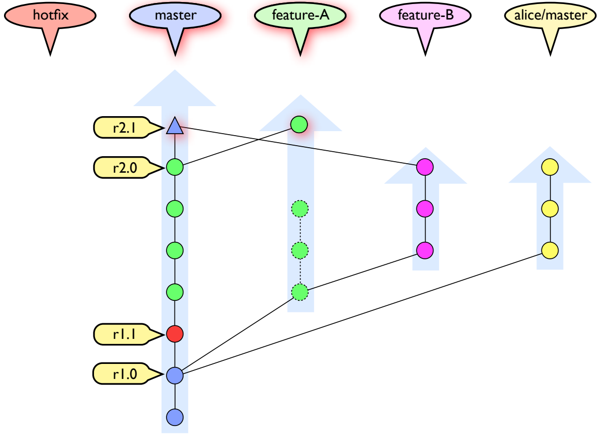


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
...resolve conflicts...
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
...resolve conflicts...
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
git checkout feature-A
git commit -a -m 'feature-A extension'
git fetch alice
```

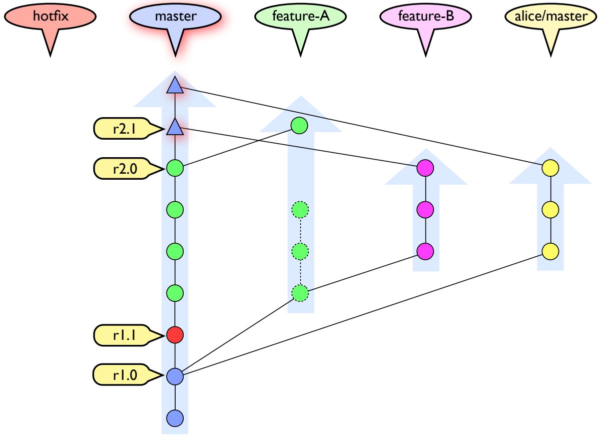


Fetching from a remote repository doesn't change your commit tree!

```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
```
...resolve conflicts...
```
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
```
...resolve conflicts...
```
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
git checkout feature-A
git commit -a -m 'feature-A extension'
git fetch alice
```
`git checkout master`

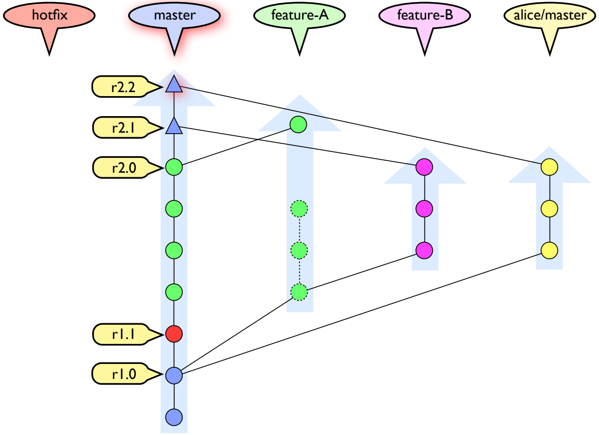


```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
…resolve conflicts…
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
…resolve conflicts…
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
git checkout feature-A
git commit -a -m 'feature-A extension'
git fetch alice
git checkout master
git merge alice/master
…resolve conflicts…
git commit -a -m 'merge alice/master into master'
```



`git pull` combines fetch and merge!

```
git checkout master
git checkout -b feature-A
git commit -a -m 'basic feature A structure'
git checkout -b feature-B
git commit -a -m 'basic feature B structure'
git commit -a -m 'debug feature B'
git checkout feature-A
git commit -a -m 'finish feature A'
git commit -a -m 'debug feature A'
git checkout -b hotfix r1.0
git commit -a -m 'keep customer happy'
git checkout master
git merge hotfix
git tag -a r1.1 -m 'security update'
git checkout feature-A
git rebase master
…resolve conflicts…
git rebase --continue
git commit -a -m 'polish feature A'
git checkout master
git merge feature-A
git tag -a 2.0 -m 'new and improved release'
git checkout feature-B
git commit -a -m 'polish feature-B'
git checkout master
git merge feature-B
…resolve conflicts…
git commit -a -m 'merge feature-B into master'
git tag -a r2.1 -m 'wow release'
git checkout feature-A
git commit -a -m 'feature-A extension'
git checkout master
git fetch alice
git merge alice/master
…resolve conflicts…
git commit -a -m 'merge alice/master into master'
```
`git tag -a r2.2 -m 'insecurity update'`



Publishing your repository

```
local ~/project/ >           ssh me@remote.com

remote ~/ >                  mkdir project.git

remote ~/project.git/ >      cd project.git

remote ~/project.git/ >      git init --bare

remote ~/project.git/ >      logout

local ~/project/ >           git remote add public_repo ssh://me@remote.com/~/project.git

local ~/project/ >           git push public_repo release_branch
```

you should NOT publish your private directories (basic security)

create a `bare repository` on a public server

push only the branches your wish to publish

USB-Stick

```
~/project/ > git clone --bare . /Volumes/usb_stick/project.git

~/project/ > git remote add usb_stick /Volumes/usb_stick/project.git

~/project/ > git push usb_stick
```

USB stick, external disk

great for ad-hoc sharing

great for backup

treat like a public repository

Which Repos Am I connected to?

```
~/project_dir/ > git remote -v

public_repo   ssh://me@remote.com/~/project.git (fetch)
public_repo   ssh://me@remote.com/~/project.git (push)
usb_stick     /Volumes/usb_stick/project.git (fetch)
usb_stick     /Volumes/usb_stick/project.git (push)
```

Updates From Multiple Repos

```
~/project/ > $EDITOR .git/config

...
[remote "steve"]
        url = ssh://steveserve.com/~/Git/project.git
        fetch = +refs/heads/*:refs/remotes/steve/*
[remote "mac"]
        url = git@github.com:mac/project.git
        fetch = +refs/heads/*:refs/remotes/mac/*
[remotes]
        buddies = steve mac
...

~/project/ > git remote update buddies
Updating steve
...
Updating mac
...
```

Working With Others

Publish your changes via a bare repository

Never push to someone else's repository

Use `git remote update` to track multiple repositories

Use `git show-branch` or `git whatchanged` to see what's new

Rewriting History

What if you want to…

…find the commit that introduced a problem…

…remove some commits from the history…

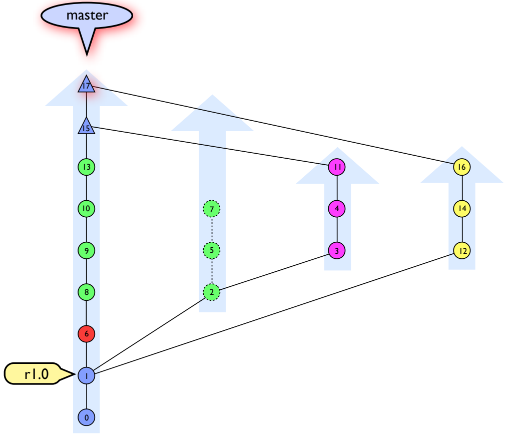
…add one or more commits from one branch to another…

…work on a branch for a long time…

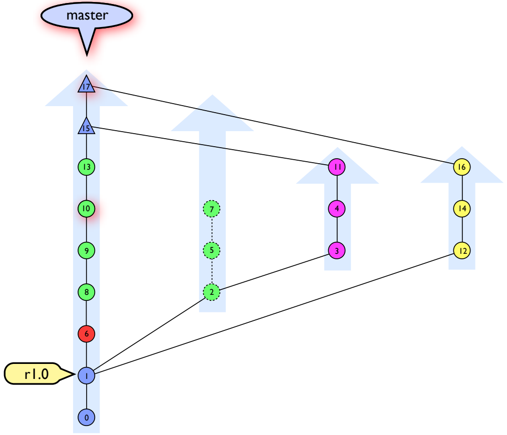
finding bad commits

git bisect

```
git checkout master
```

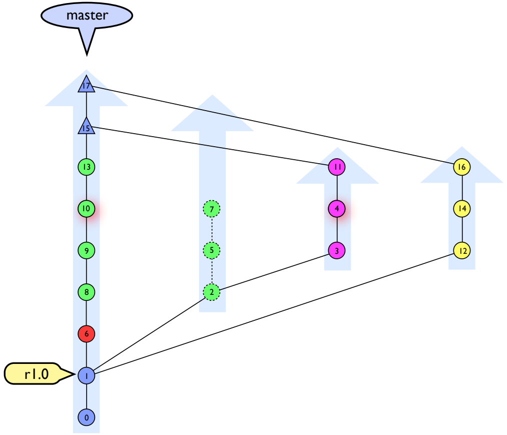


```
git checkout master
git bisect start
git bisect bad master
git bisect good r1.0

Bisecting: ## revisions left to test after this
[10] commit message
```



git has checked out a commit for you to test…
Is the problem currently checked out?
If not call `git bisect good`

```
git checkout master
git bisect start
git bisect bad master
git bisect good r1.0

Bisecting: ## revisions left to test after this
[10] commit message

git bisect good

Bisecting: ## revisions left to test after this
[4] commit message
```

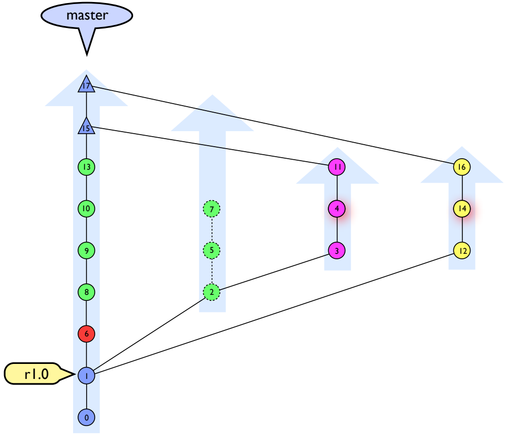


git has checked out another commit for you to test…
Can't test this version? (doesn't compile?)
If so call `git bisect skip`

```
git checkout master
git bisect start
git bisect bad master
git bisect good r1.0

Bisecting: ## revisions left to test after this
[10] commit message

git bisect good

Bisecting: ## revisions left to test after this
[4] commit message
```

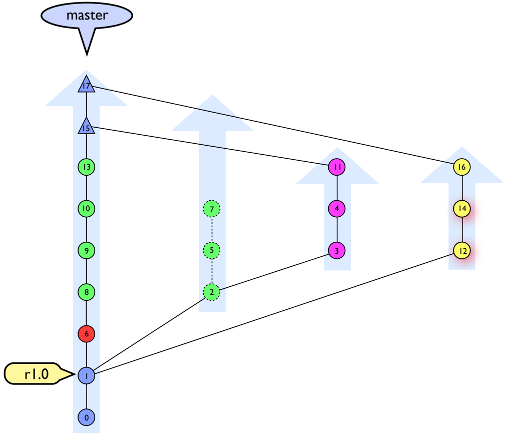
`git bisect skip`

```
Bisecting: ## revisions left to test after this
[14] commit message
```



Another commit for you to test…
Is the problem currently checked out?
If so call `git bisect bad`

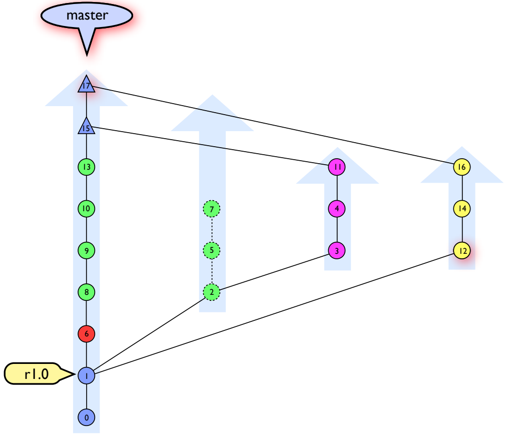
```
git checkout master
git bisect start
git bisect bad master
git bisect good r1.0

Bisecting: ## revisions left to test after this
[10] commit message

git bisect good

Bisecting: ## revisions left to test after this
[4] commit message

git bisect skip

Bisecting: ## revisions left to test after this
[14] commit message

git bisect bad

12 is the first bad commit
```



Now you know where the problem is.
Go back to your branch and fix it with a normal commit.

```
git checkout master
git bisect start
git bisect bad master
git bisect good r1.0

Bisecting: ## revisions left to test after this
[10] commit message

git bisect good

Bisecting: ## revisions left to test after this
[4] commit message

git bisect skip

Bisecting: ## revisions left to test after this
[14] commit message

git bisect bad

12 is the first bad commit
```

`git bisect reset`



```
git checkout master
git bisect start
git bisect bad master
git bisect good r1.0
```

Bisecting: ## revisions left to test after this
[10] commit message

```
git bisect good
```

Bisecting: ## revisions left to test after this
[4] commit message

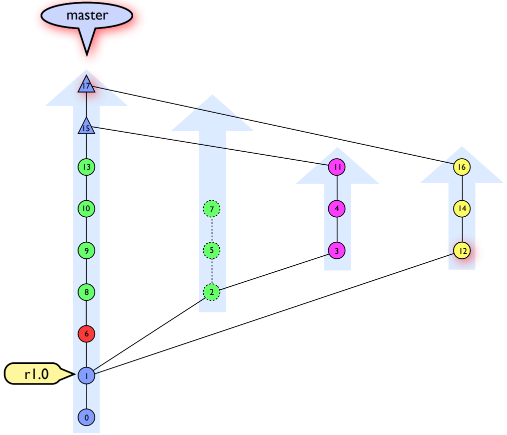
```
git bisect skip
```

Bisecting: ## revisions left to test after this
[14] commit message

```
git bisect bad
```

12 is the first bad commit

`git bisect reset`



git bisect automation

```
git bisect start bad_commit good_commit

git bisect run test_script options…
```

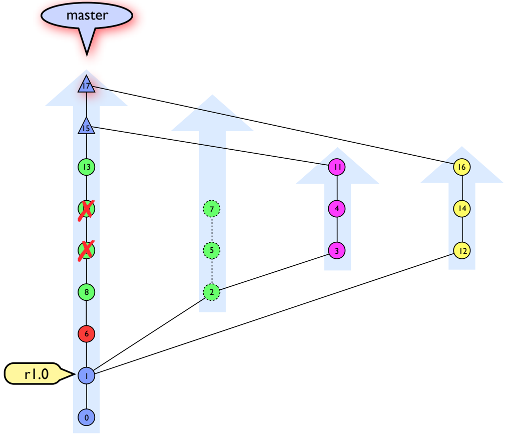
Test script exit codes:

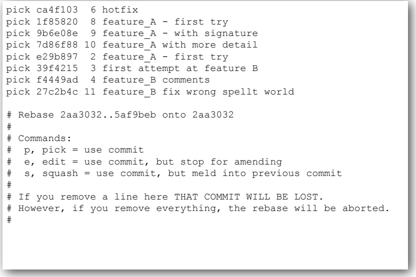
```
exit 0         => good
exit 125       => skip
exit 1 .. 127 => bad
```

removing bad commits

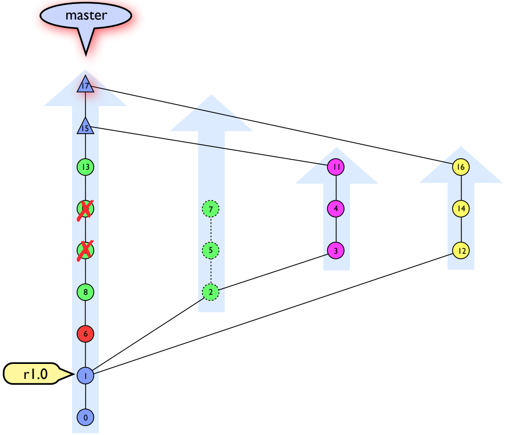
interactive rebasing

```
git checkout master
git rebase --interactive r1.0
```

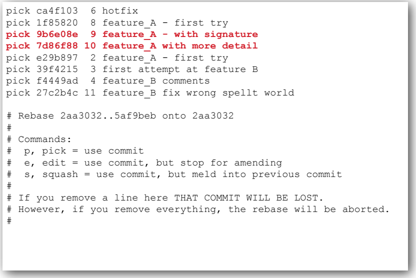




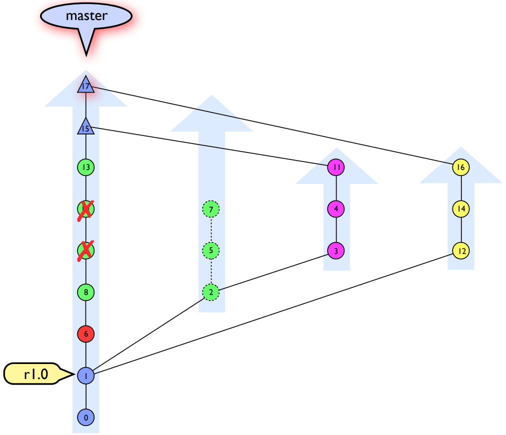
```
pick ca4f103  6 hotfix
pick 1f85820  8 feature_A - first try
pick 9b6e08e  9 feature_A - with signature
pick 7d86f88 10 feature_A with more detail
pick e29b897  2 feature_A - first try
pick 39f4215  3 first attempt at feature B
pick f4449ad  4 feature_B comments
pick 27c2b4c 11 feature_B fix wrong spellt world

# Rebase 2aa3032..5af9beb onto 2aa3032
#
# Commands:
#  p, pick = use commit
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```



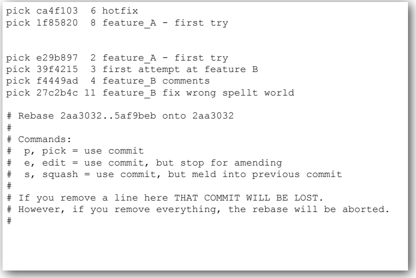
Warning!
Newest commits are at the bottom!
(most other git output has the newest commit at the top!)



```
pick ca4f103  6 hotfix
pick 1f85820  8 feature_A - first try
pick 9b6e08e  9 feature_A - with signature
pick 7d86f88 10 feature_A with more detail
pick e29b897  2 feature_A - first try
pick 39f4215  3 first attempt at feature B
pick f4449ad  4 feature_B comments
pick 27c2b4c 11 feature_B fix wrong spellt world

# Rebase 2aa3032..5af9beb onto 2aa3032
#
# Commands:
#  p, pick = use commit
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

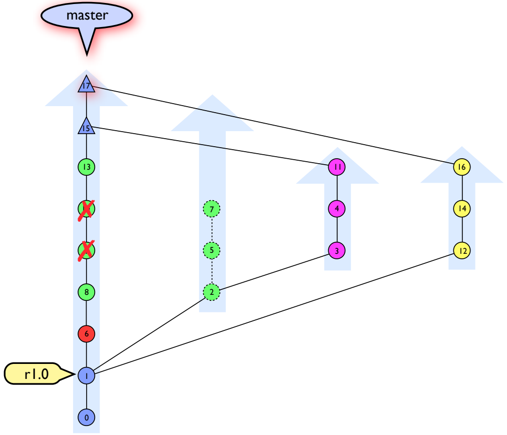


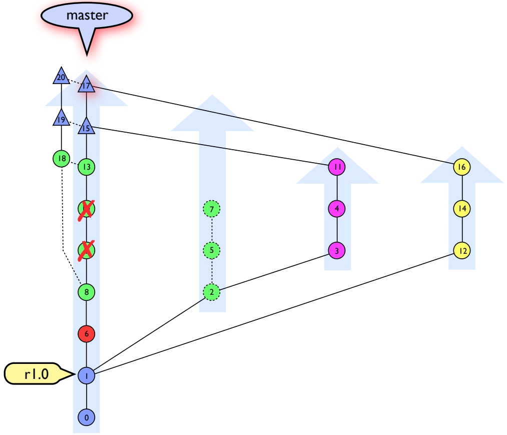
Delete the lines of commits you don't want.
Change `pick` to `squash` if you want merge commits,
or `edit` if you want to split a commit into smaller commits.

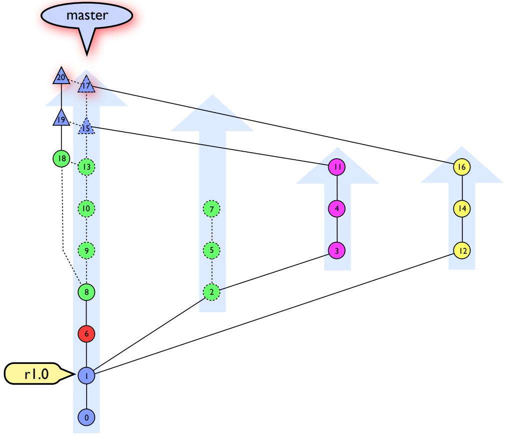


```
pick ca4f103  6 hotfix
pick 1f85820  8 feature_A - first try


pick e29b897  2 feature_A - first try
pick 39f4215  3 first attempt at feature B
pick f4449ad  4 feature_B comments
pick 27c2b4c 11 feature_B fix wrong spellt world

# Rebase 2aa3032..5af9beb onto 2aa3032
#
# Commands:
#  p, pick = use commit
#  e, edit = use commit, but stop for amending
#  s, squash = use commit, but meld into previous commit
#
# If you remove a line here THAT COMMIT WILL BE LOST.
# However, if you remove everything, the rebase will be aborted.
#
```

save the file and exit your editor…
git performs the rebase automatically




master
20
17
19
15
18
13
11
16
7
4
14
5
3
12
8
2
6
r1.0
1
0


master
20
17
19
15
18
13
10
9
8
6
r1.0
1
0
7
5
2
11
4
3
16
14
12

adding commits to other branches

cherry picking

`Add just one commit to the current branch:`

```
git cherry-pick sha1
```

rebasing onto another branch

Add a chain of commits, not the whole branch:

```
git rebase --onto target_commit first_commit last_commit
```

long-term branches

git rerere

to do…