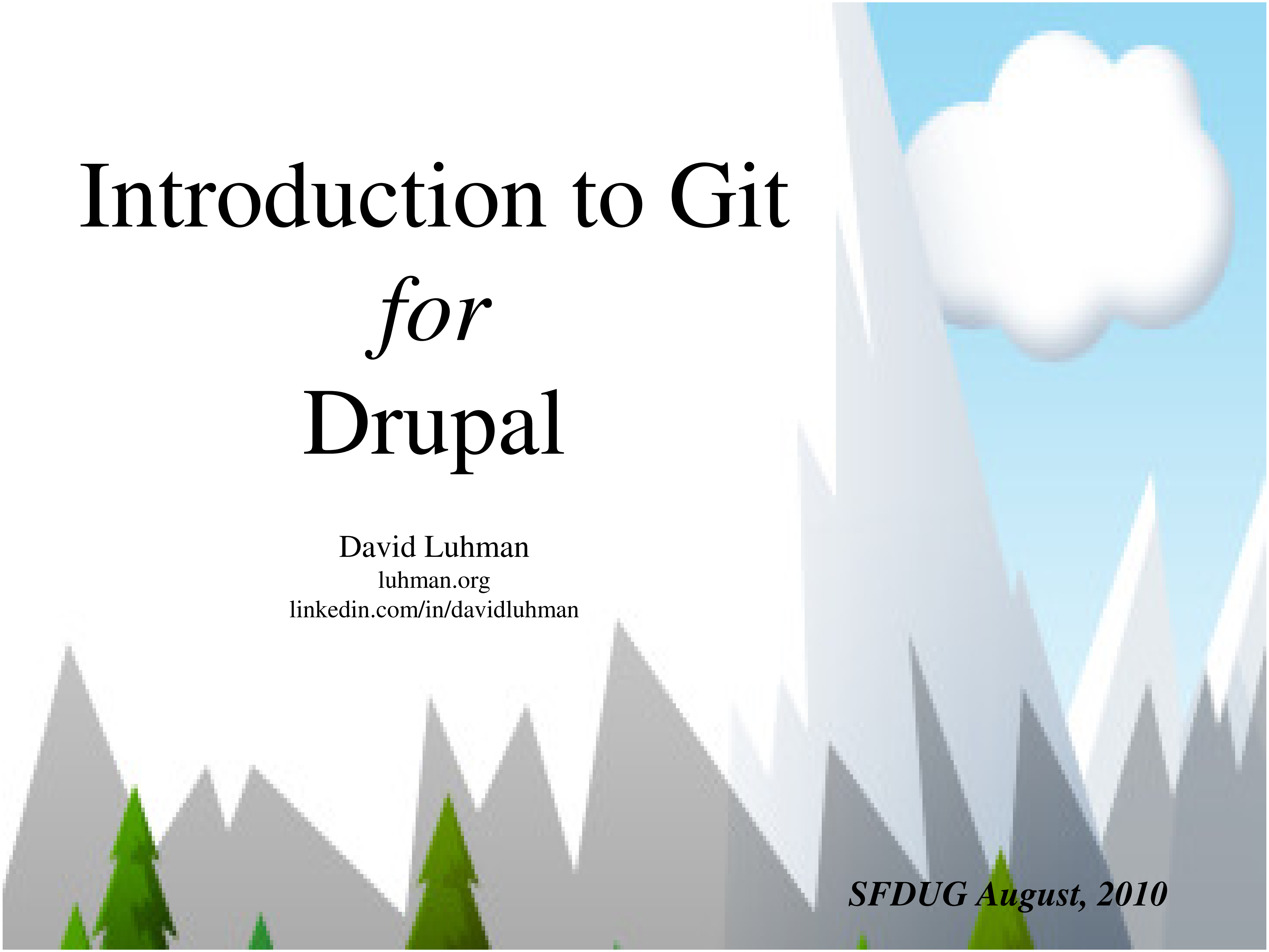
Introduction to Git *for* Drupal

David Luhman
luhman.org
linkedin.com/in/davidluhman

SFDUG August, 2010



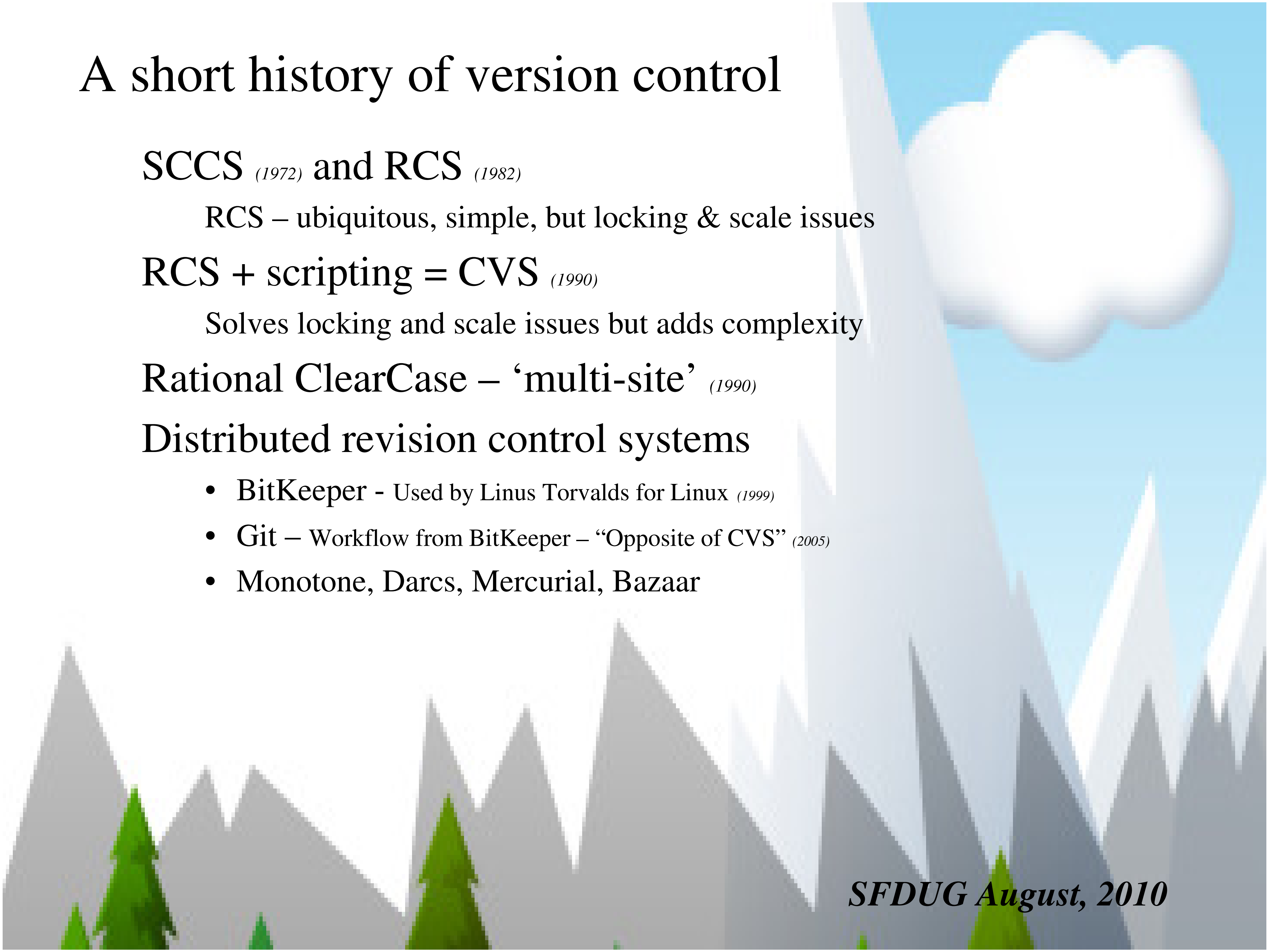
A bit about your humble presenter

- Born and raised in Colorado
- Spell ‘Ada’ ⇨ Johnson Space Center
- Spell ‘FORTRAN’ ⇨ Five years in Japan
- Back to CU Boulder for MBA
- Spell ‘日本語’ ⇨ Silicon Valley
- Spell ‘Git’ ⇨ Here today
- From Drupal 4.6, but Drupal is not my day job ☹

Why Revision Control?

- The need to track "similar but different" versions
- First test in "The Joel Test" *http://en.wikipedia.org/wiki/The_Joel_Test*

Life is too short to waste time with folks who won't use revision control



A short history of version control

SCCS *(1972)* and RCS *(1982)*

RCS – ubiquitous, simple, but locking & scale issues

RCS + scripting = CVS *(1990)*

Solves locking and scale issues but adds complexity

Rational ClearCase – ‘multi-site’ *(1990)*

Distributed revision control systems

- BitKeeper - Used by Linus Torvalds for Linux *(1999)*
- Git – Workflow from BitKeeper – “Opposite of CVS” *(2005)*
- Monotone, Darcs, Mercurial, Bazaar

Installing Git

Make sure to install Git version 1.5 or higher

- Versions 1.4.x and lower have ‘legacy’ commands
- Version 1.6.6 and higher have "smart HTTP" transport
- Current version 1.7.2 from http://git-scm.com

Ubuntu

```
# apt-get install git-core
```

Suggested packages: git-doc git-cvs git-svn git-gui gitk

Making from source is straight-forward

http://luhman.org/blog/2009/06/11/building-git-ubuntu

Mac

http://code.google.com/p/git-osx-installer

Windows

http://code.google.com/p/msysgit

- ‘Git – Bash’ and standard Tk GUIs gitk and git gui
- Don’t worry about “Preview” or “Beta” wording – works great
- Select “Git Bash only” to forgo integration with regular command line (cmd.exe)

Customize Git configuration on Linux

Edit ~/.gitconfig

```
[user]
  name = Your Name Comes Here
  email = you@yourdomain.example.com
[difftool]
  difftool=vimdiff
```

- http://www.kernel.org/pub/software/scm/git/docs/user-manual.html#telling-git-your-name

Add to ~/.bashrc

```
alias gbv='git branch -v'
alias gbva='git branch -va'
alias gs='git status'

# Git completion stuff - 2010-05-16
source ~/.git-completion.sh
#.git-completion.sh, can show dirty, unstaged (*) and staged (+)
GIT_PS1_SHOWDIRTYSTATE=true
# __git_ps1 from .git-completion.sh appends branch name to PS1
   PS1='\h:$myPWD$(__git_ps1 " (%s)") \$ ‘
```

- http://luhman.org/blog/2009/06/11/my-git-environment

SFDUG August, 2010

Sample first repository : /etc/apache2

```
/etc/apache2# git init
Initialized empty Git repository in /etc/apache2/.git/
/etc/apache2# git add .
/etc/apache2# git commit -m "Initial commit"
[master (root-commit) da3d4d3] Initial commit of /etc/apache2
  directory
 119 files changed, 2196 insertions(+), 0 deletions(-)
...
 create mode 120000 sites-enabled/000-default
/etc/apache2# ls -FA1 .git
branches/
COMMIT_EDITMSG
config
description
HEAD
hooks/
index
info/
logs/
objects/
refs/
```

Simple repository with multiple branches

Git’s killer feature: Fast, easy, compact, intra-repository branches

Think how you might do similar with Drupal

- drupal-core branch
- drupal-contrib branch
- your-custom-code branch
- master branch that merges everything together
  - http://drupal.org/node/803746
  - http://books.tag1consulting.com/scalability/drupal/start/staging

Our example : Simple ‘calculator’

- Branch for code from addition expert (branch ‘sums’)
- Branch for code from multiplication expert (branch ‘prods’)

Setup repository and create null branch

```
$ mkdir gitDemo
$ cd $_
$ git init
Initialized empty Git repository in /home/me/gitDemo/.git/
(master #) $ touch .gitignore     (note branch ‘master’  from  .git-completion.sh )
(master #) $ git add .
(master #) $ git commit -m "Initial commit"
[master (root-commit) 4b14ca0] Initial commit
 0 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .gitignore
(master) $ git tag NULL           (create tag ‘NULL’)
(master) $ git branch -va         (list all branches – ‘master’ is default name)
* master 4b14ca0 Initial commit for empty repository
(master) $ git branch -m master null     (rename master branch to ‘null’)
(null) $ git branch -va
* null 4b14ca0 Initial commit for empty repository
```

Create branch sums

```
(null) $ git checkout -b sums
Switched to a new branch "sums"
(sums) $ cat > sums.php
<?php
print "1 + 1 = 2 \n";
(sums) $ git status              (find out what needs adding, committing)
# On branch sums
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#       sums.php
nothing added to commit but untracked files present (use "git add" to track)
(sums) $ git add sums.php                    (need to add new files to track)
(sums +) $ git commit -m "Add sums.php"      (note + indicating staged [added] before commit)
[sums 85f3934] Add sums.php
 1 files changed, 2 insertions(+), 0 deletions(-)
 create mode 100644 sums.php
(sums) $                  (lost + in prompt indicating nothing staged after commit)
```

Append to sums.php

```
(sums) $ echo 'print "2 + 2 = 4 \n";' >> sums.php
(sums *) $ git status
# On branch sums
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#                                          (note how to rollback change "checkout -- sums.php")
#       modified:   sums.php
#
no changes added to commit (use "git add" and/or "git commit -a")
(sums *) $ git add sums.php                        (stage changes by adding)
(sums +) $ git commit -m "Append to sums.php"     ( +  prompt tells us have staged changes)
[sums fe76bdc] Append to sums.php                       (fe76bdc is SHA-1 commit hash)
 1 files changed, 1 insertions(+), 0 deletions(-)
(sums) $ echo 'print "3 + 3 = 6 \n";' >> sums.php
(sums *) $ git commit -a -m "Append more to sums.php"   (commit directly – no add)
[sums b89a9c9] Append more to sums.php                  (note new SHA-1 commit hash)
 1 files changed, 1 insertions(+), 0 deletions(-)
(sums) $
```

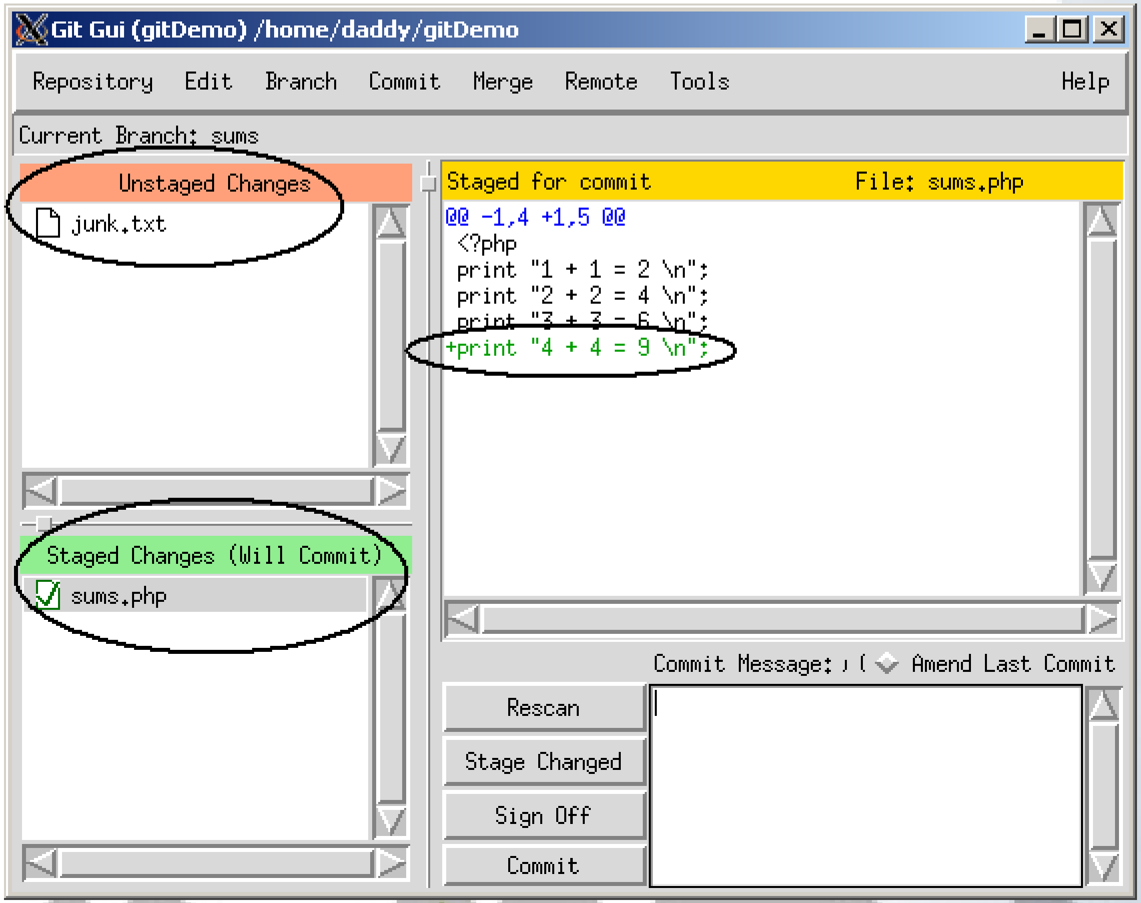
Staged versus unstaged changes

```
(sums) $ echo 'print "4 + 4 = 9 \n";' >> sums.php     (we'll fix error later)
(sums *) $ cat > junk.txt
We won't stage this file - just want to see unstaged in Git GUI
(sums *) $ git add sums.php          (stage change to sums.php)
(sums *) $ git status
# On branch sums
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
  directory)
#       modified:   sums.php
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#       junk.txt
no changes added to commit (use "git add" and/or "git commit -a")
(sums +) $
```

Staged versus unstaged via Git GUI

```
(sums +) $ git gui &
```

(we'll commit change to sums.php thru GUI)



Current Branch: sums

Unstaged Changes

junk.txt

Staged Changes (Will Commit)

☑ sums.php

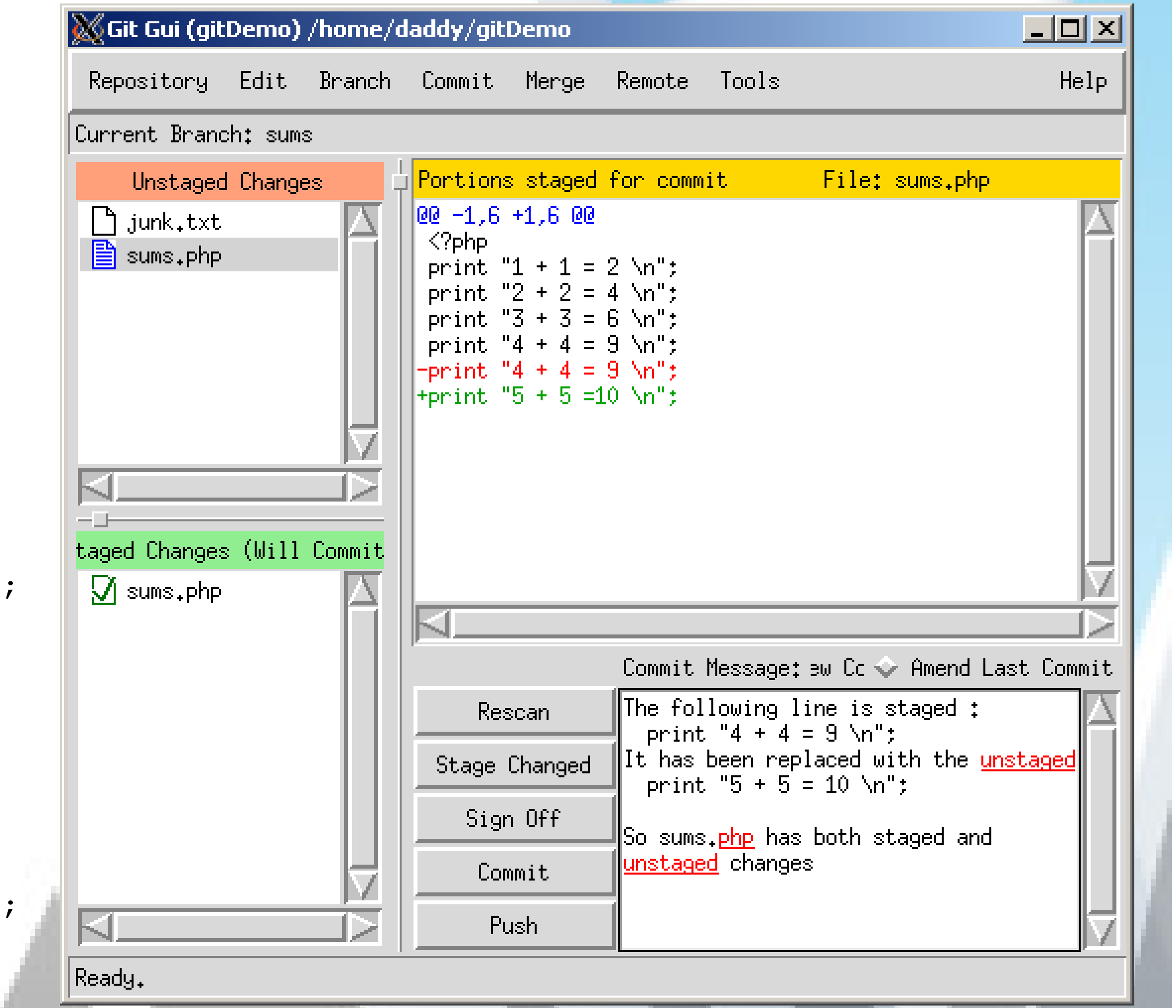
Staged for commit File: sums.php

```
@@ -1,4 +1,5 @@
 <?php
 print "1 + 1 = 2 \n";
 print "2 + 2 = 4 \n";
 print "3 + 3 = 6 \n";
+print "4 + 4 = 9 \n";
```

Commit Message:

Staging can lead to some *diff* oddities

```
(sums *+) $ git diff -U0
diff --git a/sums.php b/sums.php
index 8821e12..dfabbe4 100644
--- a/sums.php
+++ b/sums.php
@@ -6 +6 @@ print "4 + 4 = 9 \n";
-print "4 + 4 = 9 \n";
+print "5 + 5 =10 \n";
(sums *+) $ git diff --staged -U0
diff --git a/sums.php b/sums.php
index 301c44e..8821e12 100644
--- a/sums.php
+++ b/sums.php
@@ -5,0 +6 @@ print "4 + 4 = 9 \n";
+print "4 + 4 = 9 \n";
(sums *+) $ git diff HEAD -U0
diff --git a/sums.php b/sums.php
index 301c44e..dfabbe4 100644
--- a/sums.php
+++ b/sums.php
@@ -5,0 +6 @@ print "4 + 4 = 9 \n";
+print "5 + 5 =10 \n";
(sums *+) $
```



Current Branch: sums

```
@@ -1,6 +1,6 @@
 <?php
 print "1 + 1 = 2 \n";
 print "2 + 2 = 4 \n";
 print "3 + 3 = 6 \n";
 print "4 + 4 = 9 \n";
-print "4 + 4 = 9 \n";
+print "5 + 5 =10 \n";
```

The following line is staged :
  print "4 + 4 = 9 \n";
It has been replaced with the unstaged
  print "5 + 5 = 10 \n";

So sums.php has both staged and
unstaged changes

Ready.

Recommendations on staging

Commit directly and frequently when things 'work'

- When adding new code
- When making simple bug fixes
- `$ git commit -a -m "My commit message"` *(-a means 'add')*
- Instead of staging, just roll back to earlier commits if necessary

You really don't need to stage most changes

When to stage changes

- Single, complex bug
- You're "feeling your way" through bug resolution
- Make a final all-or-nothing commit for the entire bug

Seeing diffs visually

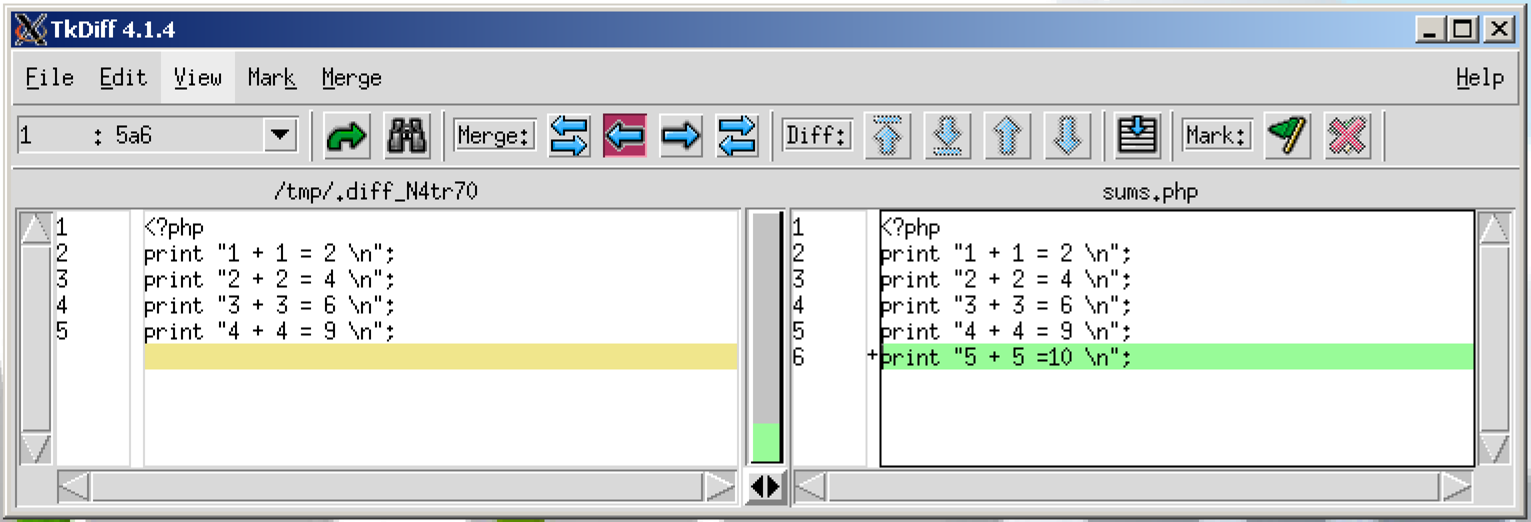
If running X-Windows, use tkdiff, kompare etc.

If terminal-based, try vimdiff

- Try to get a good set of colors for vimdiff
- http://luhman.org/blog/2009/08/25/git-difftool-and-vimdiff

```
(sums *) $ git difftool
merge tool candidates: tkdiff kompare  vimdiff
Viewing: 'sums.php'
Hit return to launch 'tkdiff':
```

(do this for each file)



```
<?php
print "1 + 1 = 2 \n";
print "2 + 2 = 4 \n";
print "3 + 3 = 6 \n";
print "4 + 4 = 9 \n";
```

```
<?php
print "1 + 1 = 2 \n";
print "2 + 2 = 4 \n";
print "3 + 3 = 6 \n";
print "4 + 4 = 9 \n";
+print "5 + 5 =10 \n";
```

Bisect your bugs away

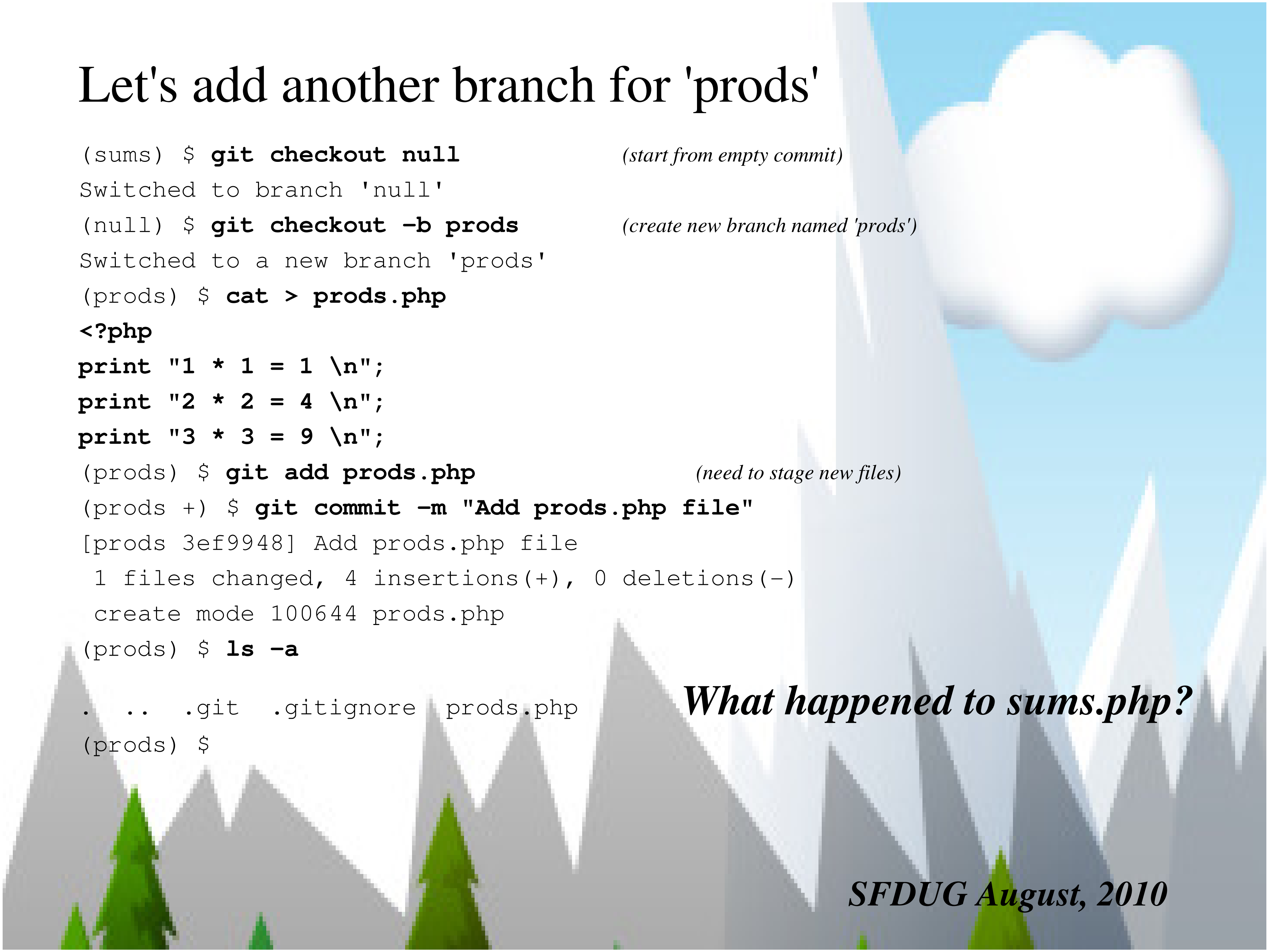
Find the commit which has "4 + 4 = 9"

```
(sums) $ git bisect start
(sums|BISECTING) $ git bisect bad          (mark current commit as bad)
(sums|BISECTING) $ git bisect good NULL    (mark initial tag as good)
Bisecting: 2 revisions left to test after this (roughly 1 steps)
[fe76bdc9f4095acb0b24fc6424640439e04e2abe] Append to sums.php
((fe76bdc...)|BISECTING) $ grep 9 sums.php    (didn't see '9', so good commit)
((fe76bdc...)|BISECTING) $ git bisect good fe76bdc9      (only need first few chars of hash)
Bisecting: 1 revisions left to test after this (roughly 1 steps)
[b9661cb236c246d8ee904778b7d7e468c330140c] Append mistaken 4+4=9.
((b9661cb...)|BISECTING) $ git bisect bad b9661cb
Bisecting: 0 revisions left to test after this (roughly 0 steps)
[b89a9c901db024cca1979550cf8c2db23bd8629a] Append more to sums.php
((b89a9c9...)|BISECTING) $ grep 9 sums.php
((b89a9c9...)|BISECTING) $ git bisect good b89a9c9
b9661cb236c246d8ee904778b7d7e468c330140c is first bad commit
commit b9661cb236c246d8ee904778b7d7e468c330140c
Author: Your name here <you@yourdomain.example.com>
Date:   Mon Jun 21 00:22:16 2010 -0700

    Append mistaken 4+4=9.

:100644 100644 28a07a97 301c44e0f1e814 M      sums.php
((b89a9c9...)|BISECTING) $ git bisect reset       (done with bisect)
Previous HEAD position was b89a9c9... Append more to sums.php
Switched to branch 'sums'
(sums) $
```

Let's add another branch for 'prods'



```
(sums) $ git checkout null                    (start from empty commit)
Switched to branch 'null'
(null) $ git checkout -b prods                (create new branch named 'prods')
Switched to a new branch 'prods'
(prods) $ cat > prods.php
<?php
print "1 * 1 = 1 \n";
print "2 * 2 = 4 \n";
print "3 * 3 = 9 \n";
(prods) $ git add prods.php                        (need to stage new files)
(prods +) $ git commit -m "Add prods.php file"
[prods 3ef9948] Add prods.php file
 1 files changed, 4 insertions(+), 0 deletions(-)
 create mode 100644 prods.php
(prods) $ ls -a

.  ..  .git  .gitignore  prods.php
(prods) $
```

What happened to sums.php?

Git switches file system under you when switching branches

Let's see both branches by merging together

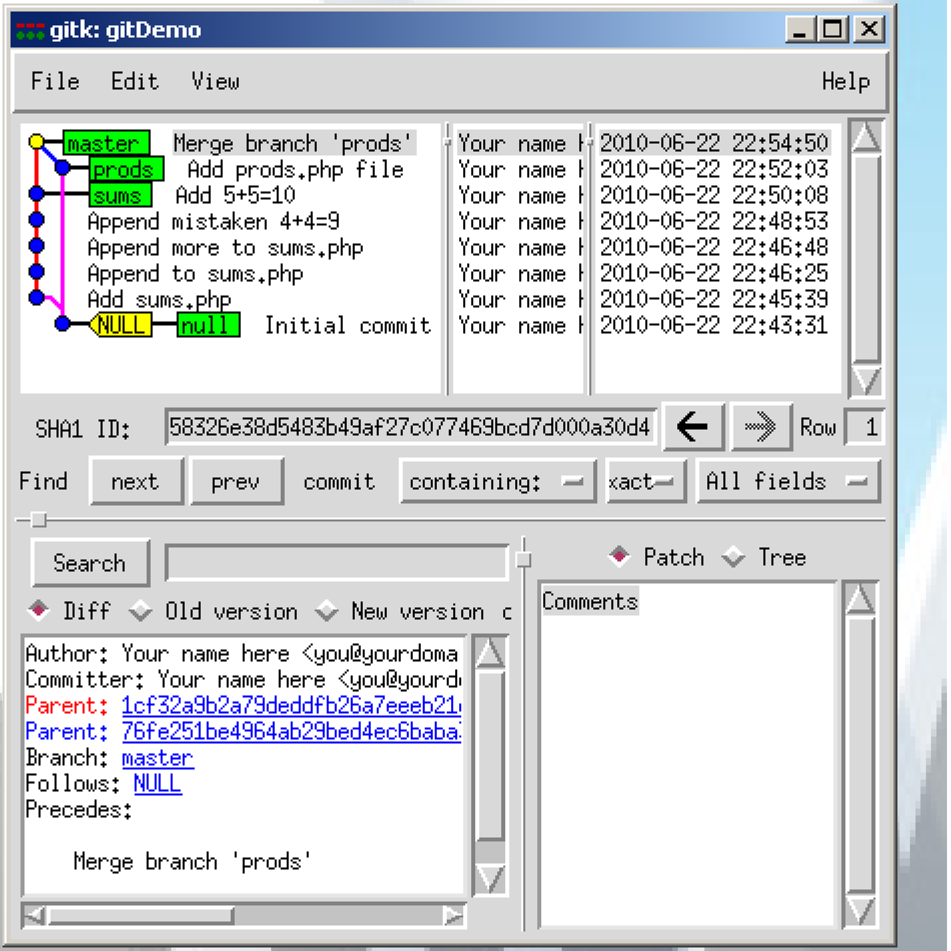
```
(prods) $ git checkout null                  (start from empty commit)
Switched to branch 'null'
(null) $ git checkout -b master              (create new branch 'master')
Switched to a new branch 'master'
(master) $ git merge sums                    (merge in the sums branch)
Updating cfdc52d..1cf32a9
Fast forward
 sums.php |    6 ++++++
 1 files changed, 6 insertions(+), 0 deletions(-)
 create mode 100644 sums.php
(master) $ git merge prods                   (merge in the prods branch)
Merge made by recursive.
 prods.php |    4 ++++
 1 files changed, 4 insertions(+), 0 deletions(-)
 create mode 100644 prods.php
(master) $ ls -a                             (now we have all files)
.  ..  .git  .gitignore  prods.php  sums.php
(master) $
```

Let's see our branches

```
(master) $ git branch -va
* master 58326e3 Merge branch 'prods'
  null   cfdc52d Initial commit
  prods  76fe251 Add prods.php file
  sums   1cf32a9 Add 5+5=10
(master) $ gitk &
(master) $
```

(verbose view of all branches)

(start GUI tool)



```
master  Merge branch 'prods'      Your name h  2010-06-22 22:54:50
prods   Add prods.php file        Your name h  2010-06-22 22:52:03
sums    Add 5+5=10                Your name h  2010-06-22 22:50:08
        Append mistaken 4+4=9     Your name h  2010-06-22 22:48:53
        Append more to sums.php   Your name h  2010-06-22 22:46:48
        Append to sums.php        Your name h  2010-06-22 22:46:25
        Add sums.php              Your name h  2010-06-22 22:45:39
NULL null  Initial commit         Your name h  2010-06-22 22:43:31

SHA1 ID: 58326e38d5483b49af27c077469bcd7d000a30d4   Row 1

Author: Your name here <you@yourdoma
Committer: Your name here <you@yourd
Parent: 1cf32a9b2a79deddfb26a7eeeb21
Parent: 76fe251be4964ab29bed4ec6baba
Branch: master
Follows: NULL
Precedes:

    Merge branch 'prods'
```

SFDUG August, 2010

Fixing bugs or adding features with 'topic branches'

- Think "New bug means new temporary branch"
- Even if fix belongs to a branch, may need to work in merged master

Drupal becomes inoperable if you switch to "drupal-contrib" with no *settings.php*

```
(master) $ git checkout -b masterAddSums          (create new, temp branch from master)
Switched to a new branch 'masterAddSums'
(masterAddSums) $ echo 'print "6 + 6 =12 \n";' >> sums.php       (add new feature)
(masterAddSums *) $ git commit -a -m "Add 6+6=12 on sums.php"
[masterAddSums ee3dc8f] Add 6+6=12 on sums.php
 1 files changed, 1 insertions(+), 0 deletions(-)
(masterAddSums) $ git diff --stat masterAddSums master           (diff --stat very useful)
 sums.php |     1 -
 1 files changed, 0 insertions(+), 1 deletions(-)
(masterAddSums) $ git log --pretty=oneline -1                    (find commit we added)
ee3dc8fba9e931a9678654602fc4d617bdd6a771 Add 6+6=12 on sums.php
(masterAddSums) $ git checkout master
Switched to branch 'master'
(master) $ git merge masterAddSums                          (merge change into master)
Updating 58326e3..ee3dc8f
1 files changed, 1 insertions(+), 0 deletions(-)
(master) $ git branch -d masterAddSums                      (no longer need temp branch)
Deleted branch masterAddSums (was ee3dc8f).
(master) $
```

Cherry pick last commit from *master* back to *sums* branch

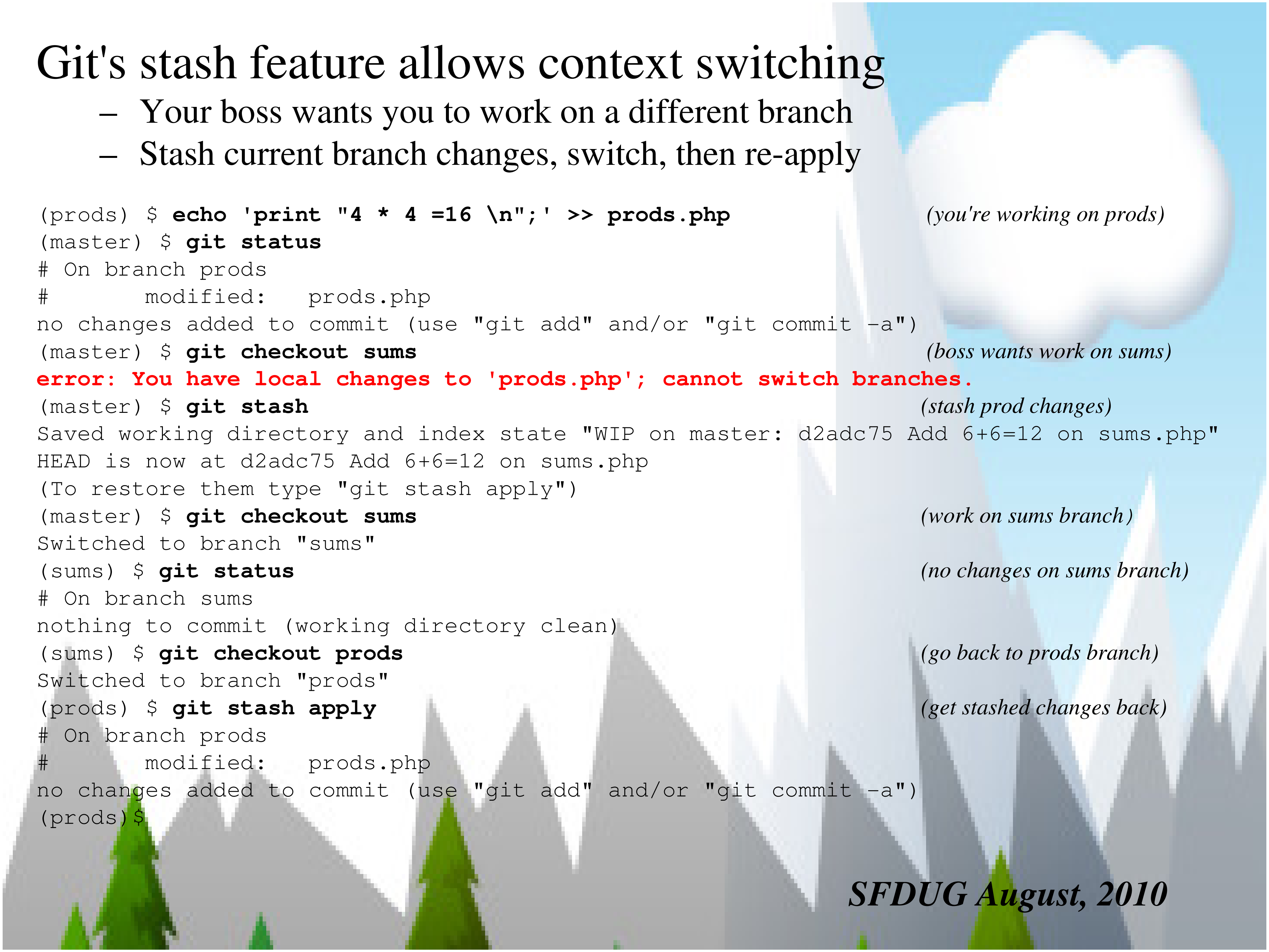
- Ensure *sums* branch reflects all changes in *master*
- This has costs & benefits
  - Cost of keeping branches up to date
  - Benefits of having a complete set of changes

```
(master) $ git checkout sums                  (switch to sums branch)
Switched to branch 'sums'
(sums) $ tail -1 sums.php                     (confirm sums.php lacks new line)
print "5 + 5 =10 \n";
(sums) $ git cherry-pick ee3dc8fba9e93        (apply commit from master – only need first chars from hash)
Finished one cherry-pick.
[sums 74af8cd] Add 6+6=12 on sums.php
 1 files changed, 1 insertions(+), 0 deletions(-)
(sums) $ tail -1 sums.php                     (confirm commit applied)
print "6 + 6 =12 \n";
(master) $
```

Git's stash feature allows context switching

- Your boss wants you to work on a different branch
- Stash current branch changes, switch, then re-apply

```
(prods) $ echo 'print "4 * 4 =16 \n";' >> prods.php              (you're working on prods)
(master) $ git status
# On branch prods
#       modified:   prods.php
no changes added to commit (use "git add" and/or "git commit -a")
(master) $ git checkout sums                                      (boss wants work on sums)
error: You have local changes to 'prods.php'; cannot switch branches.
(master) $ git stash                                              (stash prod changes)
Saved working directory and index state "WIP on master: d2adc75 Add 6+6=12 on sums.php"
HEAD is now at d2adc75 Add 6+6=12 on sums.php
(To restore them type "git stash apply")
(master) $ git checkout sums                                      (work on sums branch)
Switched to branch "sums"
(sums) $ git status                                               (no changes on sums branch)
# On branch sums
nothing to commit (working directory clean)
(sums) $ git checkout prods                                       (go back to prods branch)
Switched to branch "prods"
(prods) $ git stash apply                                         (get stashed changes back)
# On branch prods
#       modified:   prods.php
no changes added to commit (use "git add" and/or "git commit -a")
(prods)$
```



Let's move this to our prod server

Problem : *How to move to prod server, when dev server is behind a NATed firewall?*

Solution : *Copy Git repository to prod, and clone back to dev*

```
$ tar -zcf gitDemo.tar.gz gitDemo          (tar up existing repository)
$ scp gitDemo.tar.gz me@prod.com:/home/me
   ~ ~ (untar on prod.com) ~~
$ mv gitDemo gitDemoOrig      (don't clone over our original)
$ git clone ssh://me@prod.com/home/me/gitDemo
Initialized empty Git repository in /home/me/gitDemo/.git/
me@prod.com's password:
remote: Counting objects: 23, done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 23 (delta 5), reused 0 (delta 0)
Receiving objects: 100% (23/23), done.
Resolving deltas: 100% (5/5), done.
$
```

Let's see what we cloned back from prod

```
(master) $ git remote -v                             (verbose list of remote repos)
origin  ssh://me@prod.com/home/me/gitDemo
(master) $ git branch -va                            (verbose list of all branches)
* master                  58326e3 Merge branch 'prods'
  remotes/origin/HEAD     -> origin/master
  remotes/origin/master 58326e3 Merge branch 'prods'
  remotes/origin/null     cfdc52d Initial commit
  remotes/origin/prods    76fe251 Add prods.php file
  remotes/origin/sums     1cf32a9 Add 5+5=10
(master) $
```

This brings up a whole discussion of handling remote branches, and pushing to remote repositories.

http://luhman.org/blog/2009/07/28/git-push-how

A touch of advanced topics

Pre-commit hooks

- Never commit code with syntax errors
- http://luhman.org/blog/2010/02/12/cheap-php-lint-checking-git

The best way to handle merge conflicts

- Avoid them!
- Push and pull often

Continuous integration with Hudson

- Test builds with every Git commit
- http://luhman.org/blog/2009/12/16/installing-hudson-phing-phpunit-and-git-ubuntu

NetBeans or other IDE integration

CVS or SVN integration

Migrating Drupal to Git

Phase 1 : Read-only Git mirror of Drupal and contrib

Phase 2 : Drupal managed by *patches* with Git

- Limited or no branches
- Limited or no push/pull/fetch

Phase 3 : Fully embrace power of Git

http://drupal.org/community-initiatives/git

http://docs.google.com/present/view?id=dp6bhf4_6gfv8f5fq

Cloning Drupal 7

```
$ git clone  git://github.com/drupal/drupal.git
Initialized empty Git repository in /root/tmp/gitD7/drupal/.git/
...
Receiving objects: 100% (106674/106674), 21.37 MiB | 155 KiB/s, done.
Resolving deltas: 100% (79194/79194), done.
$ cd drupal
(CVS) $ git branch -va
* CVS                          386560b - Patch #295990 by mr.baileys, lilou:
  remotes/origin/CVS           386560b - Patch #295990 by mr.baileys, lilou:
  remotes/origin/DRUPAL-3-0    5e5375d - removed errors on empty meta tags:
  remotes/origin/DRUPAL-3-00   469e0dc - Renamed the SQL upgrade script.
  remotes/origin/DRUPAL-4-0    7244fbe - Made sure session.cache_limiter is
...
  remotes/origin/DRUPAL-4-7    86c491c push version
  remotes/origin/DRUPAL-5      bf60f36 Bump version number
  remotes/origin/DRUPAL-6      69b185e Now onto Drupal 6.18
  remotes/origin/HEAD          -> origin/CVS
  remotes/origin/drop          e9588cb Imported sources
(CVS) $
```

Pulling down fresh updates for Drupal 7

```
(master) # git remote -v
origin  git://github.com/drupal/drupal.git
(master) $ git pull origin CVS
From git://github.com/drupal/drupal
 * branch            CVS        -> FETCH_HEAD
Removing modules/overlay/images/loading.gif
Merge made by recursive.
 CHANGELOG.txt                                       |    8 +-
 includes/bootstrap.inc                              |   31 +-
 includes/common.inc                                 |   92 ++-
 includes/database/mysql/schema.inc                  |   10 +-
 includes/database/pgsql/database.inc                |    9 +-
...
```

Site builders : Handling contrib modules with Git

Work in progress for me but seems like three choices :

Choice 1 : Download and manage tarballs

- Simple, but loses Git tracking of mods to repositories on drupal.org
- http://drupal.org/node/803746

Choice 2 : Manage with Git's submodule feature

- Probably best tracking, but submodules can be tricky
- http://progit.org/book/ch6-6.html

Choice 3 : Manage with Git's subtree merge method

- Still tricky, but possibly easier than submodules
- http://progit.org/book/ch6-7.html

Questions
and
Thank you!

David Luhman
luhman.org
linkedin.com/in/davidluhman

SFDUG August, 2010