Git Along!

Version control to the rescue

@eporama

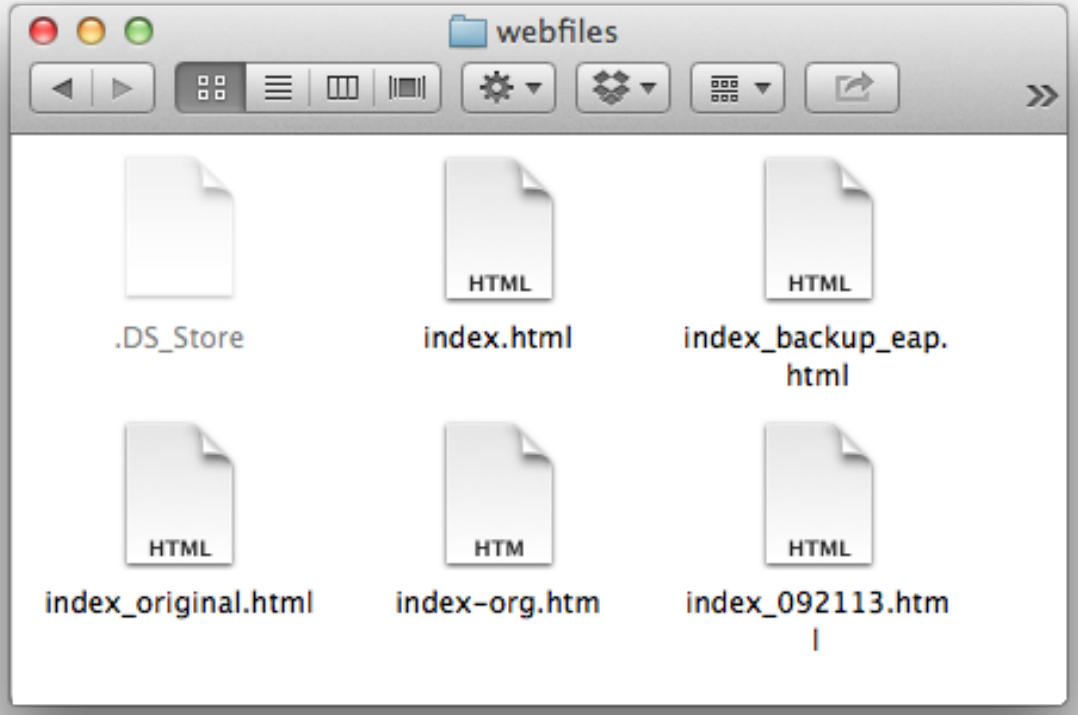
Erik Peterson

Support Manager at Acquia, Inc.

https://drupal.org/user/166970

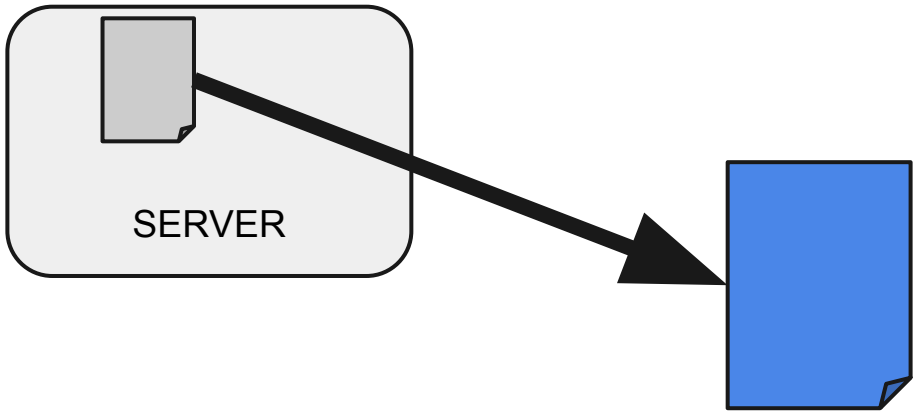
The way it was...

- Undo (multiple)
- Save As…
- Zip files
- Notes



Two’s a crowd

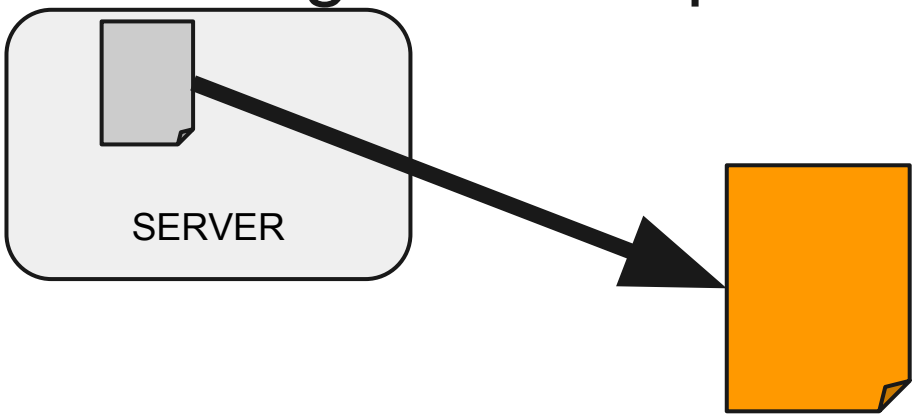
1. Erik gets a copy of layout.css from server
2. Erik starts to fix issue



Pat rocks

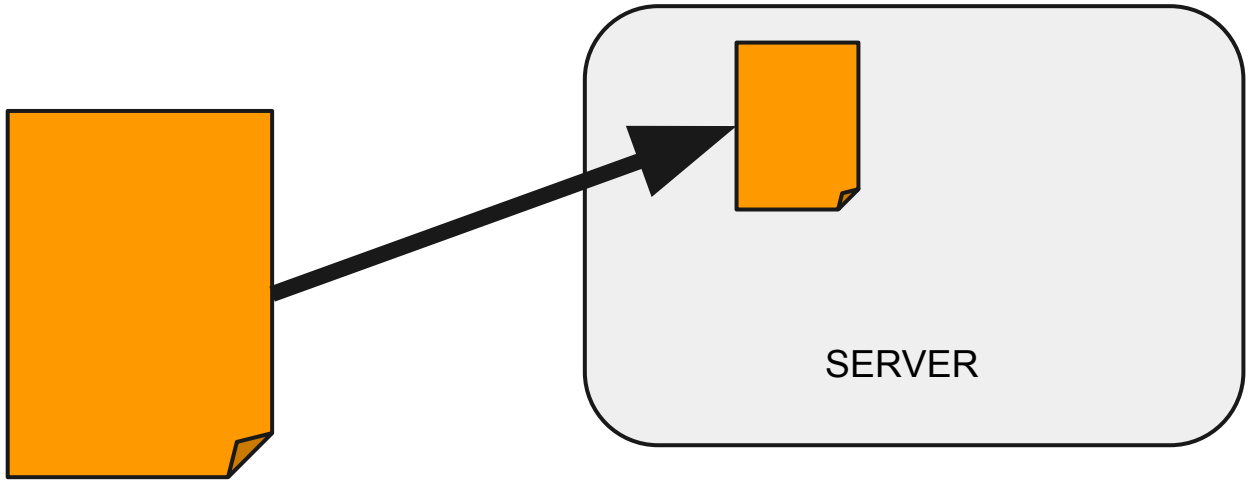
(Erik still has his file...)

1. Pat gets a copy of layout.css from server
2. Pat is working on other problem



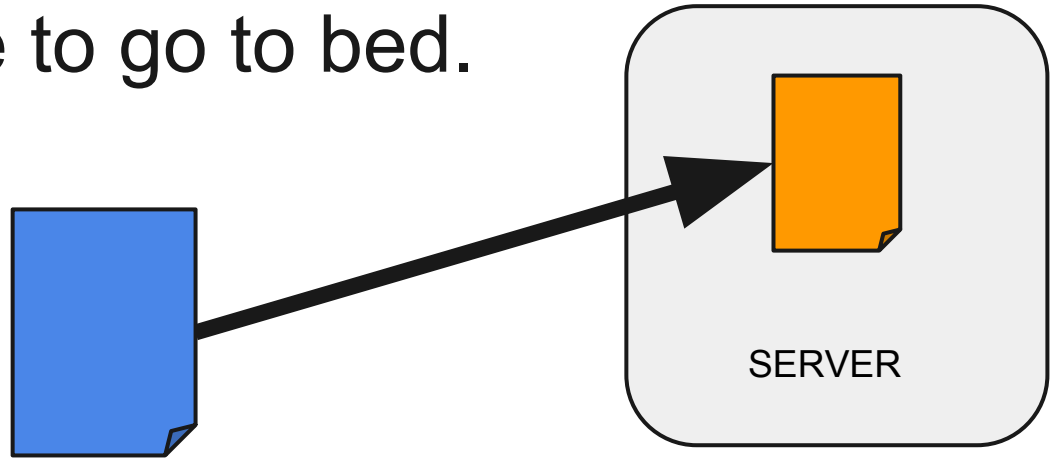
Pat's fix is quick

1. Pat saves and uploads his changes
2. Pat checks website and voilà, all good.
3. Pat goes home and has a beer.



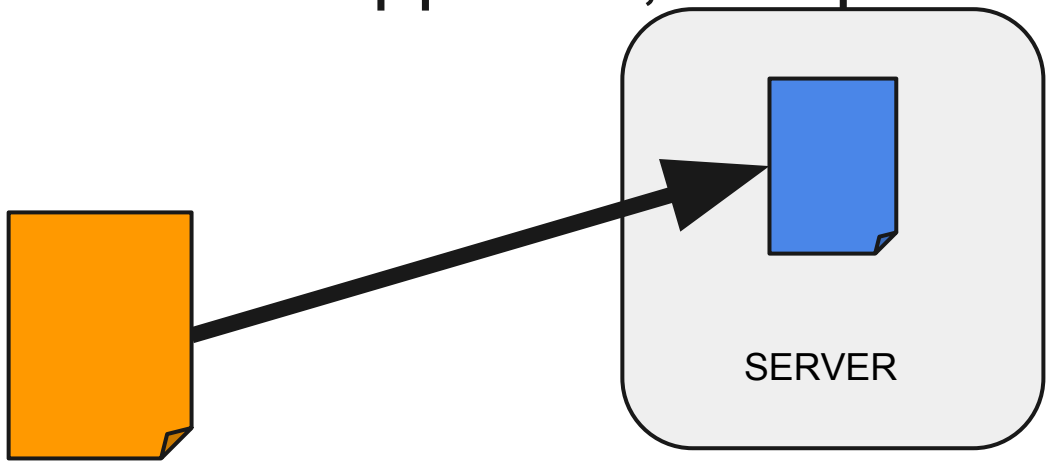
Erik finally finishes

1. Erik saves and uploads his changes, but since he didn't know Pat was working on the file, he overwrites Pat's changes.
2. Erik checks his issue out on the web and goes home to go to bed.



Pat wonders if he's crazy

1. The boss calls Pat to say "I thought you fixed this thing?"
2. Pat is sad.
3. Pat isn't sure what happened, so uploads his file again.



A groggy Erik

1. Boss wakes Erik up with a txt “Ur prob bck. Plz fix asap.”
2. Erik is very confused…

Problem repeats until boss fires both of us and hires someone who knows version control.

Version control

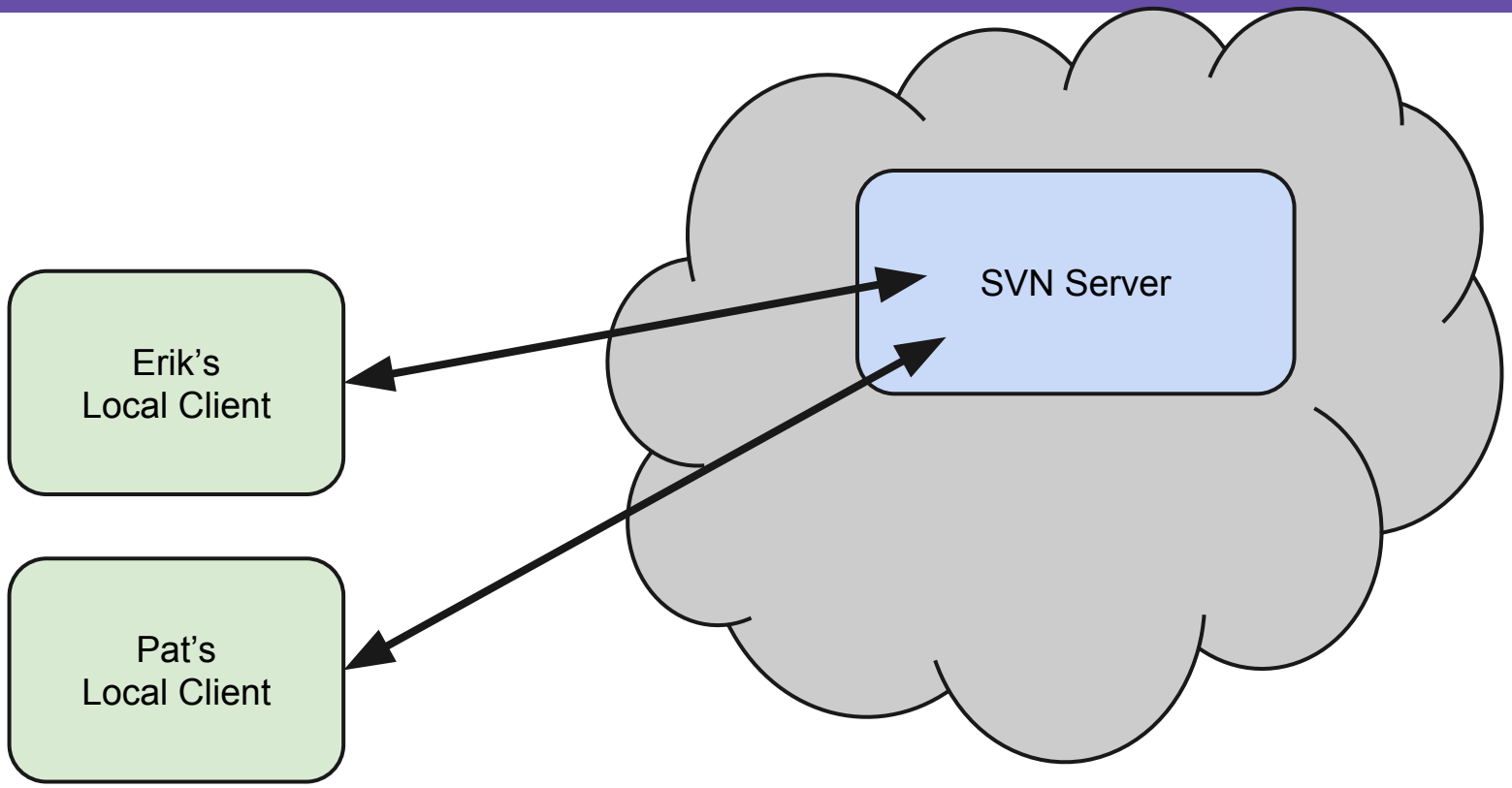
SVN, git, bzr, mercurial, CVS, …

Two main flavors:

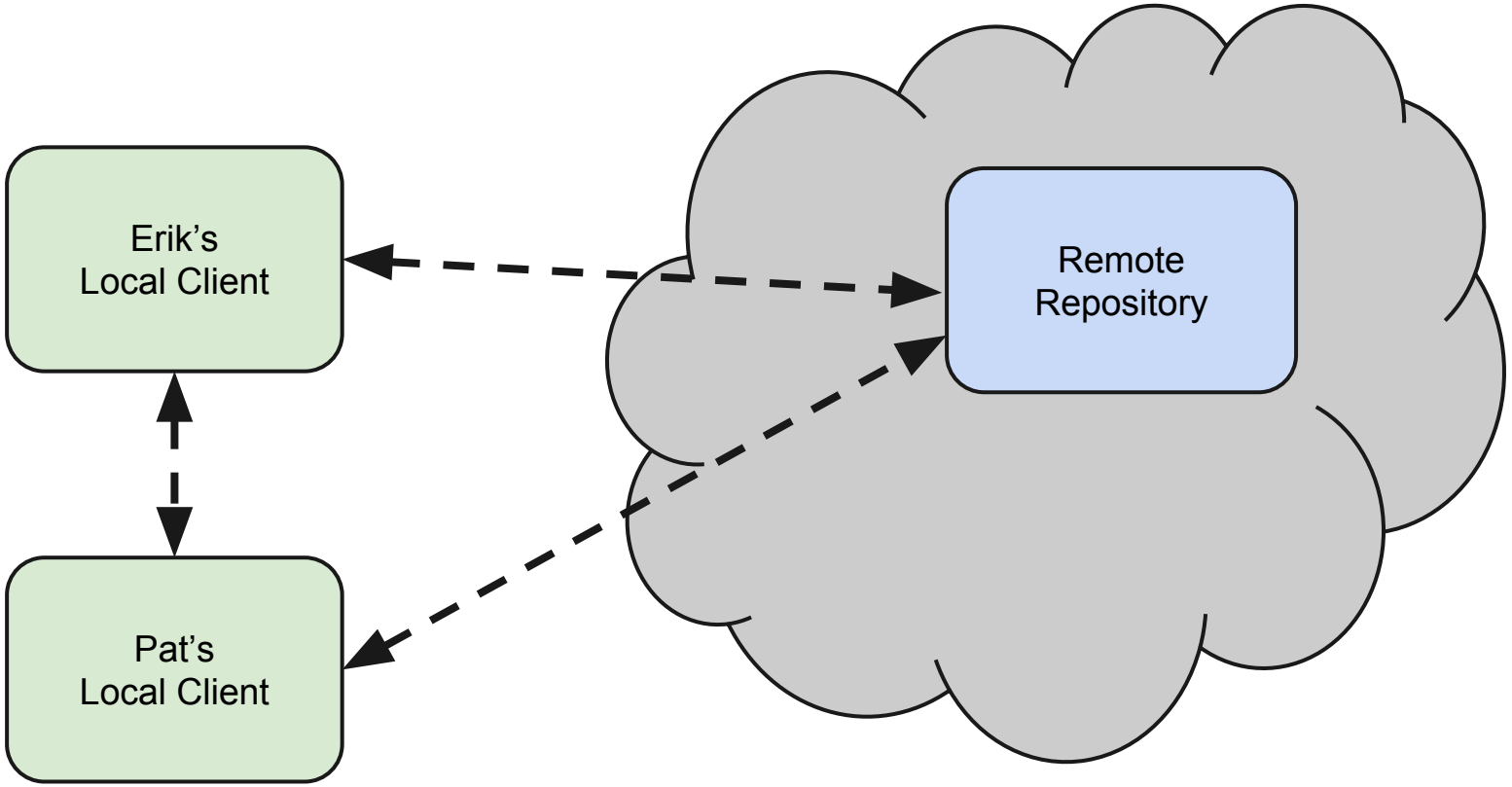
1. Client-server (SVN and CVS)

2. Distributed model (git, bzr, mercurial)

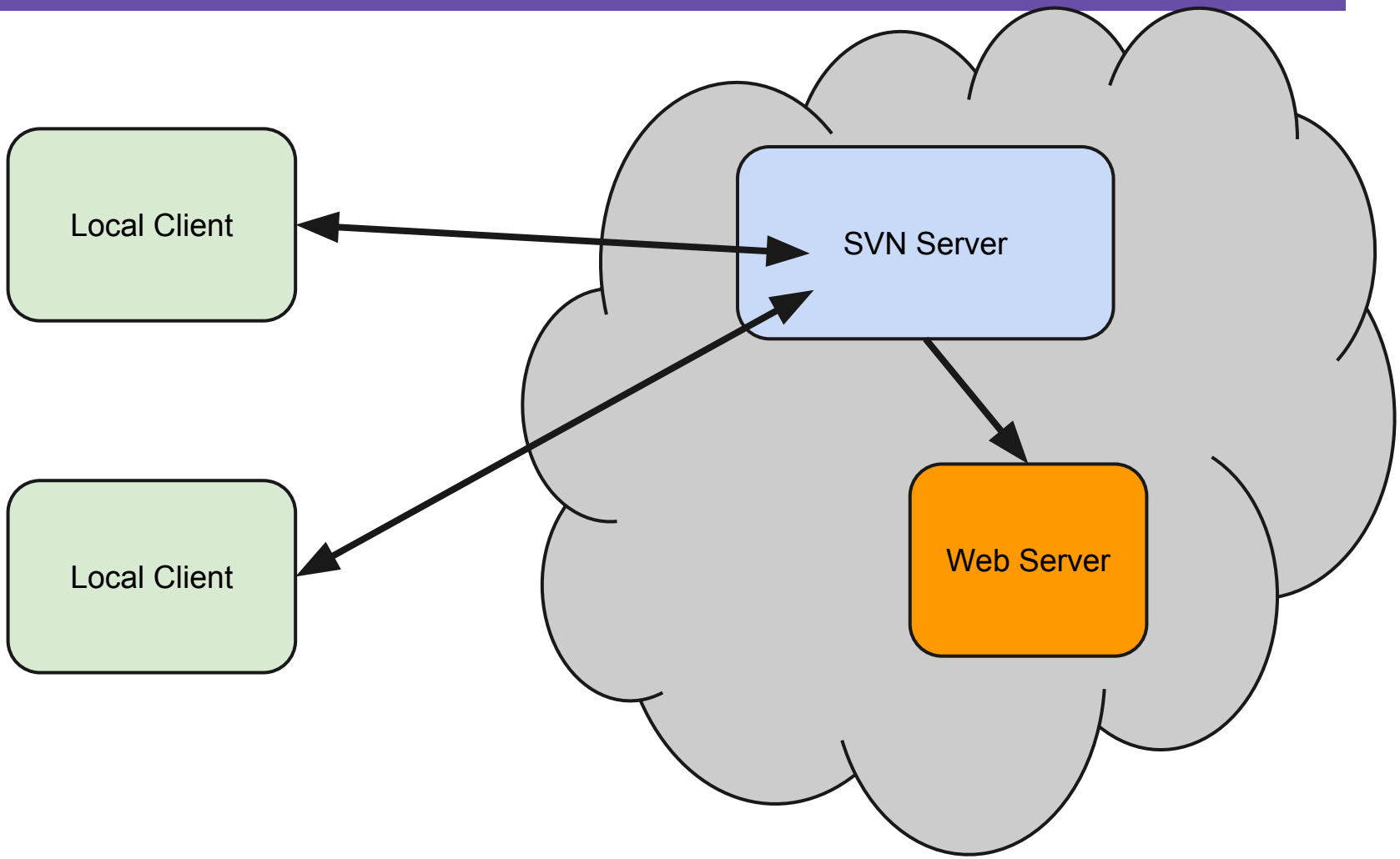
Client-Server model



Distributed Model (git)



Web Server == new client



Lines in layout.css

```
body {
  font-size: x-large;
}
```

```
body {
  font-size: x-large;
  font-weight: bold;
}
```

```
body {
  font-size: x-large;
  text-align: right;
}
```

diff

```
body {
  font-size: x-large;
}
```

```
body {
  font-size: x-large;
  font-weight: bold;
}
```

diff is:

```
“insert ‘font-weight: bold;’
   between lines 2 and 3”
```

new vs. old

```
$ diff old.css new.css
2a3
>    font-weight: bold;
```

Pretty hard to parse by humans

No context

diff -u

```
$ diff -u old.css new.css
--- old.css   2013-09-14 20:20:47.000000000
-0400
+++ new.css   2013-09-14 20:20:36.000000000
-0400
@@ -1,3 +1,4 @@
 body {
    font-size: x-large;
+   font-weight: bold;
 }
```

git diff

```
diff --git a/main.css b/main.css
index 1c29f7f..edd1da5 100644
--- a/main.css
+++ b/main.css
@@ -1,3 +1,4 @@
 body {
    font-size: x-large;
+   font-weight: bold;
 }
```

Getting started

Install git

http://git-scm.com/downloads

Homebrew

```
brew install git
```

This installs the command line things

GUIs are optional

Terminology

- Working Directory
  - The actual files on your disk that you open and edit
- Repository
  - The “history” of all the changes
  - “.git” directory in the Working Directory
- Staging area
  - sort of a “meta” area. Used to store up which changes are going to be applied at one time.

Setting up git

Set up git for tracking who you are:

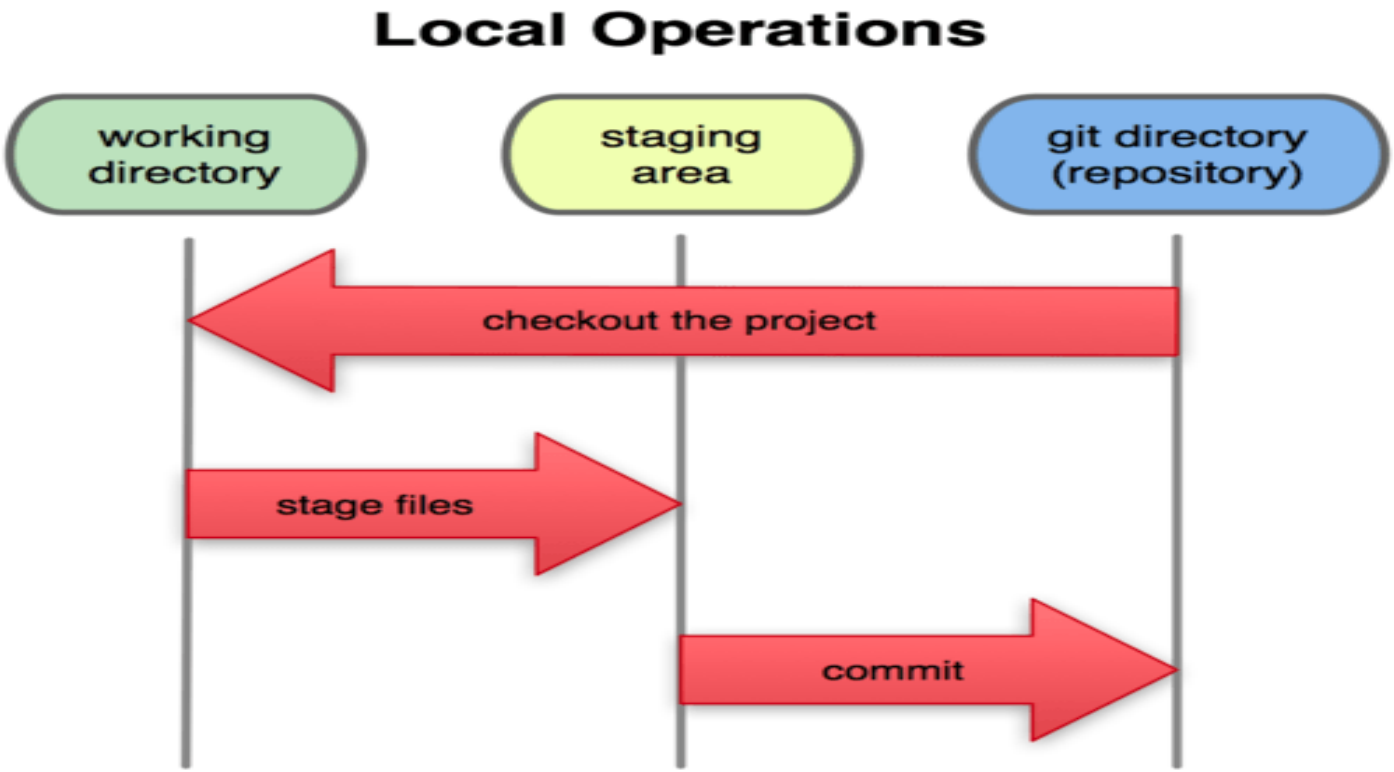
```
$ git config --global user.name "Your Name"
$ git config --global user.email "your_email@whatever.com"
```

Initialize a repo

```
cd /path/to/working_directory
git init
```

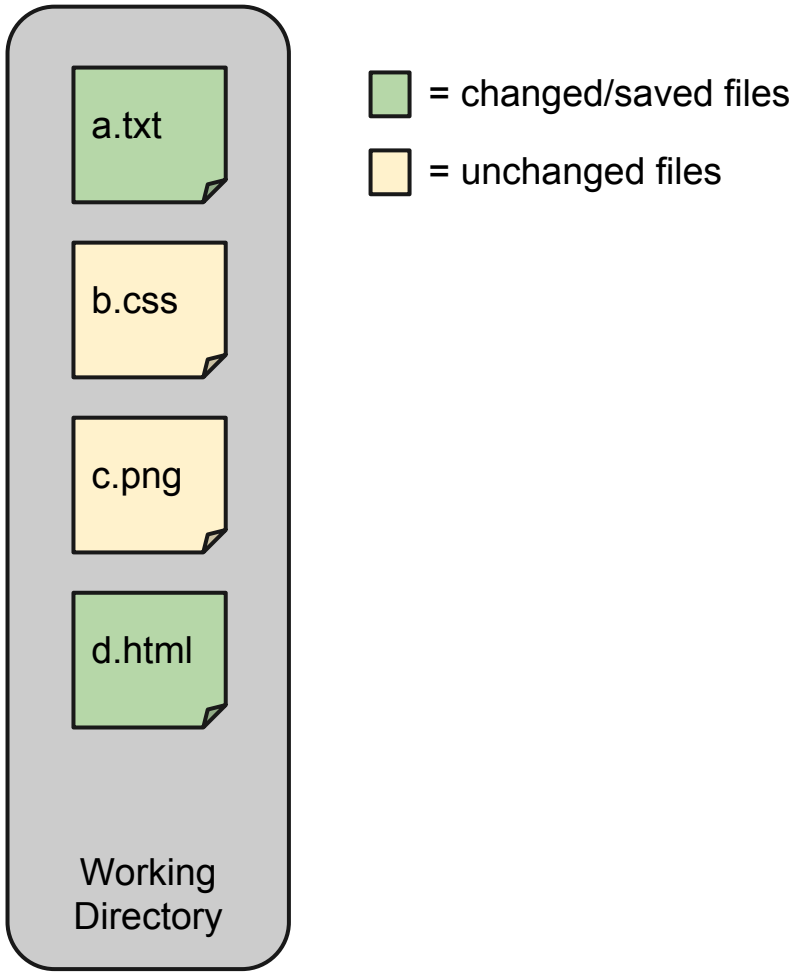
Yep, that's it.

Stages

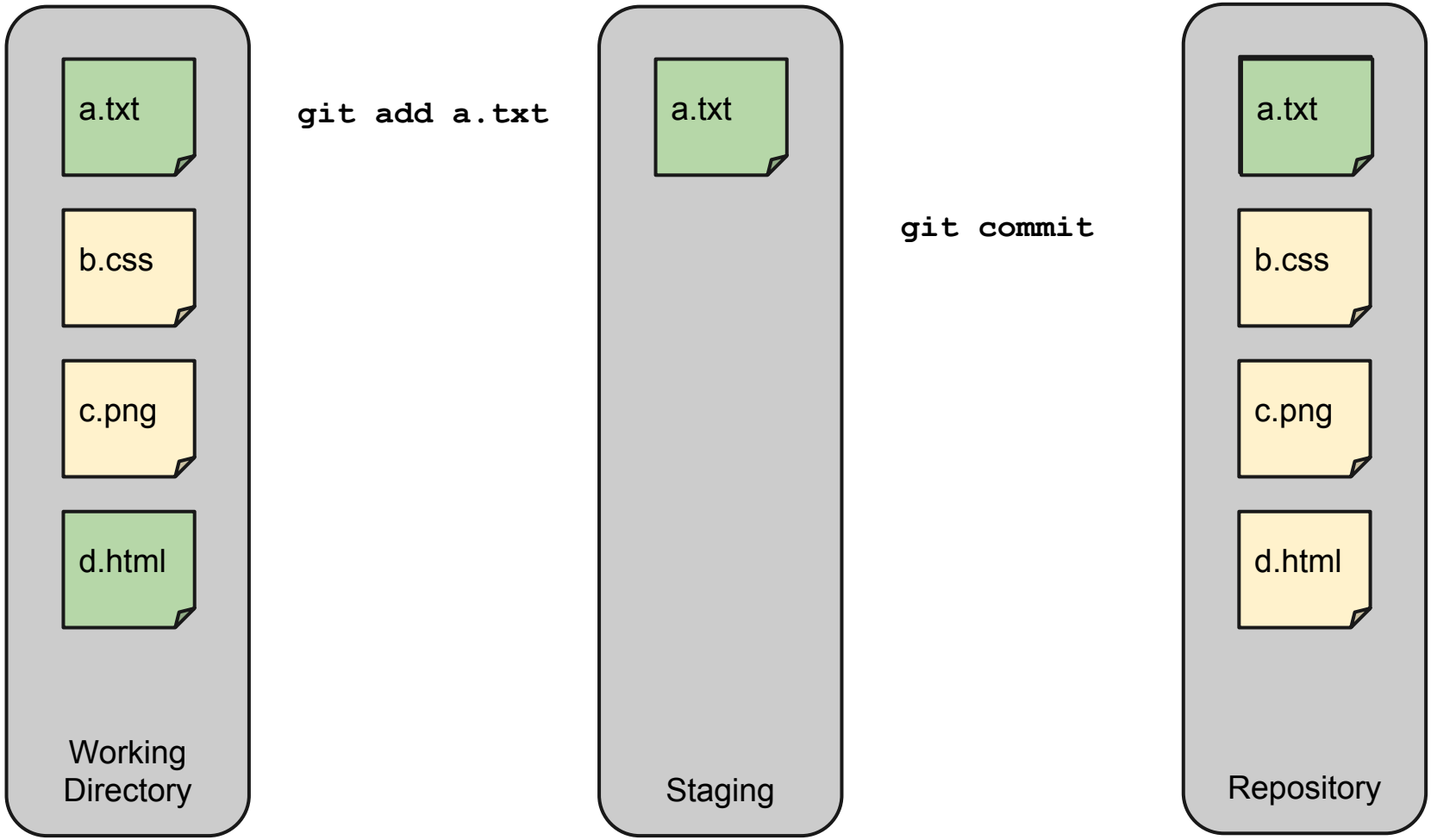


http://git-scm.com/book/en/Getting-Started-Git-Basics

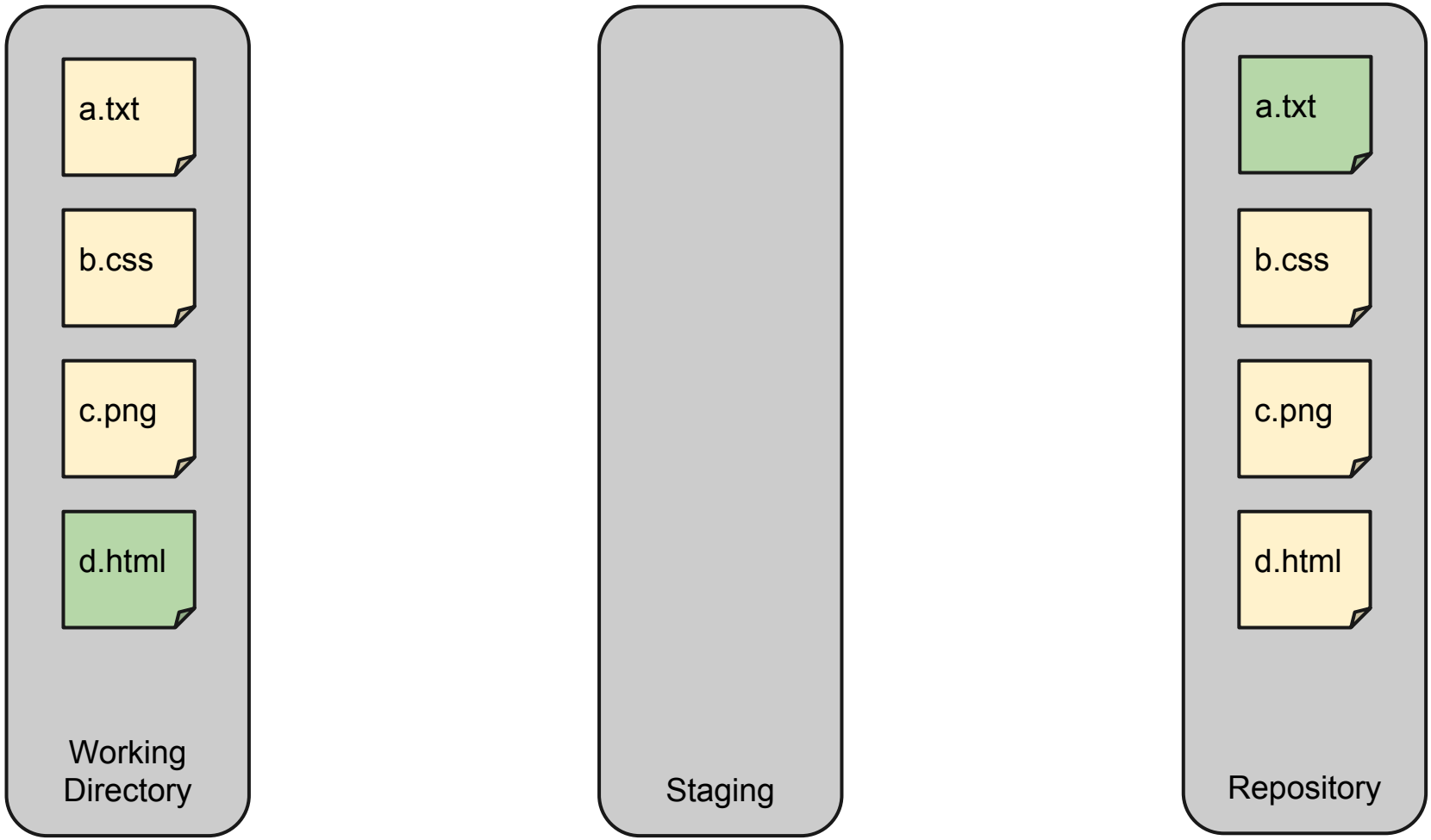
Working Directory



Staging


a.txt
b.css
c.png
d.html
Working Directory
git add a.txt
a.txt
Staging
git commit
a.txt
b.css
c.png
d.html
Repository

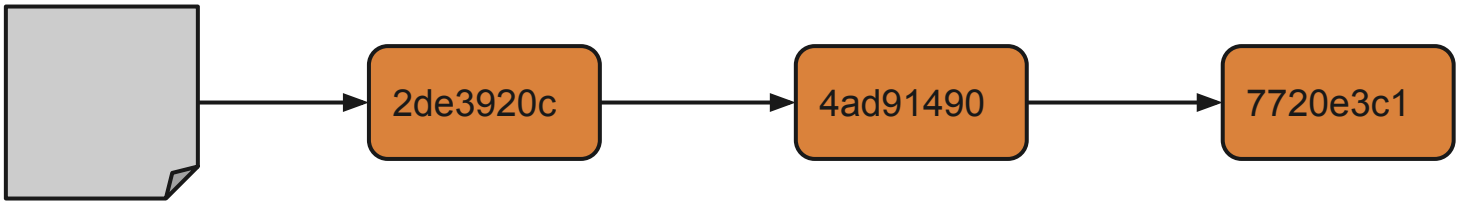
Committed


a.txt
b.css
c.png
d.html
Working Directory
Staging
a.txt
b.css
c.png
d.html
Repository

Commits

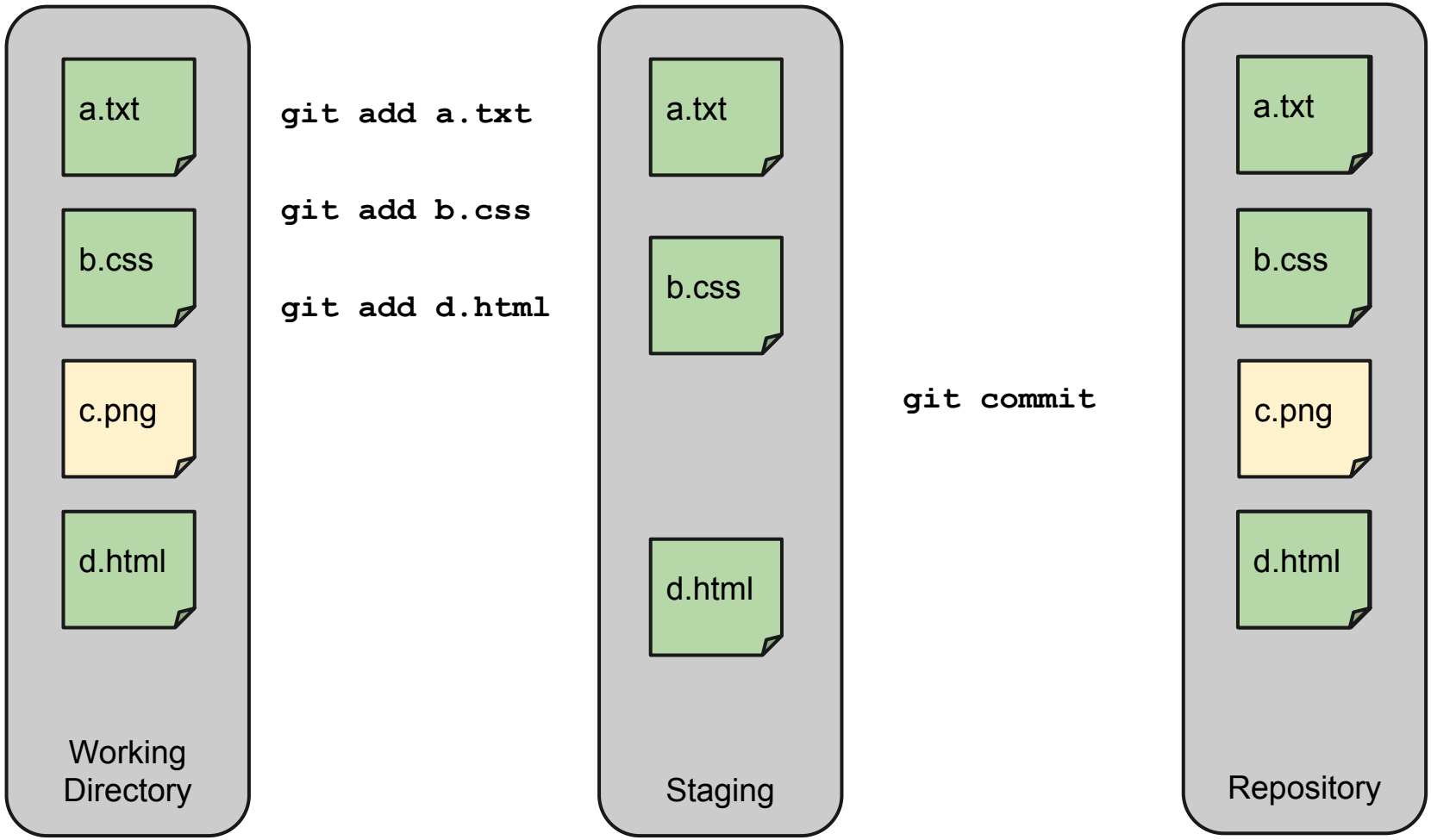
Each commit has an identifier (hash)

Commits play one after the other




```
git commit -m "An informative, yet brief message"
```

Multiple files



Tags

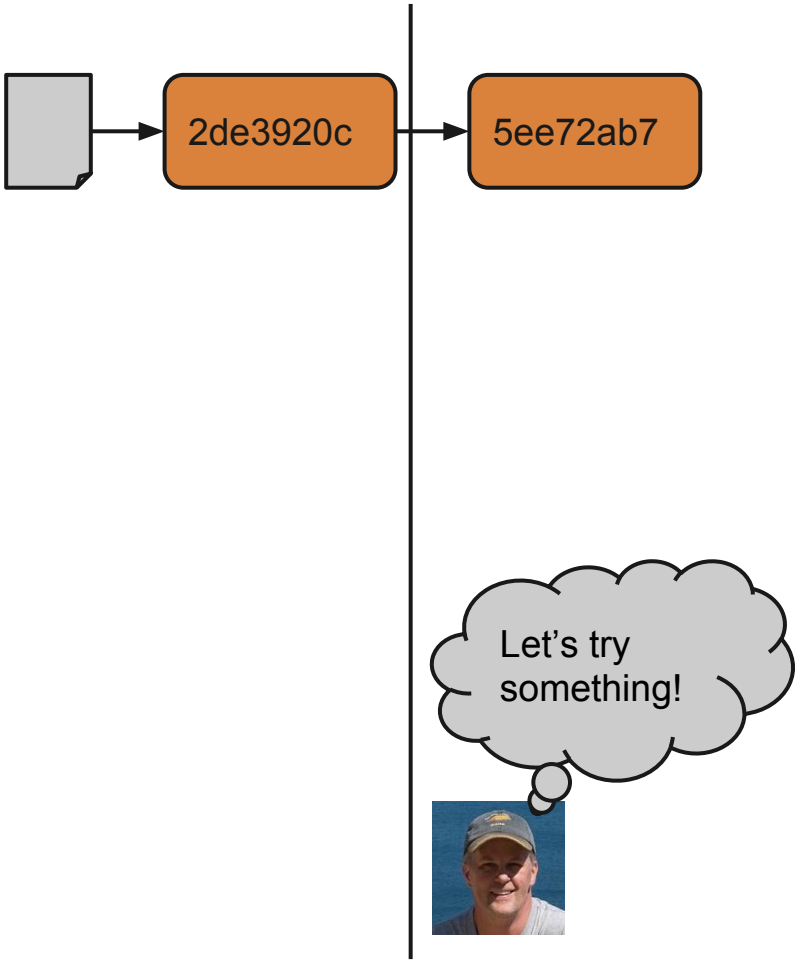
Just a friendly name to a specific commit

Commits referred to by hash

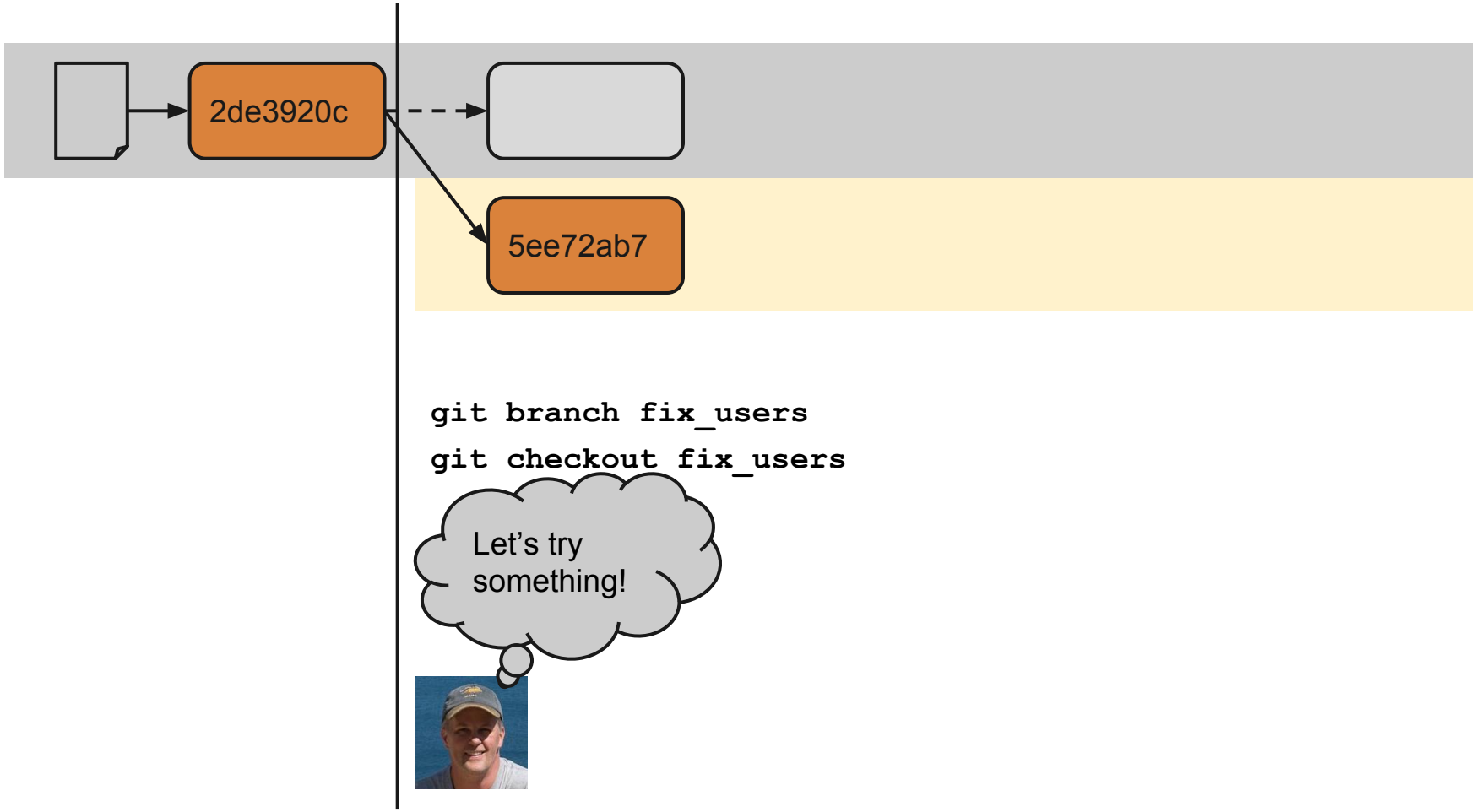
```
$ git log
commit 7720e3c1323fa752378710005775 6715c5feeab8
Author: Erik Peterson <erik.peterson@acquia.com>
Date:   Sun Sep 15 13:16:30 2013 -0400

$ git tag sept-release 7720e3c1323fa
```

Branching



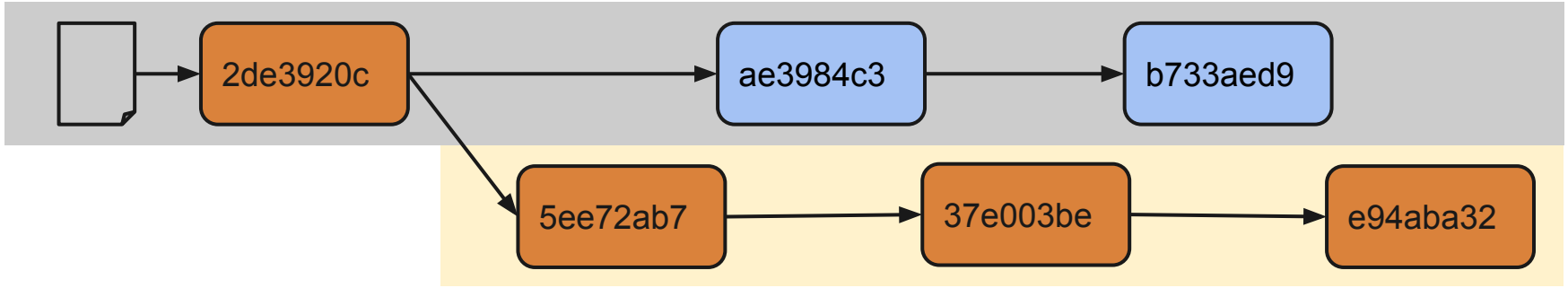
Branching



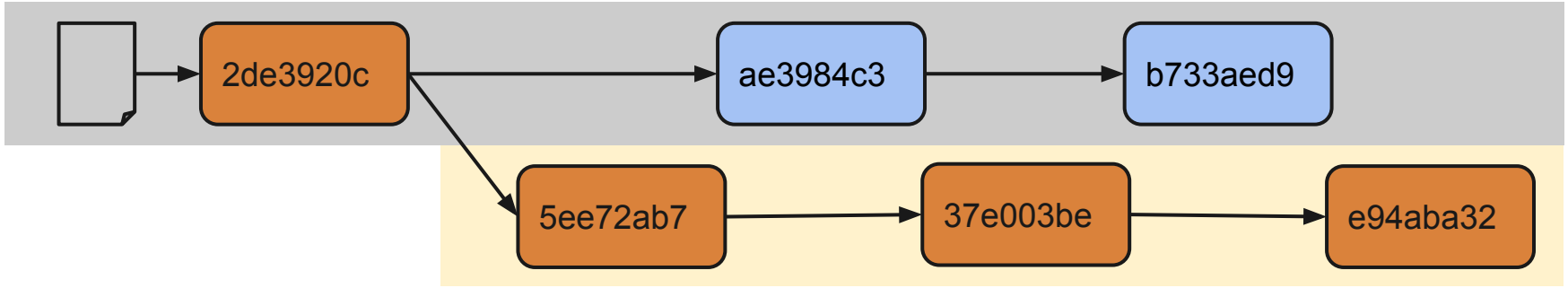

```
git branch fix_users
git checkout fix_users
```

Let's try something!

Why branch?



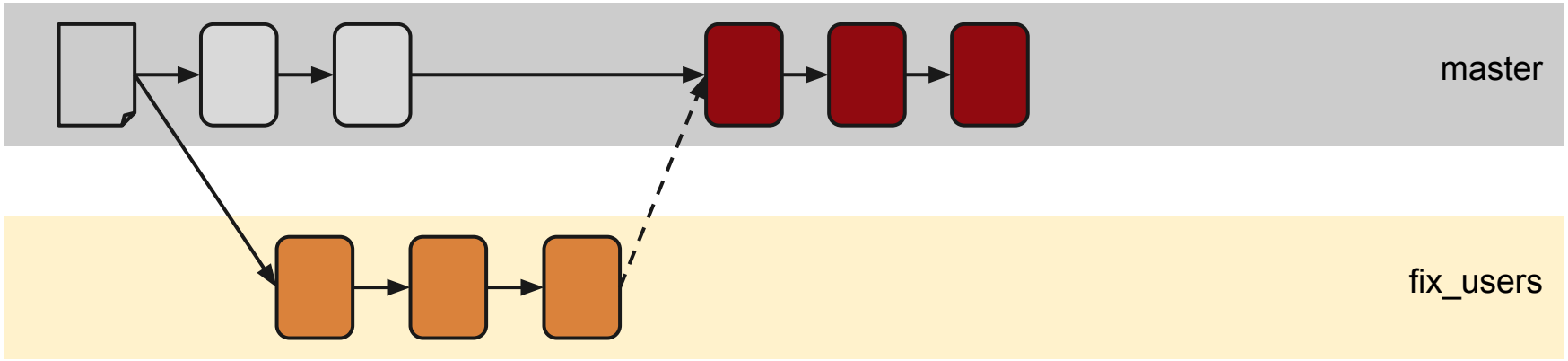
Whoops!



```
git branch -D fix_users
```

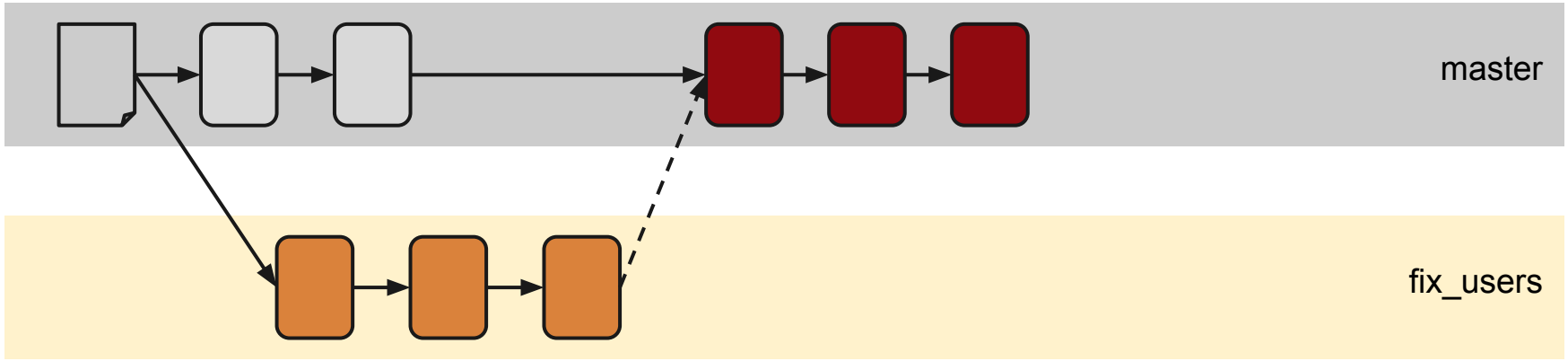
Whoops!

Merge ahead



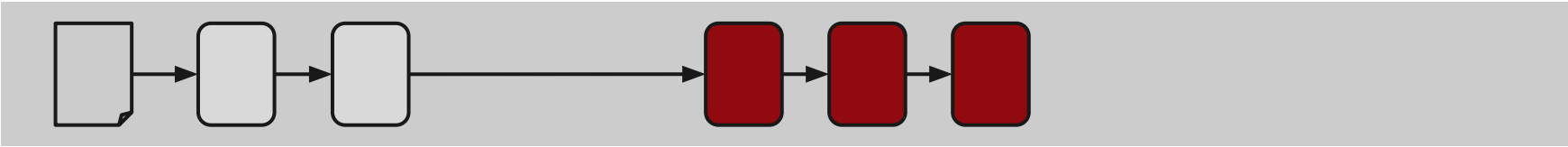

```
git checkout master
git merge fix_users
```

All Done!



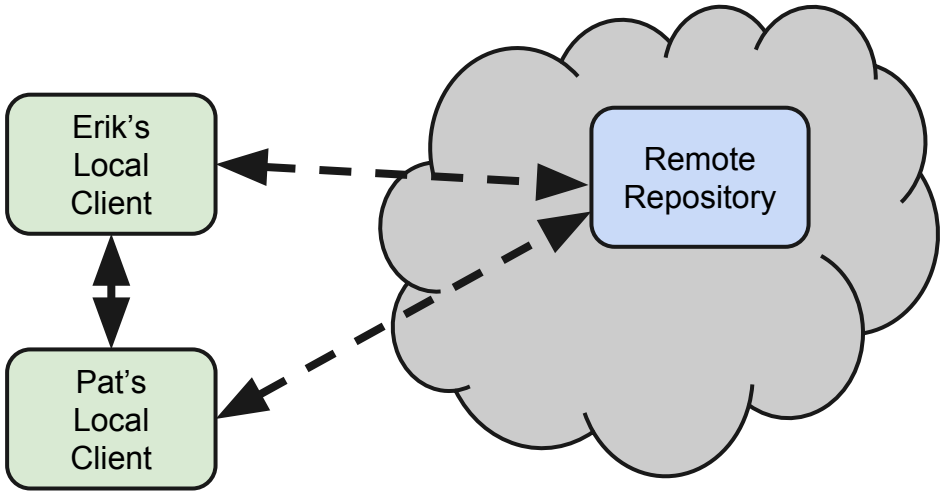

```
git checkout master
git merge fix_users
git branch -d fix_users
```

All Done!



```
git checkout master
git merge fix_users
git branch -d fix_users
```

Remotes




```
git remote add {alias} {uri}

git remote add gith git@github.com:eporama/dcnj.git
```

Pushy, pushy

```
git push {alias} {branch}
git push gith master
git push gith fix_users
```

Fetch

```
git fetch {alias}
```

gets all work done from the remote done on the branch

A clone by any other name

```
git clone {uri}
```

equivalent of a couple of functions

- init
- remote add
- fetch
- checkout

```
git clone git@github.com:eporama/blob.git
```

Push Me, Pull You

```
git pull {alias}
git pull dcnj
```

after a clone:

```
git pull
```

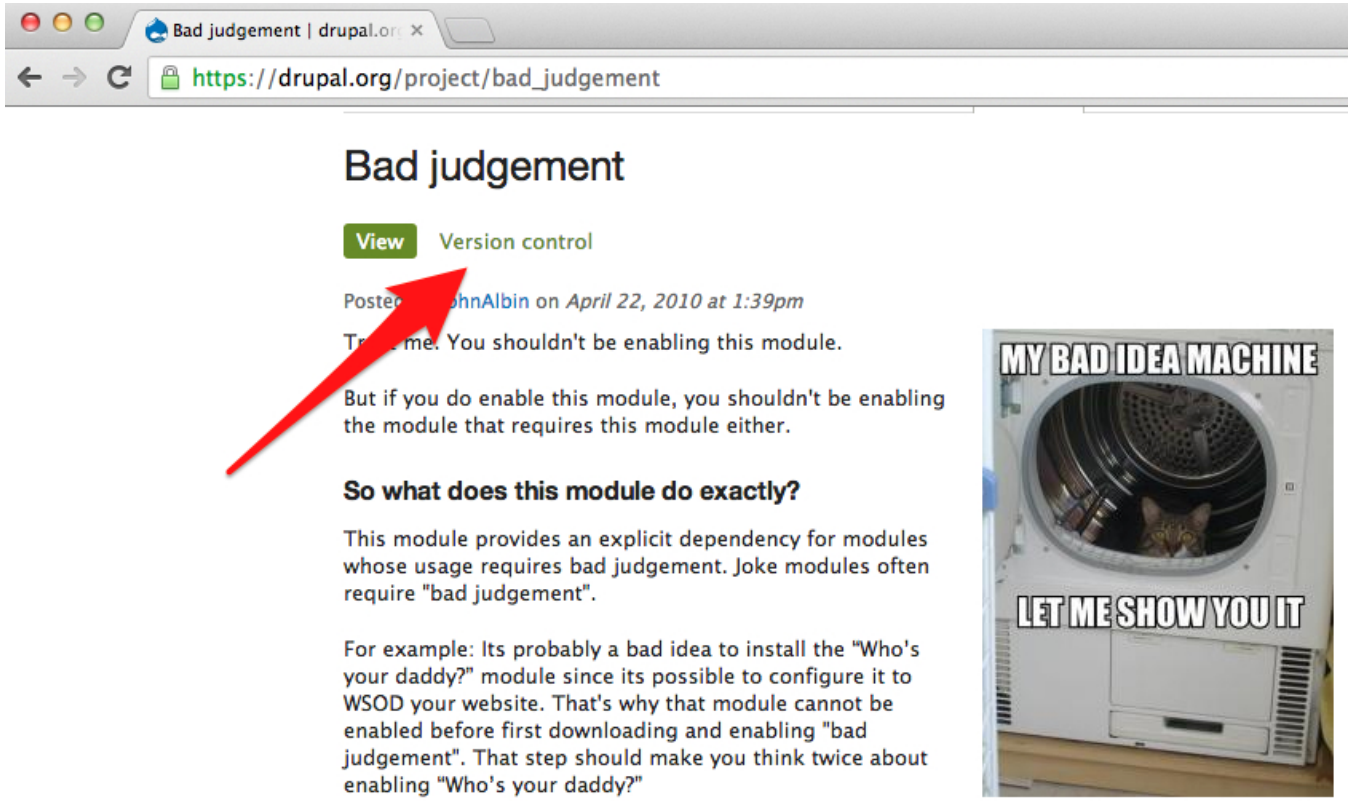
Workflows

- Always make production a tag
- Work on branches
- Merge to master
- Push branches when they’re shared

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

Git and Drupal

Each project has a “Version control” link



Acquia®

http://www.acquia.com

Want to learn more about Drupal and the systems it runs on?

We're hiring!

Acquia™

More Drupally gitness

Git Documentation

https://drupal.org/documentation/git

Contributing patches

https://drupal.org/node/707484