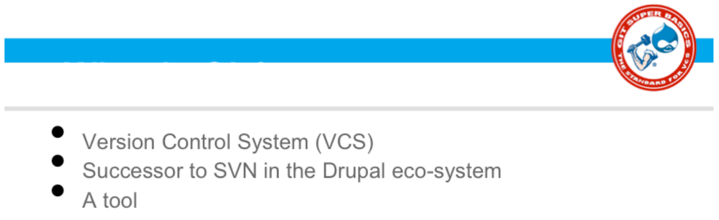
Max: Welcome to Git Super Basics. My name is Max Bronsema and I work at Western Washington University in Bellingham, WA.

Vid: My name is Vid Rowan and I work at the University of Oregon. We hope you are in the right room. As you get settled please take a look at the reference sheet to prime yourselves on the vocabulary.

Max: Vid and I use git at our jobs at WWU and the UO and what we will be sharing with you are some basic methods of using git. In git there are many ways to accomplish a task so if you have seen it done differently it is probably a valid way to do it. If you use git a lot, you may not get much from this session, but if you are new to it or have never touched it except to install it prior to this session you are in the right place.

Next Slide



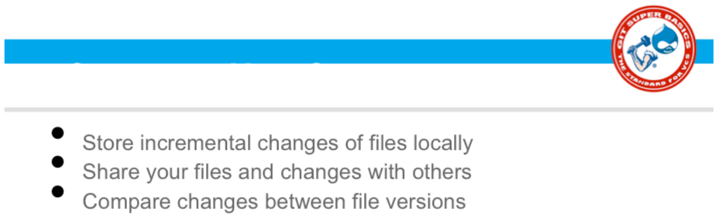
- Version Control System (VCS)
- Successor to SVN in the Drupal eco-system
- A tool

Max: Git is a version control system created by Linus Torvalds, the creator of Linux. Apparently the word is British slang for a stupid or unpleasant person. So, we are going to use a stupid content tracker today. It has rapidly increased in popularity and is used by many projects and individuals today. It replaced subversion as the version control system for the Drupal project and is now how we apply patches, pull the latest code, and generally manage projects.

Version control systems such as git simply allow us to track changes to files at any point in time. Think of saving a file name with a new file name every time you update a file. Git kind of does that but you get to keep the same file name and can see the history of all the changes made.

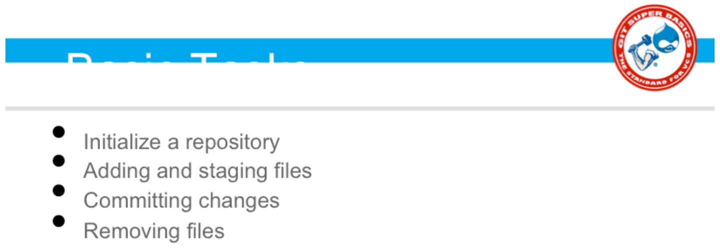
With software development and our Drupal websites git is a tool we can use to streamline workflows and allow us a safety in case we really screw up. It also allows us to collaborate with the Drupal community.

Next Slide



- Store incremental changes of files locally
- Share your files and changes with others
- Compare changes between file versions

Here we just need to discuss the common use cases. Not much detail is needed though.

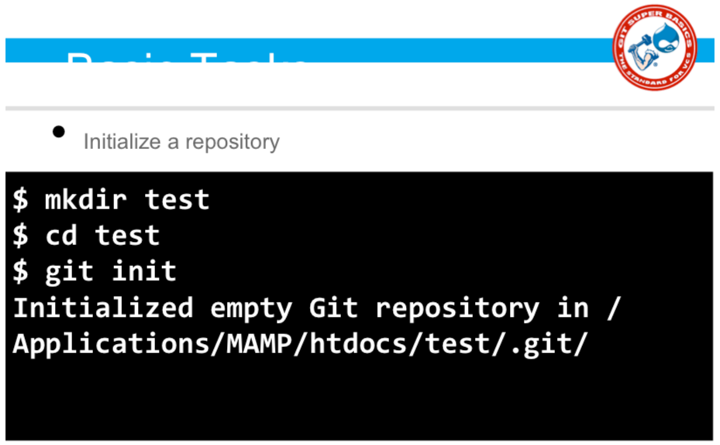


- Initialize a repository
- Adding and staging files
- Committing changes
- Removing files

Vid: To get started with git we need to learn about the following tasks. We are going to take you through a simple project that we have setup on GitHub.

Switch to Terminal

1. Make a new folder for the project and go into it.
2. Create a new repository $ git init
3. Add the HTML5 starter template file into the project folder.
4. Add a stylesheet file to the project folder.
5. $git status
   a. Discuss what the status screen is saying
6. $git add filename
7. $git status
   a. Show one file is added and another is in the directory but not added
8. $git add .
9. $git commit -m "Initial Commit"
   a. Explain that commit messages should be fairly short and descriptive
10. Add an extra file that we don't want to track like a .project file
    a. Show how to untrack the file
11. $git rm --cache filename will untrack the file



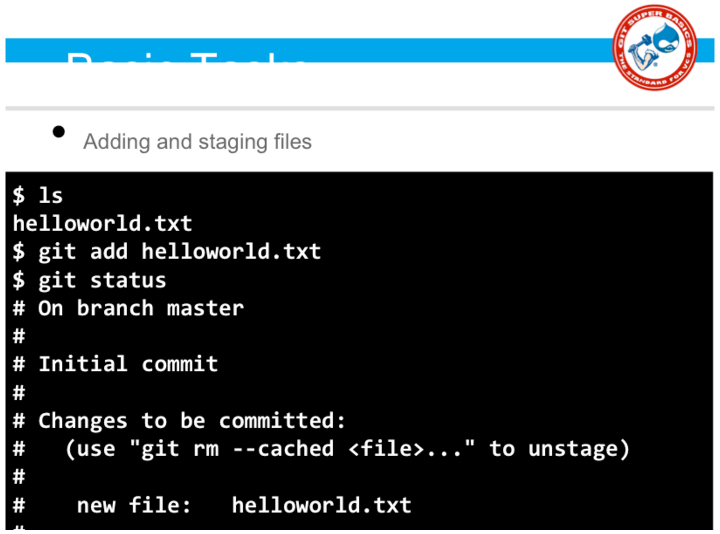
- Initialize a repository

```
$ mkdir test
$ cd test
$ git init
Initialized empty Git repository in /
Applications/MAMP/htdocs/test/.git/
```

Vid: To get started with git we need to learn about the following tasks. We are going to take you through a simple project that we have setup on GitHub.

Switch to Terminal

1. Make a new folder for the project and go into it.
2. Create a new repository $ git init
3. Add the HTML5 starter template file into the project folder.
4. Add a stylesheet file to the project folder.
5. $git status
   a. Discuss what the status screen is saying
6. $git add filename
7. $git status
   a. Show one file is added and another is in the directory but not added
8. $git add .
9. $git commit -m "Initial Commit"
   a. Explain that commit messages should be fairly short and descriptive
10. Add an extra file that we don't want to track like a .project file
    a. Show how to untrack the file
11. $git rm



Basic Tasks

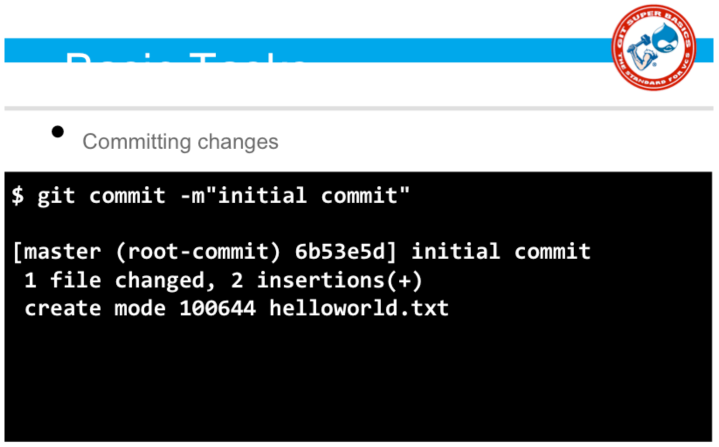
- Adding and staging files

```
$ ls
helloworld.txt
$ git add helloworld.txt
$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#    new file:   helloworld.txt
```

Vid: To get started with git we need to learn about the following tasks. We are going to take you through a simple project that we have setup on GitHub.

Switch to Terminal

1. Make a new folder for the project and go into it.
2. Create a new repository $ git init
3. Add the HTML5 starter template file into the project folder.
4. Add a stylesheet file to the project folder.
5. $git status
   a. Discuss what the status screen is saying
6. $git add filename
7. $git status
   a. Show one file is added and another is in the directory but not added
8. $git add .
9. $git commit -m "Initial Commit"
   a. Explain that commit messages should be fairly short and descriptive
10. Add an extra file that we don't want to track like a .project file
    a. Show how to untrack the file
11. $git rm



- Committing changes

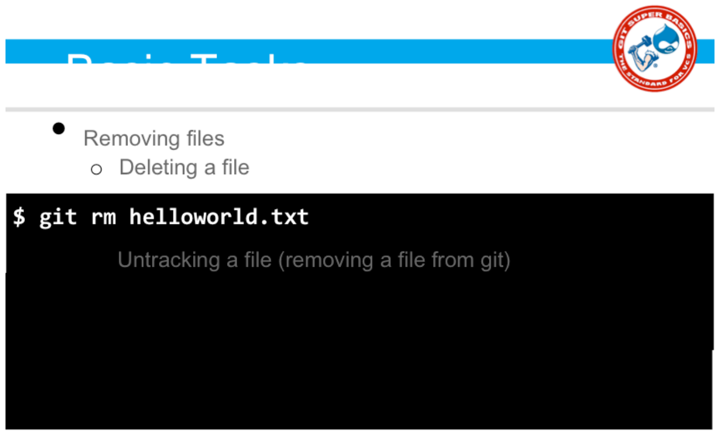
```
$ git commit -m"initial commit"

[master (root-commit) 6b53e5d] initial commit
 1 file changed, 2 insertions(+)
 create mode 100644 helloworld.txt
```

Vid: To get started with git we need to learn about the following tasks. We are going to take you through a simple project that we have setup on GitHub.

Switch to Terminal

1. Make a new folder for the project and go into it.
2. Create a new repository $ git init
3. Add the HTML5 starter template file into the project folder.
4. Add a stylesheet file to the project folder.
5. $git status
   a. Discuss what the status screen is saying
6. $git add filename
7. $git status
   a. Show one file is added and another is in the directory but not added
8. $git add .
9. $git commit -m "Initial Commit"
   a. Explain that commit messages should be fairly short and descriptive
10. Add an extra file that we don't want to track like a .project file
    a. Show how to untrack the file
11. $git rm



- Removing files
  - Deleting a file

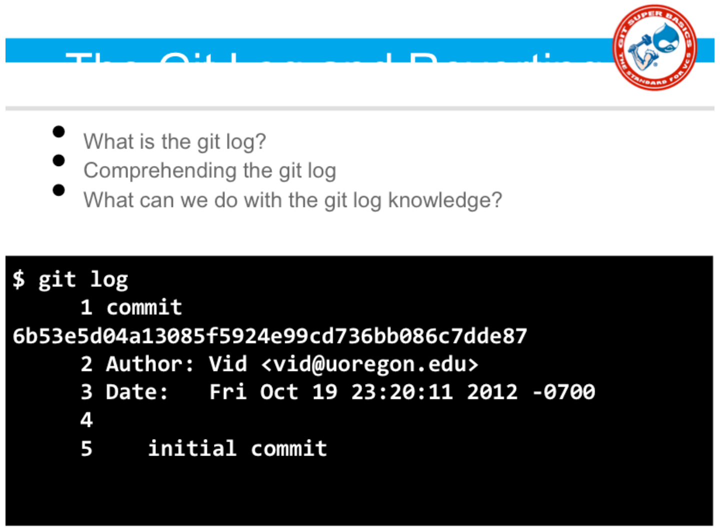
```
$ git rm helloworld.txt
```

Untracking a file (removing a file from git)

Vid: To get started with git we need to learn about the following tasks. We are going to take you through a simple project that we have setup on GitHub.

Switch to Terminal

1. Make a new folder for the project and go into it.
2. Create a new repository $ git init
3. Add the HTML5 starter template file into the project folder.
4. Add a stylesheet file to the project folder.
5. $git status
   a. Discuss what the status screen is saying
6. $git add filename
7. $git status
   a. Show one file is added and another is in the directory but not added
8. $git add .
9. $git commit -m "Initial Commit"
   a. Explain that commit messages should be fairly short and descriptive
10. Add an extra file that we don't want to track like a .project file
    a. Show how to untrack the file
11. $git rm



- What is the git log?
- Comprehending the git log
- What can we do with the git log knowledge?

```
$ git log
commit 6b53e5d04a13085f5924e99cd736bb086c7dde87
Author: Vid <vid@uoregon.edu>
Date:   Fri Oct 19 23:20:11 2012 -0700

    initial commit
```

Max: The git log allows you to look into what has been happening in the repository. Here you will be able to see the list of commits in the project and the commit message. The command is straightforward.

$git log

Now, what does all of this stuff mean. The first part of what you see is the Commit Hash. This is a unique identifier for the filesystem at the time the commit was made. You can move your file system to any commit if you know the hash.

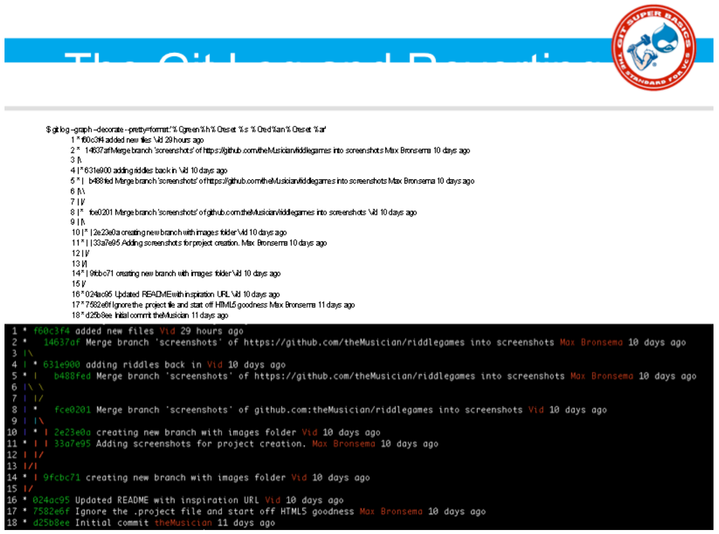
The second part is the author tag that is set by each user and their email address.

Third is the date down to the second for the commit. (Research more about the default date format)

The 4th part is the commit message itself.

Vid: We can use the git log information in the following ways.

1. Knowing the commit hash lets us revert back to that particular point in time.
2. We can see who made the commit if we are working jointly on a project.
3. We can see a diff of what changed with each commit. $ git log -p -3   (the -3 just limits the commits shown to 3)
4. Using -stat we can see stats, the number of insertions and deletes on each commit $ git log --stat



```
$ git log --graph --decorate --pretty=format:'%Cgreen%h%Creset %s %Cred%an%Creset %ar'
* f60c3f4 added new files Vid 29 hours ago
*   14637af Merge branch 'screenshots' of https://github.com/theMusician/riddlegames into screenshots Max Bronsema 10 days ago
|\
| * 631e900 adding riddles back in Vid 10 days ago
* |   b488fed Merge branch 'screenshots' of https://github.com/theMusician/riddlegames into screenshots Max Bronsema 10 days ago
|\ \
| |/
| *   fce0201 Merge branch 'screenshots' of github.com:theMusician/riddlegames into screenshots Vid 10 days ago
| |\
| * | 2e23e0a creating new branch with images folder Vid 10 days ago
* | | 33a7e95 Adding screenshots for project creation. Max Bronsema 10 days ago
| |/
|/|
* | 9fcbc71 creating new branch with images folder Vid 10 days ago
|/
* 024ac95 Updated README with inspiration URL Vid 10 days ago
* 7582e6f Ignore the .project file and start off HTML5 goodness Max Bronsema 11 days ago
* d25b8ee Initial commit theMusician 11 days ago
```

```
* f60c3f4 added new files Vid 29 hours ago
*   14637af Merge branch 'screenshots' of https://github.com/theMusician/riddlegames into screenshots Max Bronsema 10 days ago
|\
| * 631e900 adding riddles back in Vid 10 days ago
* |   b488fed Merge branch 'screenshots' of https://github.com/theMusician/riddlegames into screenshots Max Bronsema 10 days ago
|\ \
| |/
| *   fce0201 Merge branch 'screenshots' of github.com:theMusician/riddlegames into screenshots Vid 10 days ago
| |\
| * | 2e23e0a creating new branch with images folder Vid 10 days ago
* | | 33a7e95 Adding screenshots for project creation. Max Bronsema 10 days ago
| |/
|/|
* | 9fcbc71 creating new branch with images folder Vid 10 days ago
|/
* 024ac95 Updated README with inspiration URL Vid 10 days ago
* 7582e6f Ignore the .project file and start off HTML5 goodness Max Bronsema 10 days ago
* d25b8ee Initial commit theMusician 11 days ago
```

Max: The git log allows you to look into what has been happening in the repository. Here you will be able to see the list of commits in the project and the commit message. The command is straightforward.

$git log

Now, what does all of this stuff mean. The first part of what you see is the Commit Hash. This is a unique identifier for the filesystem at the time the commit was made. You can move your file system to any commit if you know the hash.

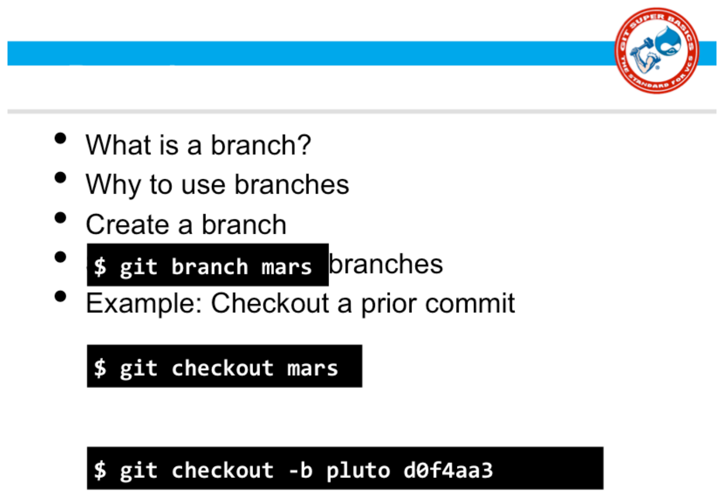
The second part is the author tag that is set by each user and their email address.

Third is the date down to the second for the commit. (Research more about the default date format)

The 4th part is the commit message itself.

Vid: We can use the git log information in the following ways.

1. Knowing the commit hash lets us revert back to that particular point in time.
2. We can see who made the commit if we are working jointly on a project.
3. We can see a diff of what changed with each commit. $ git log -p -3   (the -3 just limits the commits shown to 3)
4. Using -stat we can see stats, the number of insertions and deletes on each commit $ git log --stat



- What is a branch?
- Why to use branches
- Create a branch
- `$ git branch mars` branches
- Example: Checkout a prior commit

```
$ git checkout mars
```

```
$ git checkout -b pluto d0f4aa3
```

Note: When creating a local branch you'll want to set the upstream branch to track: 'git branch --set-upstream screenshots origin/screenshots'

Vid: A git branch takes a snapshot of your present code and then allows you to make changes without the code at the time you create the branch. When you are done working in a branch you can switch to another branch and if necessary, merge the changes together. In Drupal branches are often used when you are trying to create a patch for a module or core issue.

Max: To create a new branch, first make sure the branch you are currently on has no outstanding commits, or staged files.

```
        $ git status
```

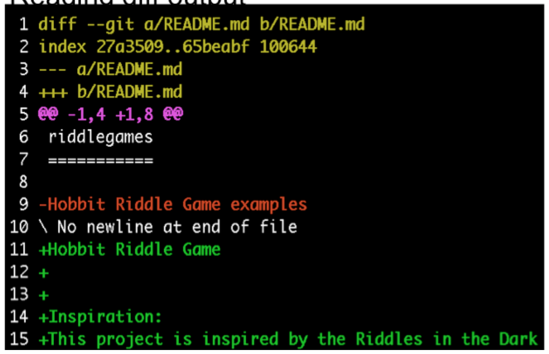
If there are changes, make sure to commit the changes before proceeding.

```
$ git commit . -am "Change description."
```

The shorthand for creating a branch and automatically switching to it is

```
$ git checkout -b branchName
```

- What is a diff?
- Reading diff output
-



```
diff --git a/README.md b/README.md
index 27a3509..65beabf 100644
--- a/README.md
+++ b/README.md
@@ -1,4 +1,8 @@
 riddlegames
 ===========
 
-Hobbit Riddle Game examples
\ No newline at end of file
+Hobbit Riddle Game
+
+
+Inspiration:
+This project is inspired by the Riddles in the Dark
```

Vid: A diff is a comparison between two files which shows the differences. Using git you can merge the changes between a file on different branches. You sometimes need to do this when moving your changes into a branch that has been updated.

$ git diff

with no options will show the differences between the files that are not yet staged.

$ git dff --staged

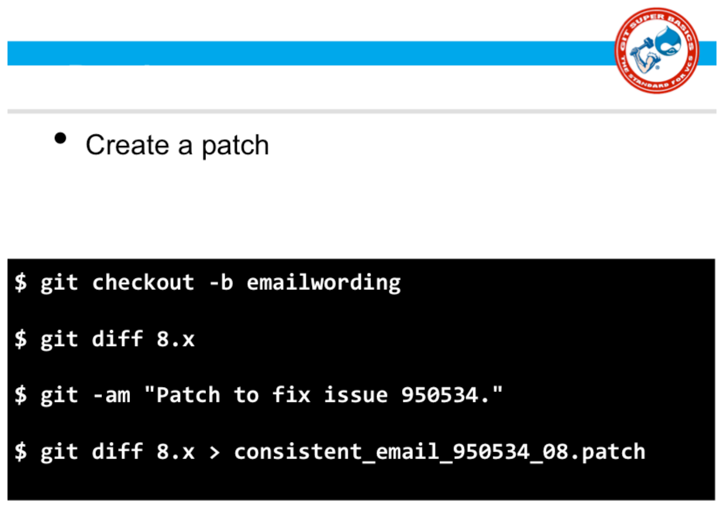
shows the diff of files that have been staged.

$git diff hash1 hash2

will show the difference between the two commits specified by the hash. --stat will give you the stats and not the diff.

The default diff tool can be a bit clunky. I prefer to use diffmerge. e.g. $ git difftool -t diffmerge file|hash|branch

To go back one commit after another, use this nice line created by Vid"



- Create a patch

```
$ git checkout -b emailwording

$ git diff 8.x

$ git -am "Patch to fix issue 950534."

$ git diff 8.x > consistent_email_950534_08.patch
```

Max: Now that you can create branches and generate diffs patches can make some sense. Patches in git simply identify files and then make changes to them. Patches are convenient because you can be working in a branch and then diff your branch against the production branch to create a patch. Now anyone else with the same production branch can apply the patch over a myriad of files. No need to make changes by hand.

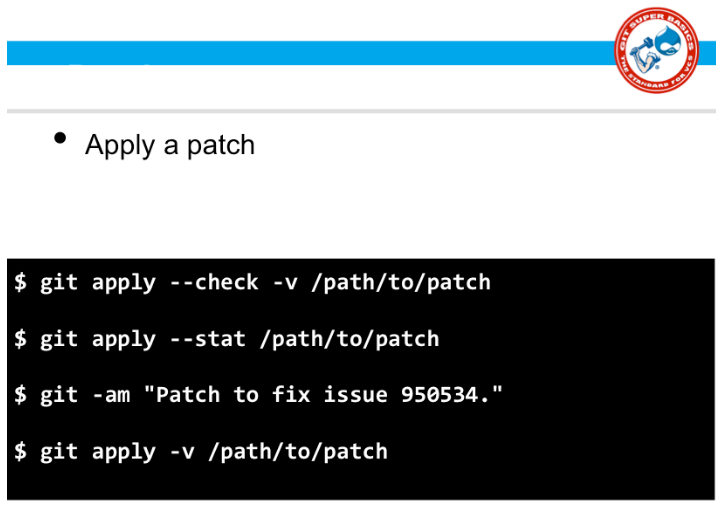
To create a patch (The Drupal Way http://drupal.org/node/707484):

(Make a patch from alternatestyle to styles)

1. Ensure you have the latest version of the code you wish to create a patch for.
2. Create a new branch with the following format. $ git branch [IssueNumber]-[IssueDescription]
    a. The issue number is the node number on drupal.org
3. Make your changes. When you are done making the changes use diff to see the changes that will go into the main code. $ git diff branchToDiffAgainst
4. If everything looks good. Stage and commit your changes. $ git -am "Patch to fix issue # 123456789."
5. git diff branchToDiffAgainst > [project_name]-[short_description]-[issue-number]-[comment-number].patch

Applying a patch:

1. Make sure you have committed any current changes, git status should be clean.



- Apply a patch

```
$ git apply --check -v /path/to/patch

$ git apply --stat /path/to/patch

$ git -am "Patch to fix issue 950534."

$ git apply -v /path/to/patch
```

Max: Now that you can create branches and generate diffs patches can make some sense. Patches in git simply identify files and then make changes to them. Patches are convenient because you can be working in a branch and then diff your branch against the production branch to create a patch. Now anyone else with the same production branch can apply the patch over a myriad of files. No need to make changes by hand.

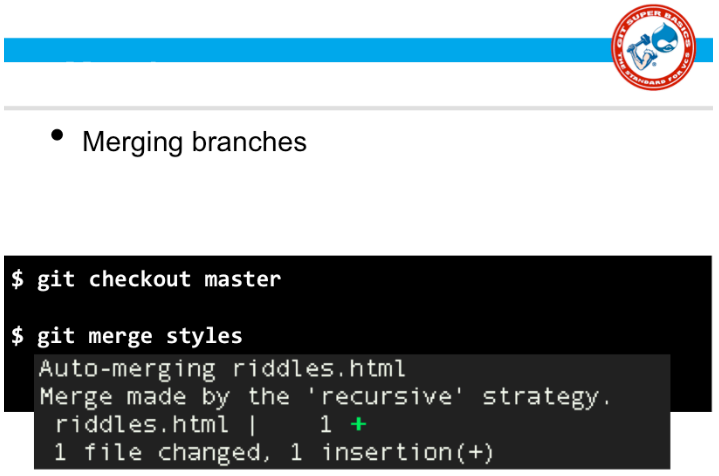
To create a patch (The Drupal Way http://drupal.org/node/707484):

(Make a patch from alternatestyle to styles)

1. Ensure you have the latest version of the code you wish to create a patch for.
2. Create a new branch with the following format. $ git branch [IssueNumber]-[IssueDescription]
   a. The issue number is the node number on drupal.org
3. Make your changes. When you are done making the changes use diff to see the changes that will go into the main code. $ git diff branchToDiffAgainst
4. If everything looks good. Stage and commit your changes. $ git -am "Patch to fix issue # 123456789."
5. git diff branchToDiffAgainst > [project_name]-[short_description]-[issue-number]-[comment-number].patch

Applying a patch:

1. Make sure you have committed any current changes, git status should be clean.



- Merging branches

```
$ git checkout master

$ git merge styles
 Auto-merging riddles.html
 Merge made by the 'recursive' strategy.
  riddles.html |     1 +
  1 file changed, 1 insertion(+)
```

Vid: Merging two branches means simply taking the changes from one branch and joining them with the other branch. Often a merge will be necessary after you created a branch to fix an issue and you want to share that fix with the production branch.

Let's say that the styles branch is deemed to be the one we want to launch with. To merge those changes we use the merge command on the branch we want to merge with. For this example lets checkout the master branch.

$ git checkout master

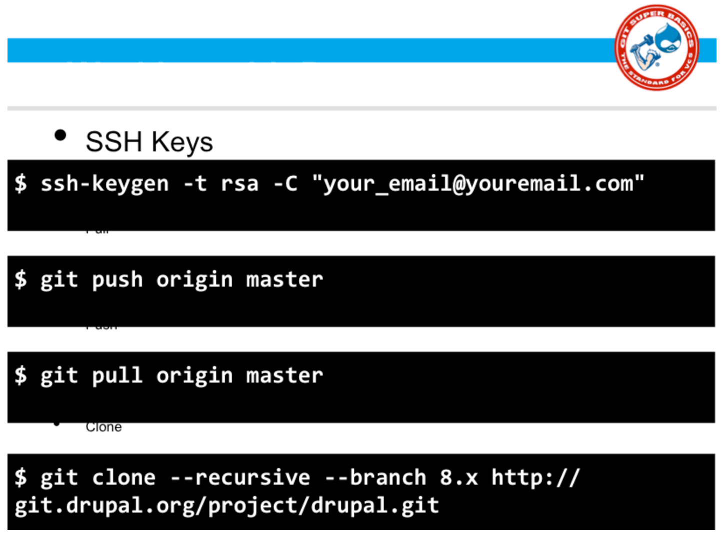
Then we identify the branch we want to merge and use the merge command.

$ git merge styles

You will then often see the words Fast Forward or "Merge made by recursive" and the stat of what files changed. In this case riddles.html and styles.css

Basic Git Merge Conflicts:

Next Slide



- SSH Keys

```
$ ssh-keygen -t rsa -C "your_email@youremail.com"
```

```
$ git push origin master
```

```
$ git pull origin master
```

- Clone

```
$ git clone --recursive --branch 8.x http://
git.drupal.org/project/drupal.git
```

Max: The great thing about git is that you can work with others and share code from a central resource. This is what github, bitbucket, and firewalled solutions like gitolite and stash are all about. To do this effectively we need to talk with those resources and push and pull code from it.

SSH Keys are needed with many systems, though some HTTPS authentication systems are gaining popularity. For this session we will use SSH keys. Generating a key is pretty straightforward and in doing so you create a private key which you keep on your system and a public key that you can share with others. If you have keys already, make sure you know the passphrase and give us a few moments to generate them for others.

From your prompt quickly check that no key exists.

```
$ cd ~/.ssh
$ ls
```

If something appears it means you have keys. If you were not aware of that, make a backup quickly in case they are important. (These steps are from gitHub's great write-up)

```
mkdir key_backup
cp id_rsa* key_backup
rm id_rsa*
```

Now we are going to generate a key.

- Thank You
- Questions?
- Links:
  2012.pnwdrupalsummit.org/sessions/git-super-basics

Max Bronsema - http://maxbronsema.com
@theMusician

Vid Rowan - http://uoregon.edu/~vid