Recap: Developer's Environment

There's no audio yet.

We'll start at 2PM (Eastern Time).
Standardizing Your Test Environment

• VirtualBox -> machine inside a machine “the oracle thing”
• Chef -> configuration scripts for how the machine inside the machine is setup
• Vagrant -> wrapper for Chef + VirtualBox
• Putty (or SSH) -> to log into Baby Ubuntu
VirtualBox Without Vagrant
VirtualBox Without Vagrant (windows)
Administering Machines with VirtualBox
Shared Folders

On the Display page, you have assigned less than 9 MB of video memory which is the minimum amount required to switch the virtual machine to fullscreen or seamless mode.
Logging Into Baby Ubuntu

- Windows -> putty
- OSX -> command line

```
$ vagrant ssh
Username: vagrant
Password: vagrant
```
Inside Baby Ubuntu

login as: vagrant
vagrant@127.0.0.1's password:
Linux lucid32 2.6.32-38-generic #83-Ubuntu SMP Wed Jan 4 11:13:04 UTC 2012 i686
GNU/Linux
Ubuntu 10.04.4 LTS

Welcome to Ubuntu!
* Documentation: https://help.ubuntu.com/
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Tue Nov 20 18:46:11 2012 from 10.0.2.2
vagrant@lucid32:~$
Getting the Right Start Point

- Jeff Eaton
  https://github.com/eaton/vagrant-chef-dlamp

- Vagrant Project
  http://drupal.org/project/vagrant/git-instructions

- Mark Sonnabaum
  https://github.com/msonnabaum/drush-ci-chef

- Patrick Connolly
  https://github.com/myplanetdigital/vagrant-ariadne/
Version Control Basics
Benefits of Version Control

- Backup and restore
- Syncronization across multiple systems
- Short-term undo to test implications
- Long-term undo to reverse bugs
- Track changes to see why/how software evolved
- Track ownership to give ‘credit’ to change makers
- Sandboxing our code to test changes without affecting others
There is no excuse for not having version control.

The cheapest way to get version control is to use an automated backup system, like Dropbox, for your code.
Terminology

• **Repository.** The database of changes to your files.

• **Server.** The computer storing the repository.

• **Client.** The computer connecting to the repository.

• **Working copy.** Your local copy, where changes are made.

• **Trunk** (or “main”). The current, primary source for unchanged code.

• **Head.** The latest revision in the repository.
Basic Actions

- **Add.** Put a file into the repo.
- **Revision.** Checks what version a file is on.
- **Check out.** Download files from the main repository.
- **Check in.** Upload changed files to the main repository.
- **Changelog.** A list of changes made to a file since it was created.
- **Update/sync.** Synchronize your files with the ones from the main repository.
- **Revert.** Throw away your local changes and reload the latest version from the repository.
Workflow: Centralized
no local commits
Workflow: The Solo Developer

Create project → Add files → Do your work → Upload files

Do more work
Installing Git

- Command line
- Desktop integration
- Graphical interface

http://msysgit.googlecode.com/
Git GUIs

- SourceTree, OSX

- SmartGit, cross-platform
My First Version-Controlled Project

There are three steps needed to version your work:

1. identify a folder as a git repository.
   ```
cd <my_project_folder>
git init
   ```

2. notify git of new files you would like it to monitor.
   ```
git add <filename>
   ```

3. commit your changes to the repository.
   ```
git commit -m “your message about the changes”
   ```
The diagram shows the basic check-ins for the work definition. The main trunk is labeled with the following items:

- r1: Milk
- r2: Milk and Eggs
- r3: Milk, Eggs, and Juice
- r4: Milk, Eggs, and Soup
Diffs Show the Difference Between Two Versions of a Project
define:work
Use Tags to Identify Milestones

James’s Final Grocery List
Emma’s Additions
Groceries Purchased

Main Trunk

Milk
Eggs
Soup

Milk
Eggs
Soup
Bread

Milk
Eggs
Soup
Bread
M&Ms
define:work
Use Branches to Identify Deviations

Main Trunk

Candy Store

Milk
Eggs
Soup
Bread
M&Ms

r4

Milk
Eggs
Soup

r7

Bread
Workflow: Read-only Projects

“Fork Me” on GitHub

My Fork
(branch)

Improved version of project

Random Project on the Internet
Workflow: Partner

1. James starts a grocery list

2. Emma already had a grocery list started. She asks to see James’s list.

3. James remembers a couple more items.

4. Emma adds a few things from her list.

5. The grocery lists are combined and James goes shopping.
Collaborative Actions

- Branch. Create a separate copy of a repository for personal use.
- Diff/change/delta. Identifies the differences between two versions of a file.
- Merge/patch. Apply the changes from one version of a file, to another.
- Conflict. When two versions of a file have proposed changes at the same place.
- Resolve. Deciding which version of conflicting changes should be applied, and which should be discarded.
Merging

Milk
Eggs
Soup

+ M&Ms

Milk
Eggs
Soup
M&Ms

Main Trunk (James’s List)

Milk
Eggs
Soup
+ Bread

Milk
Eggs
Soup
Bread

Milk
Eggs
Soup
M&Ms

Milk
Eggs
Soup
Bread
M&Ms
Sample Project

http://betterexplained.com/articles/a-visual-guide-to-version-control/
Workflow:
Decentralized with a shared mainline
Homework

• Create a new repository for one of your own projects.

• In your notebook of problems that you started yesterday, create a list of “similar” clients that might benefit from having a single repository.

• Explore the four vagrant scripts discussed in class.