Make it Cooler

With Decentralized Version Control

Indy Nagpal

Straker Software

CF.Objective()

A bit about me

- CTO, Straker Software, New Zealand
- Lots of CF and Flex experience
- In love with Groovy
- Tackling challenges of creating cloud-based software
- nagpals.com/blog

Once upon a time,
in a galaxy not too far away...

There was CVS

And it was a little... well... painful

Subversion made it a little easier

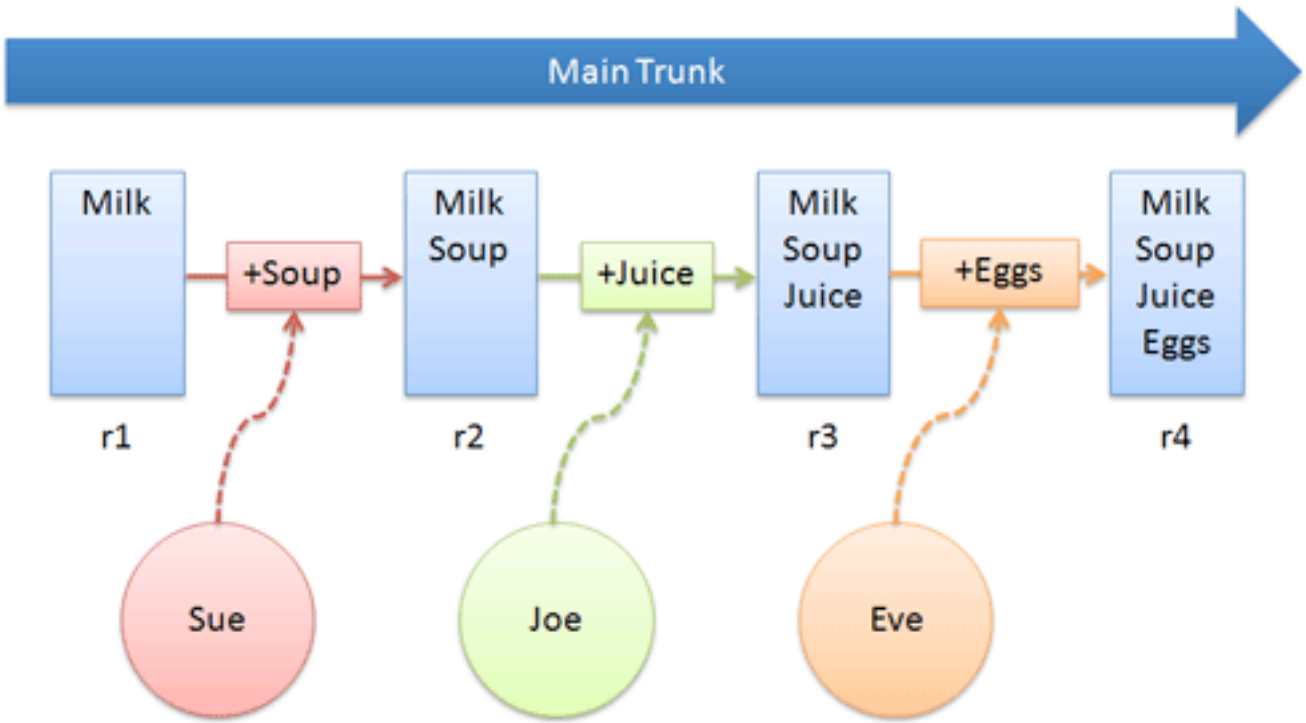
Both are centralized
version control systems

Central Server

Network in between

Clients

Centralized VCS



http://tinyurl.com/dcvs-explained

A little history

- Emerged in 1970s
  - Yes, that's about 40 years back
- That was the time of
  - weak clients
  - grunty mainframes
- Centralized teams

Centralized Version Control

- Simpler
- Works nicely for
  - backup
  - undo
  - synchronization
- "Everyone in my team knows how to use it"

So what's the problem?

- Reliance on being connected
- Single point of failure
- Branching and merging -- a big pain
- Growing projects
- Diffused teams

Enter...

Decentralized Version Control Systems

Not too long a ago
on a planet called Earth
Lived a nerd called Linus Torvalds

And then there was Linux
and Git

Other DVCS Systems

Mercurial

Bazaar

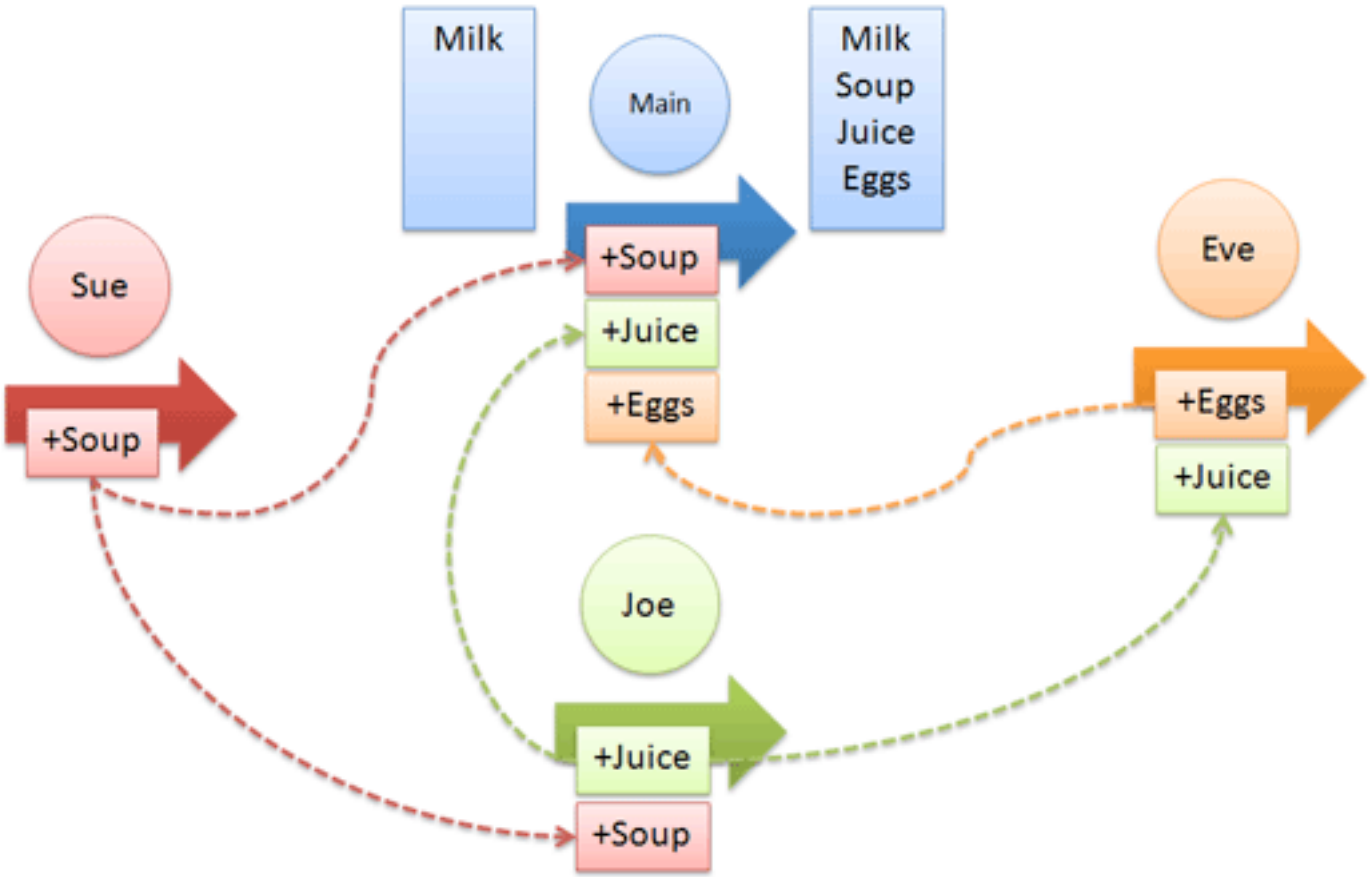
Main repository

Multiple clients clone the repository, including all history

Commit changes locally

Push to main repository

Distributed VCS



http://tinyurl.com/dcvs-explained

Decentralized Version Control

- Relatively recent innovation
- Focus on sharing changes
- Recording changes and Applying changes are separated
- Fast
- Efficient

Decentralized Version Control

- Works offline
- Branching is cheap
- Merging a pleasure
- Support for non-linear development
- Distributed development

Considerations in switching from Centralized to Decentralized

IDE Support

- git-gui, egit (Eclipse), gitx (Mac)
- Command-line works very well
- Configurable to work with other diff-ing/ merging and editing utilities
- Nothing as simple as TortoiseSVN

Repository Hosting Infrastructure

- Third party hosting works well
  - Unfuddle, Assembla
  - repo.or.cz, github
- Host own server
- *nix friendly, but have Windows ports

Learning Curve

- There is one! And it is a little steep
- Getting everyone to change development practice is a biggie
- Need a champion who doubles up as trainer
- If business finds reason, development teams have limited choice

Existing Repos

- Size of repositories
- Most DVCS can import SVN repos, including history
- Process of migration needs to be planned and implemented

Integration with Third-party Tools

- Code repos are tied to other systems:
  - Bug Tracking
  - Continuous Integration Systems
  - Code Review Systems
- Switching requires updates to those processes
- Do-able

Eternal Truth #1

The longer you use a system,
the harder it is to switch

Eternal Truth #2

Inertia does not help

Eternal Truth #3

Lack of a business case does not help either

Eternal Truth #4

Change is the only constant

So where does that leave us?

Choose Git

- Seems to have more legs than others
- A little more difficult to grasp and work with
- But once you've got your head around it, and worked with it for a while, it is easy

git-svn

- Very useful feature
- Checkout a SVN repo as a Git repo
- Work with locally as a Git repo
- Branch/merge locally as much as you like
- Commit locally, and then push to SVN repo

A few other useful things

- Only one .git folder, not a zillion .svn folders
- Create local repos for things you normally wouldn't
- Efficiency
  - 1.5 GB svn repo = 700 MB git repo
- Speed
  - Of local commits and pull/push from main repo

Resources

- Git
  - git-scm.com
- Peepcode
  - peepcode.com/products/git
- Pragmatic Programmers
  - pragprog.com/titles/tsgit/

One Last Truth

Automation is the key

On Automation and Developers

Given a choice between spending an hour doing a task manually, or spending three hours writing a program to do it automatically... a geek will write the program, every single time. And, if not given the choice, if explicitly ordered to do the job manually, we'll disobey and write the program anyway.

Catherine Devlin (http://tinyurl.com/devautomation)

Hook DVCS up with Testing and CI

Unit Testing

- MXUnit works well
- mxunit.org/

- Writing testable code -- key to automation

- Retro-fitting tests does not work

- Have to write tests *before* writing code

- Makes the code more robust

What is testable code

- Marc Esher's presentation on Testable Code:
  - Extracted
  - Abstracted
  - Injected

Git with Unit Testing

- Agnostic
- Key issues are:
  - Where do you store tests
  - How do you run them

Continuous Integration

- Hudson works extremely well
- https://hudson.dev.java.net/

Setting up CI Workflow

- Set up project
- Ping Git repo / Implement callback
- Checkout if new changes
- Call MXUnit Unit Ant Task to run tests
- Publish report
- Notify

Thank you!

indy@strakersoftware.com
strakersoftware.com
nagpals.com/blog