A Tour of Git, with Java and Eclipse

Kyle Cordes
Oasis Digital Solutions Inc.
kyle@kylecordes.com

St. Louis Java SIG
Oct. 9, 2008
http://kylecordes.com

Introduction and Agenda

Linus Torvalds wanted a replacement for BitKeeper. He wrote the first version of Git in a few weeks. A few years later, Git is now the leading contender among distributed source control tools: http://git.or.cz/

This will not be a typical "introduction" talk; there are ample introductions to Git on the WWW. Instead, I'll wander through some source control use cases, pointing out Git features, strengths, and weaknesses along the way.

I use Git on both Linux and Windows, and I use both the command line and GUI tools. For this presentation, I will show how it works on Windows, and mostly use the included GUI tools.

We'll end a bit early, to leave time for questions and ah doc demos.

Why Git?

- Git promises to keep your files exactly as they are, and uses hashes to make it happen.
- Git is very fast; speed matters a lot more than you might think.
- Git is distributed and local, it works very well offline as well as online.
- Git handles the realities of branching and merging.
- Git has an enormous set of very useful features.
- Git is clearly a tool made by people who love great tools.

Get Git

Deb/Ubuntu: apt-get install git-core
Windows: http://code.google.com/p/msysgit/

Version Some Files, Look at Them

git init	git add	git-gui
git commit	.gitignore	gitk

Use Git with Eclipse

There is a git-Eclipse project in development:

http://repo.or.cz/w/egit.git

However, I will show how to use Git and Eclipse without any integration. This turns out to be much less problematic than you might initially guess, and has advantages:

- If you use git with multiple IDEs, you only need to learn one interface.
- You can use the tool in ways that the IDE does not anticipate, such as by versioning a whole workspace as a unit.

Branch and Merge

git branch	git checkout -b newbranch
git diff	git merge

Rebase

1. The **best** thing in the world, a **great** capability.
2. The **world** thing in the world, **never** do this.

My advice: Have it in your toolbox, but be careful.

Review and Search History

The sample repos from tonight's demo don't have much history to show; but with git, and without relying on a network connection, we can see git's review and search capabilities on projects "in the wild".

Rearranging Files

Git tracks content, and will follow that content as it moves around, without you telling it about the moves. This is extremely helpful if you have a pile of files to rearrange.

http://kylecordes.com/2008/07/18/rearrange-file-svn-git/

Blame!

Poll: **Do you use Blame**? How often? Does it help?

Blaming is a terrible way to interact with other people… but get over the name, because "blame" is a **fantastic** source control feature.

Host Your Repo on Your Server

Git has a built in ability to host repos, and also includes gitweb.

Gitosis is also a great tool: http://eagain.net/gitweb/?p=gitosis.git

GitHub

Don't want to mess with hosting? http://github.com/

Enterpriseynessility

Git does not have all the same feature as whatever enterprisey tool you may be using. It has a different set of features. YMMV, but in my opinion it is a "disruptive innovation".

Git-SVN

Git makes a great front end for SVN; I use it this way regularly on Linux. Msysgit does not currently include git-svn, so no demo.

Tips

Initial Setup:

```
git config --global color.diff auto
git config --global color.status auto
git config --global color.branch auto
export PS1="\u@\h \W\$(git-branch 2> /dev/null | grep -e '\* ' | sed 's/^..\(.*\)/ {\1}/')\$ "
```

Remove from source control, the files you have removed from the file system:

```
git ls-files -z --deleted | git update-index -z --remove --stdin
```

Using Git in a Closed Team

There are many ways to use Git; you can think of it as mostly a superset of what other systems do. You can use Git to achieve many possible workflows. Here is one way to use it effectively:

- Set up a server everyone can pull from and push to.
- Use Gitosis or similar.
- Have one(+) repo per developer, and one(+) for leads to all push to.
- Everyone pushes to their repo.
- Leads pull from others' repos, and push to the main repo.
- Server performance is not important, a slow machine can serve many users.
- Backup is not all that important, you will not actually lose anything if the machine dies.
- Make your initial repo fork for each person, on the server, for speed and space sharing.

Using Git for Open Source

- Clone the project's repo.
- Hack.
- Push your changes to your public repo (perhaps on Github)
- Get someone to pull them.
- git format-patch, if you want to send them.

Problems in the Git World

On Linux, things are going very well. But on Windows, not so well: the Windows port itself is advancing nicely, but it appears to suffer somewhat badly from the "I am just doing this for fun, if you want it fix, make a patch" problem.

My opinion: to really thrive on Windows, Git needs a firm to create a commercial offering around it, and feed lots of core fixes back to the project.

Questions?