Git Workshop

Introduce Git

1. Who made it
2. Why did they make it
3. Discuss differences between cvc and dcvc systems

A Short History of Git

As with many great things in life, Git began with a bit of creative destruction and fiery controversy.

The Linux kernel is an open source software project of fairly large scope. For most of the lifetime of the Linux kernel maintenance (1991–2002), changes to the software were passed around as patches and archived files. In 2002, the Linux kernel project began using a proprietary DVCS called BitKeeper.

In 2005, the relationship between the community that developed the Linux kernel and the commercial company that developed BitKeeper broke down, and the tool's free-of-charge status was revoked. This prompted the Linux development community (and in particular Linus Torvalds, the creator of Linux) to develop their own tool based on some of the lessons they learned while using BitKeeper. Some of the goals of the new system were as follows:

- Speed
- Simple design
- Strong support for non-linear development (thousands of parallel branches)
- Fully distributed
- Able to handle large projects like the Linux kernel efficiently (speed and data size)

Since its birth in 2005, Git has evolved and matured to be easy to use and yet retain these initial qualities. It's amazingly fast, it's very efficient with large projects, and it has an incredible branching system for non-linear development (See Git Branching).

Install Git

1. EULA
2. Select Options
3. Choose your editor (not the default)
4. Choose how to run Git
5. Choose your ssh exe
6. Choose SSL lib for https
7. Line endings
8. Terminal choice
9. Other options
10. Finally, the install!

Create New Repo

1. Git Init
2. Edit a file
3. Git add
4. Git commit
5. Talk a little about remote repositories

Init

Checkout

Add

Status

Fetch

Merge

Pull (The combo of Fetch then Merge)

Commit

Push

Create a project in Visual Studio

Open Team Explorer

Create a new SSIS project

Connect to Visual Studio and push

Delete project locally and pull

Branching in Merging

Branches are cheap

Branches are cool

Merge and the merge strategies. Fast-forward vs 3-way

Reset

Discuss the file system

Stage (Index)

Commits

Reset --soft Only move pointer

Reset --mixed Will also clear index back

Reset --hard Most destructive, can loose data because it will effect working dir

Can be used with specific dirs or files

Other Useful Commands

Stash - Do it before merging

Clean – remove junk

Commit --amend

Rebase – makes it look as though it were never branched

GitHub for Forking

Have people sign up for Git Hub and fork my project.

Make some changes

Create a pull request back to me

Approve some, reject others.