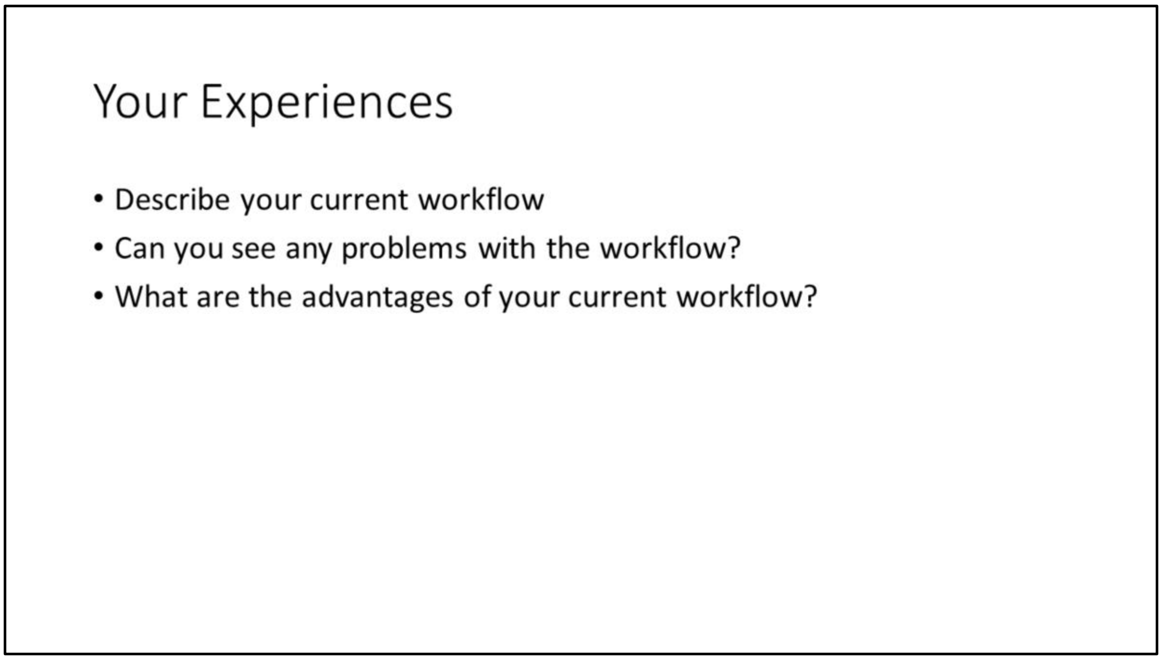
git SCM

What it is and why should we care?

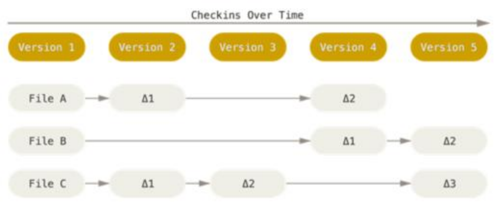
Jack Concanon



Your Experiences

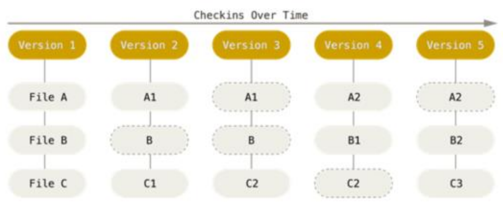
- Describe your current workflow
- Can you see any problems with the workflow?
- What are the advantages of your current workflow?

Git Fundamentals



SVN, Perforce etc.

Stores file deltas from a base version

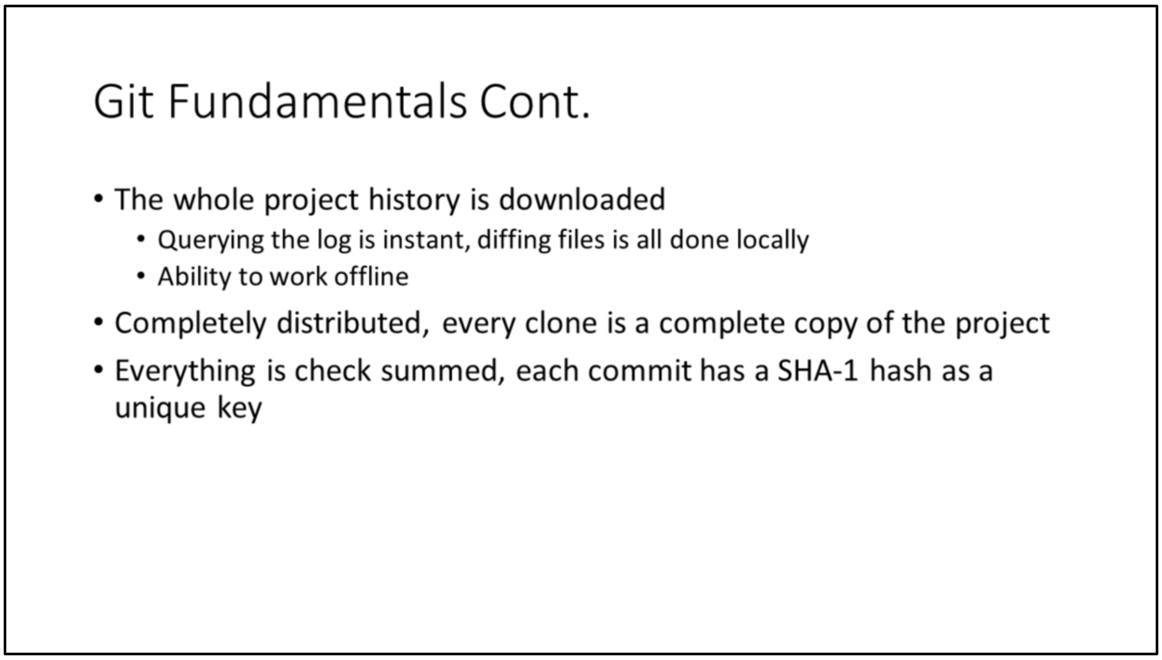


Git

Stores information as a snapshot – becomes a stream of snapshots

SVN, perforce etc. store changes as a series of deltas from the beginning of the repo, git uses snapshots of every file at the point of commit. If no files have changed then the previous pointer for that file is used otherwise a new pointer is made and updated.

This means that in git all operations become pointer operations, for example checking out a specific version of a file is just grabbing the snapshot pointed to by the pointer. Creating a branch is creating a new snapshot based on a pointer (writing 40 bytes to a file)

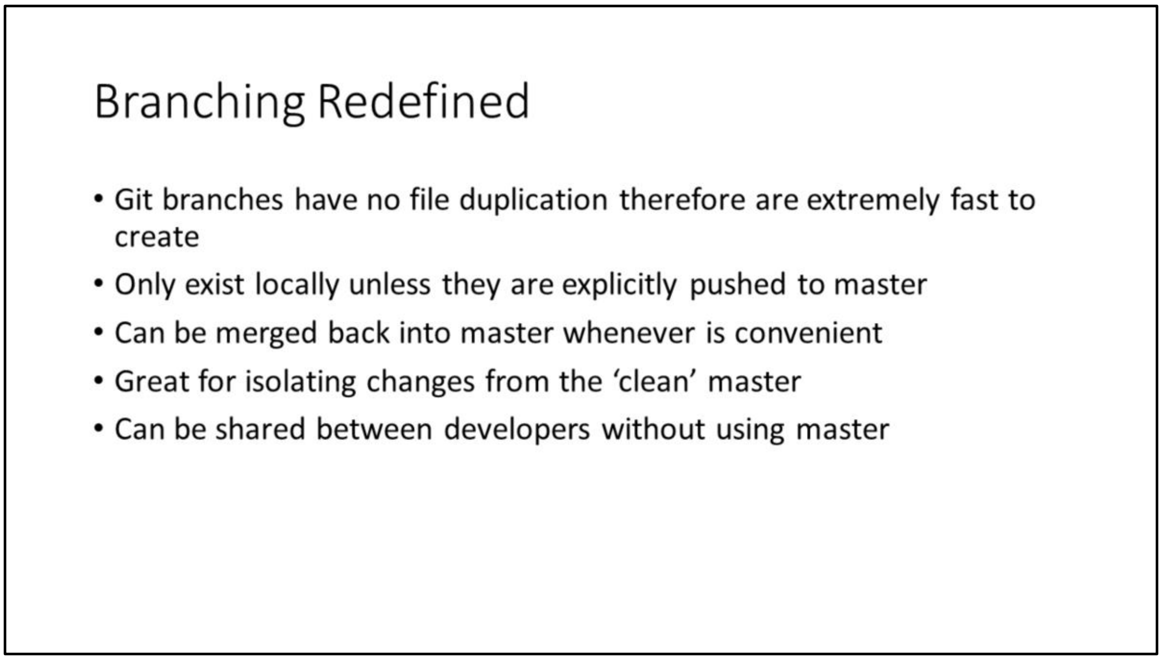


Git Fundamentals Cont.

- The whole project history is downloaded
  - Querying the log is instant, diffing files is all done locally
  - Ability to work offline
- Completely distributed, every clone is a complete copy of the project
- Everything is check summed, each commit has a SHA-1 hash as a unique key

A git project is entirely self contained, no communication with a central server is required at all. This means that operations such as diff against any point in time can be done offline/locally with a huge speed advantage.

Every clone of a git repository is a backup of the master as long as it is kept up to date.


Branching Redefined

- Git branches have no file duplication therefore are extremely fast to create
- Only exist locally unless they are explicitly pushed to master
- Can be merged back into master whenever is convenient
- Great for isolating changes from the 'clean' master
- Can be shared between developers without using master

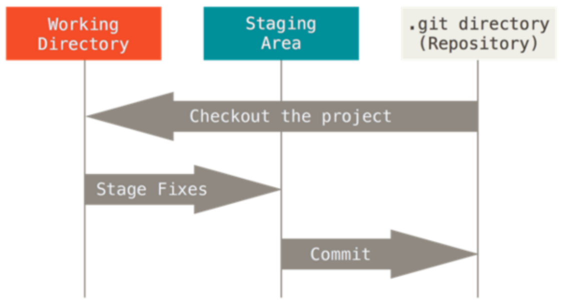
One of the biggest advantages over other version control systems is the ability to cheaply create branches, this encourages new workflows that move away from a rigid central repository workflow.

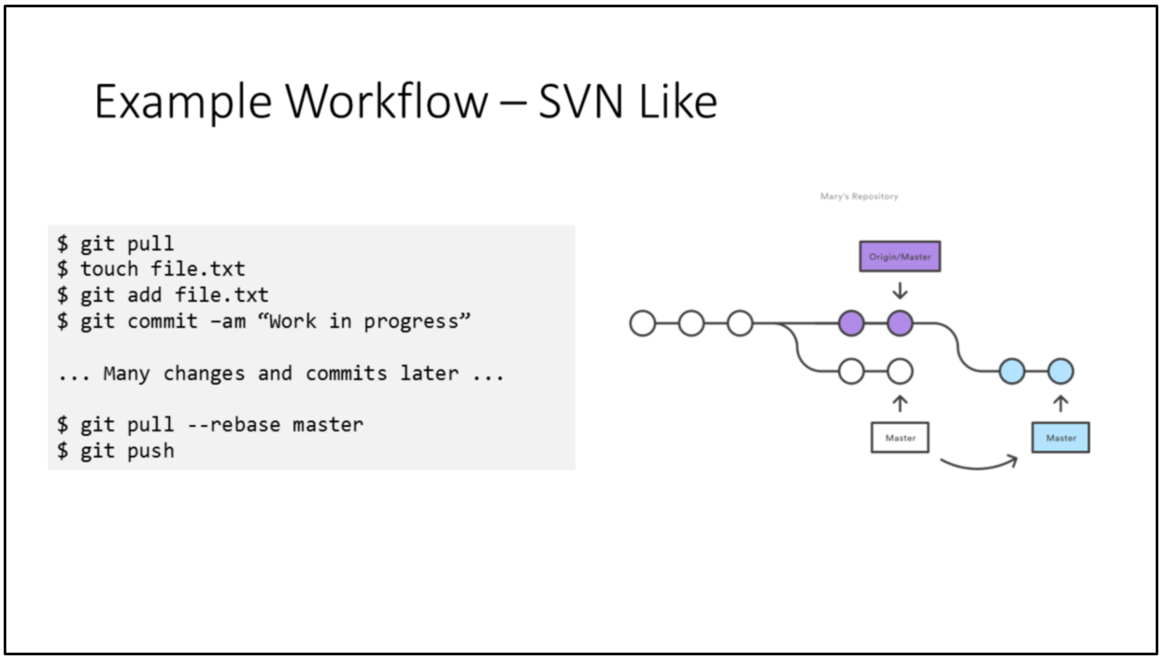
Branches can be merged with other branches relatively easily, even if the other branch has changed and modified existing files. Merging in git is usually automatic and painless (not always though).

Using branches as features

Three Main States

- Project is cloned from a master repository to a working directory
- Changes are made to working directory, these become modified files and move to staging area
- A commit is made which pushes changes to the master repository

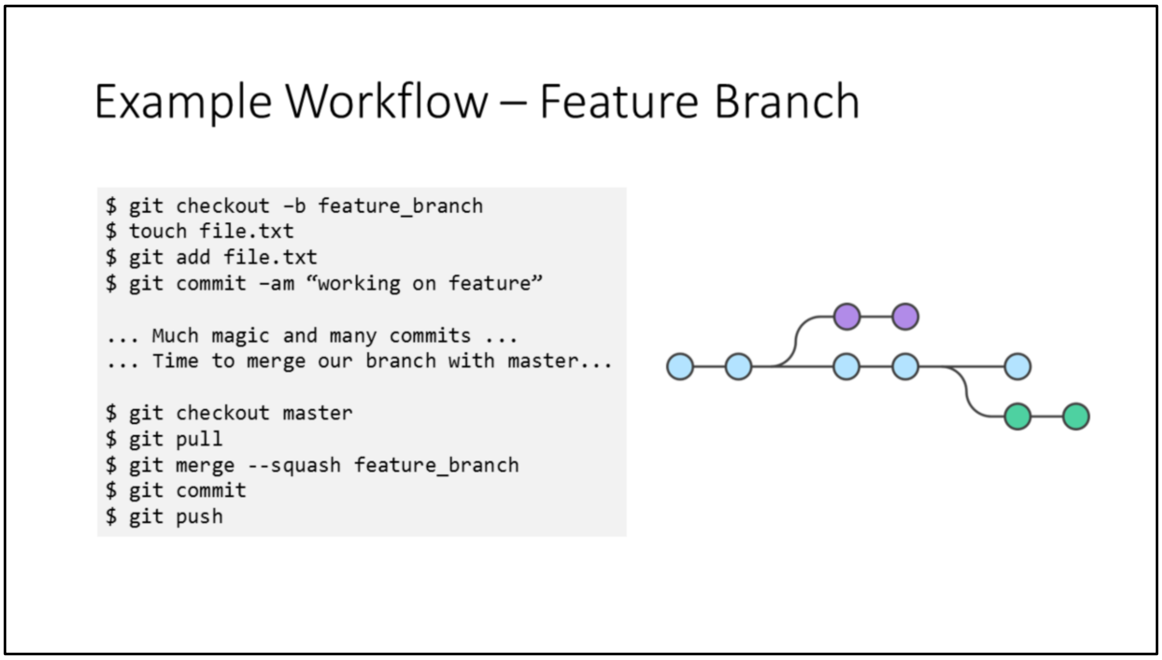




Example Workflow – SVN Like

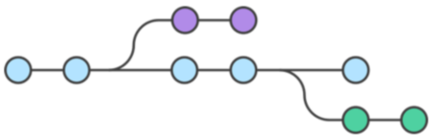
```
$ git pull
$ touch file.txt
$ git add file.txt
$ git commit -am "Work in progress"

... Many changes and commits later ...

$ git pull --rebase master
$ git push
```

Git pull –rebase master will avoid having to do a separate merge commit. Always a good idea to do a pull before pushing otherwise you will likely get merge conflicts.



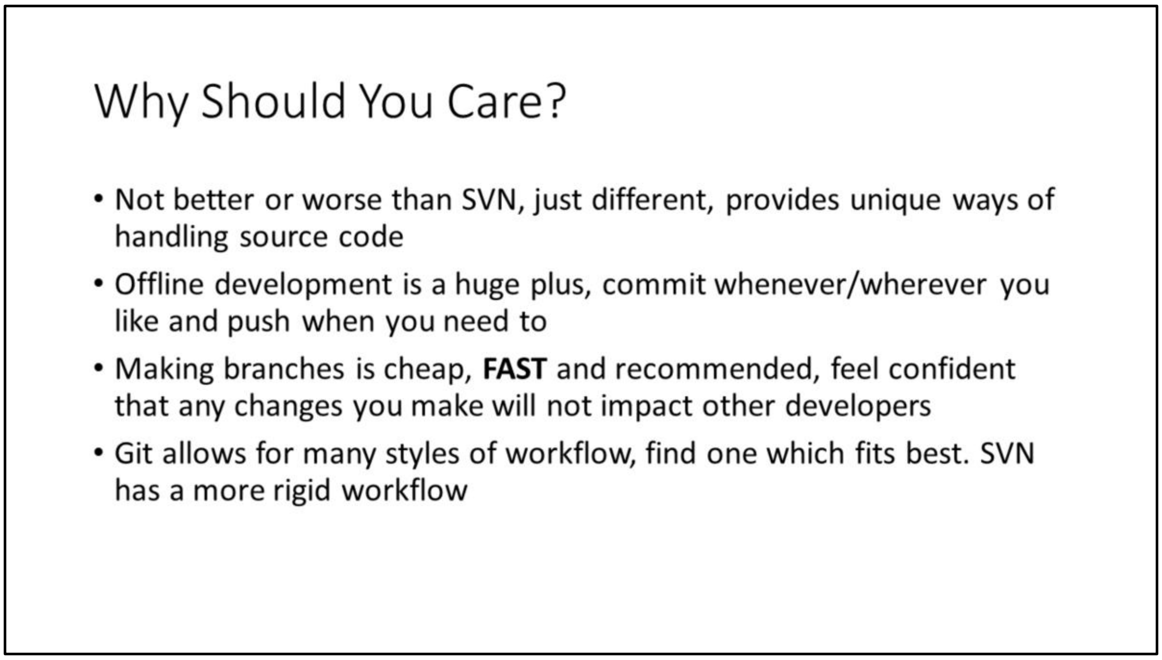
Example Workflow – Feature Branch

```
$ git checkout –b feature_branch
$ touch file.txt
$ git add file.txt
$ git commit –am “working on feature”

... Much magic and many commits ...
... Time to merge our branch with master...

$ git checkout master
$ git pull
$ git merge --squash feature_branch
$ git commit
$ git push
```



Use branches whenever you want to work on new features, this gives you versioned code without having to pollute master. Master should represent a working snapshot of code. Allows for sharing of feature branches.

Can implement some sort of code review before anything is merged back into master by using pull requests.



Why Should You Care?

- Not better or worse than SVN, just different, provides unique ways of handling source code
- Offline development is a huge plus, commit whenever/wherever you like and push when you need to
- Making branches is cheap, **FAST** and recommended, feel confident that any changes you make will not impact other developers
- Git allows for many styles of workflow, find one which fits best. SVN has a more rigid workflow

Great Links

- https://www.atlassian.com/git/tutorials/what-is-git
- https://try.github.io/levels/1/challenges/1 (Interactive online git training)
- www.github.com – online hosting of Git repositories
- www.bitbucket.com – similar to GitHub