Source Control 101

with GitHub

Rebecca Hill
Luke Ryan

Beautiful accounting software

rites
sktop
wnloads
cent Places

ries
cuments
usic
tures
leos

puter

work

Documents library

Project 123

Arrange by: Folder


Version 1


Version 1 - Copy


Version 1 - from flashdrive


Version 1.1


Version 1.1 but with the changes Barry did


Version 1.1 test that Idea that Bob had


Version 1.2


Version 2.0


Version Final hand it in

Imagine...

Imagine you worked on a team with a product that had hundreds of thousands of lines of code over thousands of files with hundreds of developers all around the world?

How would you work together as a team? How would you share files?

Why source control?

1. Teamwork

- “I’m working on file X, nobody else touch it”
- Multiple people can work on the same files, then you can merge the changes together at integration time.
- Full history of who changed what, when. No lost code.

Why source control?

2. Backups

Repository of code - one source of truth, backed up.

“I lost my flashdrive...”

“I deleted the shared directory…”

“The production server died…”

Why source control?

3. Change Management

- History of revisions over time - who changed what and when
- When was a defect introduced?
- What code is running on environment x?
- The only way to maintain changes to multiple different versions at the same time
- No need to comment out code!

Why source control?

4. Rapid Innovation

With your source code backed up, you can do crazy things.

“What happens if I delete all of these dependencies then re-write this class…”

Why source control?

5. Automate all the things

Once you have a centralised repository of your code you can automate builds, testing, deployment from it. Devops.

Everything is triggered by code changes.

There are lots of source control systems

- It’s been around for ages
- You may have heard of
  - Git
  - Subversion
  - Mercurial
  - Team Foundation Server

How does it work

- The source of truth for your code is held in a *repository*
- You get this code
- You change it
- You *commit* and send your stuff back to the server

Git

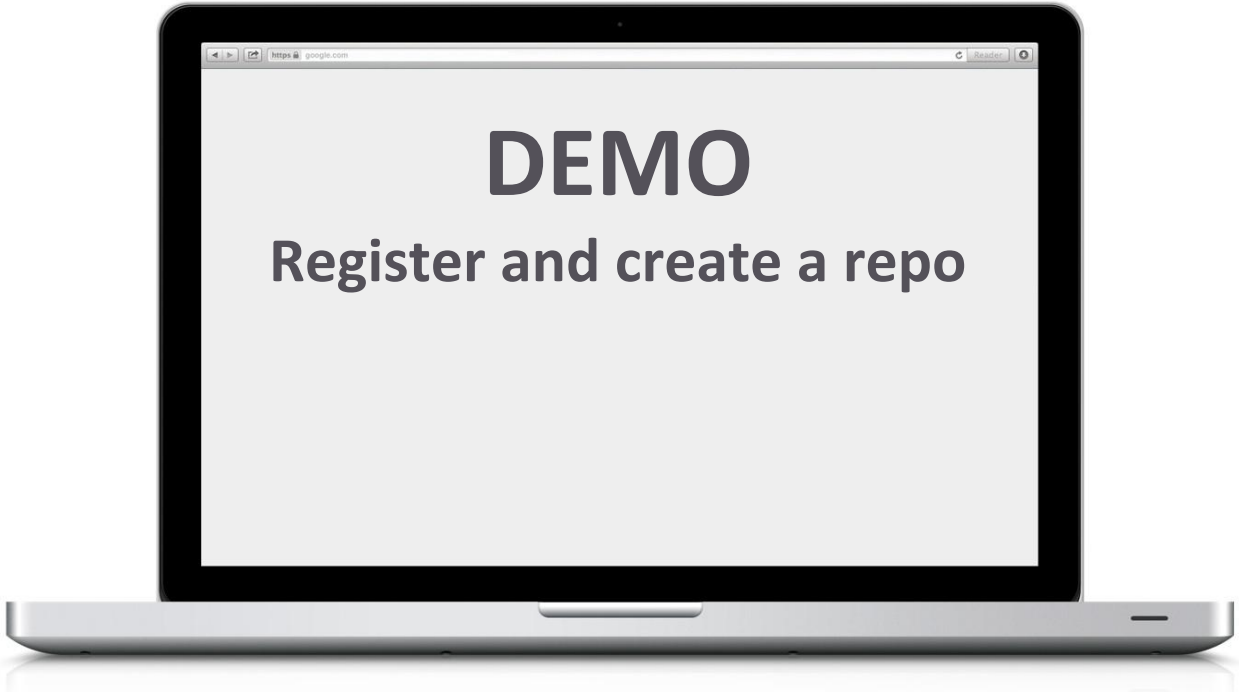
- Distributed source control vs centralised
- You have a local copy of the repository
- Can push to remote server repository

Written by Linus Torvalds...

...in three weeks

GitHub

- Online version of git
- Makes social coding easy (pull requests)
- Open source community



DEMO

Register and create a repo

Your turn

Create a github account at http://github.com

Create a new repository

- Name it
- Make it public
- Tick to initialise with ReadMe

Git clients

- Git Bash (Command line)
- GitHub for Windows/Mac
- GitExtensions
- SourceTree

Today we are using SourceTree and Git Bash

DEMO

Clone the repo

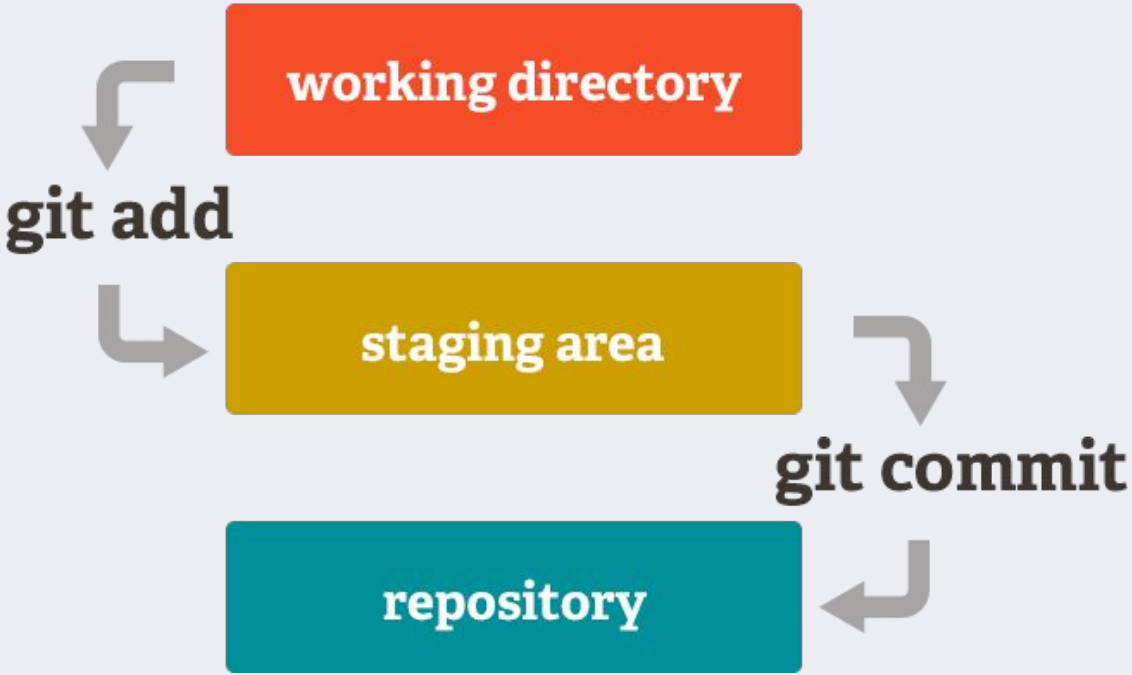
Your turn

- Clone the github repository you just created to your local machine
- If you're using command line:
  - git clone [HTTPS URL HERE]

Commit

- Has a unique hash
- Set of changes to many files
- Represents a piece of work from one person
- Has a comment - can include issue numbers

Example of a commit


working directory
git add
staging area
git commit
repository

DEMO

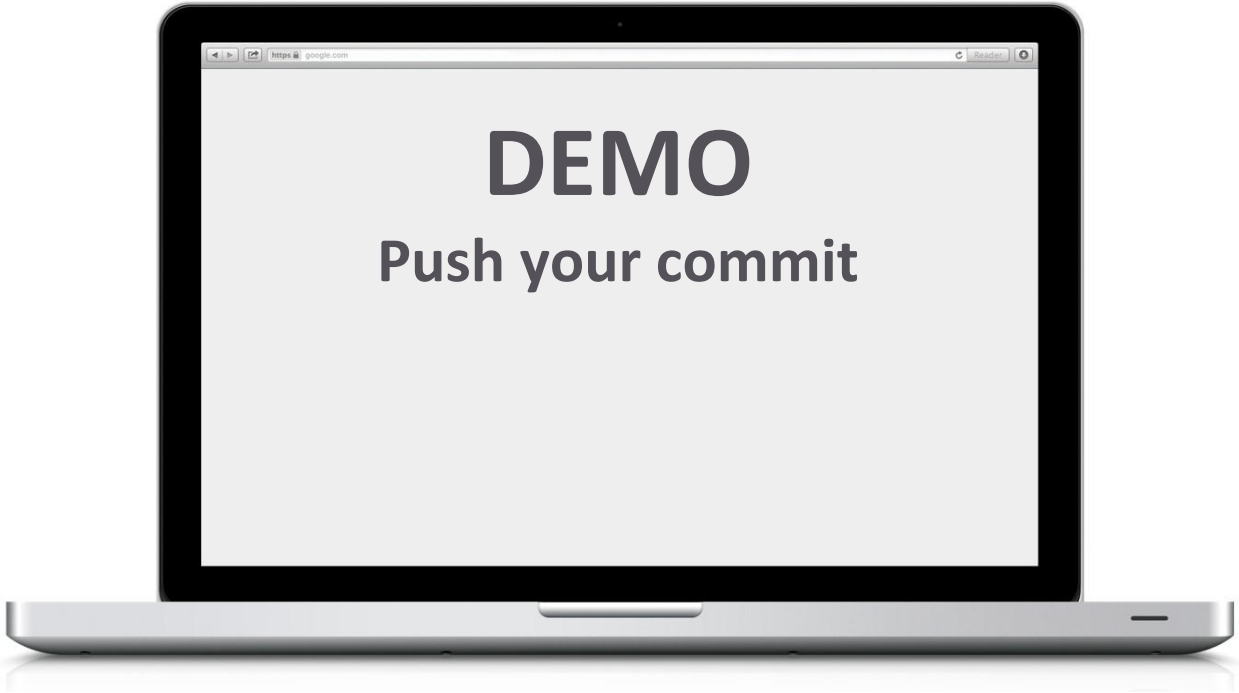
Make a change and commit it

Your turn

- Change the readme.MD file from the repo you cloned earlier
- If you're using command line:
  - git status
  - git add *
  - git status
  - git commit -m "Some awesome message"

Push

Sends your changes (commits) back to the server (your repo)



DEMO

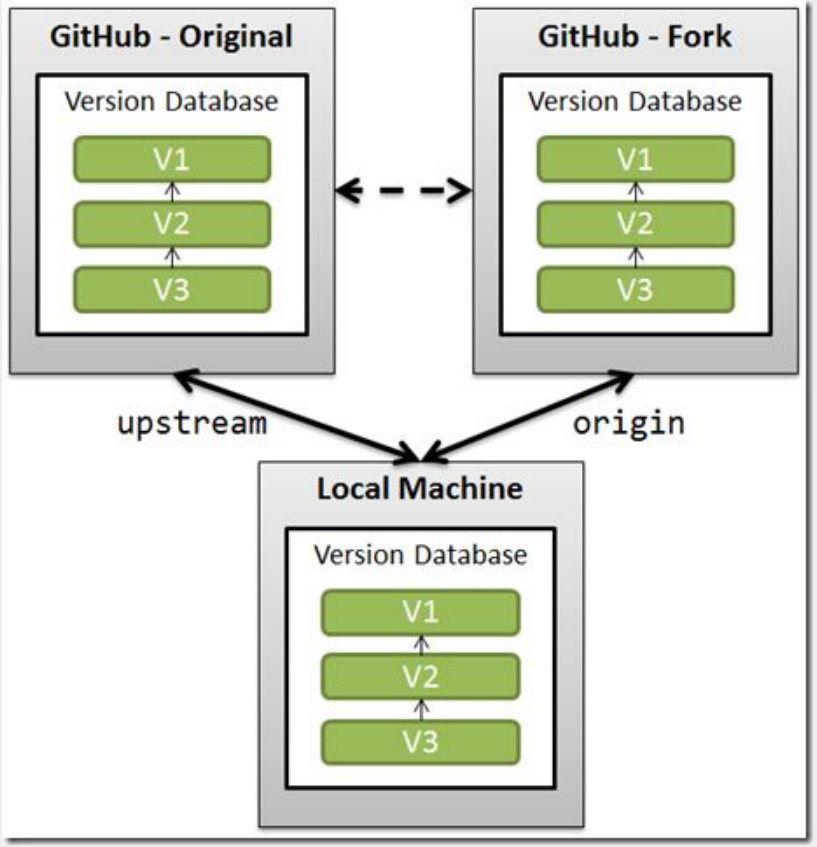
Push your commit

Your turn

- Push your commit and have a look at the edited repo
- If you're using command line:
  - git push origin

Forks

- Isolated server copy of the repository
- Useful if you don’t have permission to push to the remote repository



Pull Requests

- Request to pull changes from our Fork to the master Repo.
- Internally we use this for code reviews

Example of a pull request

Your turn

- Fork the summeroftech repo on Github
- Clone your fork
- Add your name to the README
- Commit
- Push your change to your fork
- Make a pull request

Merging and conflicts

A.txt <-> B.txt <-> C.txt - KDiff3

File Edit Directory Movement Diffview Merge Window Settings Help

A (Base): C:\Users\max.henderson\Desktop\A.txt
Top line 1 Encoding: System Line end style: DOS

```
public int Mutiply(int a, int b)
{
        return a*b;
}
```

B: C:\Users\max.henderson\Desktop\B.txt
Top line 1 Encoding: System Line end style: DOS

```
public int Add(int x, int y)
{
        return x+y;
}

public int Mutiply(int a, int b)
{
        return a * b;
}
```

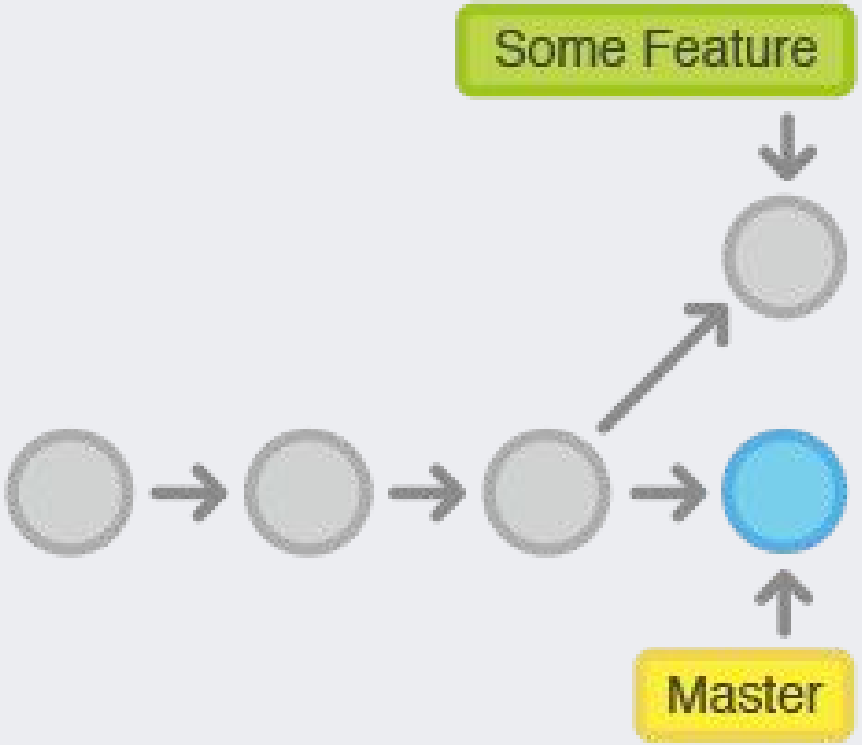
C: C:\Users\max.henderson\Desktop\C.txt
Top line 1 Encoding: System Line end style: DOS

```
public int Mutiply(int a, int b)
{
        return a*b;
}

public int Subtract(int i, int j)
{
        return i-j;
}
```

Output: C:\Users\max.henderson\Desktop\C.txt
Encoding for saving: Codec from C: System Line end style: DOS (A, B, C)

```
B public int Add(int x, int y)
B {
B         return x+y;
B }

  public int Mutiply(int a, int b)
  {
? <Merge Conflict (Whitespace only)>
C }
C
C public int Subtract(int i, int j)
C {
C         return i-j;
  }
```

Branching



Resources

https://try.github.io

http://gitimmersion.com/

http://git-scm.com/book/en/Getting-Started-Git-Basics

https://www.youtube.com/watch?v=1ffBJ4sVUb4 1:40:00

http://gitref.org/index.html

https://www.atlassian.com/pt/git/tutorial/git-branches#!checkout