Git 101: A Crash Course for Productive `git` Usage

for the University of Maryland Cybersecurity Club

William Woodruff

Agenda

1. What is Git?
2. Why should I care?
3. How do I use `git`?
4. How do I use `git` *properly*?

Some background on Git

Prior to 2005, the Linux kernel used BitKeeper for source control management ("SCM").

In 2005, Andrew Tridgell (of Samba and `rsync` fame) reverse engineered parts of BitKeeper's (proprietary) protocol, causing BitKeeper to withdraw free use of their SCM.*

Git was (initially) hacked together as a replacement, with some features tailored to the Linux community:

- Decentralized, unlike CVS or SVN (which both use a client-server model)
- Performant, even on extremely large source trees ("patching should take no more than 3 seconds")
- Resistant to accidental or intentional corruption

These characteristics, plus its adoption by sites like GitHub and Bitbucket, have made Git the dominant SCM.

* The actual story is longer, and more interesting.

What does Git actually do?

Git is a "distributed version control system", which means very little.

For our purposes, Git:

- Manages source code by breaking it into discrete groups of changes ("commits")
- Manages commits by breaking them into discrete groups ("branches")
- Manages branches by associating them with different copies of the source code ("remotes" and "clones")

There are many ways to interact with Git (like `gitk` and the GitHub website), but we'll use the reference `git` CLI for this presentation. At the end of the day, each has its place and use cases.

Why should I care?

- An increasingly large amount of open source development is done on GitHub and similar platforms, and employers *love* to see open source contributions.
  - Contributions make the most impact when they're delivered well:
    - Clear, concise commit messages
    - Discrete changes broken across different commits
    - Descriptive branch names, &c
- Even if you don't like open source (which is fine), there's a good chance your future employer is using Git. You'll be expected to be comfortable with using Git (or a very similar SCM) on a daily basis.
- Lots of interesting incidental topics: file diffing, conflict resolution (not the HR kind), proper project planning, &c.

How do I use git?

First, you'll have to have it installed. You can install it and follow along with these slides by interacting with a local repository, if you'd like.

To get started, let's copy ("clone") a repository:

```
$ git clone https://github.com/UMD-CSEC/git-101
$ # go into the directory we just cloned
$ cd git-101
```

If we weren't cloning an extant repository, we'd use `mkdir` and `git init` to create a new one:

```
$ mkdir my-repo
$ cd my-repo
$ git init # creates a new repository inside "my-repo"
```

Adding, removing, and modifying files

We now have a local copy of the repository, which we can modify to our hearts' content.

Looking around

Now that we're in a repository, we can look around a bit:

- `git status` - Show unstaged changes to the repository
    - DIY: Create a new file in the repo, and run `git status`. What changes?
- `git diff` - Show unstaged *modifications* to files
    - DIY: Change an extant file in the repo, and run `git diff`. What does it show?
- `git log` - Show the commit logs for the repository
    - DIY: