Let's Git this Party Started

An introduction to Git and GitHub

@Kim_Moir

Tonight's agenda

- Introduction
- A short history of open source version control systems
- Why Git and GitHub
- Hands on exercises: Using Git, GitHub
- Git concepts

-poll people to ask what operating systems they are running
-verify that everyone has wireless working
-mention break half way through
-give out GitHub stickers and collect names for door prize
-pass around USB key

Introduction

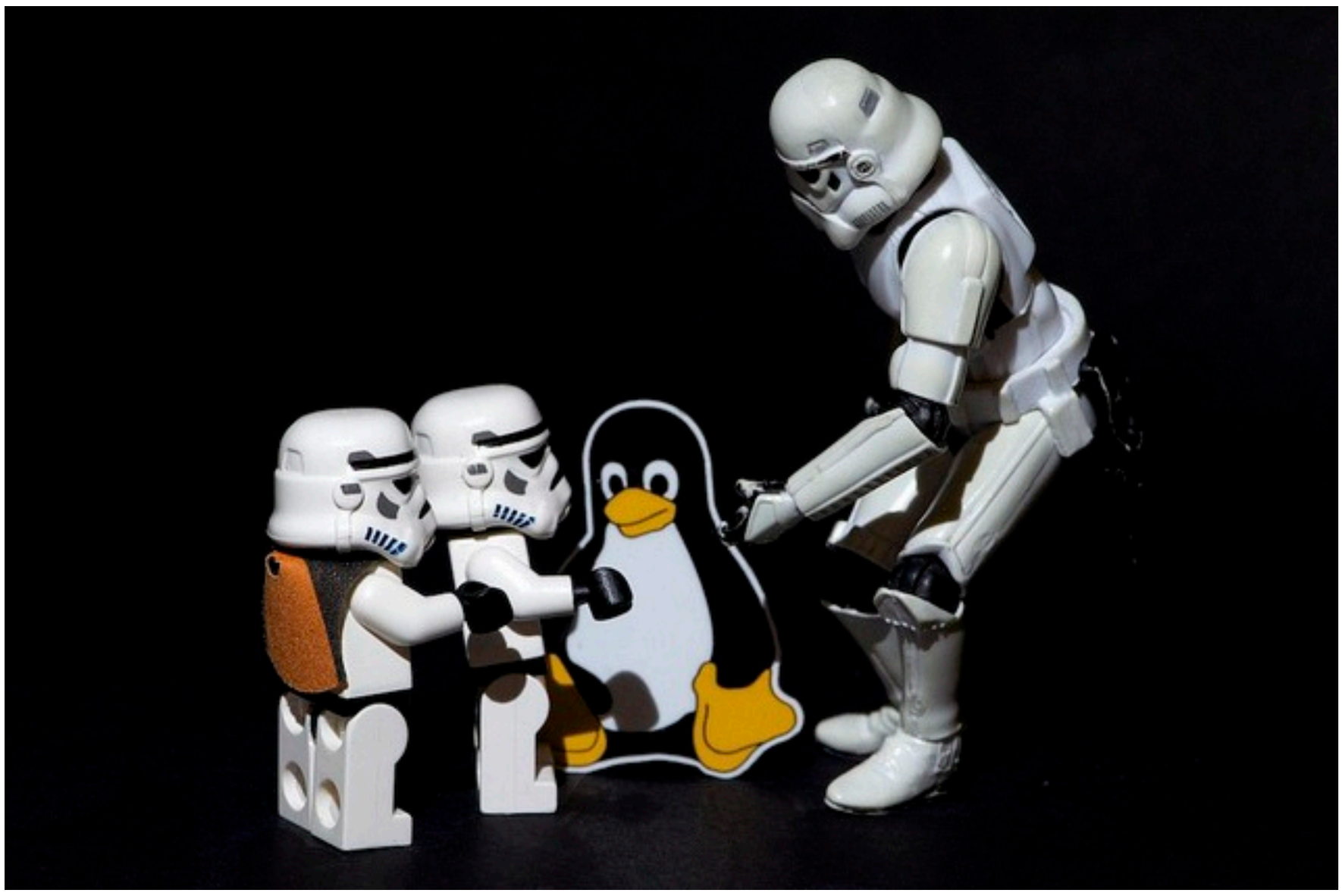
- About you?
- About me

I'm honoured to be here tonight. This is my first time attending Girls Develop It Ottawa. How many of you are here for the first time?

Who has used version control systems before?
Who has used Git or GitHub before?

My name is Kim Moir and I've been involved in the open source Eclipse community for 10 years. I'll be starting a job as a release engineer at Mozilla in a few weeks. Anybody hear of Eclipse before? Eclipse is an open source community that produces software tools for developers. The Eclipse Foundation office is in Ottawa, on Centrepointe drive. 1000 committers, 70+ projects, millions of downloads. Release engineering "is concerned with the compilation, assembly, and delivery of source code into finished products or other software components." (Wikipedia definition). As a result, just like software developers we spend a lot of time interacting with the version control system. I have a business degree, not a computer science degree, so like many of you, I've learned a lot a long the way during my career. Outside of work, I'm a long distance runner.

What is open source



-Code is developed in the open under an open source license that allows people to redistribute and modify the code
-Code can be inspected, downloaded and compiled by anyone
-Contributions are welcome from the community
-People can build upon this code and deliver commercial products that are based upon it

Open source communities

The Apache Software Foundation

Community-led development since 1999.

THE LINUX FOUNDATION

Examples of open source communities

-develop Apache web server, Linux operating system, Eclipse IDE, Mozilla Firefox and so on
-Other examples Gentoo, Gnome, SPI etc
-Open source foundations provide governance models, intellectual property management, marketing, and infrastructure for these communities
-Git is also developed as a open source project (software freedom conservatory, hosted by GitHub)

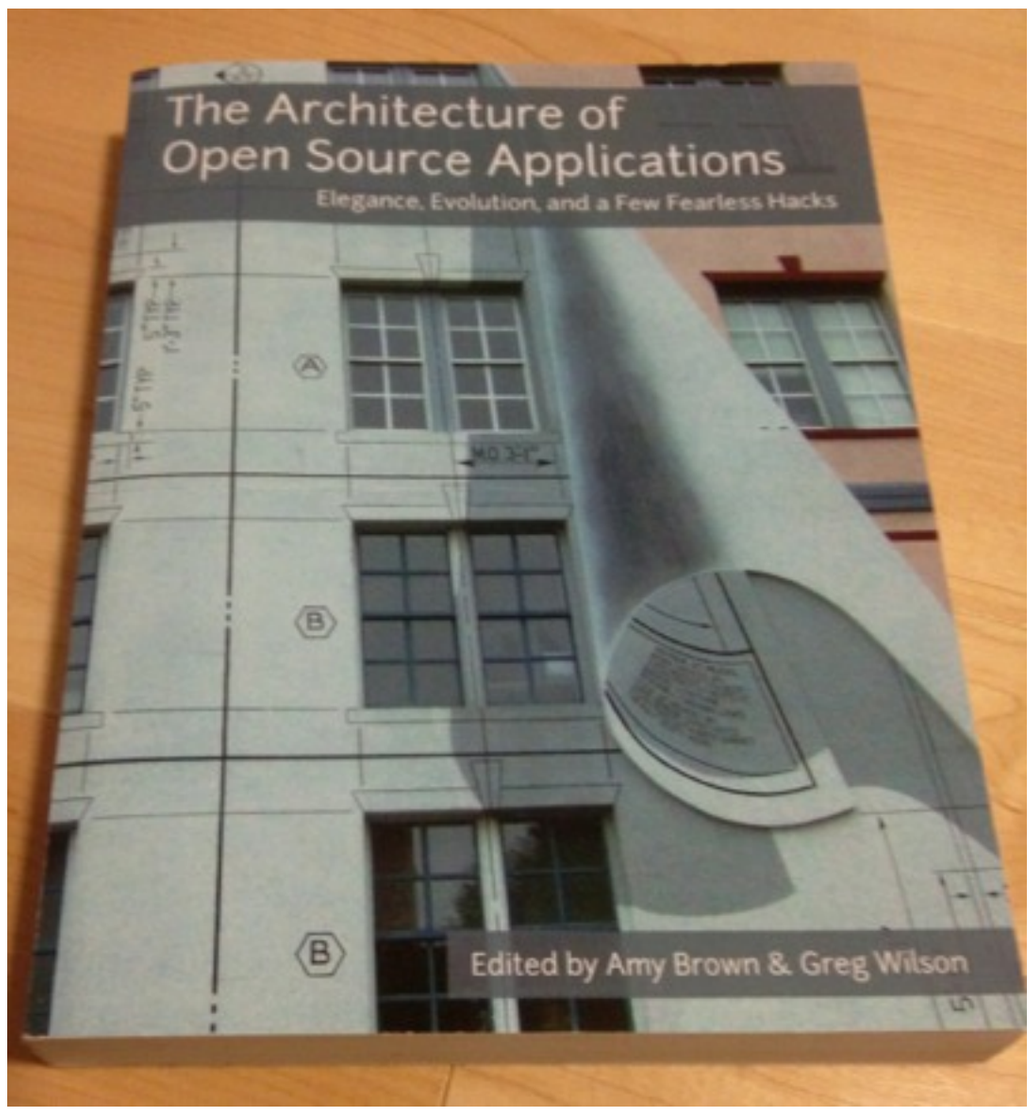
What is version control

- A system to manage changes to digital artifacts
- Examples: code, documentation, the possibilities are endless

It's a system that that tracks and provides control over changes to source code, or other types of digital artifacts. Examples of source control you may have seen before. Wikipedia where you can look at the revision history of a document. Any other examples that people may have used?

Why do we need version control systems? We need to be able to track changes to our code. For instance, look back at the history and see what changes were made and who actually made them.. Be able to remove a change. And contribute changes of your own. For example, if the operating system on your laptop had a security bug that made it vulnerable, the software developers need to be able to go use the exact version of the code base that was used to build that version, and apply a patch to fix it. Version control systems let you do that. Without version control, there would be chaos.

Example: Book collaboration



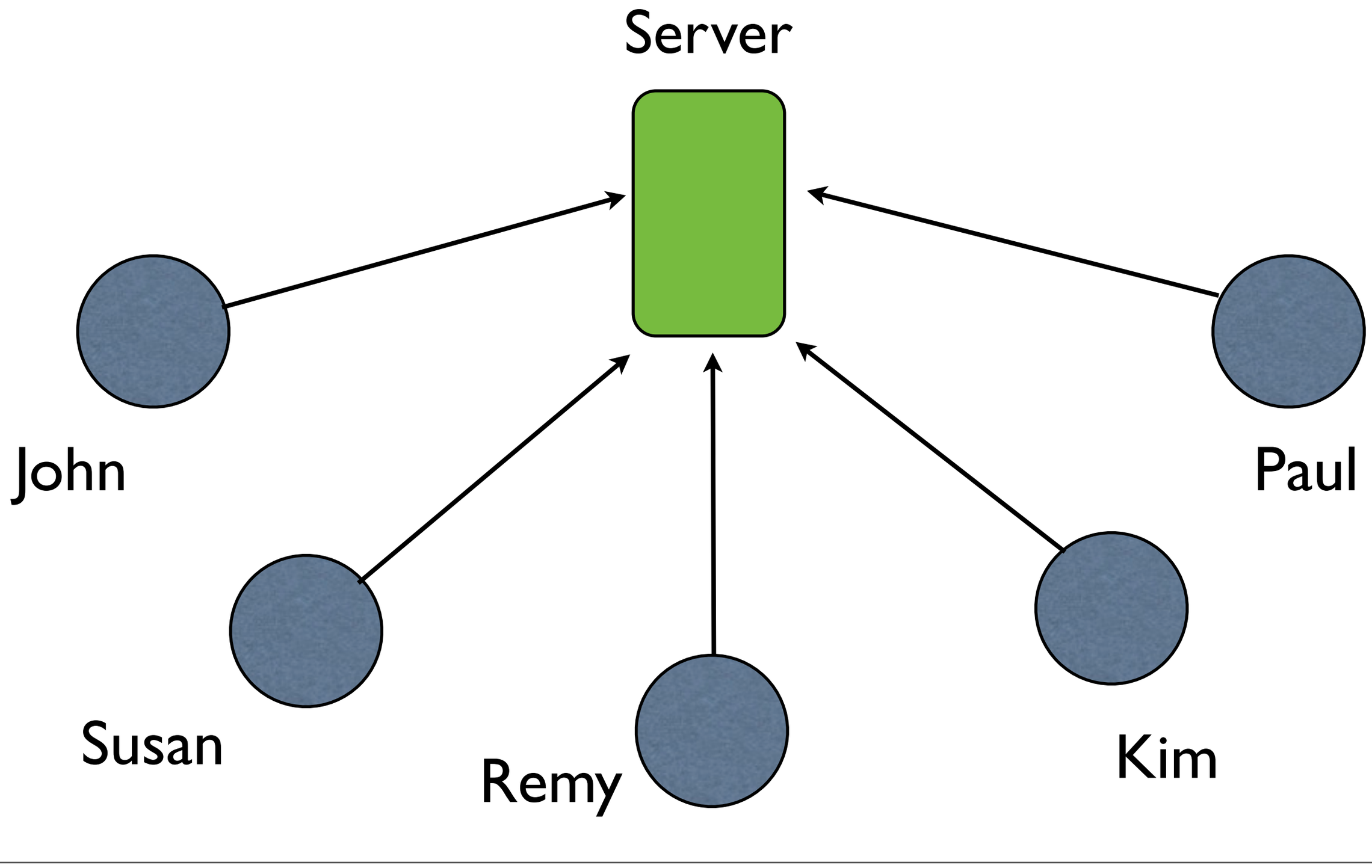
-People in many different timezones are collaborating on a book. They want to be able to make changes to their chapter without overwriting other author's changes.
-Also, it would be interesting to see what changes were made and when in the document
-If there are changes that the editor doesn't agree with, she can revert them back to the previous version
-The editor can also see if people are actually working on the book :-)

History of Version Control

In the beginning, people didn't have version control systems. They just passed files around via email. Very painful to merge changes.

Imagine if you were a student and working on a group paper. You all started with a one page summary of what you were going to do and then added the content to your liking. The same thing would apply to a group of developers working on a common code base.

CVS and SVN



In the 1990s, CVS was invented. It was basically a model that allowed developers to check out copy of a centralized repository make local changes, and release or merge these changes back into the central server. In the early 2000s, SVN was released and was supposed to overcome some of the problems with CVS.
Limitations of these centralized source control systems: hard to track changes, difficult to branch and merge, everything you do is public since the public server controls all. Also, you have to deal with network latency when you contact the central server each time.

2005: Git

Thus in 2005, Git was invented by Linus Torvalds who is famous for inventing something else. Any guesses? Linux. Linux is an open source operating system that's very popular around the world. Most web sites run on the Linux operating system.

Why Git was developed
The Linux development team used to use a version control system called BitKeeper which they quite liked. However, the authors of BitKeeper withdrew their open source license and made it proprietary. Thus the Linux kernel team had to find a new version control system. And if you've spent your spare time writing an operating system, perhaps it's not difficult to write a distributed version control system. When asked why he named it Git, he said

"I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git."

http://en.wikipedia.org/wiki/Git_%28software%29

Git : English slang for a difficult or unpleasant person.

"I'm an egotistical bastard, and I name all my projects after myself. First Linux, now git."

--Linus Torvalds

Source Control

- Centralized all use a central repository: CVS, Subversion, Perforce
- Distributed (DVCS) allow you to reconcile your local repository with a centralized one: Git, Mercurial

Git Design Goals

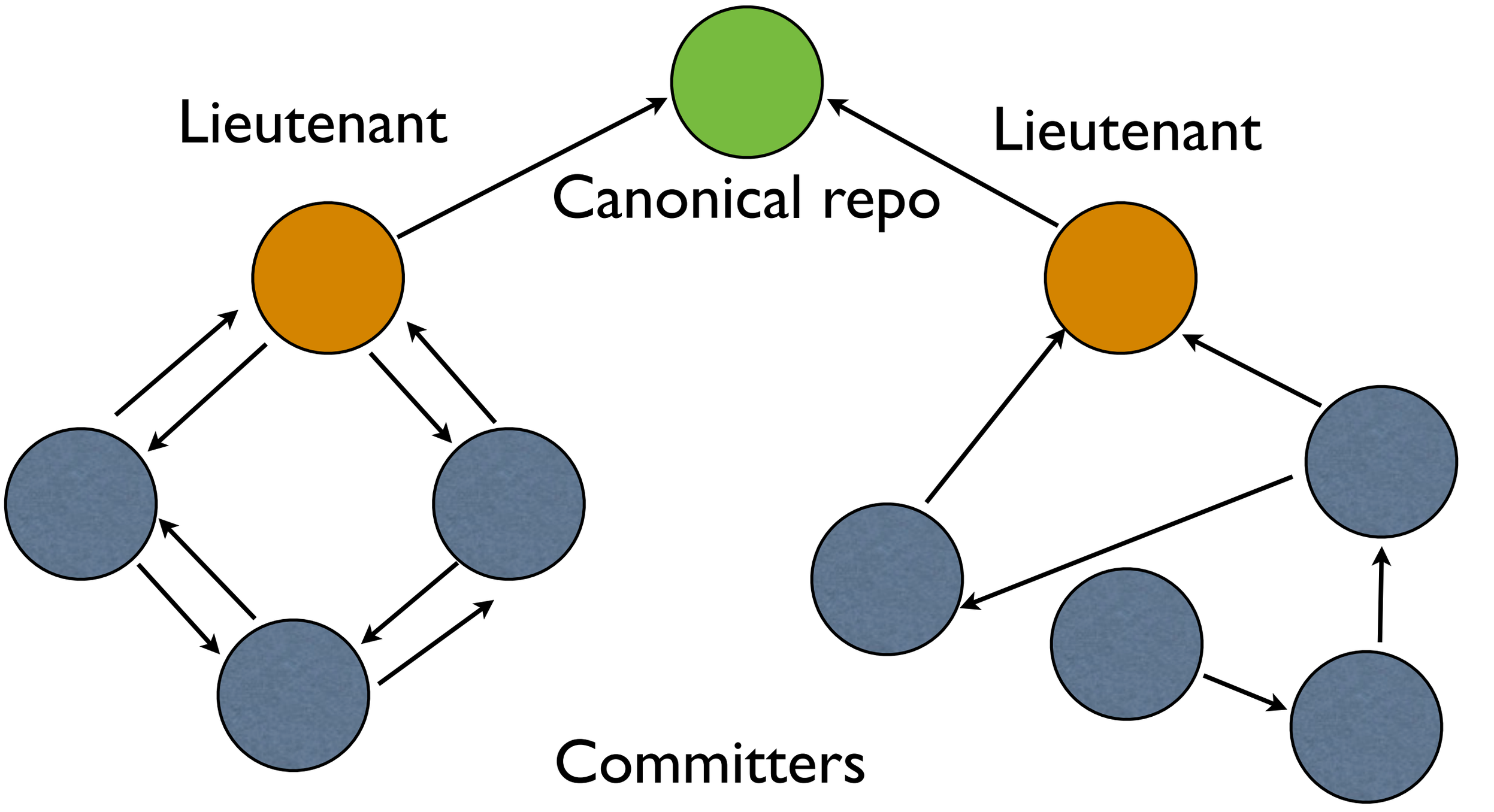
- Fast
- Distributed
- Each commit has a corresponding hash key
- Everyone has a local copy of the history

It's quite popular with many open source projects such as Android, Apache, Linux, Eclipse and more. It's also used within corporations.

Advantages and disadvantages of Git
-distributed, fast, everyone has a copy of the local repo and its history
-Since it contains all history, it's easy to revert to a historical point in your source code
-many models for organization for the development team

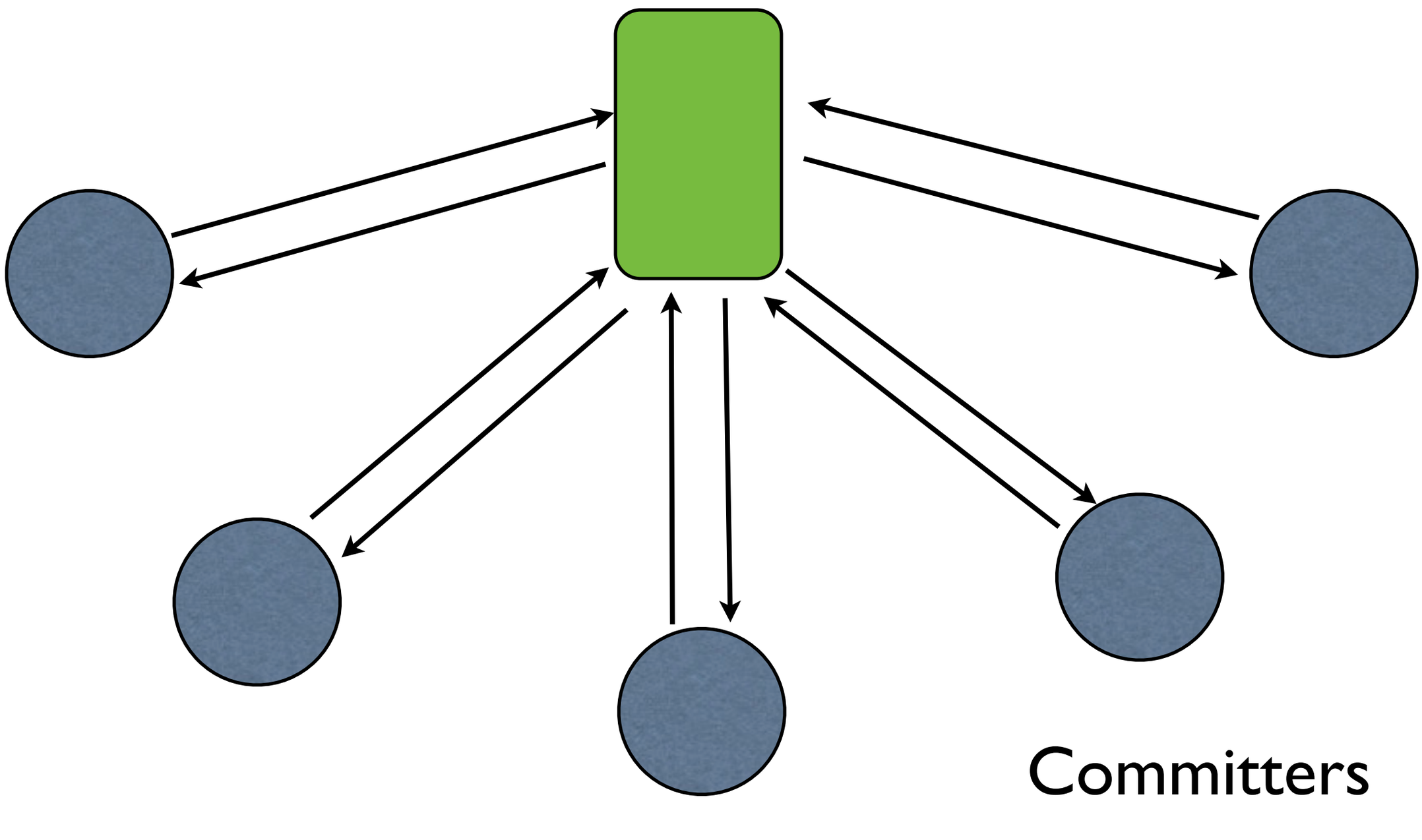
Disadvantages
-have to clone the entire repo if you want to just check out one project
-learning curve

Development models



define committers
For example: Linux kernel
Very flexible

Another model



Eclipse SDK project uses canonical repo which committers push and pull from, not each other explicitly

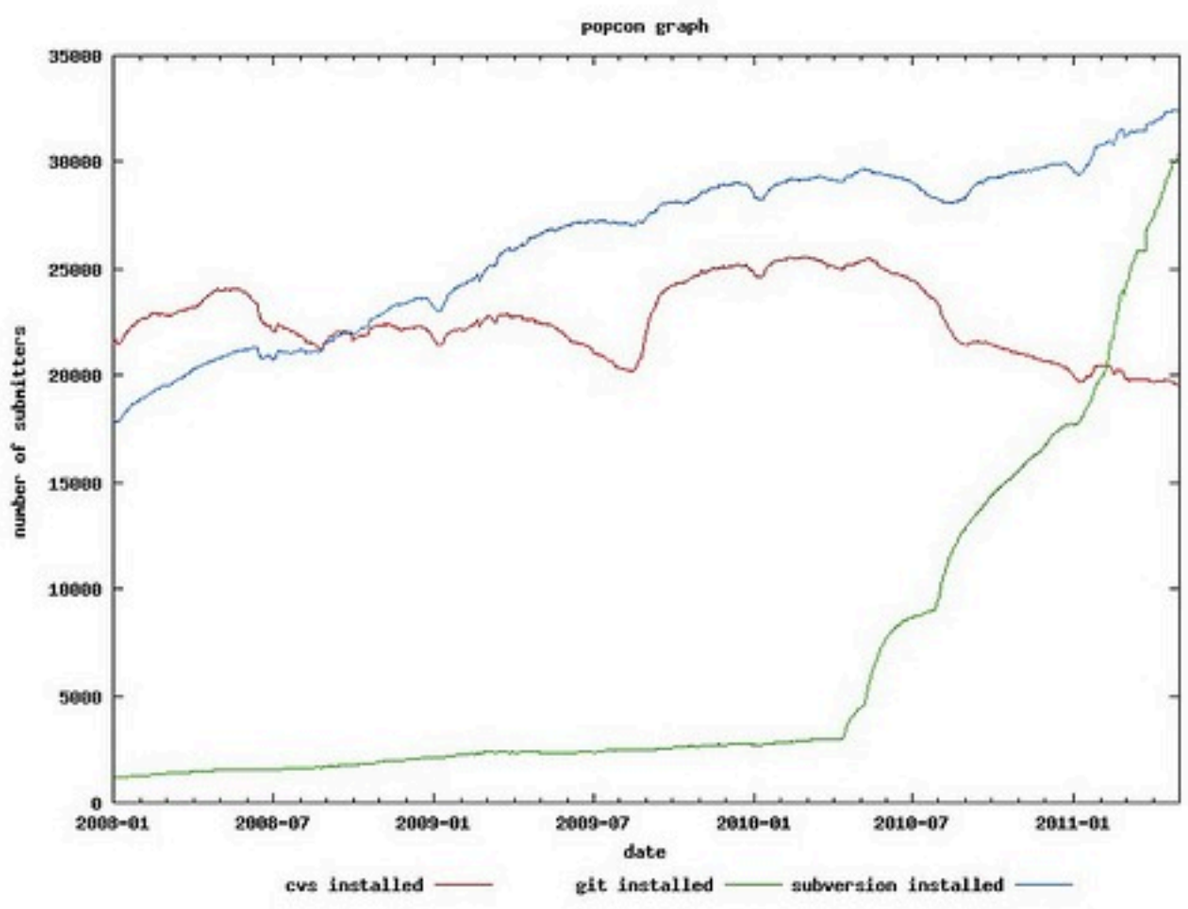
As with any DCVS, an agreement is reached within the team on regarding how they would like to work

Git Usage

tecosystems

because technology is just another ecosystem

You Won't Get Fired for Using Apache



Source Stephen O'Grady Redmonk

Git usage is rising rapidly so it's a good skill to learn

At the Eclipse foundation, we have been using CVS for 10 years, and half the projects recently finished migrating to Git.

Installation steps

- Install git
- Setup ssh keys
- Create github account and add ssh key
- Setup your name and email in gitconfig
- git config --list

Check your installation first

- Is git installed? Yes, if this command runs

```
git
```

- Is ssh installed and a key created? Yes, if this file exists

```
ls ~/.ssh/id_rsa.pub
```

Installing Git

- Windows: http://help.github.com/win-set-up-git/
- Linux: http://help.github.com/linux-set-up-git/
- Mac: http://help.github.com/mac-set-up-git/

1. Install git. (If you don't already have it installed already)
2. Setup ssh keys (if you don't already have one generated on your machine) and copy to GitHub
3. Setup name and email in git config
4. API token setup isn't necessary for this tutorial, can be skipped if you want

Create a local repository

Change to your home directory

`cd ~/`

create a working directory

`mkdir my_repo`

`cd my_repo`

`ls -la` to look at the directory

`git init`

`ls -la`

Should see `.git`

To check the status of your repo

`git status`

Key concepts in Git - cloning, branching, merging, committing, push to remote, pull, log, help, creating branches, switching branches
-explain local versus remote repos
-checksums are associated with every commit
-Git client and server on various operating systems
-look at git server config - git config --list
-git protocols:git, ssh, http
-concept of commit rights since you have permissions on the repo, you commit to it
-remote repos - you will have to have the correct permissions to push
-can only push via ssh

Git-ting help

`man <command>` (print the online manual)

`program help` (git help)

`<program> help command` (git help init)

Add files to your repo

Create a file

```
touch hello_world.txt
```

Check repo status

```
git status
```

Git only tracks files we tell it to

Add the file to the repo

```
git add hello_world.txt
git status
```

File should now be tracked

Changes and commits

Open hello_world.txt and add some more text

```
git status
```

Stage the change

```
git add hello_world.txt
```

Commit the change

```
git commit -m “added text”
```

Create another file

```
touch newfile.txt
```

Add all untracked files in the current directory

```
git add .
```

Commit the change

```
git commit -m “added newfile”
```


Difference between staging and committing

Why do you need to add files?

- Git tracks your changes, not your files
- We are telling git to track the current state of the file when we add it
- Staging is adding the file to the repository to be tracked
- If you change a file, you also have to tell git to track the changed file

What have you done?

```
git log
git log | grep newfile
git log --pretty=oneline
gitk
```


git log can be used to look at the history of the repo
git log --pretty=oneline shows the hash tags associated with commits to the repo with the most recent commit first
gitk - graphical view of the repo

Git archeology

add text to hello_world.txt

```
git add .
git commit -m “added text”
```

Find the commit hash

```
git log --pretty=oneline
```

checkout a previous commit

```
git checkout <hash>
cat hello_world.txt
```

To return to the current version

```
git checkout master
```


By default, master is the current branch in a git repo
git log --pretty=oneline shows commit history with most recent commit first
You can have many branches in a repo, we’ll talk more about that later

Tags are better for Humans

- The reality is that nobody types hashes to check out a commit
- One approach is to use tags
- Tags as a general practice are considered immutable
- Tags are often used to record a moment in time for administrative purposes

Tags are a snapshot of a repository at a point in time
For instance, at Eclipse, we tagged the repos with a timestamp that reflected the time the build started. So we could go back in time and see what went into each build.
Each tag tags the whole repo, not just a single file or directory

Tags

Tag your current repo as version_one

```
cat hello_world.txt
git tag version_one
```

checkout previous changeset

```
git checkout version_one^
cat hello_world.txt
```

tag the previous changeset as a RC

```
git tag rc
git checkout version_one
```

Use gitk and git tag to view tags

To find your tag in a repo with a large number of tags
git tag -l | grep <yourtag>

Undoing Local changes

Return to master branch

```
git checkout master
```

If you haven't committed

```
change hello_world.txt
git checkout hello_world.txt
```

look at local copy of hello_world.txt

Version of file is overwritten with the one from the repo

Undoing Commits (1)

If staged
change hello_world.txt

```
git add hello_world.txt
```

```
git reset HEAD hello_world.txt
```

This will unstage the change

```
git checkout hello_world.txt
```

roll back unstaged changes

Undoing Commits (2)

```
cat hello_world.txt
```

Find the commit hash

```
git log --pretty=oneline
git revert <hash>
cat hello_world.txt
```

Your change is reverted to the previous commit

Can revert your revert if you like :-)

git diff

git diff shows the difference between staged and unstaged changes

change hello_world.txt

```
git diff
git add .
git commit -m “update text”
git diff
git status
```

What is a branch?

A branch in Git is a different version of the source code. It starts as the copy of an existing branch. The default branch is called master. For instance, you may want to try to implement a new feature that will radically change how your product works. So you may want to create a new branch so you can explore this work without disrupting the work of others. When you're sure that this feature is working and won't break the build, you can merge the changes from this branch into the main development stream. Another example: You need to fix a bug for a customer. So you can branch the code base from the release branch and implement that one fix for a new release. Ongoing work isn't touched.

Branching

- Why branch?
  - Branch to conduct some exploratory work without impacting the work that others are doing in the master branch
  - Branch to backport changes to a old release

How to branch

Create a new branch called v2

```
git checkout -b v2
```

add a new file

```
touch v2.txt
```

change hello_world.txt

stage your changes

```
git add .
```

commit to the new branch

```
git commit -m “changes to v2 branch”
```


Look at branches in gitk
yellow: tags
green: branches

Switching branches is easy breezy

List all branches

```
git branch -a
```

Branch with * is active

switch to master

```
git checkout master
```

ls to view files

switch to v2

```
git checkout v2
```

ls to view files

Branching is Git is cheap

Can have local branches that the server doesn't know about

When you branch in Git, the entire repo is branched

In CVS, you can branch just a single file or directory. Not in Git.

Commit to a branch

Switch to master

```
git checkout master
touch master.txt
git add .
git commit -m “added master.txt”
git checkout v2
ls
```


Must commit to the branch before switching to another branch or your changes will be lost

Merging is the opposite of branching

While you’re working in your local branch on your awesome new feature, other people on your team will also be making changes.

You want to keep your local branch current so that you have all the changes that other team members are contributing

How to Merge

What branch are we in?

```
git status
```

If not v2,

```
git checkout v2
```

Merge changes from master into v2

```
git merge master
ls
git status
```

Merge conflicts

```
git checkout master
edit hello_world.txt
git commit -am “update text in master”
git checkout v2
```

edit hello_world.txt and add different text

```
git commit -am “hello_world.txt in v2”
```

merge from master to v2

```
git merge master
```

you’ll be notified of a conflict - edit hello_world.txt to fix

```
git commit -am “hello_world.txt in v2”
git merge master
```


The same file may be changed by two different people at the same time in their version of the local repository
Attempting to merge them may not work – you have a conflict
You need to modify the files before you move on

Git blame

- Shows who last modified the file or made the last commit

```
git blame hello_world.txt
```

GitHub

- GitHub is a commercial site that allows you to host Git repositories
- Many open source projects host or mirror their repositories to GitHub
- Allows you to showcase your code to potential employers
- Possibly attract new contributors
- Learn from others!

-history of GitHub - open source and commercial aspects. GitHub was launched in 2008 and is a web based service for hosting git repositories. It's a leader in social coding. Just like you can follow someone on Twitter, you can follow a developer on GitHub and see the work that they are doing. There is also issue tracking, so you can open bugs or feature requests against people's code. You can also look a graphs and see how project's develop and who is contributing to them. For example:
Who's contributing
https://github.com/eclipse/orion.server/contributors

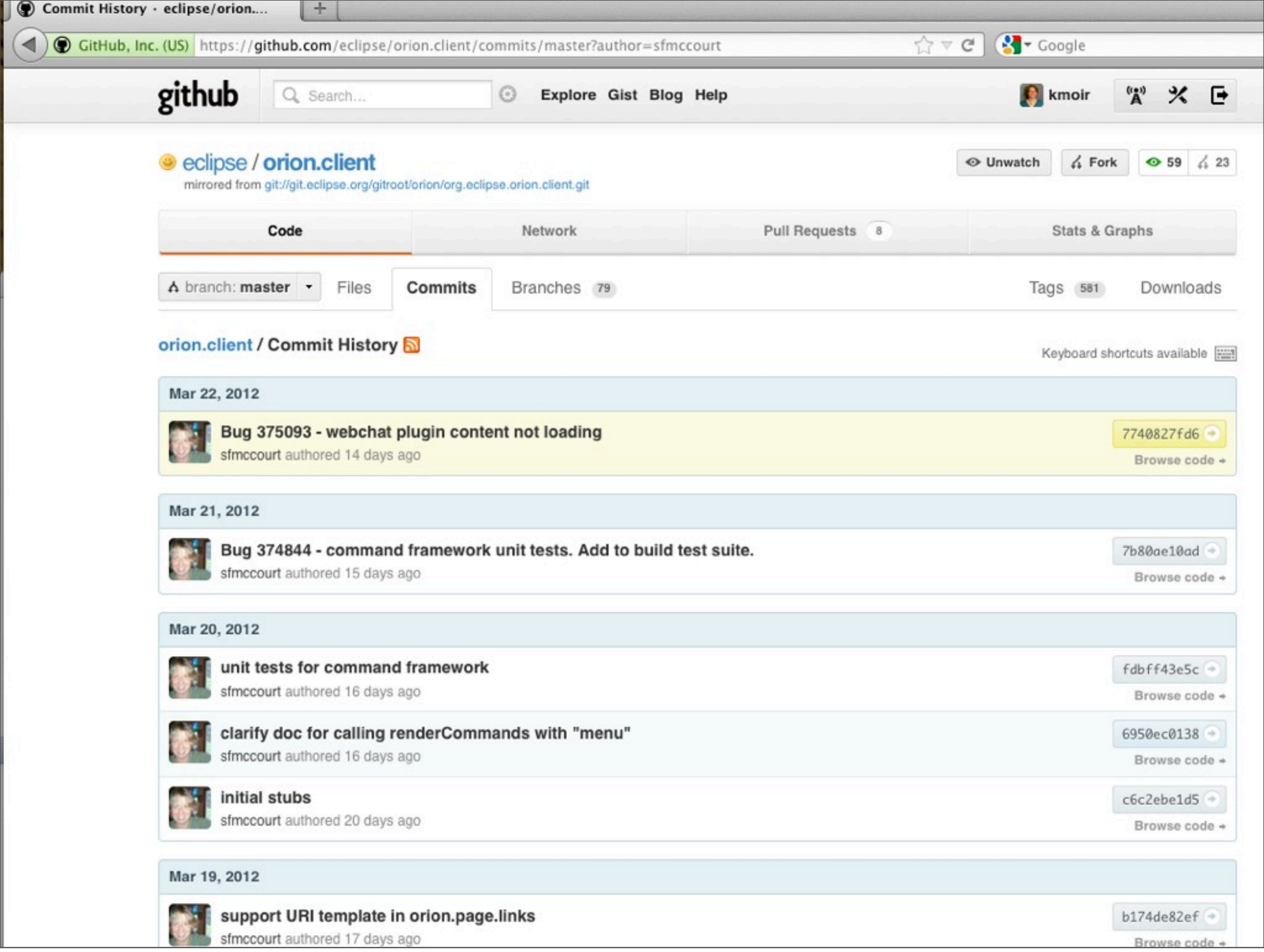
Impact graph for Eclipse Orion
https://github.com/eclipse/orion.client/graphs/impact

-mention that code there is open source, what does open source mean, what benefits does it offer

-very quick mention of open source business models, how this giving away software for money can make money

Why release your code as open source on GitHub?
-Shows a portfolio of your work for potential employers
-showcase your open source project to attempt to get new contributors providing fixes to your project.
-Open Source projects are often mirrored to GitHub even though the primary development may take place on another server. For instance, for legal reasons, all eclipse code is hosted on git.eclipse.org. However, these projects are mirrored to GitHub.
-Allows you to focus on writing code, not managing infrastructure, just like any hosting infrastructure

-many open source projects host mirror their projects to GitHub, give examples
-can mirror both personal and professional projects (although this may depend on your



-Software is social
-Look at graphs in GitHub to see commit activity and code submitted
For example, look at Susan McCourt, Orion committer
https://github.com/eclipse/orion.client/commits/master?author=sfmccourt
https://github.com/sfmccourt folllow
-Watch a project or follow a person, learn in the open
-Fork the code if desired

Git clone

A git clone is a local copy of a remote repository. All history is copied locally. You can then modify the local copy, and if you have permissions, push the changes up to a remote repository. If you don't have rights, you can create a pull request to ask the repository owner to accept the changes on your behalf and add them to the repository.


Hey girl.
Nice pull request .
Let’s merge it in.

Clone the course repository

Browse to this git tutorial
https://github.com/kmoir/Git-tutorial
find the clone url

```
cd /~
git clone git@github.com:kmoir/
Git-tutorial.git
cd Git-tutorial
git status
ls
```

Remote Repos

View active remotes

```
git remote -v
```

Get latest version and merge updates from Remote

```
git pull <remote name> <local branch>
git pull origin master
```


could update content for class

How to play well with others

Commit local changes

```
git commit -am “my latest commit”
```

pull any changes from remote repo

```
git pull origin master
```

push local changes to the remote

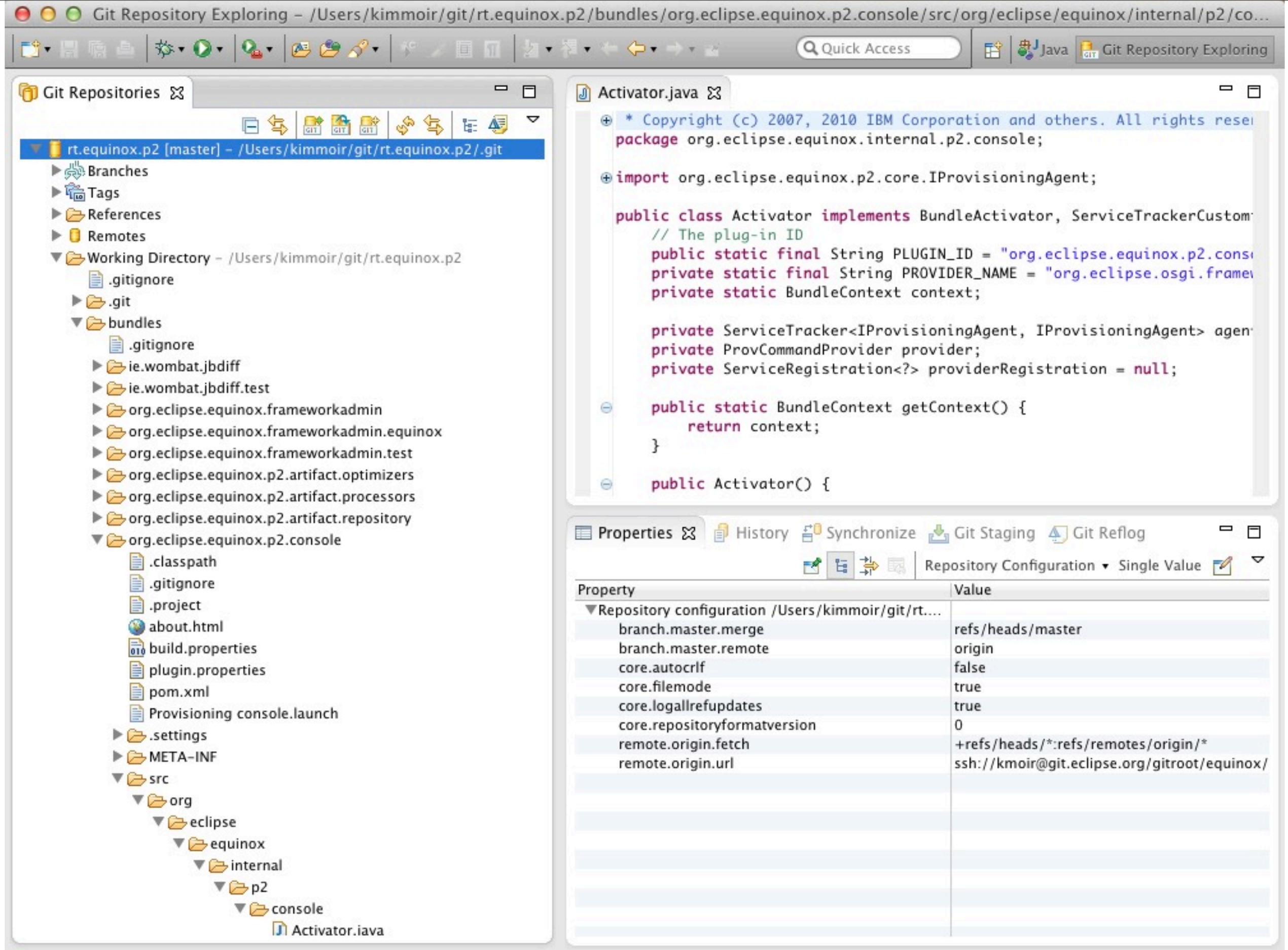
```
git push origin master
```


You must always integrate changes from the remote repository into your local copy before you can push up your changes

Beyond the command line

- Many IDEs have Git integration
- Example Eclipse + EGit
- htttp://download.eclipse.org
- Allow you to work seamlessly with Git repos from within your IDE

Confession: I don't use the command line when I use Git to interact with the code repositories I work with
Brief history of IDEs
IDEs are integrated development environments i.e. Eclipse, NetBeans, IntelliJ. Many open source and commercial IDEs are available.
Basically, they are desktop applications that providing tooling to you to make it easier to develop software i.e. automatic compilation, connecting to source code repositories, content assist, refactoring etc.



-Clone repos, commit code, branch tag etc. all within your IDE

Eclipse Orion

- demo connecting to GitHub via Orion
- Request password here http://www.eclipse.org/orion
- Login here http://orionhub.org
- Clone the repo https://github.com/eclipse/orion.client.git
- http://wiki.eclipse.org/Orion/Getting_Started_with_Orion

Photo Credits

- Linus Torvalds Wikipedia http://en.wikipedia.org/wiki/File:Linus_Torvalds.jpeg
- Git utilization graph http://redmonk.com/sograady/2011/11/28/you-wont-get-fired-for-using-apache/
- Mozilla pins http://www.flickr.com/photos/flod/2221300134/sizes/z/in/photostream/
- Women with laptops at Pop Life http://www.flickr.com/photos/yourdon/5157431891/sizes/l/in/photostream/
- Merge http://www.flickr.com/photos/lex/43631705/sizes/o/in/photostream/
- Merging Ryan Gosling http://programmerryangosling.tumblr.com/post/16921926583
- X wing clones http://www.flickr.com/photos/pmiaki/5520399713/
- Branches http://www.flickr.com/photos/e_phots/3012896283/
- Stormtroopers with Linux http://www.flickr.com/photos/kalexanderson/6261634332/in/photostream/

Legal

- Copyright Kim Moir 2012. All rights reserved. This presentation and source code are available under the Creative Commons Att. Nc Cd 3.0 License
- Eclipse and the Eclipse logo are trademarks of the Eclipse Foundation, Inc.
- Apache and the Apache logo are trademarks of the Apache Software Foundation
- Linux and the Linux foundation logo are trademarks of the Linux foundation
- Mozilla, Firefox, Thunderbird and their associated logos are trademarks of the Mozilla foundation

References

- Pro Git book http://progit.org/
- The Git parable http://tom.preston-werner.com/2009/05/19/the-git-parable.html
- Girls Develop IT NYC Version Control with Git https://github.com/GDIAdmin/Git-Introduction
- Git Hub help http://help.github.com
- The Designer's Guide to Git or: How I Learned to Stop Worrying and Love the Repository http://www.sitepoint.com/the-designers-guide-to-git-or-how-i-learned-to-stop-worrying-and-love-the-repository/
- Git: The Lean, Mean, Distributed Machine http://www.slideshare.net/err/git-machine
- You won't get fired for Using Apache http://redmonk.com/sogrady/2011/11/28/you-wont-get-fired-for-using-apache/
- The Rise of Git http://www.infoworld.com/d/application-development/torvaldss-git-the-it-technology-software-version-control-1`67799
- Lars Vogel's Git tutorial http://www.vogella.de/articles/Git/article.html