

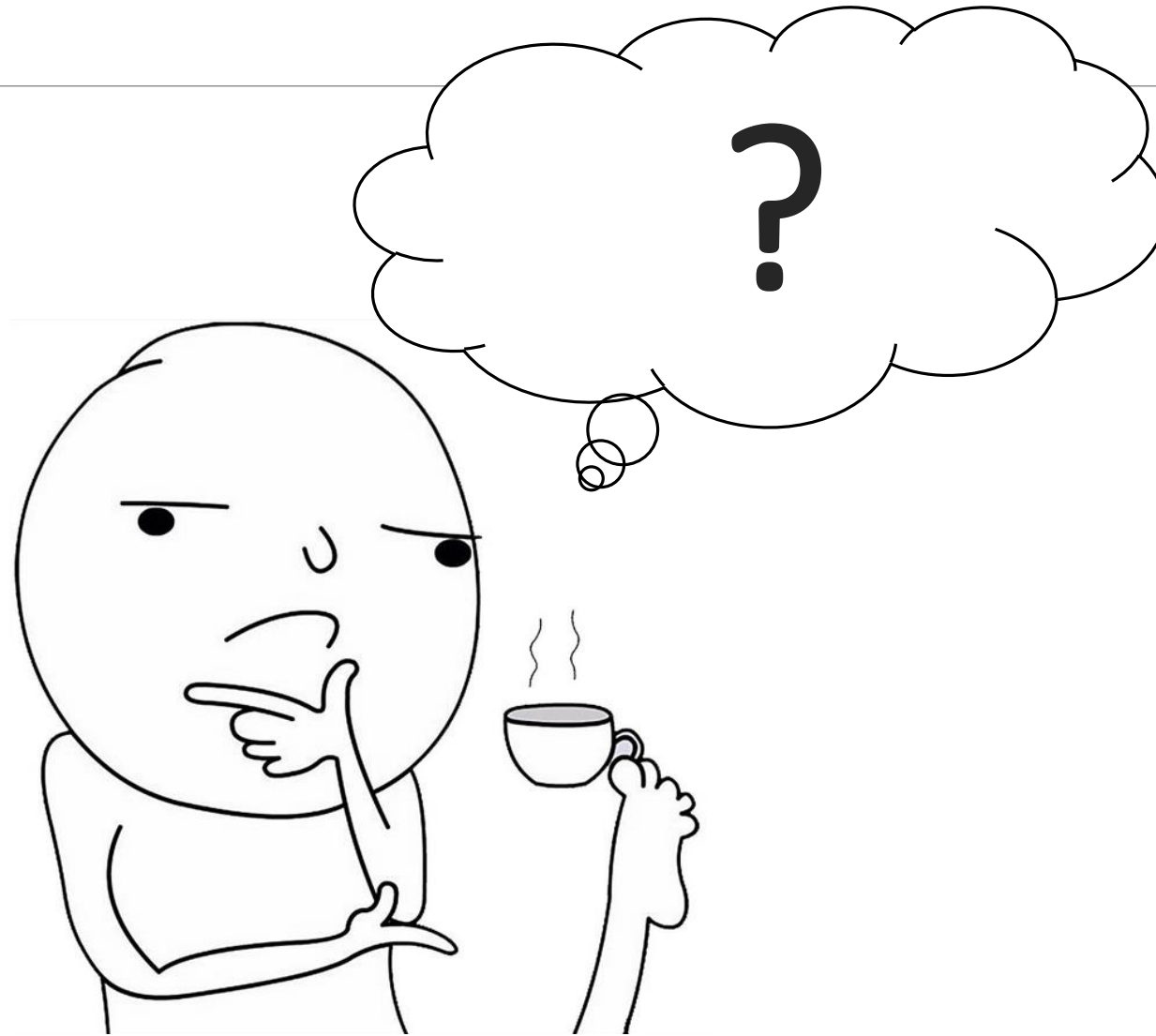
Introduction to



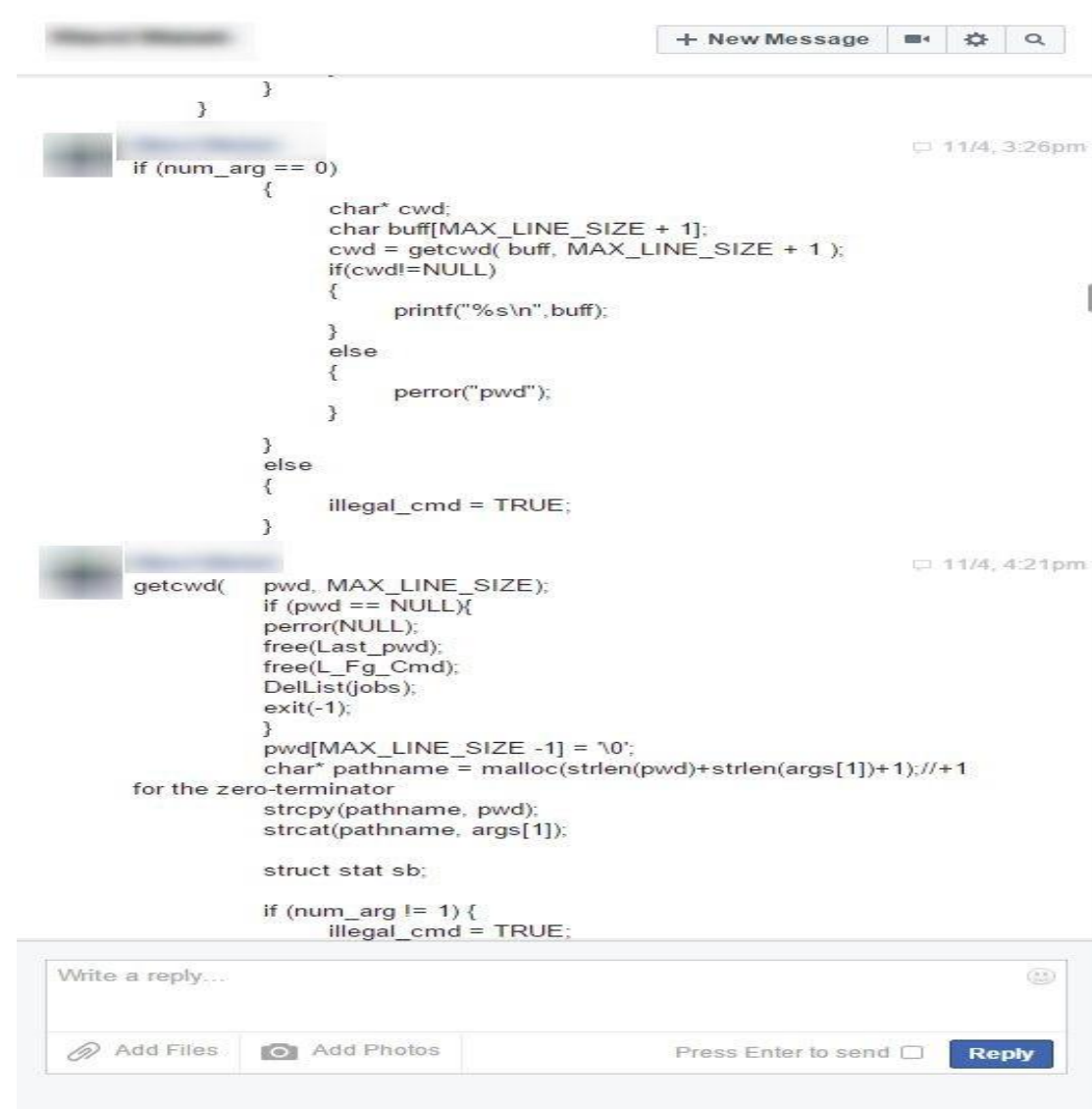
MD HASIBUL HUQ

mdhasibul.huq@mail.concordia.ca

Why git is useful?



Use cases



Use cases



matam_hw_3 - Final 2



matam_hw_3 - Final 3



matam_hw_3 - v4 -Ready for submission



matam_hw_3 - submit



matam_hw_3 - Final 3 - no leaks



matam_hw_3 - Final



matam_hw_3 - v4



matam_hw_3 - submit - Copy



matam_hw_3 - Final 3.3- enhanced prin...

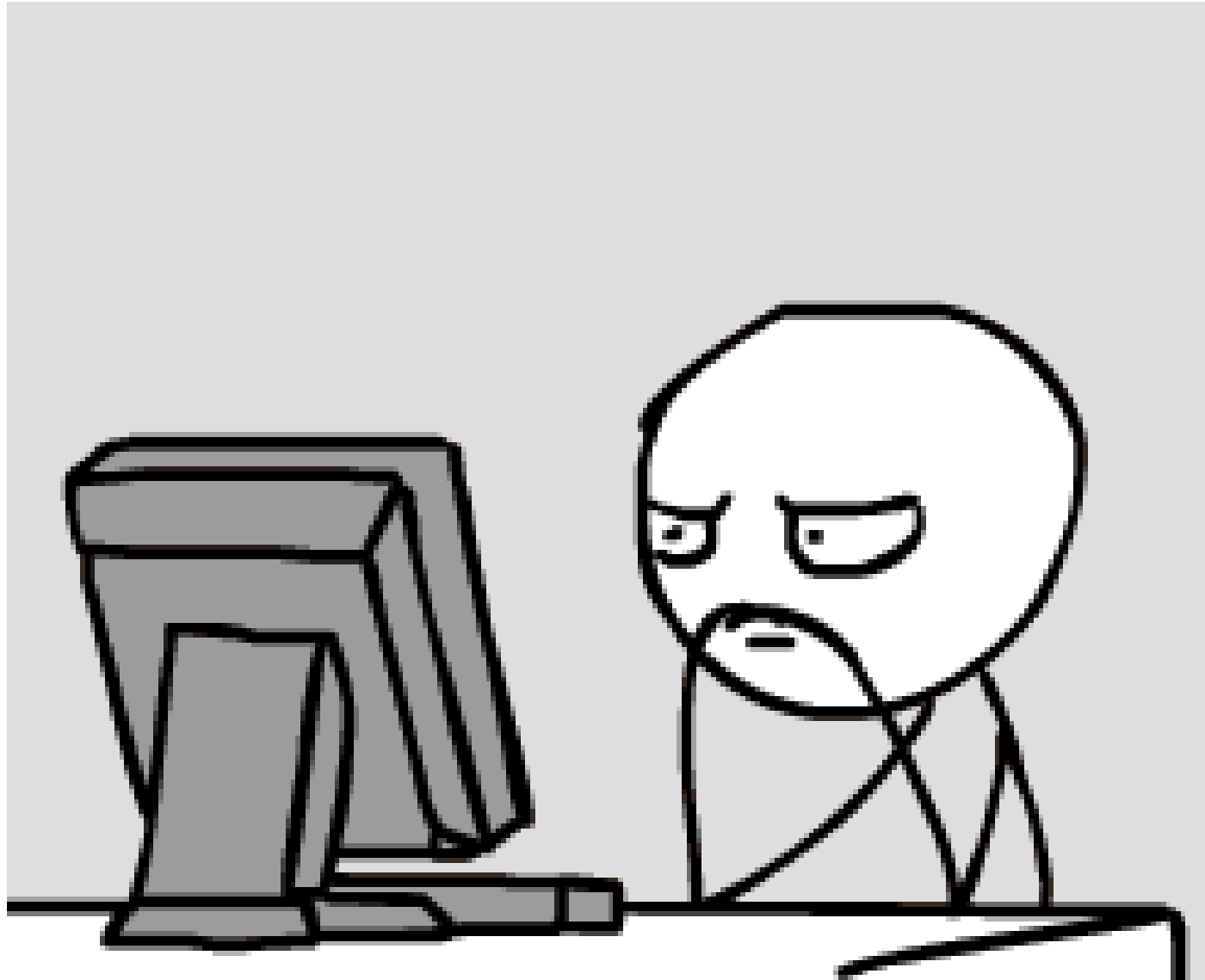


matam_hw_3 - v4 -Fixed typos

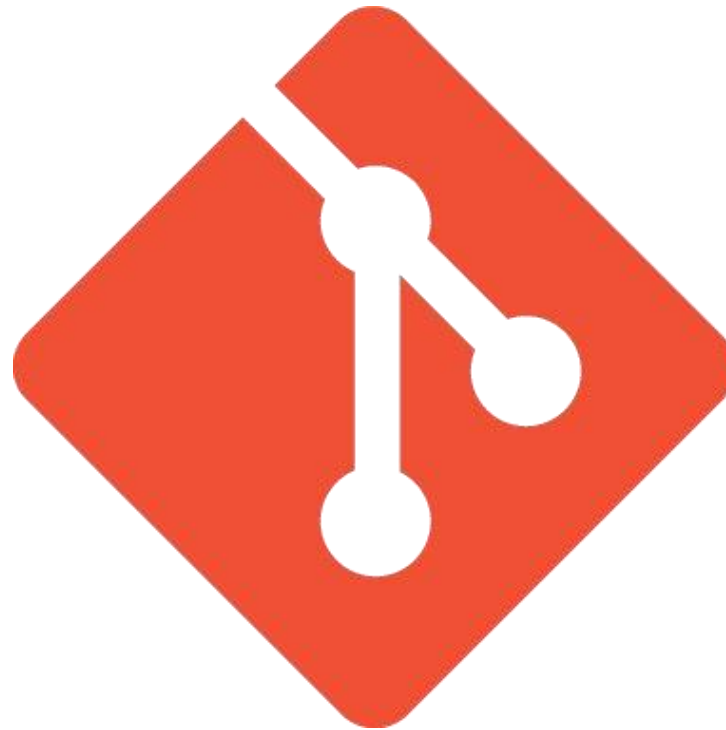


matam_hw_3

Now you realize that **CTRL + Z** won't solve your problems



What is git?



What is git?

- Open source project originally developed in 2005 by Linus Torvalds
- A command line utility
- You can imagine **git** as something that sits on top of your file system and manipulates files.
- A distributed version control system - **DCVS**



What is “distributed version control system” ?

Version control system is a system that records changes to a file or set of files over time so that you can recall specific versions later

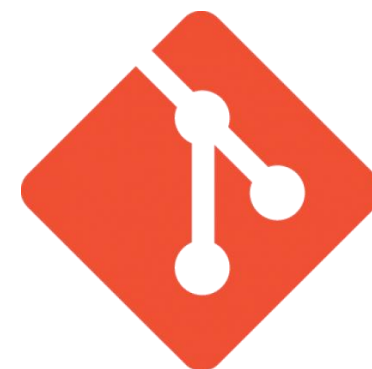
Distributed means that there is no main server and all of the full history of the project is available once you cloned the project.

A brief history

- In 2002, the Linux kernel project began using a **DVCS** called BitKeeper
- In 2005, the commercial company that developed BitKeeper broke down, and the tool's free-of-charge status was revoked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 - **git**

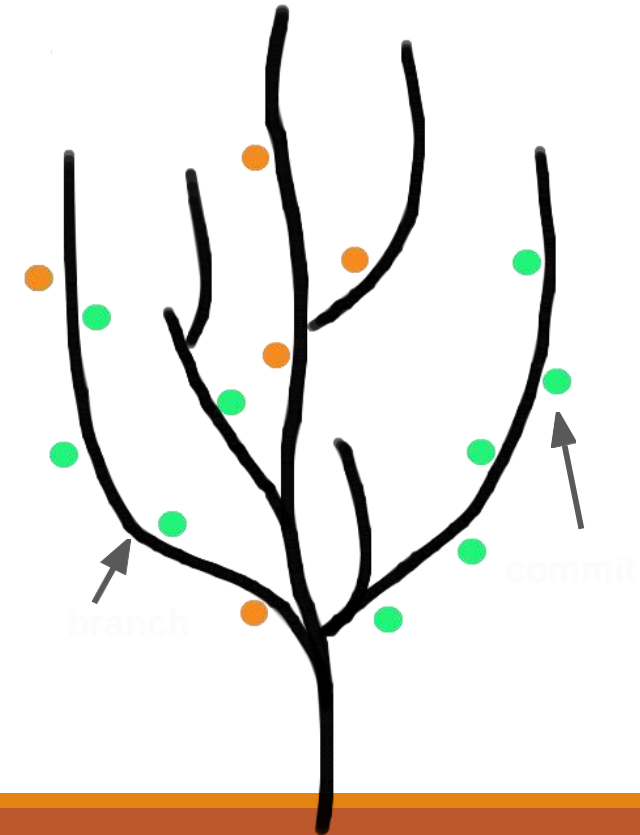
About Git

- Created by Linus Torvalds, creator of Linux, in 2005
 - Came out of Linux development community
 - Designed to do version control on Linux kernel
- Goals of Git:
 - Speed
 - Support for non-linear development (thousands of parallel branches)
 - Fully distributed
 - Able to handle large projects efficiently
 - *(A "git" is a cranky old man. Linus meant himself.)*

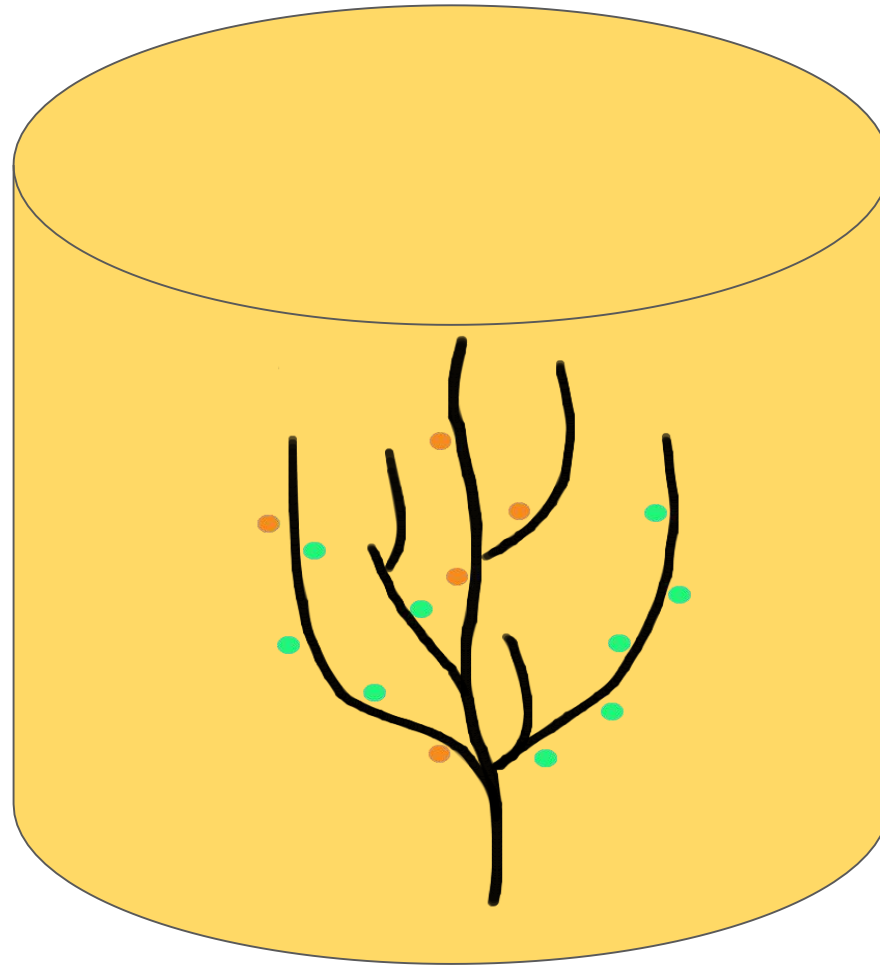


git

- You can imagine **git** as something that sits on top of your file system and manipulates files.
- This “something” is a **tree** structure where each **commit** creates a new node in that tree.
- Nearly all **git** commands actually serve to navigate on this tree and to manipulate it accordingly.

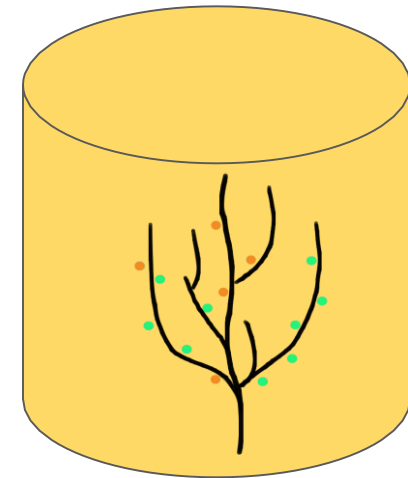


git repository



git repository

- The purpose of **git** is to manage a project, or a set of files, as they change over time. **Git** stores this information in a data structure called a **repository**
- A **git repository** contains, mainly:
 - A set of **commits**



Commit

- A **commit** object mainly contains three things:
 - A set of **changes** the **commit** introduces
 - **Commit message** describing the changes
 - A **hash**, a 40-character string that uniquely identifies the commit object

Commit

```
commit 0c7c3fe66f4cc43f875be2fb4e5fde5f27fcfb86
```

Commit id (hash)

```
Author: Sameeh Jubran <sameeh@daynix.com>
```

```
Date: Thu Feb 18 11:55:36 2016 +0200
```

```
Fixed a typo.
```

Commit message

```
Signed-off-by: Sameeh Jubran <sameeh@daynix.com>
```

```
diff --git a/guest_tools/KitAutosetup/KitSetup.sh b/guest_tools/KitAutosetup/KitSetup.sh
index 1e41969..89ef9c5 100755
```

```
--- a/guest_tools/KitAutosetup/KitSetup.sh
```

```
+++ b/guest_tools/KitAutosetup/KitSetup.sh
```

```
@@ -4,7 +4,7 @@ SCRIPTS_DIR=`dirname $0`
```

```
##### Settings #####
```

The change the commit introduces

```
-#Frequwntly changed
```

```
+#Frequently changed
```

```
cl1Name='CL1-2012R2X64'
```

```
cl2Name='CL2-2012R2X64'
```

Installing/learning Git

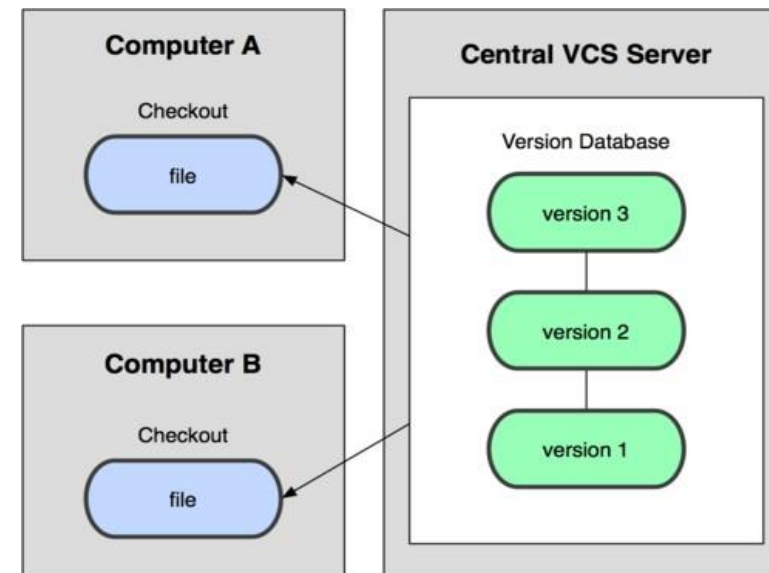
- Git website: <http://git-scm.com/>
 - Free on-line book: <http://git-scm.com/book>
 - Reference page for Git: <http://gitref.org/index.html>
 - Git tutorial: <http://schacon.github.com/git/gittutorial.html>
 - Git for Computer Scientists:
 - <http://eagain.net/articles/git-for-computer-scientists/>
- At command line: (*where verb = config, add, commit, etc.*)
 - `git help verb`

Git Tutorials

- Git tutorial: <https://www.youtube.com/watch?v=USjZcfj8yxE>
 - notes : [Introduction to Git \(notion.so\)](#)
- Github tutorial: <https://www.youtube.com/watch?v=nhNq2kIvi9s>
 - notes : [Introduction to GitHub \(notion.so\)](#)
- Git Crash Course: https://www.youtube.com/watch?v=SWYqp7iY_Tc
- Git drag and drop: <https://youtu.be/xmK1Q5uzH4wAnd>
- Download Git: <https://git-scm.com/downloads>
- Check Github: <https://github.com/>

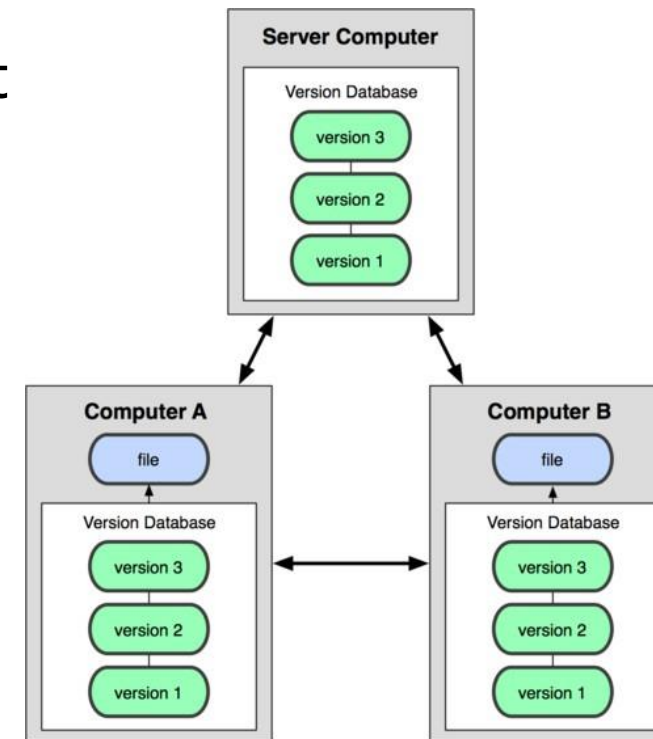
Centralized VCS

- In Subversion, CVS, Perforce, etc.
A central server repository (repo) holds the "official copy" of the code
 - the server maintains the sole version history of the repo
- You make "checkouts" of it to your local copy
 - you make local modifications
 - your changes are not versioned
- When you're done, you "check in" back to the server
 - your checkin increments the repo's version



Distributed VCS (Git)

- In git, mercurial, etc., you don't "checkout" from a central repo
 - you "clone" it and "pull" changes from it
- Your local repo is a complete copy of everything on the remote server
 - yours is "just as good" as theirs
- Many operations are local:
 - check in/out from *local* repo
 - commit changes to *local* repo
 - local repo keeps version history
- When you're ready, you can "push" changes back to server

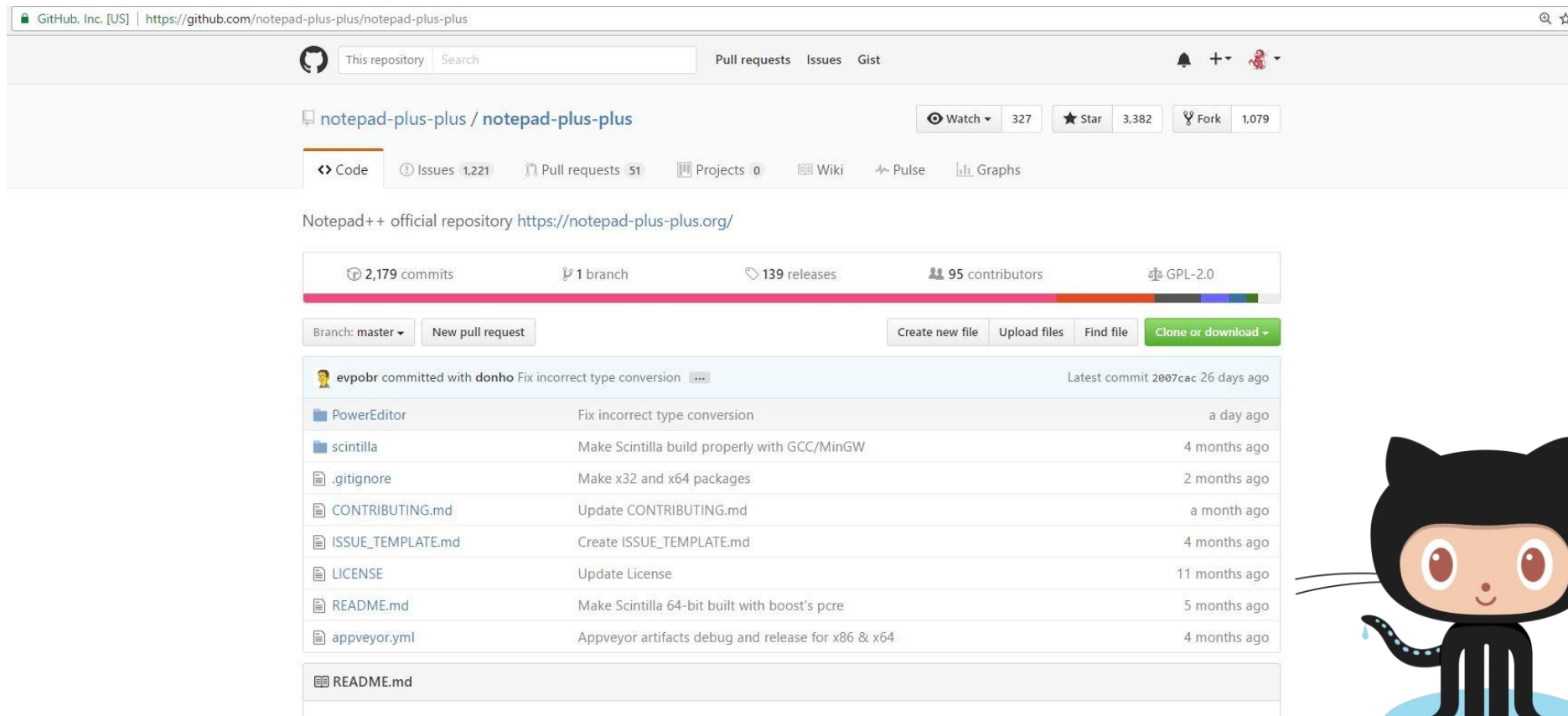


Github



Github

- GitHub is a web-based Git repository hosting service



The screenshot shows the GitHub repository page for `notepad-plus-plus / notepad-plus-plus`. The repository has 327 watchers, 3,382 stars, and 1,079 forks. It is the official repository for Notepad++, with a link to <https://notepad-plus-plus.org/>. The repository statistics show 2,179 commits, 1 branch, 139 releases, 95 contributors, and the GPL-2.0 license. The commit history table lists recent changes:

Commit	Message	Time
evpobr committed with donho	Fix incorrect type conversion	a day ago
scintilla	Make Scintilla build properly with GCC/MinGW	4 months ago
.gitignore	Make x32 and x64 packages	2 months ago
CONTRIBUTING.md	Update CONTRIBUTING.md	a month ago
ISSUE_TEMPLATE.md	Create ISSUE_TEMPLATE.md	4 months ago
LICENSE	Update License	11 months ago
README.md	Make Scintilla 64-bit built with boost's pcre	5 months ago
appveyor.yml	Appveyor artifacts debug and release for x86 & x64	4 months ago

The README.md file is also visible at the bottom of the list.



Initial Git configuration

- Set the name and email for Git to use when you commit:
 - `git config --global user.name "Bugs Bunny"`
 - `git config --global user.emai bugs@gmail.com`
 - You can call `git config -l` to verify these are set.
- Set the editor that is used for writing commit messages:
 - `git config --global core.editor nano`
 - (it is vim by default)

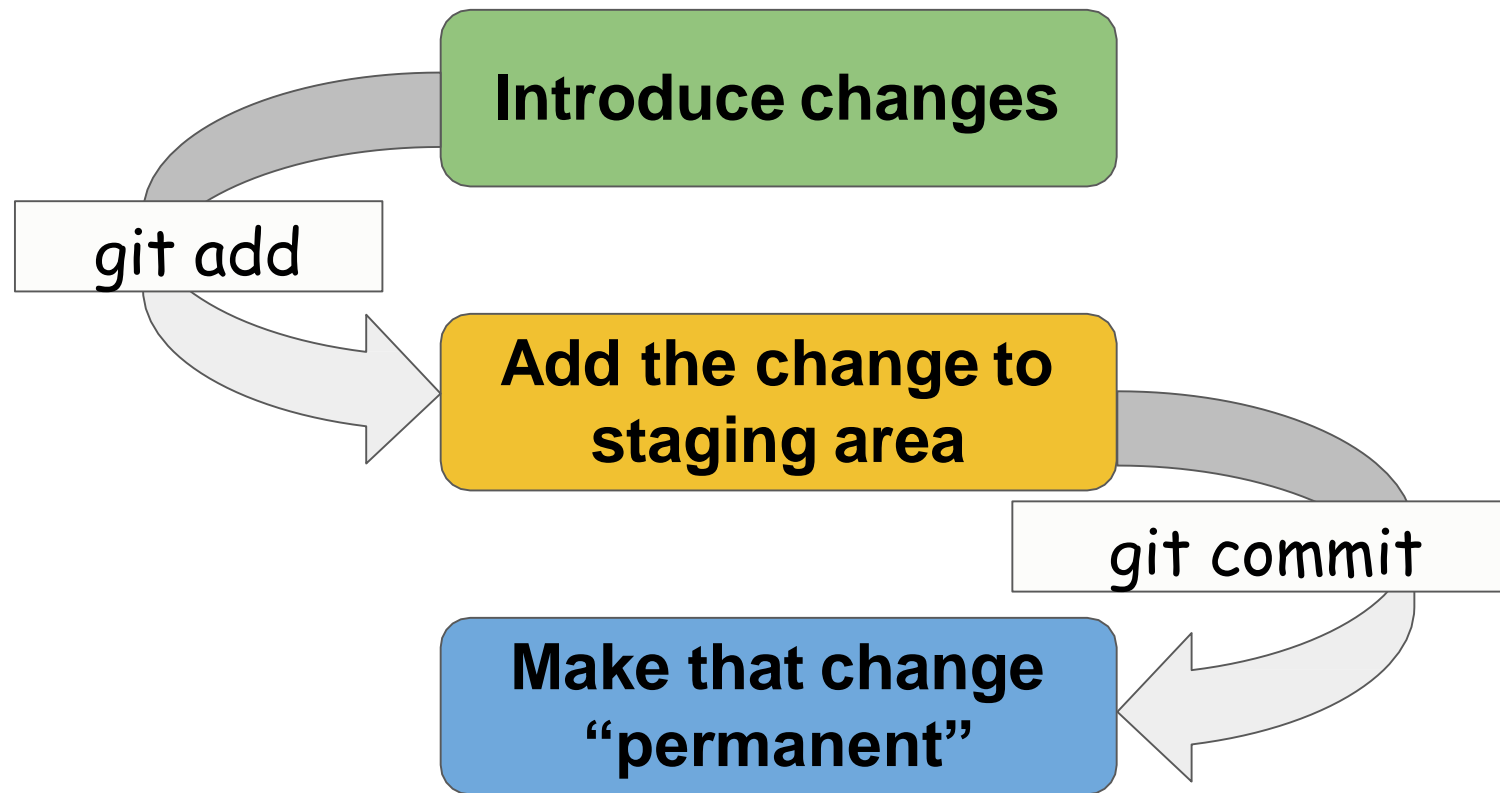
Creating a Git repo

Two common scenarios: (only do one of these)

- To create a new **local Git repo** in your current directory:
 - `git init`
 - This will create a `.git` directory in your current directory.
 - Then you can commit files in that directory into the repo.
 - `git add filename`
 - `git commit -m "commit message"`
- To **clone a remote repo** to your current directory:
 - `git clone url localDirectoryName`
 - This will create the given local directory, containing a working copy of the files from the repo, and a `.git` directory (used to hold the staging area and your actual local repo)

git workflow

How commits are created?

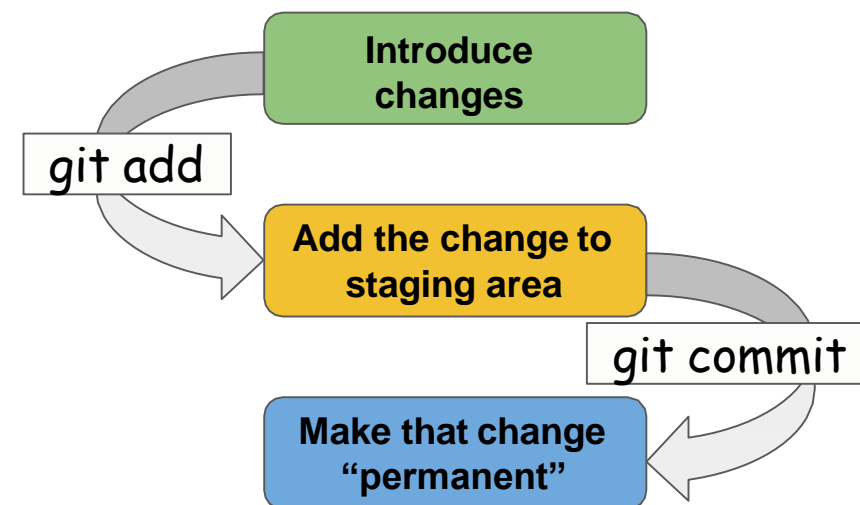


The three steps of git

Introduce a change: introduce a change to a file that is being tracked by git

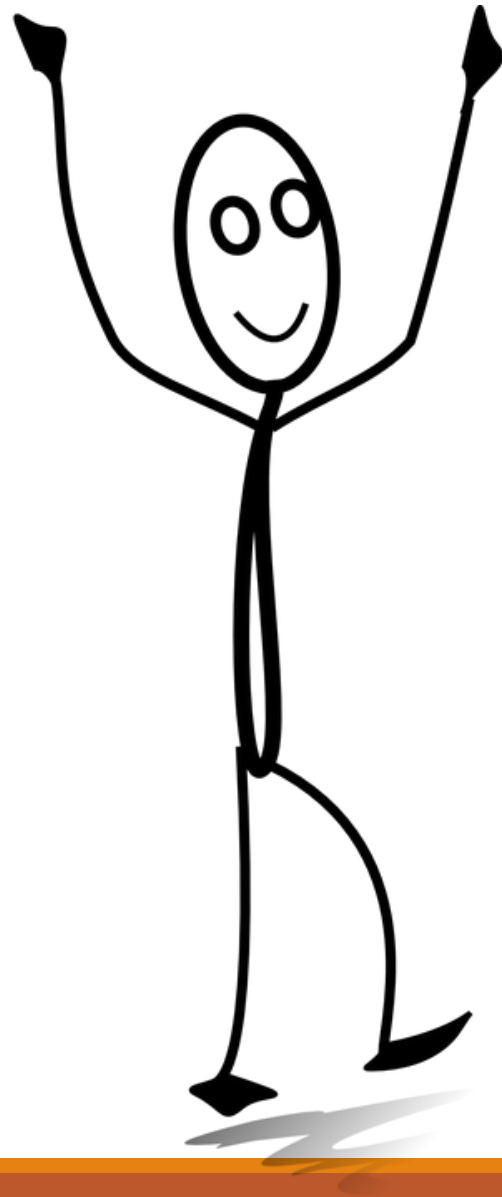
Add the actual change to staging area: Add the change you actually want using “git add”

Commit: Commit the change that has been added using git commit

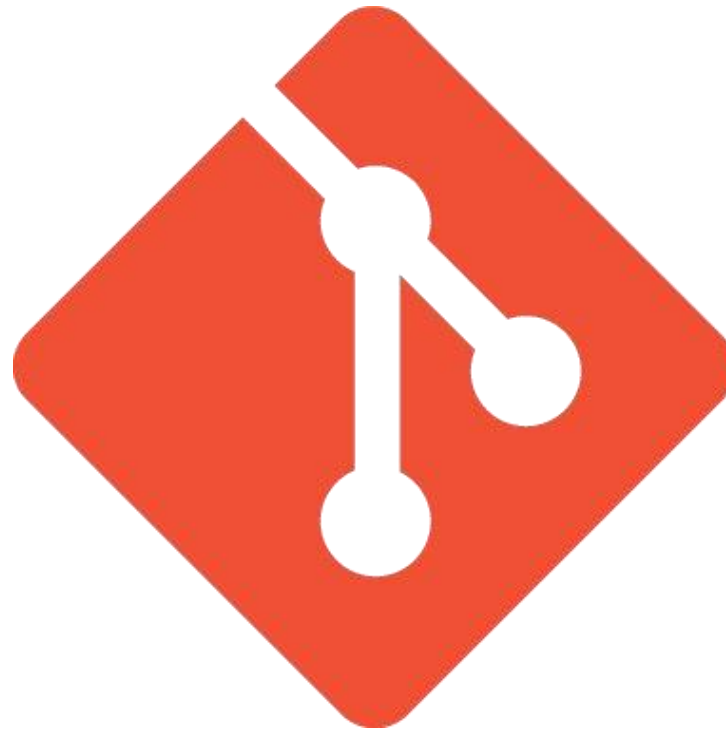


Let's Start with **git** GUI

Let's create a commit

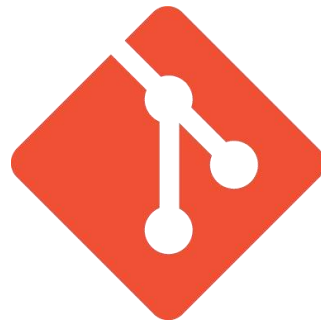


git commands



git commands

- For most of the basic interactions with git you'll mainly use **7 commands** that we'll cover here



git commands

- git init
- git clone
- git log
- git diff
- git status
- git add
- git commit

git init

- Creates a new **git repository**
- Can be used to convert an existing, unversioned project to a **git repository** or initialize a new **empty repository**

git clone

- Copies an existing **git repository**

Shows the **commit** logs

```
sameeh@bark:~/Builds/VirtHCK$ git log
commit 6f3d6e0e6db1c98716deb29040ef516a9d7f38dd
Author: Bishara AbuHattoum <bishara@daynix.com>
Date: Tue Aug 2 14:00:52 2016 +0300

    Add configuration for file system filter drivers

    Signed-off-by: Bishara AbuHattoum <bishara@daynix.com>
    Signed-off-by: Dmitry Fleytman <dmitry@daynix.com>

commit bad56e1b8f48faf02753d6083641f967a07b0a2a
Author: Bishara AbuHattoum <bishara@daynix.com>
Date: Tue Aug 2 14:00:53 2016 +0300

    Extend size of storage test images to 30G

    Accrding to requirements of the latest HLK.

    Signed-off-by: Bishara AbuHattoum <bishara@daynix.com>
    Signed-off-by: Dmitry Fleytman <dmitry@daynix.com>

commit 3acb92c3fd0c0058aeb9fbf2e457a69237f90314
Author: Bishara AbuHattoum <bishara@daynix.com>
Date: Tue Aug 2 13:46:11 2016 +0300

    Adding the auto partitioning script that prepares clients for storage tests.

    Partitioning is done accrding to:
    https://msdn.microsoft.com/en-us/library/windows/hardware/jj125194(v=vs.85).aspx

    Signed-off-by: Bishara AbuHattoum <bishara@daynix.com>
    Signed-off-by: Dmitry Fleytman <dmitry@daynix.com>

commit 19c98a231141efe18da987464d269585af2a0513
Author: Denis Gersten <denisg@daynix.com>
Date: Tue Jul 26 15:03:52 2016 +0300

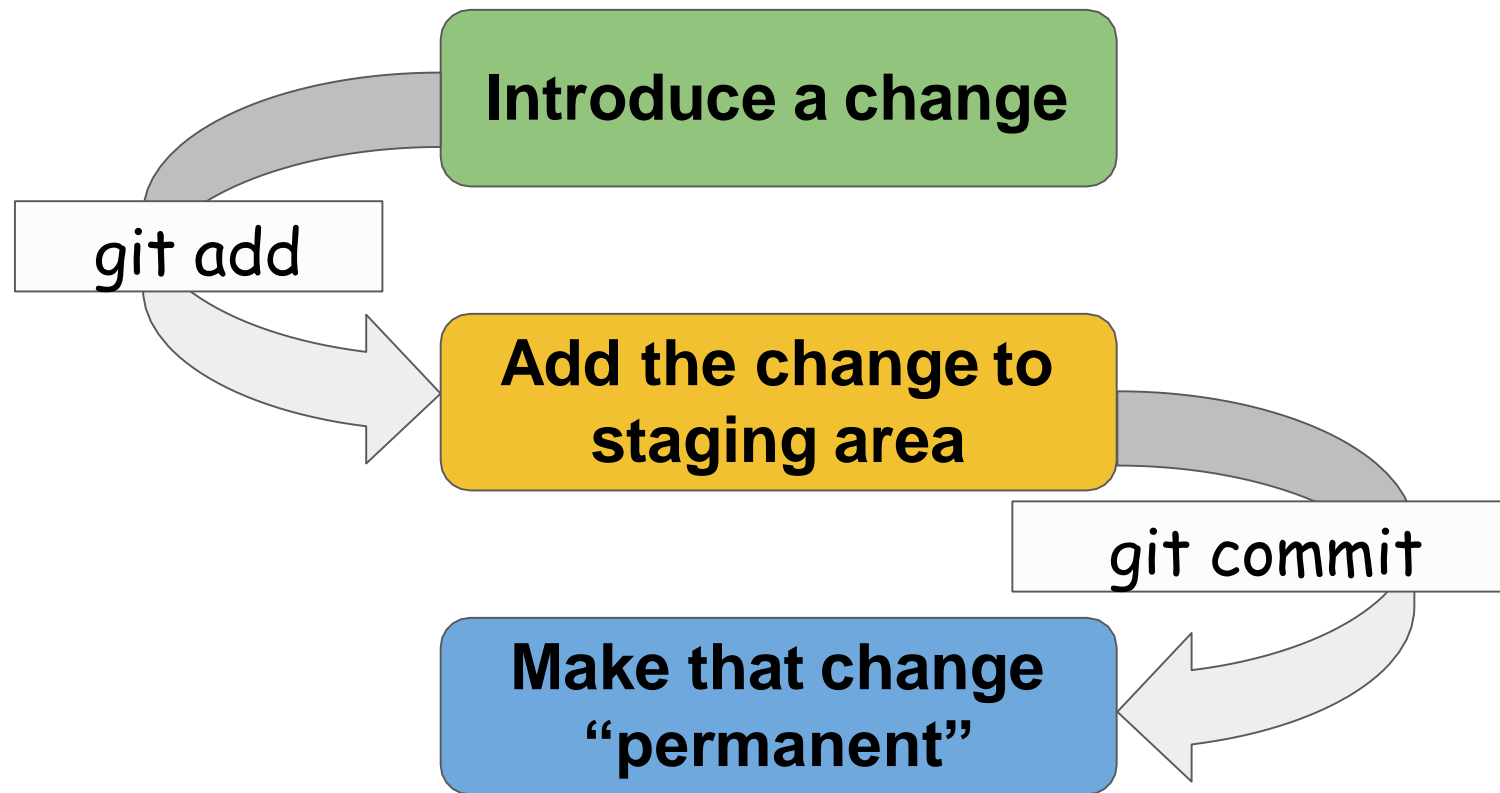
    VirtHCK: Attach 2 drives for USB storage tests

    The latest HCK tests for USB devices require 2
    test devices attached.

    Signed-off-by: Denis Gersten <denisg@daynix.com>
    Signed-off-by: Dmitry Fleytman <dmitry@daynix.com>
```

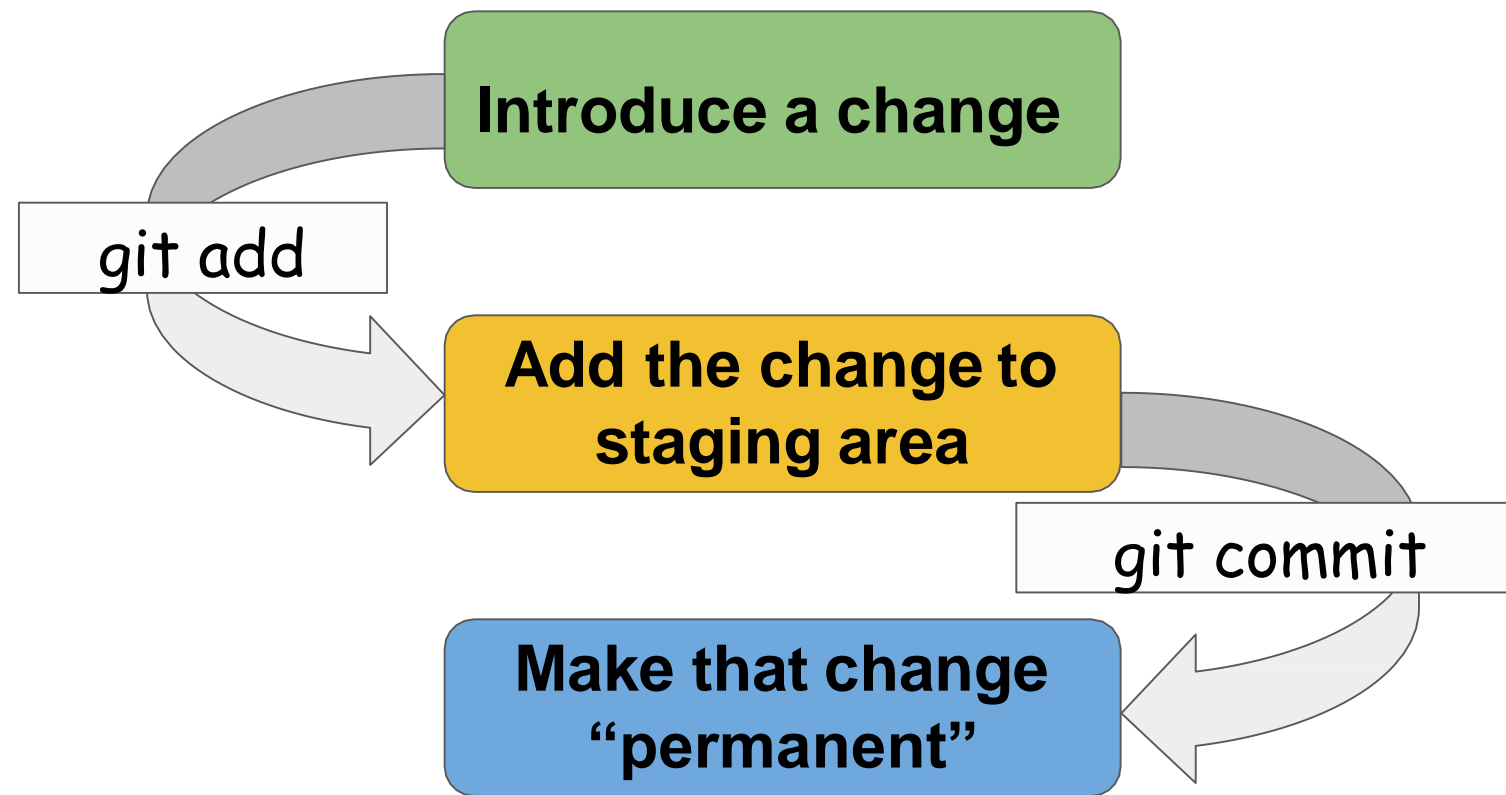
git add

- Adds changes



git commit

- Creates a **commit** out of the changes that had been added



git diff

- Displays the change that was introduced

Useful flag:

- `--cached`:
Displays the change that was added using “git add”

git status

- Displays the file names that has been modified, added and untracked

Bonus command: `git checkout`

- Checking out a commit makes the entire working directory match that commit

Credits

1. **Yan Vugenfirer** – CEO - daynix.com – for this amazing Slide content
2. **Learn more about GIT :University of Washington Open Course material.**
<https://courses.cs.washington.edu/courses/cse403/13au/lectures/git.ppt.pdf>
3. Git install : <https://git-scm.com/downloads>
4. GIT GUI for windows: https://matthew-brett.github.io/pydagogue/git_gui_windows.html
5. Useful tools:
 1. Best Cross platform git tool: [GitKraken](#)

Git Cheat Sheet

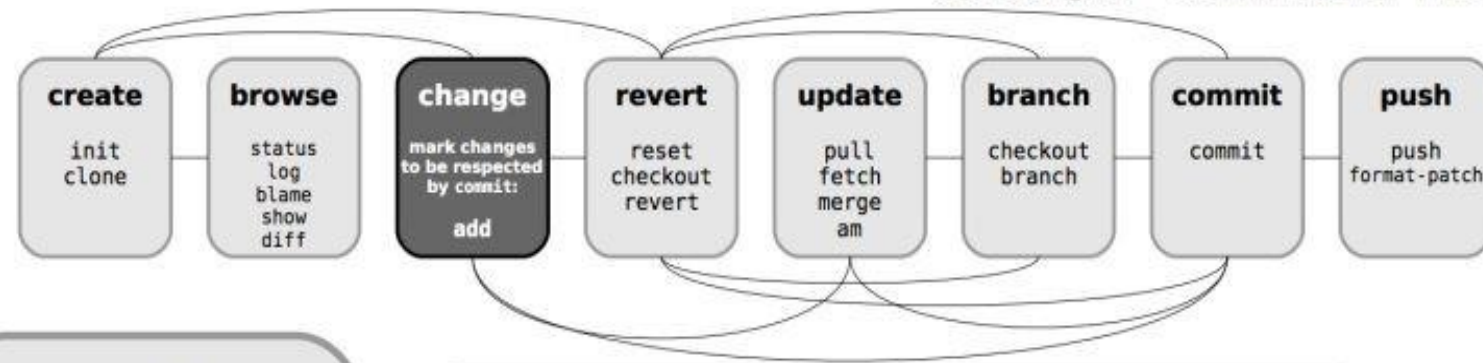
by Jan Krüger <jk@jk.gs>, <http://jan-krueger.net/git/>
Based on work by Zack Rusin

Basics

Use `git help [command]` if you're stuck.

master	default devel branch
origin	default upstream branch
HEAD	current branch
HEAD^	parent of HEAD
HEAD~4	great-great grandparent of HEAD
foo..bar	from branch <code>foo</code> to branch <code>bar</code>

(left to right) Command Flow



Create

From existing files

```
git init
git add .
```

From existing repository

```
git clone ~/old ~/new
git clone git://...
git clone ssh://...
```

Publish

In Git, `commit` only respects changes that have been marked explicitly with `add`.

```
git commit [-a]
(-a: add changed files automatically)
git format-patch origin
(create set of diffs)
git push remote
(push to origin or remote)
git tag foo
(mark current version)
```

Useful Tools

```
git archive
Create release tarball
git bisect
Binary search for defects
git cherry-pick
Take single commit from elsewhere
git fsck
Check tree
git gc
Compress metadata (performance)
git rebase
Forward-port local changes to remote branch
git remote add URL
Register a new remote repository for this tree
git stash
Temporarily set aside changes
git tag
(there's more to it)
gitk
Tk GUI for Git
```

Tracking Files

```
git add files
git mv old new
git rm files
git rm --cached files
(stop tracking but keep files in working dir)
```

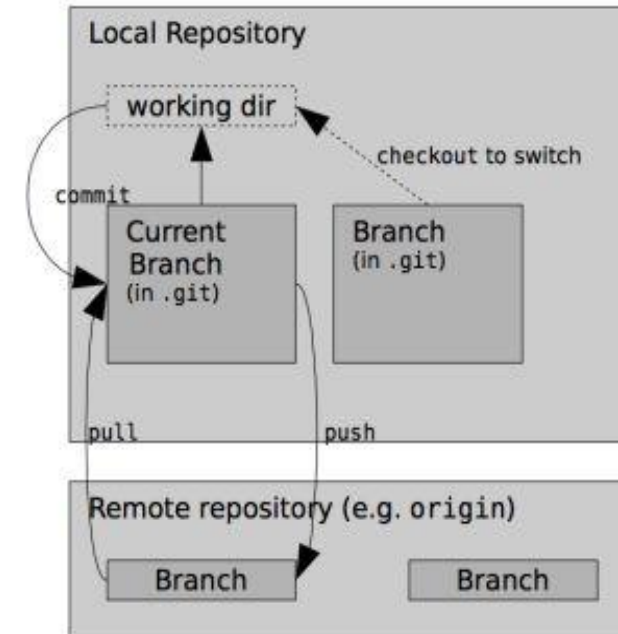
View

```
git status
git diff [oldid newid]
git log [-p] [file|dir]
git blame file
git show id (meta data + diff)
git show id:file
git branch (shows list, * = current)
git tag -l (shows list)
```

Update

```
git fetch (from def. upstream)
git fetch remote
git pull (= fetch & merge)
git am -3 patch.mbox
git apply patch.diff
```

Structure Overview



Revert

In Git, `revert` usually describes a new commit that undoes previous commits.

```
git reset --hard (NO UNDO)
(reset to last commit)
git revert branch
git commit -a --amend
(replaces prev. commit)
git checkout id file
```

Branch

```
git checkout branch
(Switch working dir to branch)
git merge branch
(merge into current)
git branch branch
(branch current)
git checkout -b new other
(branch new from other and switch to it)
```

Conflicts

Use `add` to mark files as resolved.

```
git diff [--base]
git diff --ours
git diff --theirs
git log --merge
gitk --merge
```