Introduction to

Sameeh Jubran, Sameeh@daynix.com

Why git is useful?

Use cases

+ New Message

```
                }
        }
```

11/4, 3:26pm

```
if (num_arg == 0)
                {
                        char* cwd;
                        char buff[MAX_LINE_SIZE + 1];
                        cwd = getcwd( buff, MAX_LINE_SIZE + 1 );
                        if(cwd!=NULL)
                        {
                                printf("%s\n",buff);
                        }
                        else
                        {
                                perror("pwd");
                        }

                }
                else
                {
                        illegal_cmd = TRUE;
                }
```

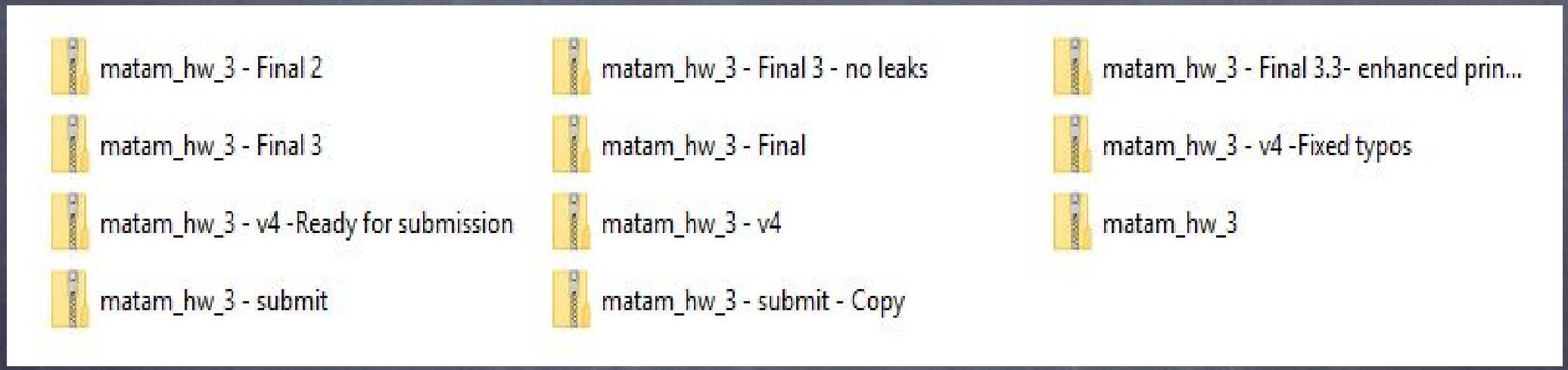
11/4, 4:21pm

```
getcwd(         pwd, MAX_LINE_SIZE);
                if (pwd == NULL){
                perror(NULL);
                free(Last_pwd);
                free(L_Fg_Cmd);
                DelList(jobs);
                exit(-1);
                }
                pwd[MAX_LINE_SIZE -1] = '\0';
                char* pathname = malloc(strlen(pwd)+strlen(args[1])+1);//+1
for the zero-terminator
                strcpy(pathname, pwd);
                strcat(pathname, args[1]);

                struct stat sb;

                if (num_arg != 1) {
                        illegal_cmd = TRUE;
```

Write a reply...

Add Files Add Photos Press Enter to send Reply

Use cases


matam_hw_3 - Final 2
matam_hw_3 - Final 3 - no leaks
matam_hw_3 - Final 3.3- enhanced prin...
matam_hw_3 - Final 3
matam_hw_3 - Final
matam_hw_3 - v4 -Fixed typos
matam_hw_3 - v4 -Ready for submission
matam_hw_3 - v4
matam_hw_3
matam_hw_3 - submit
matam_hw_3 - submit - Copy

Use cases

Consider this scenario:

- You have a homework submission in Matam for today and the assignment is ready for submission
- While testing it you discovered a minor bug and decided to fix it

Use cases

- After attempting to do so, you accidentally changed a working code and got yourself in a big mess
- You no longer remember what was and what wasn’t there
- It is 23:58 PM


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.

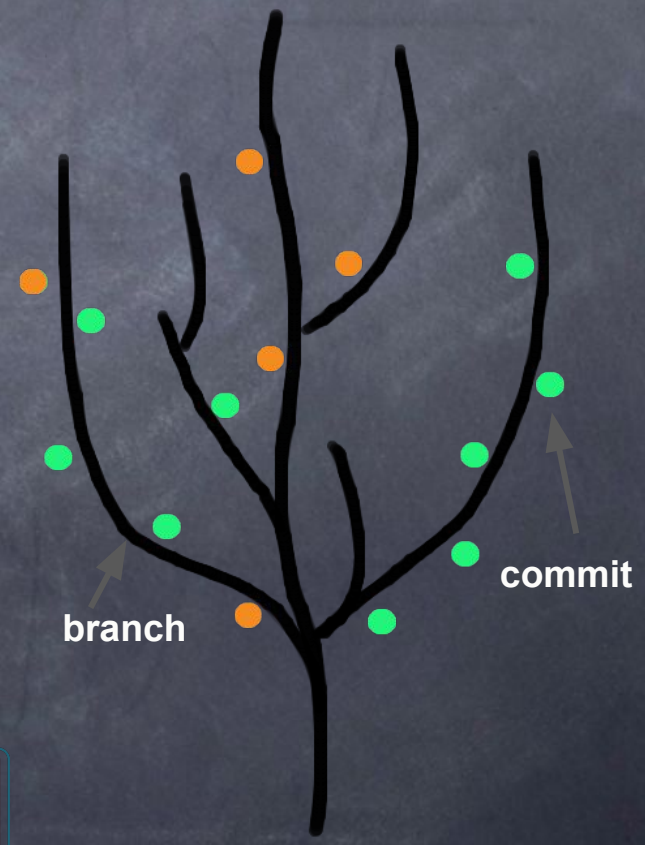
A brief history

- 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**

Demo

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.

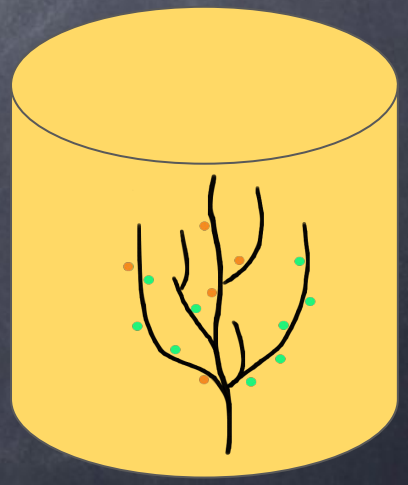


[DAYNIX]

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
Author: Sameeh Jubran <sameeh@daynix.com>
Date:   Thu Feb 18 11:55:36 2016 +0200

    Fixed a typo.

    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 ###################

-#Frequwntly changed
+#Frequently changed
 cl1Name='CL1-2012R2X64'
 cl2Name='CL2-2012R2X64'
```

Commit id (hash)

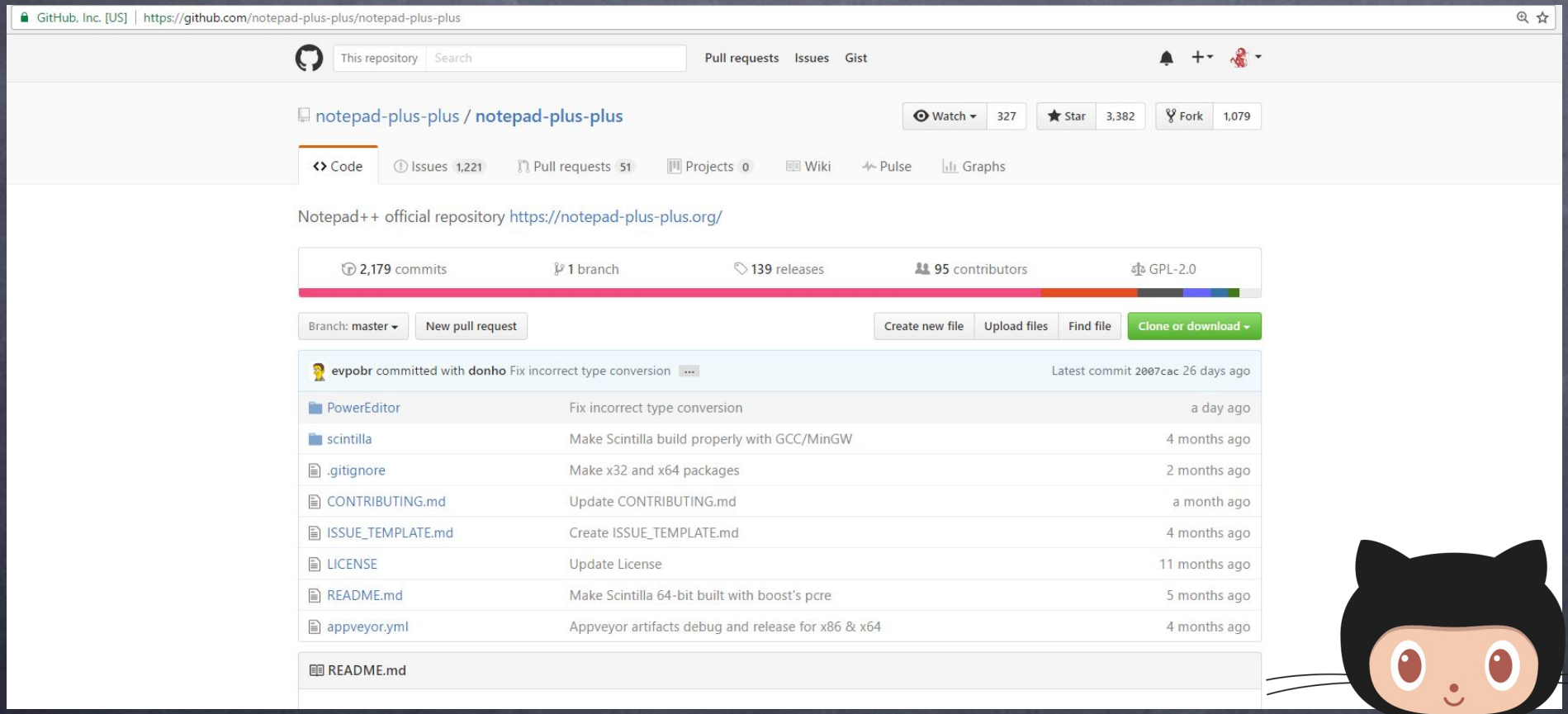
Commit message

The change the commit introduces

Github

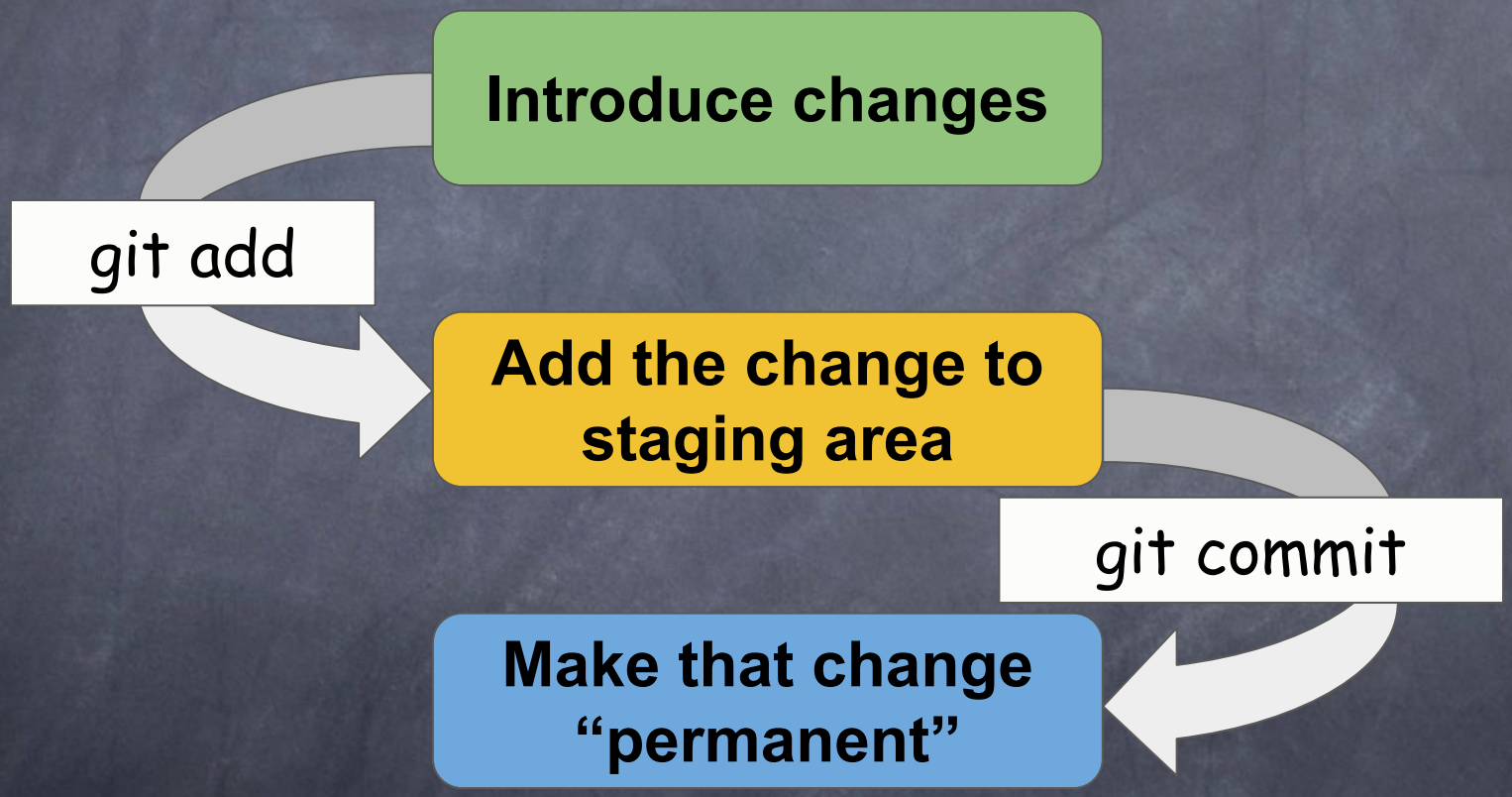
Github

- GitHub is a web-based Git repository hosting service



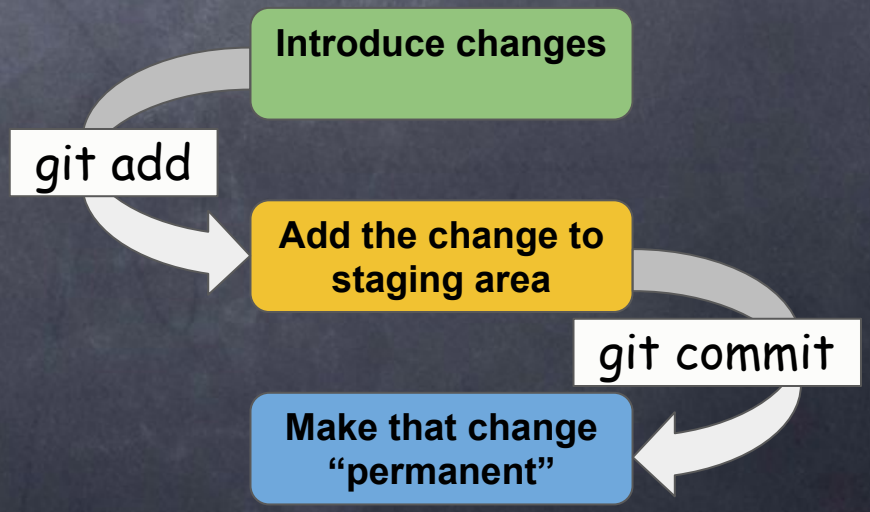
git workflow

How commits are created?

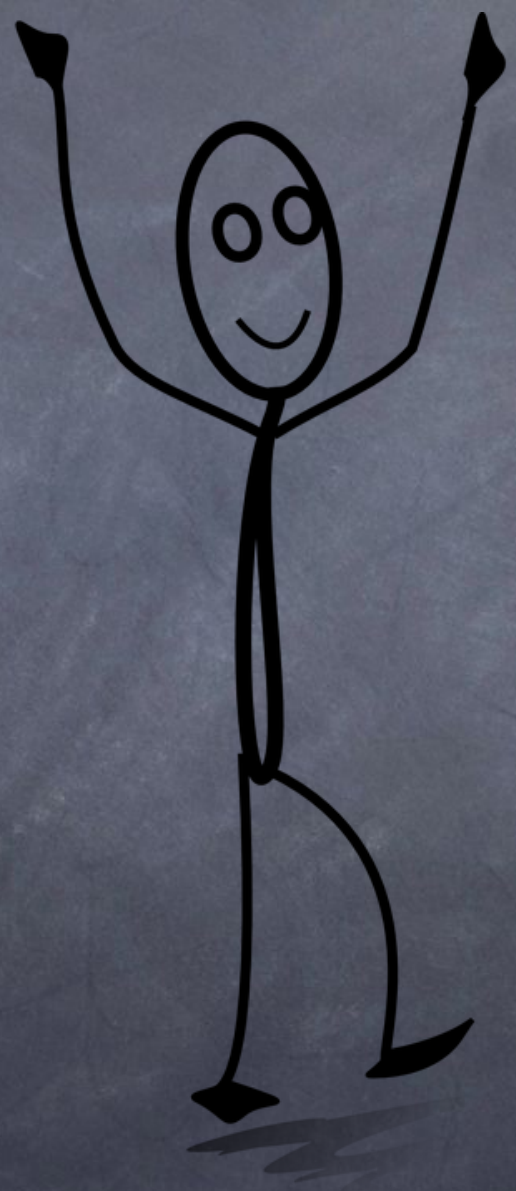


The three steps of git

- **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 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**

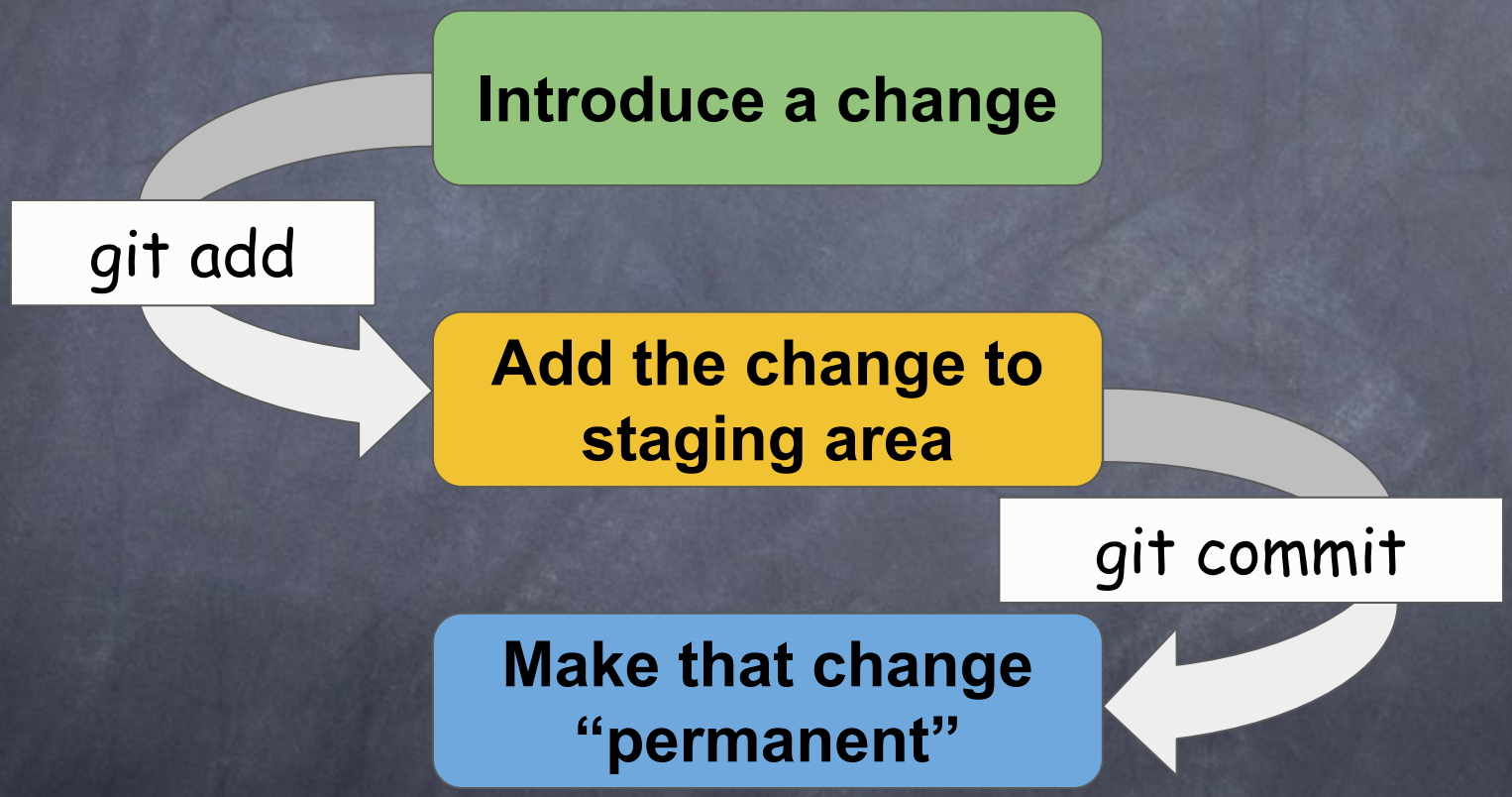
git log

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>
```

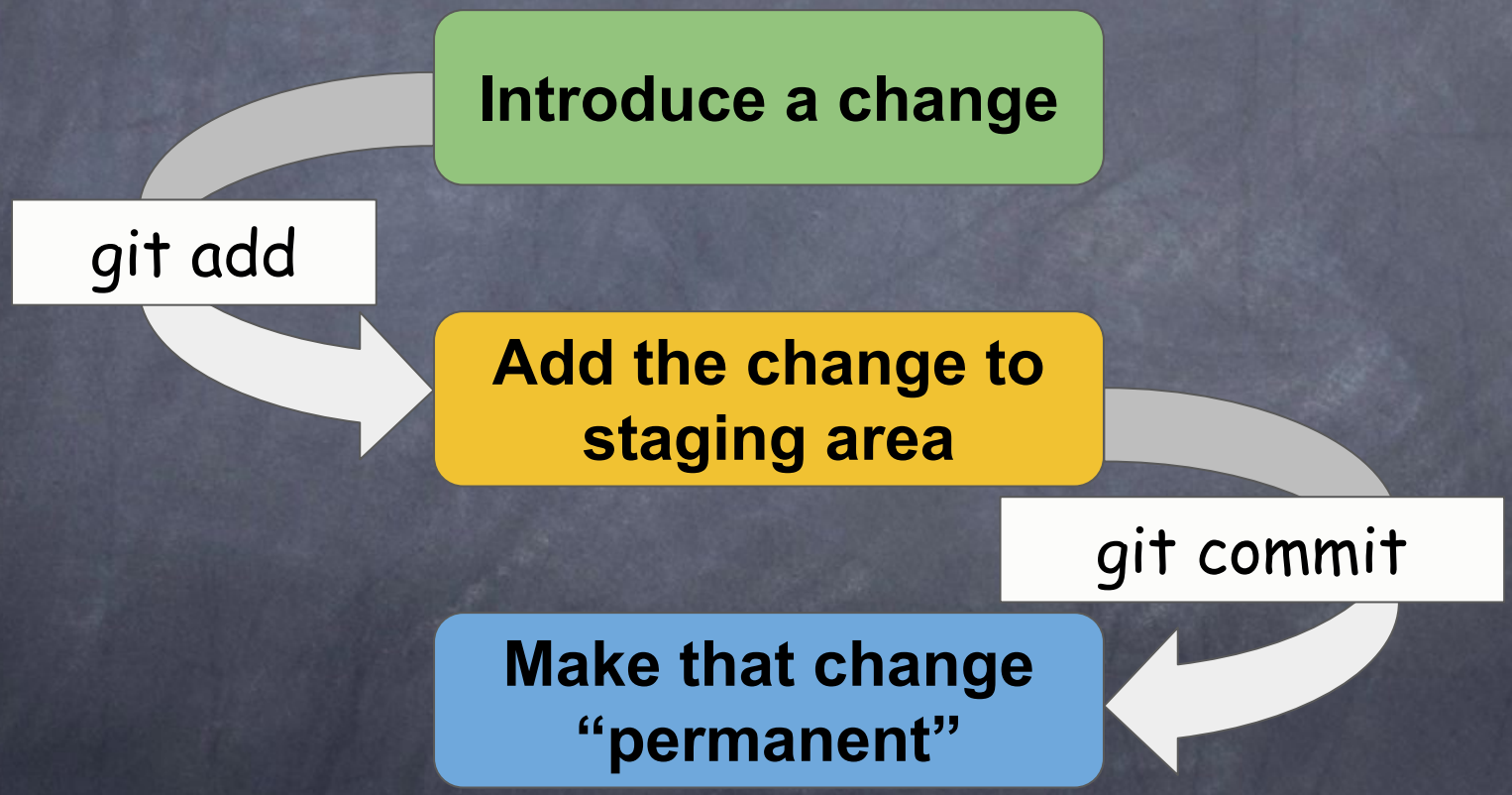
git add

- 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

Q&A

Git Cheat Sheet

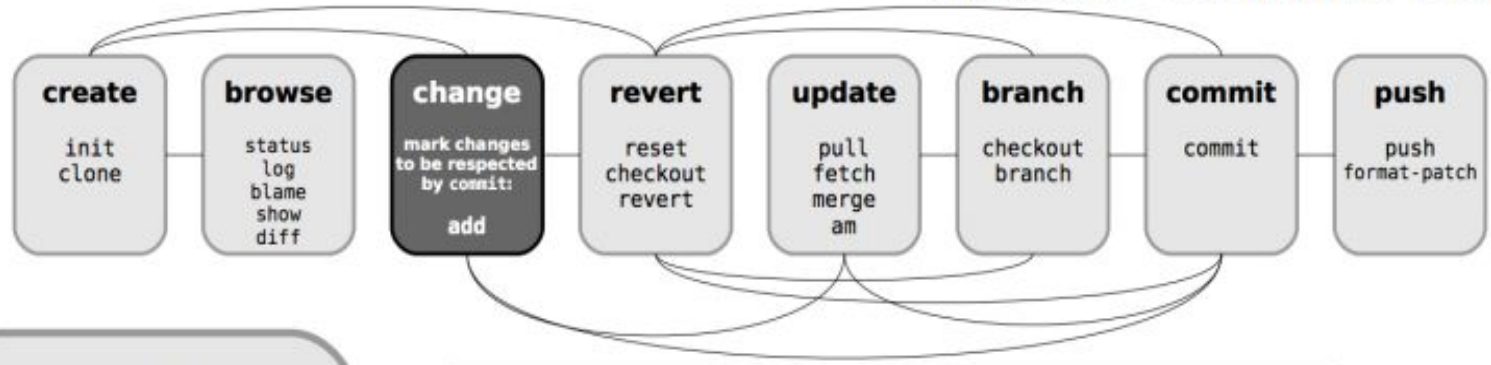
by Jan Krüger <jk@jk.gs>, http://jan-krueger.net/git/
Based on work by Zack Rusin

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 *foo* to branch *bar*

Command Flow (left to right)



Create

From existing files

```
git init
git add .
```

From existing repository

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

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)
```

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
```

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)
```

Update

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

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)
```

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
```

Conflicts

Use add to mark files as resolved.

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

Tracking Files

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

Structure Overview