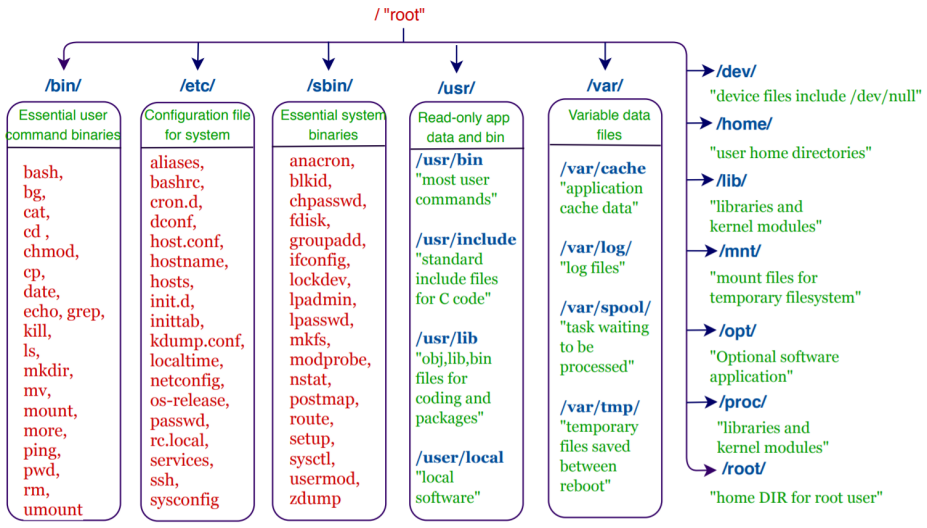
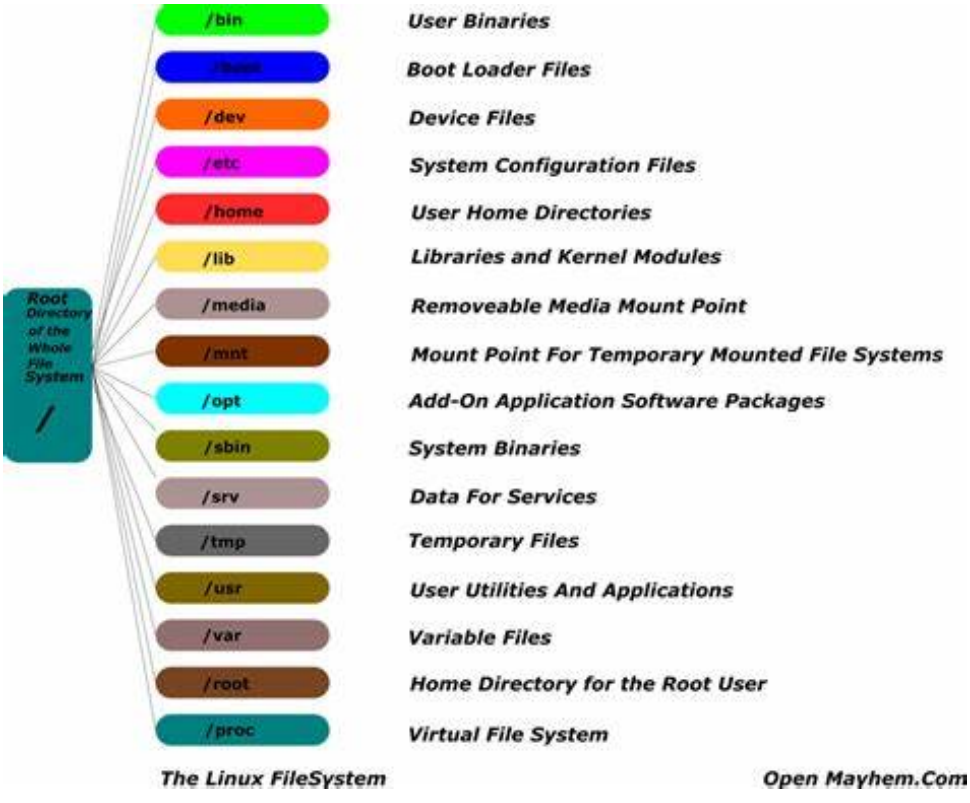
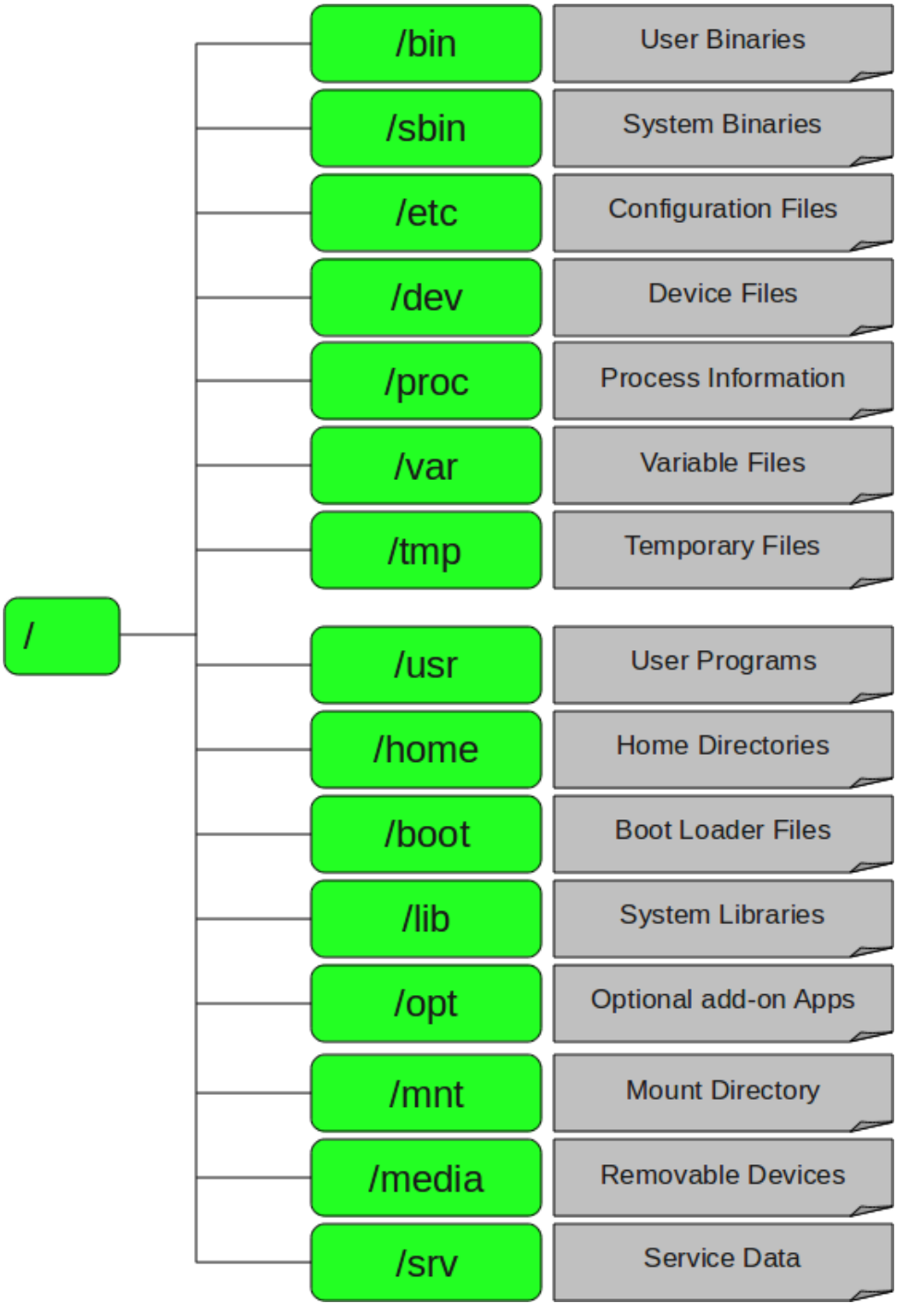
Memo - DevOps - Git and GitHub

Git Bash

- Bash Terminal Command






/
/bin
User Binaries
/sbin
System Binaries
/etc
Configuration Files
/dev
Device Files
/proc
Process Information
/var
Variable Files
/tmp
Temporary Files
/usr
User Programs
/home
Home Directories
/boot
Boot Loader Files
/lib
System Libraries
/opt
Optional add-on Apps
/mnt
Mount Directory
/media
Removable Devices
/srv
Service Data

Unix/Linux Command

HTTPS://ITSKILLSUPPORT.BLOGSPOT.COM

CREATED BY:- IT SUPPORT

File Commands

ls – directory listing
ls -al – formatted listing with hidden files
cd *dir* - change directory to *dir*
cd – change to home
pwd – show current directory
mkdir *dir* – create a directory *dir*
rm *file* – delete *file*
rm -r *dir* – delete directory *dir*
rm -f *file* – force remove *file*
rm -rf *dir* – force remove directory *dir* *
cp *file1 file2* – copy *file1* to *file2*
cp -r *dir1 dir2* – copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* – rename or move *file1* to *file2*
if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* – create symbolic link *link* to *file*
touch *file* – create or update *file*
cat > *file* – places standard input into *file*
more *file* – output the contents of *file*
head *file* – output the first 10 lines of *file*
tail *file* – output the last 10 lines of *file*
tail -f *file* – output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps – display your currently active processes
top – display all running processes
kill *pid* – kill process id *pid*
killall *proc* – kill all processes named *proc* *
bg – lists stopped or background jobs; resume a stopped job in the background
fg – brings the most recent job to foreground
fg *n* – brings job *n* to the foreground

File Permissions

chmod *octal file* – change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 – read (r)
- 2 – write (w)
- 1 – execute (x)

Examples:
chmod 777 – read, write, execute for all
chmod 755 – rwx for owner, rx for group and world
For more options, see **man chmod**.

SSH

ssh *user@host* – connect to *host* as *user*
ssh -p *port user@host* – connect to *host* on port *port* as *user*
ssh-copy-id *user@host* – add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* – search for *pattern* in *files*
grep -r *pattern dir* – search recursively for *pattern* in *dir*
command **|** **grep *pattern*** – search for *pattern* in the output of *command*
locate *file* – find all instances of *file*

System Info

date – show the current date and time
cal – show this month's calendar
uptime – show current uptime
w – display who is online
whoami – who you are logged in as
finger *user* – display information about *user*
uname -a – show kernel information
cat /proc/cpuinfo – cpu information
cat /proc/meminfo – memory information
man *command* – show the manual for *command*
df – show disk usage
du – show directory space usage
free – show memory and swap usage
whereis *app* – show possible locations of *app*
which *app* – show which *app* will be run by default

Compression

tar cf *file.tar files* – create a tar named *file.tar* containing *files*
tar xf *file.tar* – extract the files from *file.tar*
tar czf *file.tar.gz files* – create a tar with Gzip compression
tar xzf *file.tar.gz* – extract a tar using Gzip
tar cjf *file.tar.bz2* – create a tar with Bzip2 compression
tar xjf *file.tar.bz2* – extract a tar using Bzip2
gzip *file* – compresses *file* and renames it to *file.gz*
gzip -d *file.gz* – decompresses *file.gz* back to *file*

Network

ping *host* – ping *host* and output results
whois *domain* – get whois information for *domain*
dig *domain* – get DNS information for *domain*
dig -x *host* – reverse lookup *host*
wget *file* – download *file*
wget -c *file* – continue a stopped download

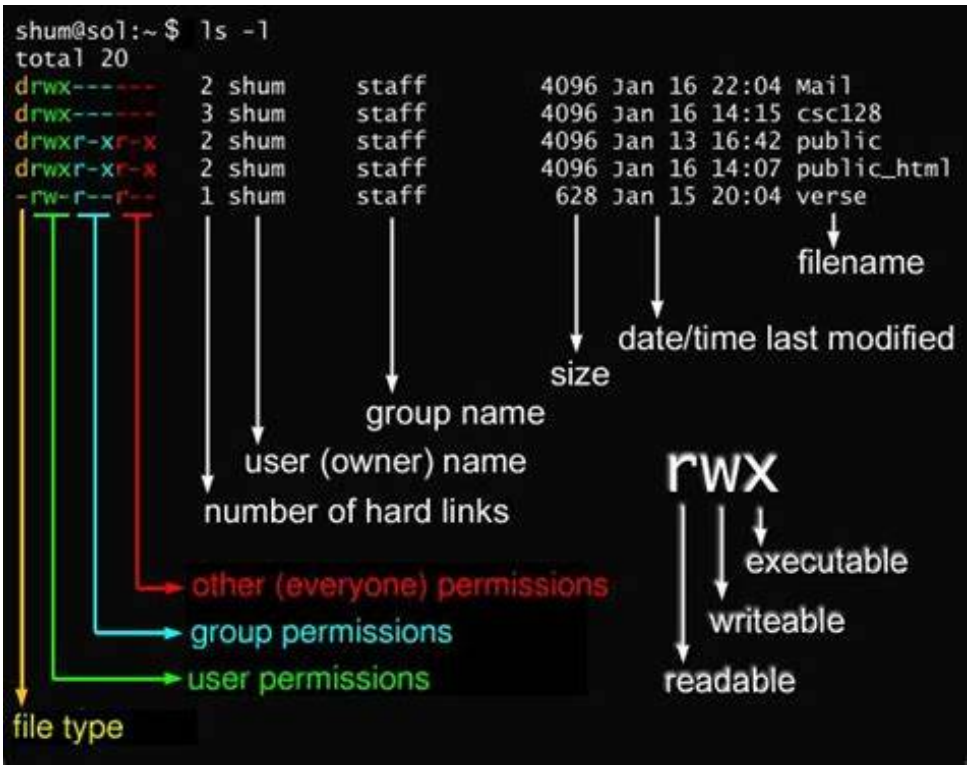
Installation

Install from source:
./configure
make
make install
dpkg -i *pkg.deb* – install a package (Debian)
rpm -Uvh *pkg.rpm* – install a package (RPM)

Shortcuts

Ctrl+C – halts the current command
Ctrl+Z – stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D – log out of current session, similar to **exit**
Ctrl+W – erases one word in the current line
Ctrl+U – erases the whole line
Ctrl+R – type to bring up a recent command
!! - repeats the last command
exit – log out of current session

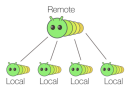
* use with extreme caution.


shum@sol:~$ ls -l
total 20
drwx------ 2 shum staff 4096 Jan 16 22:04 Mail
drwx------ 3 shum staff 4096 Jan 16 14:15 csc128
drwxr-xr-x 2 shum staff 4096 Jan 13 16:42 public
drwxr-xr-x 2 shum staff 4096 Jan 16 14:07 public_html
-rw-r--r-- 1 shum staff 628 Jan 15 20:04 verse
filename
date/time last modified
size
group name
user (owner) name
number of hard links
rwx
executable
writeable
readable
other (everyone) permissions
group permissions
user permissions
file type

- Git Hub

Git Cheat Sheet

Good things to know



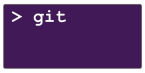
Git is a **distributed** version control system

Everyone that has a local copy of the repository at all times. Cannot be partially checked out, it's all or nothing.

Git ≠ GitHub

Git **is not** Github

Git is a repository, where files are stored and managed. Github is a hosting service that keeps your Git repo in the cloud.



Git CLI is **very powerful**.

OK, that's not a "fact", but once you get used to it, you'll be able to use the commands anywhere. You can also use CLI and GUIs interchangeably on the same Git repository.

There are **many ways** to get there

People use Git in many different ways, and projects may follow different Git flows. Don't be shy, ask around.

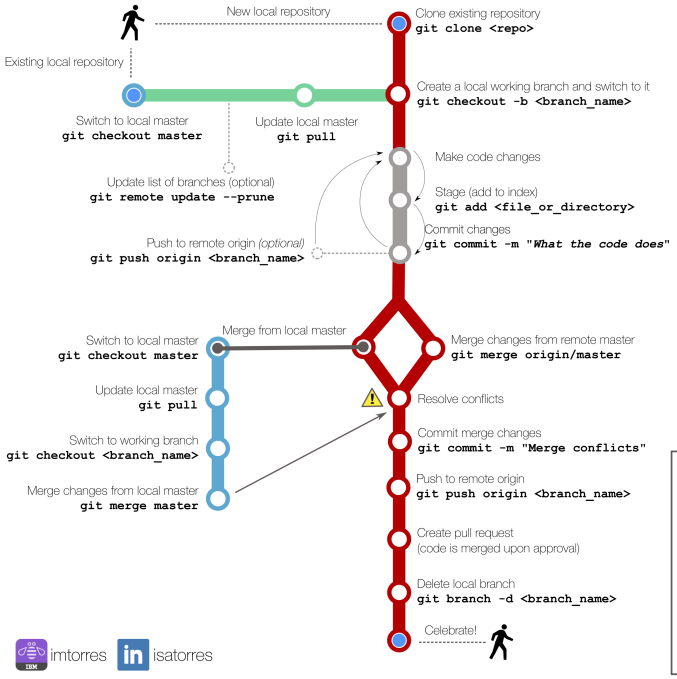
Terminology

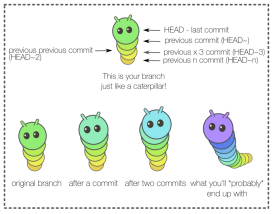
repository - *where files are stored, can be remote or local*
remote repository - repository in hosting server, also referred to as origin
local repository - repository in local development machine
branch - *a stream of work where commits are kept, can be remote or local*
remote branch - branch with published commits (commits that have been pushed)
local branch - branch with unpublished commits, only developer can see, not shared
commit - *a change unit, it is a scope of changes that are kept in sequential order*

master branch - *main stream of work, must always be stable*
working branch - *developer's stream of work, sandbox*
staging area/index - *keeps files to include in next commit*
push - *publish changes to remote*
pull - *merge remote changes into local changes*
pull request - *code review process to allow a change be integrated to the master branch*

Simple flow of getting things done with Git

Scenario: developer working in individual branch, team code (stable) lives in the master branch





FREE!

Commands that you can use anytime

See the status of the working branch
`git status`
See the commit history
`git log`
See all branches (remote - since last remote update)
`git branch -a`

⚠ Resolve conflicts

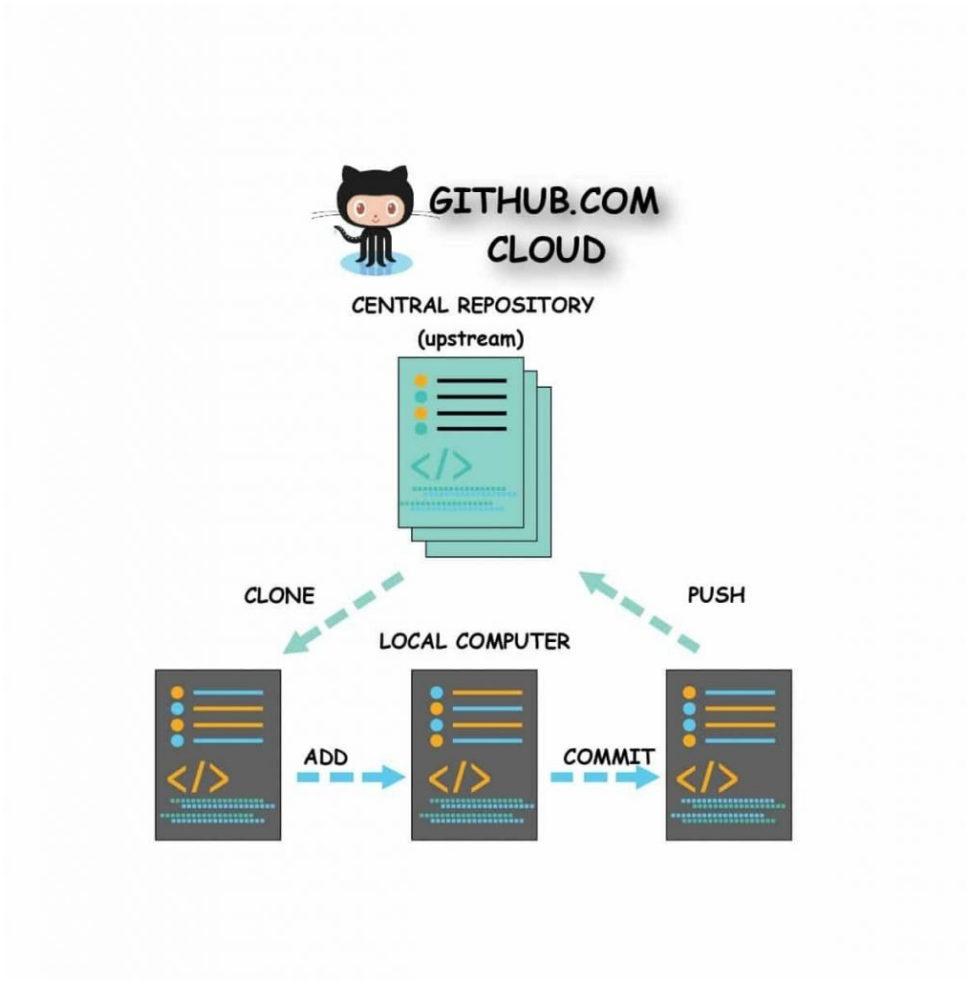
No need to panic! Use a merge tool, or...

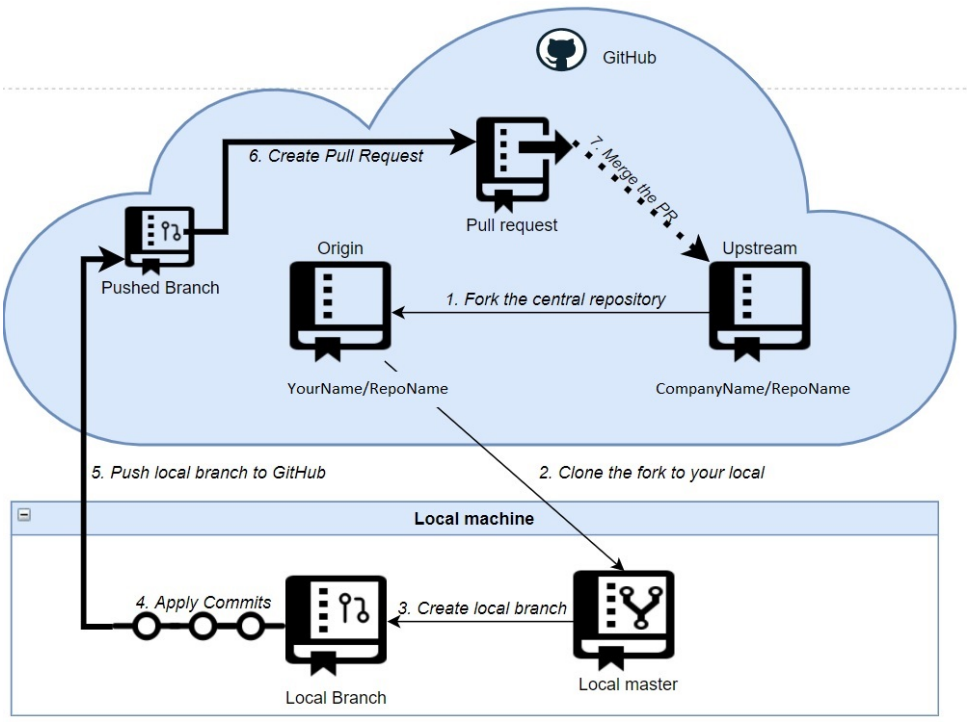
```
(remove) <<<<<<< HEAD
(merge)   your local code
(remove) =======
(merge)   code in master
(remove) >>>>>>> master
```

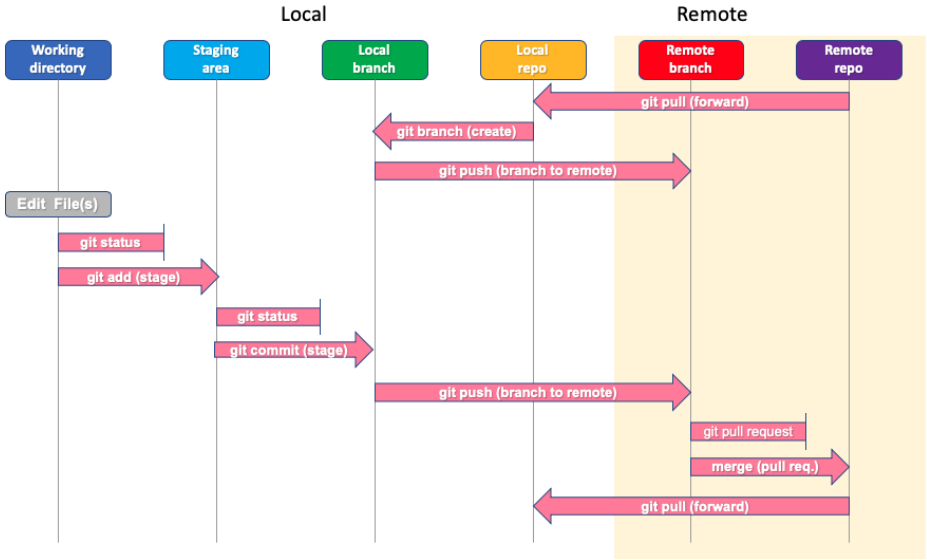
Mark the conflict resolved
`git add <file>`

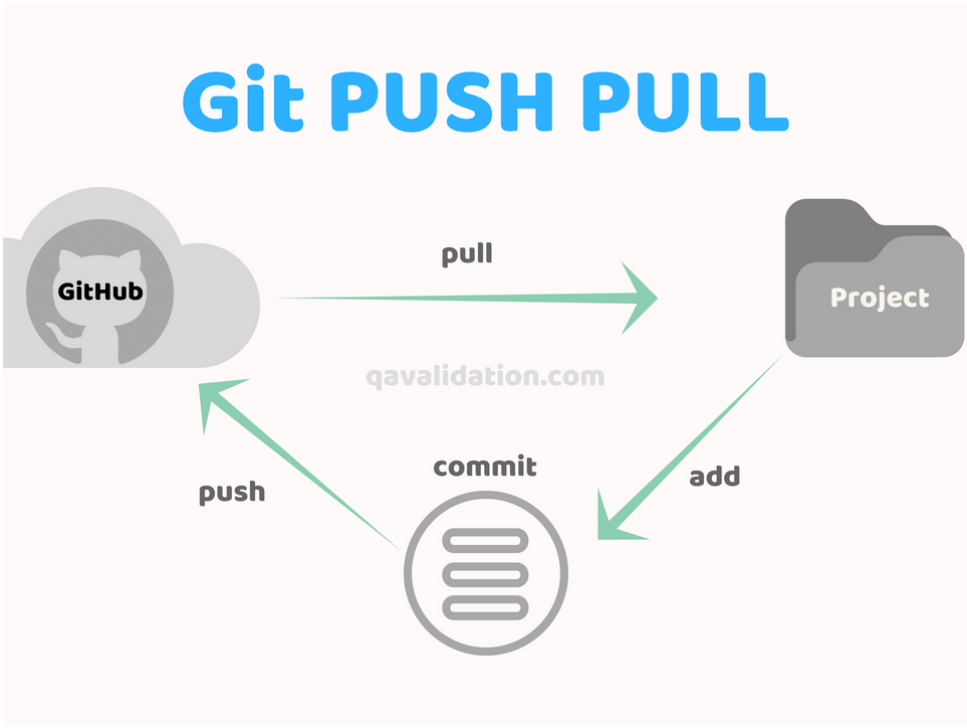
Want to start over? Roll back all the merged changes
`git reset --hard HEAD`

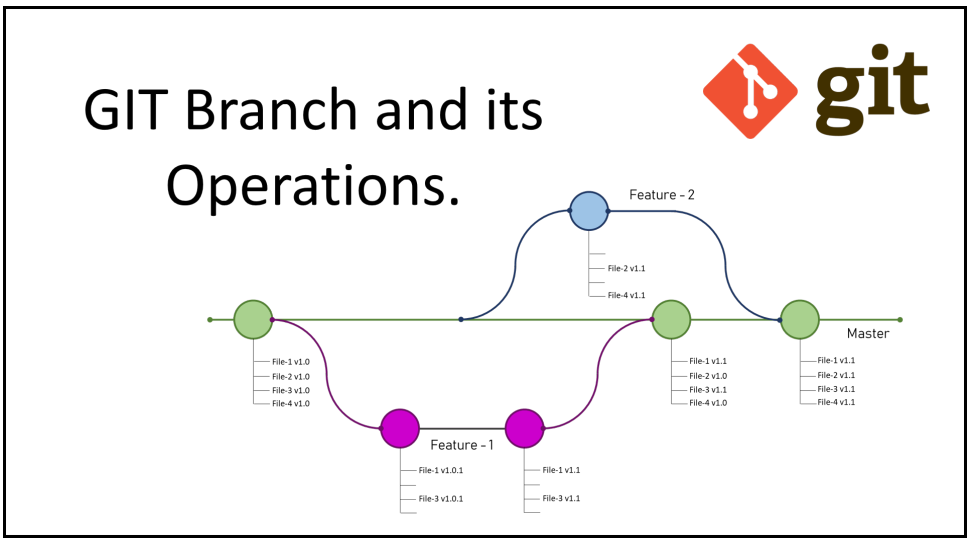
imtorres isatorres

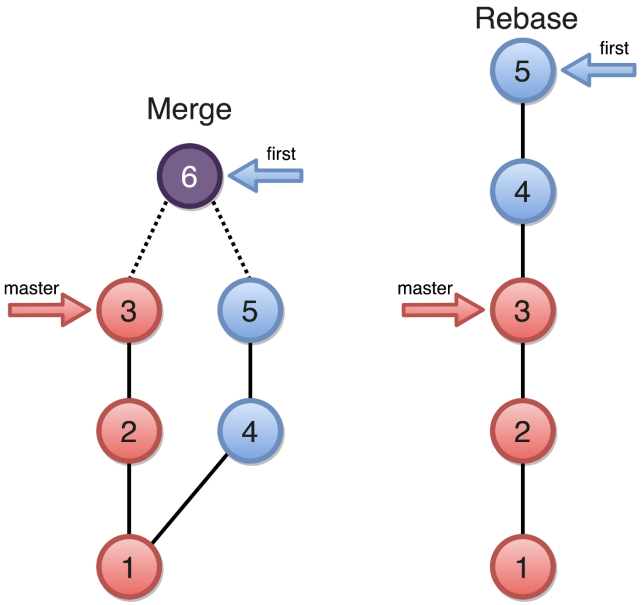

GITHUB.COM
CLOUD
CENTRAL REPOSITORY
(upstream)
CLONE
PUSH
LOCAL COMPUTER
ADD
COMMIT

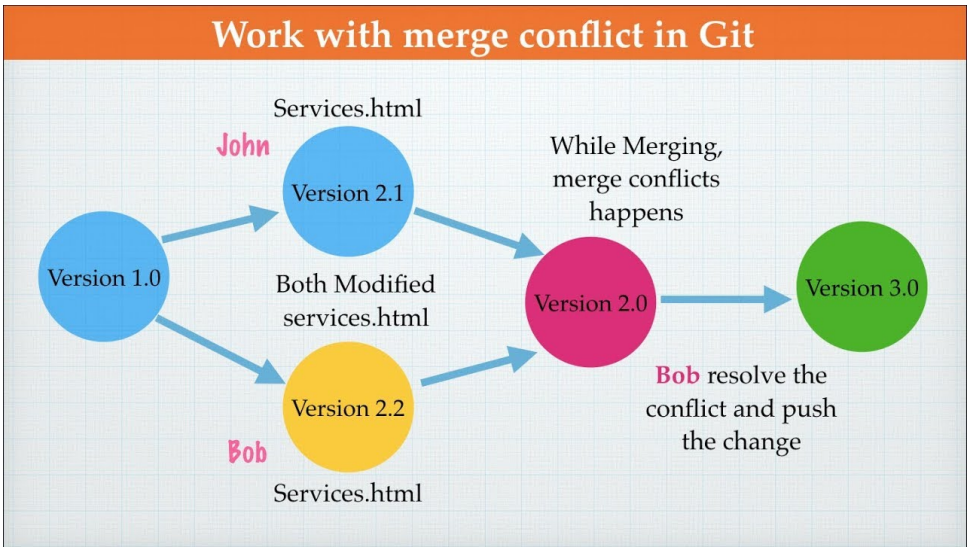

GitHub
6. Create Pull Request
7. Merge the PR
Pull request
Pushed Branch
Origin
Upstream
1. Fork the central repository
YourName/RepoName
CompanyName/RepoName
5. Push local branch to GitHub
2. Clone the fork to your local
Local machine
4. Apply Commits
3. Create local branch
Local Branch
Local master


Local
Remote
Working directory
Staging area
Local branch
Local repo
Remote branch
Remote repo
git pull (forward)
git branch (create)
git push (branch to remote)
Edit File(s)
git status
git add (stage)
git status
git commit (stage)
git push (branch to remote)
git pull request
merge (pull req.)
git pull (forward)


Git PUSH PULL
GitHub
pull
Project
qavalidation.com
push
commit
add


GIT Branch and its Operations.
git
Feature - 2
File-2 v1.1
File-4 v1.1
Master
File-1 v1.0
File-2 v1.0
File-3 v1.0
File-4 v1.0
File-1 v1.1
File-2 v1.0
File-3 v1.1
File-4 v1.0
File-1 v1.1
File-2 v1.1
File-3 v1.1
File-4 v1.1
Feature - 1
File-1 v1.0.1
File-3 v1.0.1
File-1 v1.1
File-3 v1.1





git basic

- git init
- git status
- git add .
- git add all
- git reset
- git restore filename
- git commit -m "xxxxxx"
- git commit -v --amend (for changing the commit message)
- git restore --staged filename
- git push
- git pull
- git log
- git log --oneline
- git remote remove

git branch

- git branch --list
- git branch
- git branch -r
- git branch -a
- git branch newbranch(create newbranch)
- git checkout newbranch(switch to newbranch)
- git checkout -b newbrach(create newbranch and switch to newbranch)
- git branch -D branch(delete branch)
- git checkout -
- git branch -m oldbranchname newbranchname

git merage

- git checkout main
- git merge branch_A*(merge branch_A to main)

-> `conflict`
-> git merge --abort -> git pull + amend file + git add file + git commit -m "xxx"+ git push

git rebase

- git rebase --continue

Undo

- git checkout log_id(switch to log_id)
- git revert log_id(automerge -> undo to log_is status)
- git reset log_id(unstage changes after log_id)
- git reset log_id --hard(delete commits after log_id)

git cheat sheet

learn more about git the simple way at rogerdudler.github.com/git-guide/
cheat sheet created by Nina Jaeschke of ninagrafik.com

create & clone

create new repository	*git init*
clone local repository	*git clone /path/to/repository*
clone remote repository	*git clone username@host:/path/to/repository*

add & remove

add changes to INDEX	*git add <filename>*
add all changes to INDEX	*git add **
remove/delete	*git rm <filename>*

commit & synchronize

commit changes	*git commit -m "Commit message"*
push changes to remote repository	*git push origin master*
connect local repository to remote repository	*git remote add origin <server>*
update local repository with remote changes	*git pull*

branches

create new branch	*git checkout -b <branch>* *e.g. git checkout -b feature_x*
switch to master branch	*git checkout master*
delete branch	*git branch -d <branch>*
push branch to remote repository	*git push origin <branch>*

merge

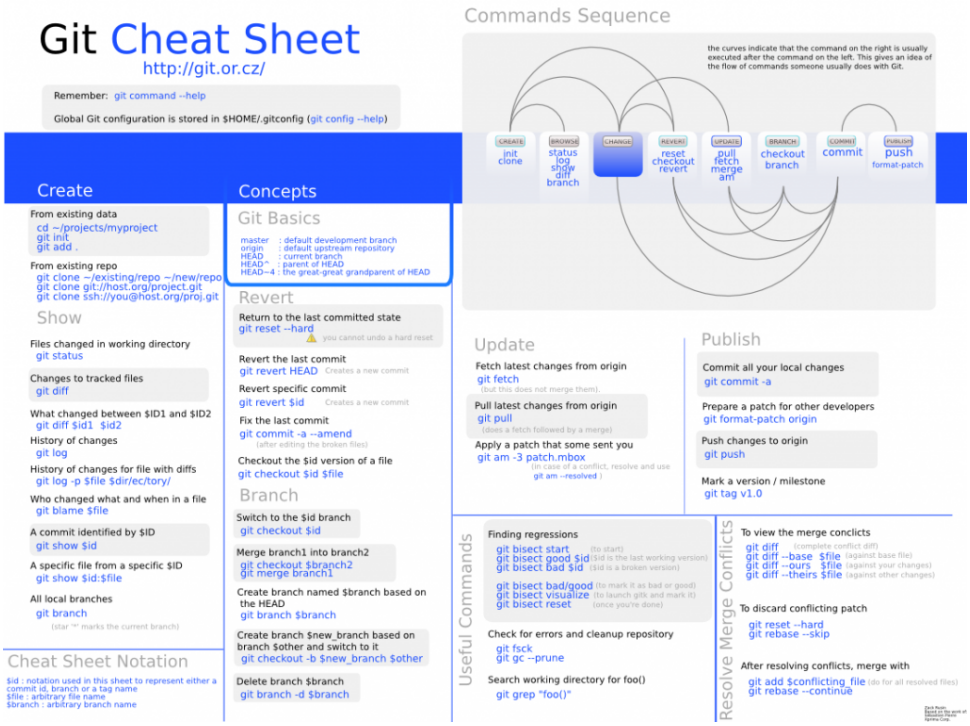
merge changes from another branch	*git merge <branch>*
view changes between two branches	*git diff <source_branch> <target_branch>* *e.g. git diff feature_x feature_y*

tagging

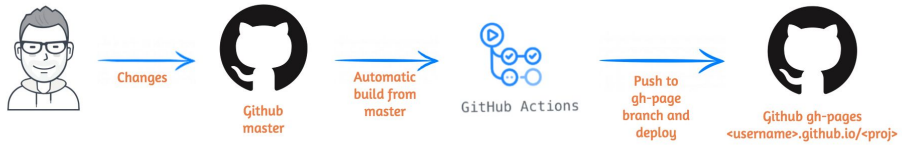
create tag	*git tag <tag> <commit ID>* *e.g. git tag 1.0.0 1b2e1d63ff*
get commit IDs	*git log*

restore

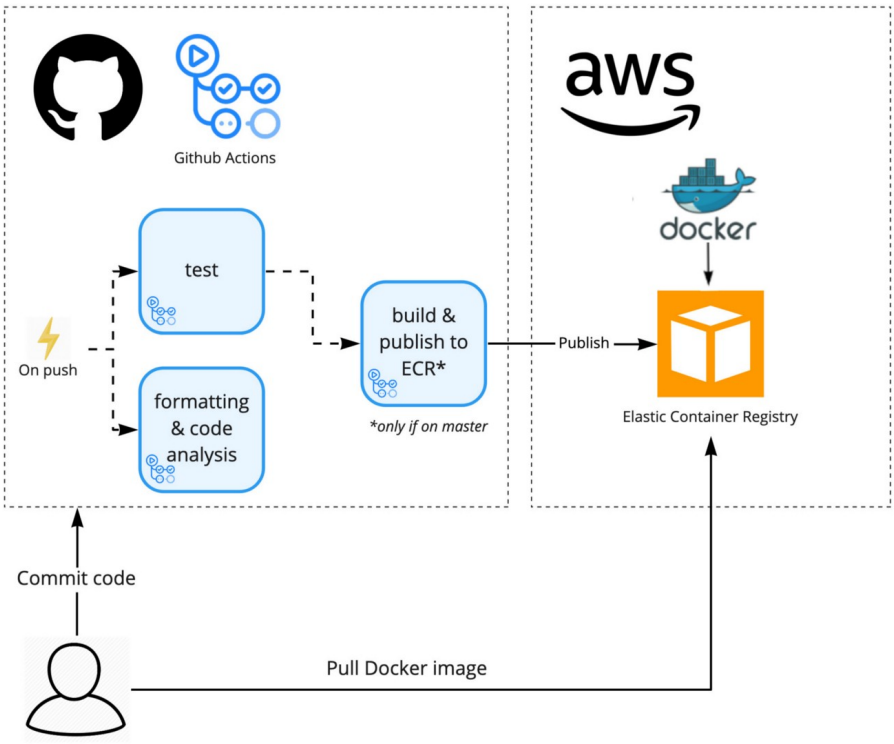
replace working copy with latest from HEAD	*git checkout -- <filename>*

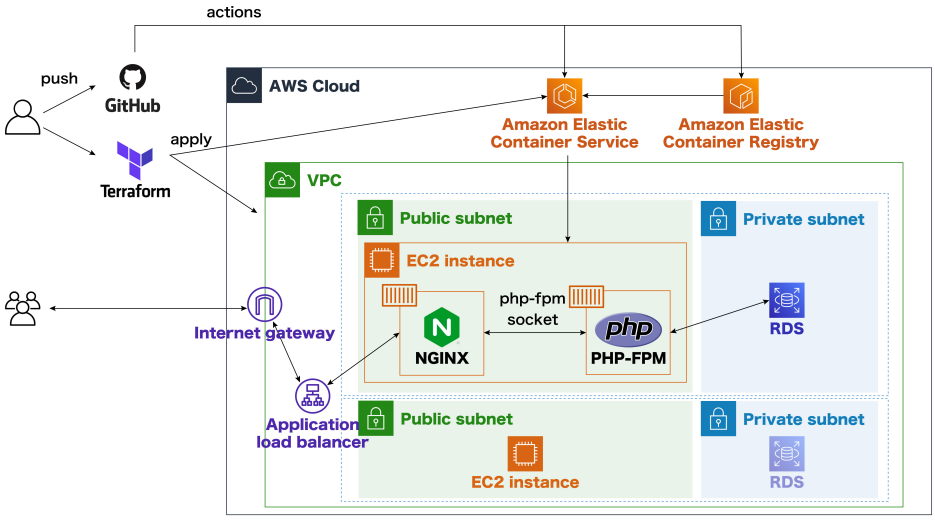


- GitHub Pages



- GitHub Actions


Github Actions
aws
docker
test
build &
publish to
ECR*
*only if on master
Publish
On push
formatting
& code
analysis
Elastic Container Registry
Commit code
Pull Docker image


actions
push
GitHub
apply
Terraform
AWS Cloud
Amazon Elastic
Container Service
Amazon Elastic
Container Registry
VPC
Public subnet
Private subnet
EC2 instance
php-fpm
socket
NGINX
PHP-FPM
RDS
Internet gateway
Application
load balancer
Public subnet
Private subnet
EC2 instance
RDS

-- Memo End --