GIT && GITHUB

Date / Time : 17.06.2016 / 6 P.M to 7 P.M (IST)

Presented by: Sivaram.N

Agenda

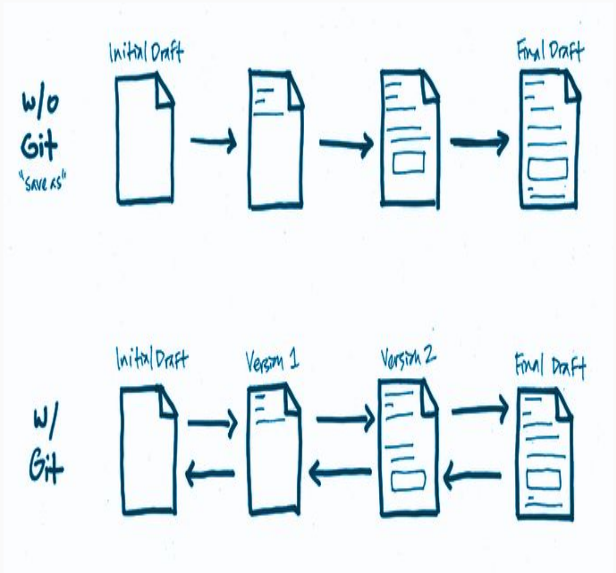
- ❏ Introduction
- ❏ Why Git
- ❏ Install && Setup Git
- ❏ Git Workflow
- ❏ Git && Git Hub && Gist
- ❏ GitHub Pages (gh-pages)
- ❏ Git Commands
- ❏ Where Can I Learn More
- ❏ Misc

Introduction

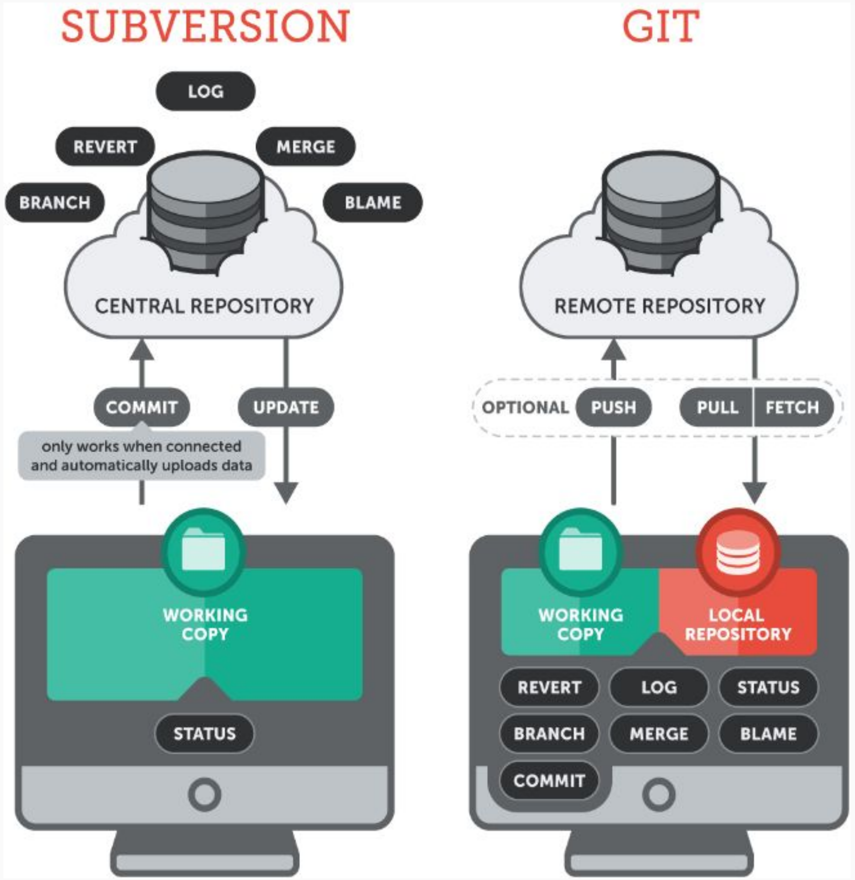
- Distributed version control system
  - Eg:http://www.compuquip.com/2009/11/20/centralized-vs-distributed-computing/
- Developed by **Linus Torvalds** for the development of the Linux Kernel in 2005
  - Eg:https://www.atlassian.com/git/articles/10-years-of-git/
- A version control application keeps track of all the changes that you do in the files of your project.
- Eg: **VCS** : svn, cvs .., **DVCS** : git, Mercurial...

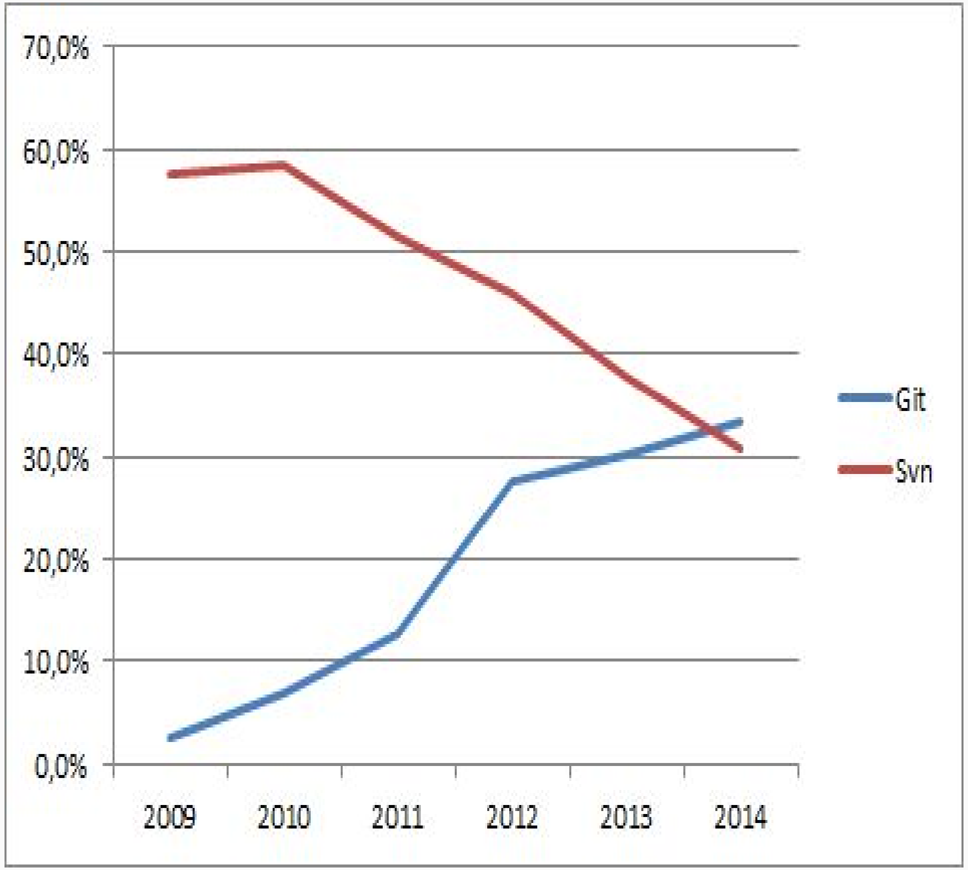
Why Git

- Free and OpenSource
- It is distributed not centralised
- Everything is local / Offline Usage
- Ignoring Certain Files
- CI friendly repos
- Git hooks for CI
- Collaborative/Individual Development



Why Git


SUBVERSION
LOG
REVERT
MERGE
BRANCH
BLAME
CENTRAL REPOSITORY
COMMIT
UPDATE
only works when connected and automatically uploads data
WORKING COPY
STATUS
GIT
REMOTE REPOSITORY
OPTIONAL
PUSH
PULL
FETCH
WORKING COPY
LOCAL REPOSITORY
REVERT
LOG
STATUS
BRANCH
MERGE
BLAME
COMMIT


70,0%
60,0%
50,0%
40,0%
30,0%
20,0%
10,0%
0,0%
2009
2010
2011
2012
2013
2014
Git
Svn

Install && Setup

Install

- Linux
  - $ sudo apt-get install git-all
  - $ git --version # To check version
  - $ git help # To get help

- Mac : https://git-scm.com/download/mac

- Windows: http://windows.github.com

Install && Setup

First time Setup : [$git config --list] ['--global' is optional]

- Identity
  - $ git config --global user.name "<username>"
  - $ git config --global user.email user@example.com

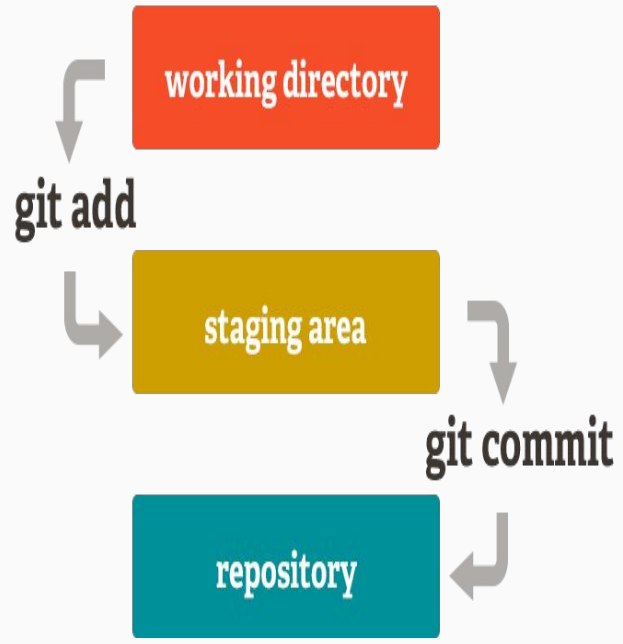
- Editor
  - $ git config --global core.editor <emacs,vi,nano>

- Color
  - $ git config --global color.ui auto

- Checking Configuration
  - $ git config --list

Git Workflow

- Obtain repository (init/clone/fork)
- Make Some edits/additions (modified)
- Stage your changes (add)
- Commit your work (commit)
- Push to remote (push)



Git && GitHub && Gist

Git

- The version control tool which is CLI
- Decentralized source code management protocol

GitHub

- Free web based software project hosting
- Git is used as underlying scm protocol

Gist

- Gists are a particular service offered on that site, namely code snippets

GitHub Pages (gh-pages)

- Hosting static site on GitHub

Github pages (gh-pages)

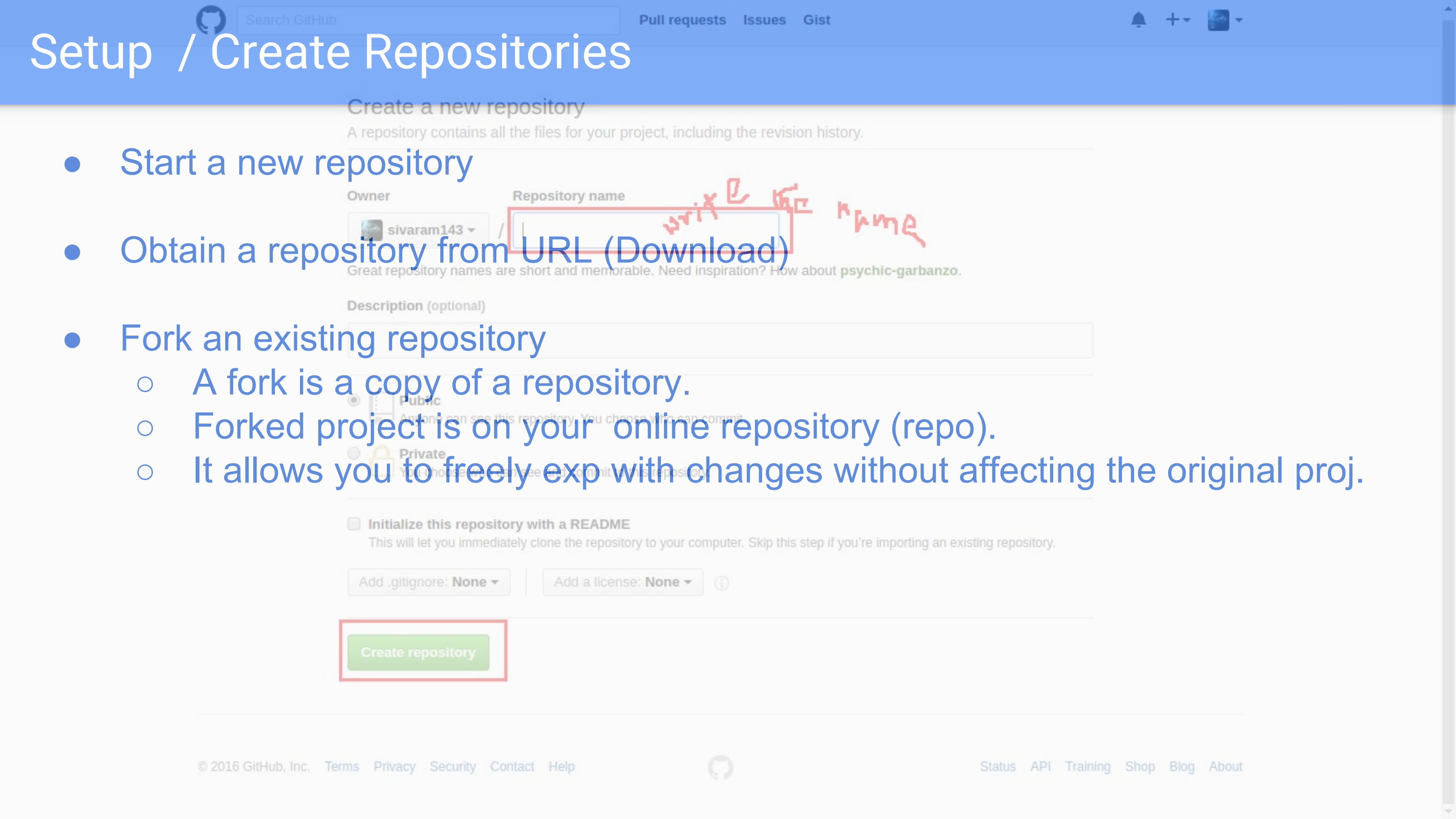
- Steps to create Github Pages
  - Create new repo
  - Add gh-pages branch
  - Pick Bootstrap template and customize or use Jekyll to generate site
  - Commit the site to the gh-pages branch
- Ref
  - https://pages.github.com/
  - https://guides.github.com/features/pages/

Git Commands

- ❑ Setup / Create Repositories
- ❑ Saving Changes (add / remove / move / commit)
- ❑ Viewing History / Logs
- ❑ Branching / Merging
- ❑ Undoing Changes
- ❑ Stashing
- ❑ Remote Commands
- ❑ Release & Tags
- ❑ Others

Setup / Create Repositories

- Start a new repository
- Obtain a repository from URL (Download)
- Fork an existing repository
  - A fork is a copy of a repository.
  - Forked project is on your online repository (repo).
  - It allows you to freely exp with changes without affecting the original proj.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: sivaram143 / Repository name:

write the name

Great repository names are short and memorable. Need inspiration? How about psychic-garbanzo.

Description (optional)

Public — Anyone can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None   Add a license: None

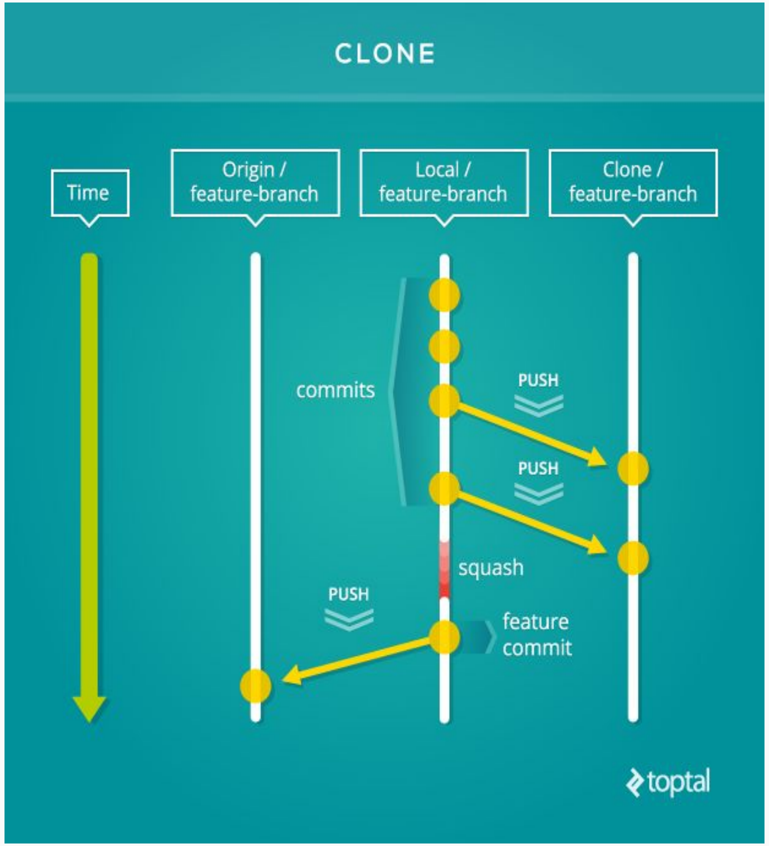
Start a new repository

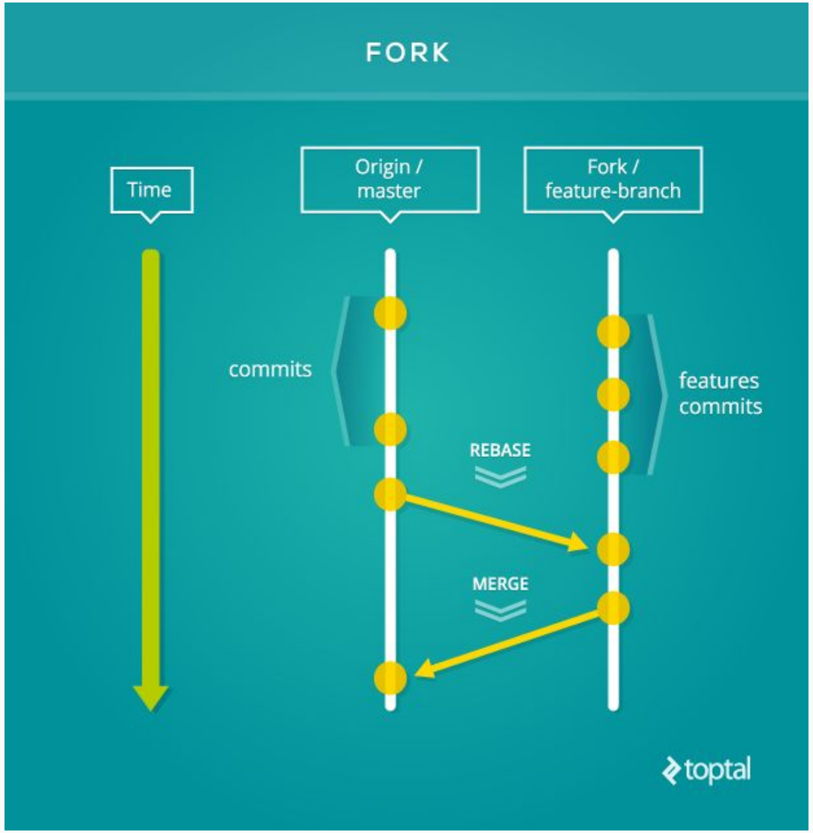
- Adding local project to github repository
  - Create a new repo from github (gui)
  - $ mkdir <name>
  - $ cd <name>
  - $ git init #initialize local directory as git repo
  - $ ls -a # observes '.git' in current directory
  - $ git add . #Adds the files in the local repository
  - $ git commit -m "First commit" #Commits the tracked changes
  - $ git remote add origin remote repository URL # sets remote repo
  - $ git remote -v # Verifies the new remote URL
  - $ git push origin master
- Create a new repo directly from CLI
  - https://coderwall.com/p/mnwcog/create-new-github-repo-from-command-line
  - https://www.viget.com/articles/create-a-github-repo-from-the-command-line

Obtain a repository from URL (Download)

- $ git clone <URL> # URL is either ssh / https
- $ git log # viewing history
- $ git add . #Adds the files in the local repository
- $ git status # checking status
- $ git diff # checking difference
- $ git commit -m "First commit" #Commits the tracked changes
- $ git push origin master

Clone && Fork


CLONE
Time
Origin / feature-branch
Local / feature-branch
Clone / feature-branch
commits
PUSH
PUSH
squash
PUSH
feature commit
toptal


FORK
Time
Origin / master
Fork / feature-branch
commits
features commits
REBASE
MERGE
toptal

Saving Changes

- $ git status
- $ touch <file_name>
- $ git add <file_name>
- $ git commit -m “added message here…”
- $ git log

- $ git rm <file>
- $ git status
- $ git commit -m “remove message here…”
- $ git mv <old> <new> # renaming file

Viewing History / Logs

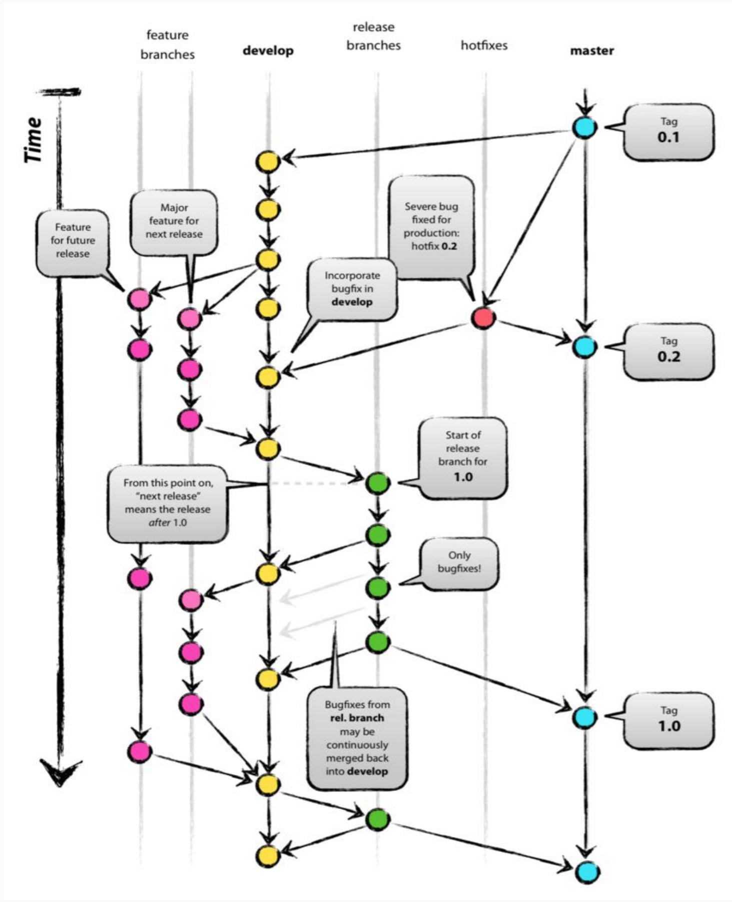
- $ git show <commit>

- $ git diff / $ git diff <file1> <file2> .... / $ git diff --name-only

- $ git log
- $ git log -p -2 #show diff & last2
- $ git log --stat # show stats (file changes)
- $ git log --pretty=online
- git log --pretty=format:"%h-%an, %ar: %s

Branching / Merging

- $ git checkout <branch>
- $ git checkout -b <new_branch> # creating new branch
- $ git branch # show the current branch
- $ git branch -d <branch> # deleting the branch, we must away from group


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Undoing Changes

- $ git reset (--hard, --soft)
- $ git revert
- $ git rebase
- $ git cherry-pick # picking specific commit from any branch

Stashing

- $ git stash save # temporary saves the local changes to stack
- $ git stash list # save the local changes
- $ git stash apply/pop # grab the item from stash list)
- $ git stash drop # droping stash changes

Remote

- $ git remote
- $ git push   # push the changes to remote repo
- $ git pull   # pull the changes from remote repo
- $ git fetch # Fetch all the recent changes from remote repo
- $ git apply <patch> # Apply patch

Release && Tag

- $ git tag <tag_release> # creating tag
- $ git fetch origin --tags # fetching tags
- $ git name-rev --tag --name-only # show current tag

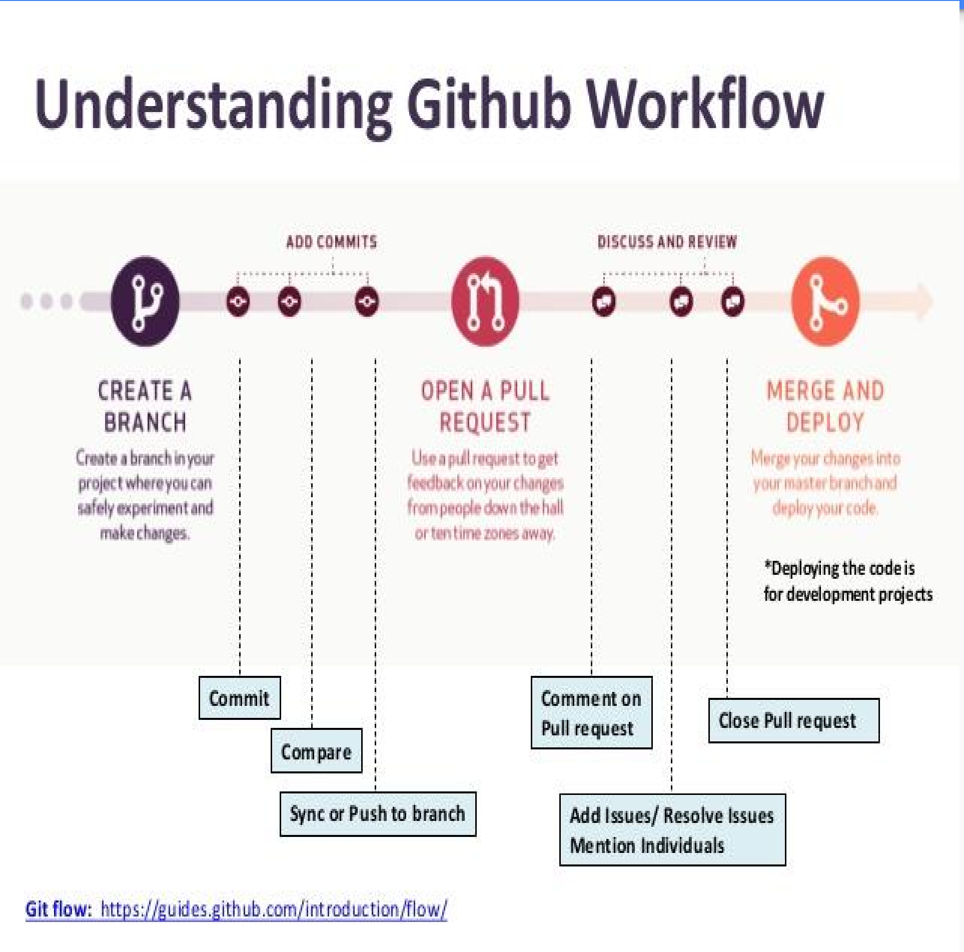
Others

- Pull-requests
- SSH Agent Forwarding
    - https://gist.github.com/apoo/2279196
    - http://cakebox.readthedocs.io/en/latest/ tutorials/connecting-your-github-ssh-key/
- Git Hookups with CI && CD

Where Can I Learn More

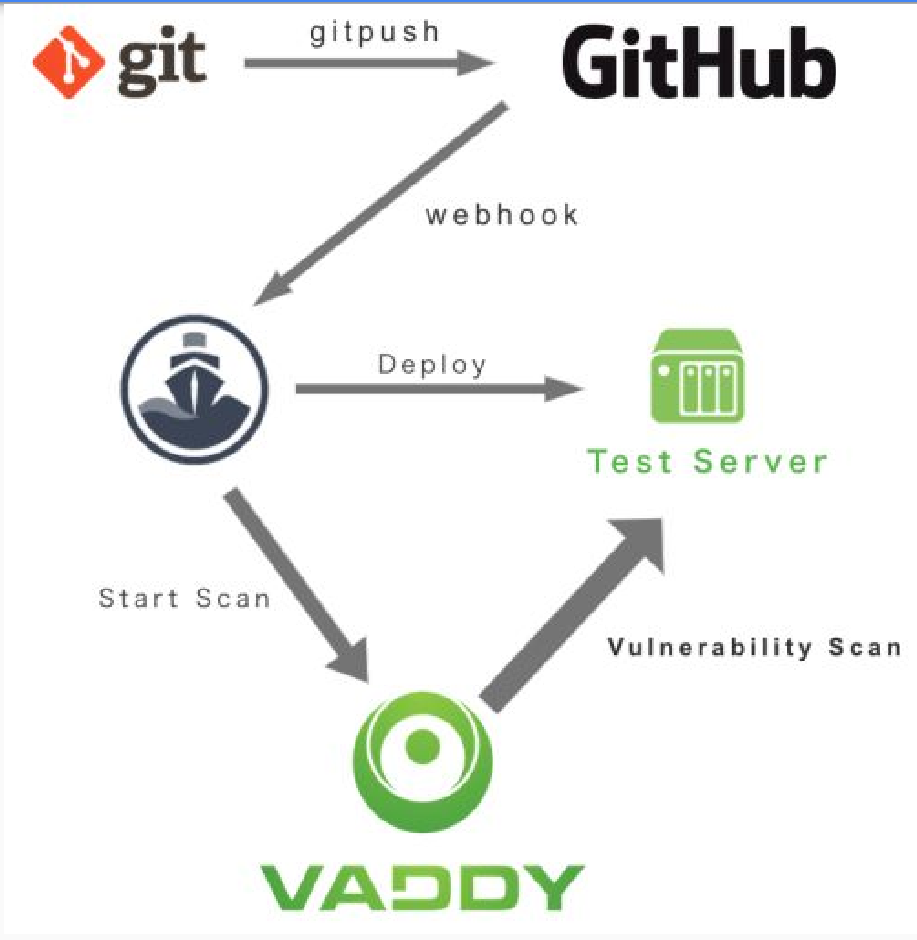
- -- Official Site: https://git-scm.com/
- Github: https://github.com/
- Slideshare: http://www.slideshare.net/chacon/getting-git
- Try Git : https://try.github.io/levels/1/challenges/1
- Git Test : https://help.github.com/articles/about-gists/
- Code tutorials: http://code.tutsplus.com/tutorials, https://www.atlassian.com/git/tutorials/
- Deploy : https://about.gitlab.com/gitlab-ci/, http://blogs.atlassian.com/2014/04/practical-continuous-deployment/

Misc


Understanding Github Workflow

ADD COMMITS
DISCUSS AND REVIEW
CREATE A BRANCH
Create a branch in your project where you can safely experiment and make changes.
OPEN A PULL REQUEST
Use a pull request to get feedback on your changes from people down the hall or ten time zones away.
MERGE AND DEPLOY
Merge your changes into your master branch and deploy your code.
*Deploying the code is for development projects
Commit
Compare
Sync or Push to branch
Comment on Pull request
Add Issues/ Resolve Issues Mention Individuals
Close Pull request

Git flow: https://guides.github.com/introduction/flow/



Thanks for listening ...