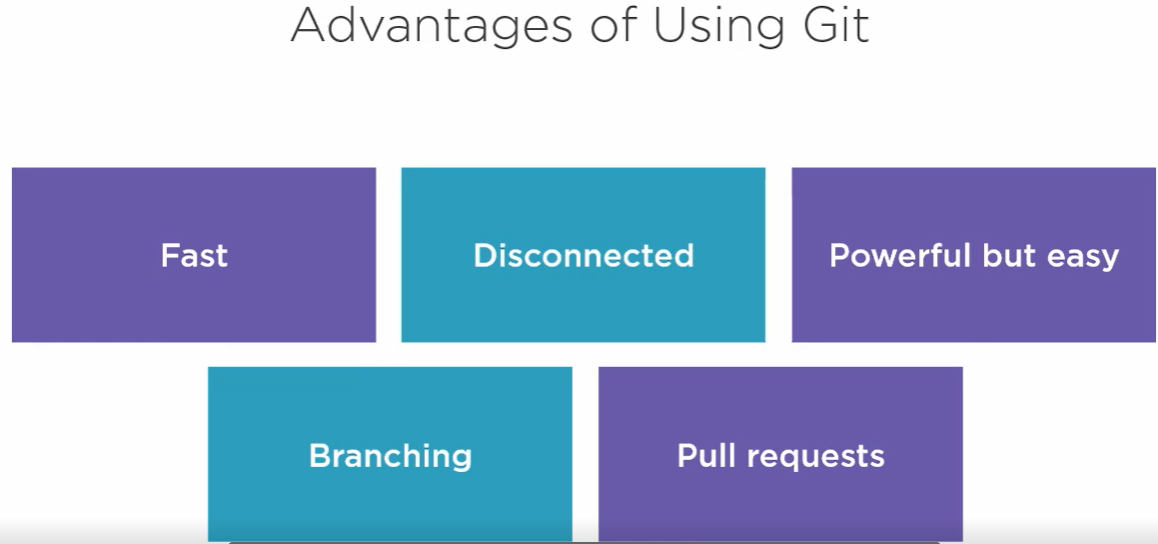
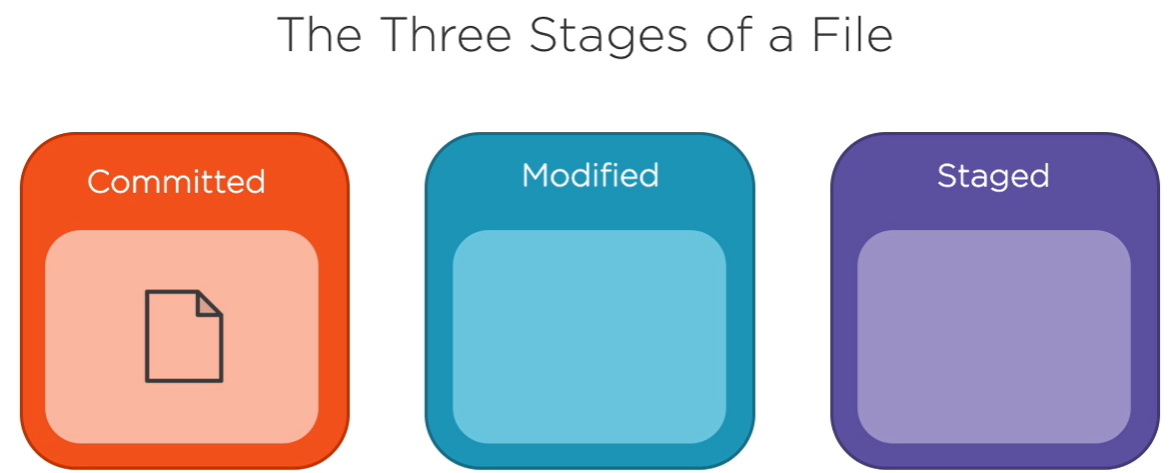
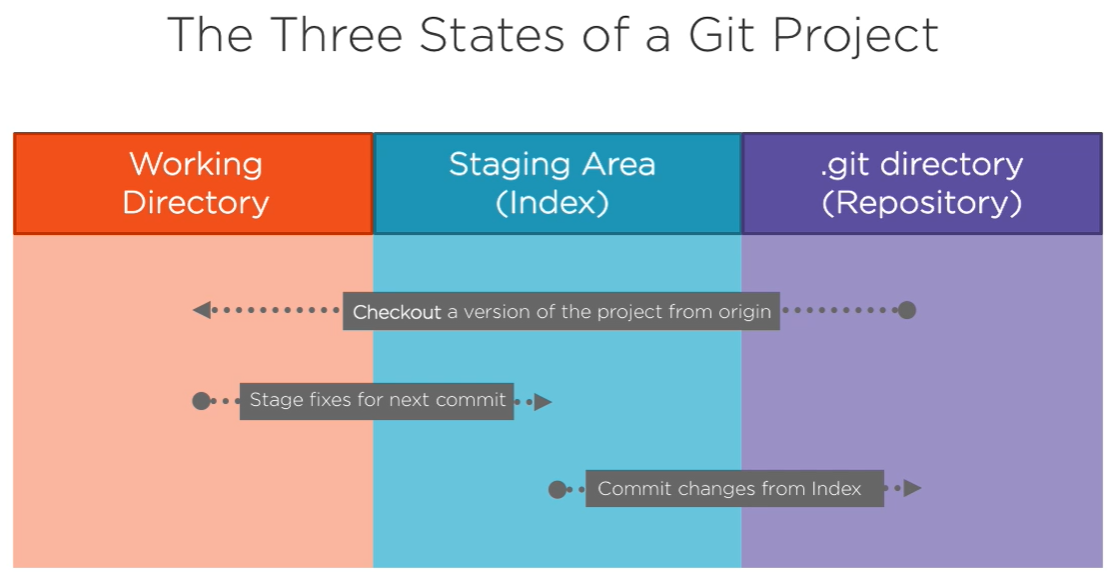
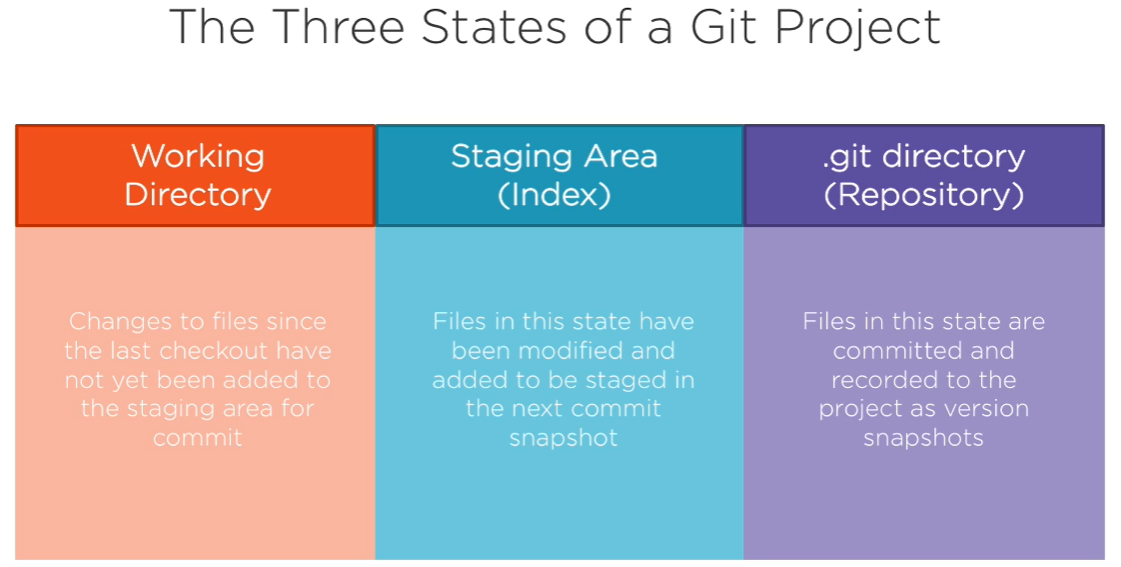
**Git**

**It is a source control system or VCS (Version Control System)**

****









**Code Hosting Providers**

Send your code, to code hosting provider



1. GitHub
2. GitLab
3. BitBucket



**Git bash commands**

* git --version
* git config --global user.name “ “
* git config --global user.email “ “
* git config –edit --global
* git config --list
* git config --user.name

**#Initialize a new git repository**

(be in the project directory)

* git init

**#(GIT\_DIR!)**

* cd .git

#This command will add all of the files to our staging area

* git add .

**#commit**

* git commit -m “message”

**#here -a option automatically stages all changes in working directory that are being tracked by git**

* git commit -a -m ”message”

**#create link between local git repository to GitHub**

* git remote add origin <http link>

**#pushing local git repository to GitHub**

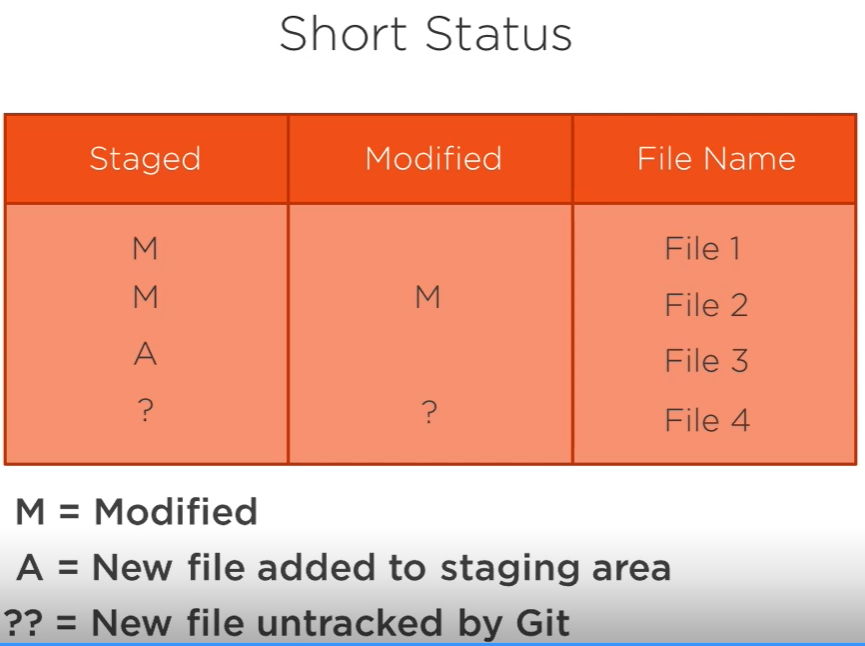
* git push -u origin master (or)
* git push origin master

**#git status**

* git status

**#short status**

* git status -s (or) git status --short



**#open text editor notepad++**

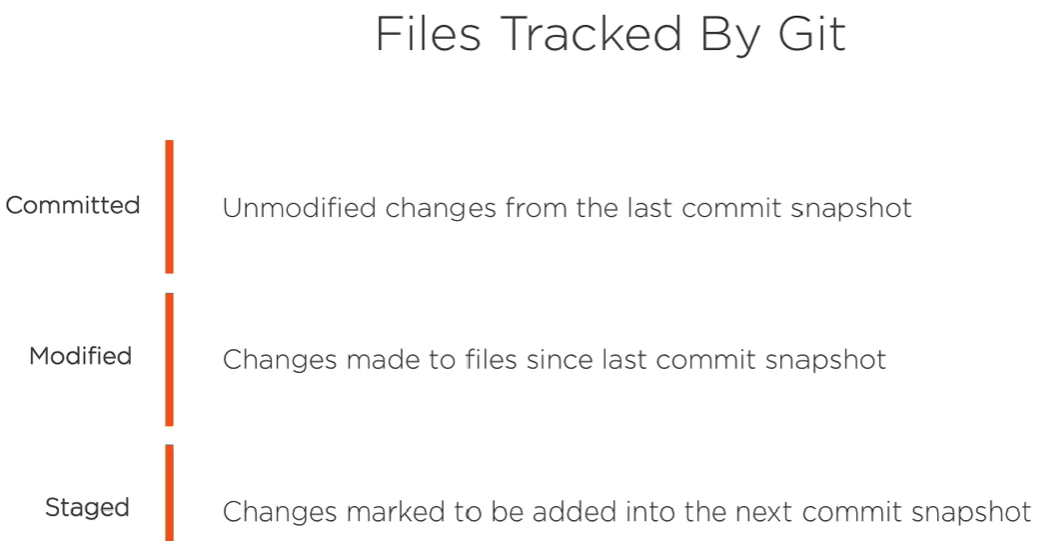
**#For notepad++, set path in environment variables**

* git config core.editor “notepad++ -multiInst -nosession”
* notepad++ <file-name>

**#For default editor**

* start <file-name>

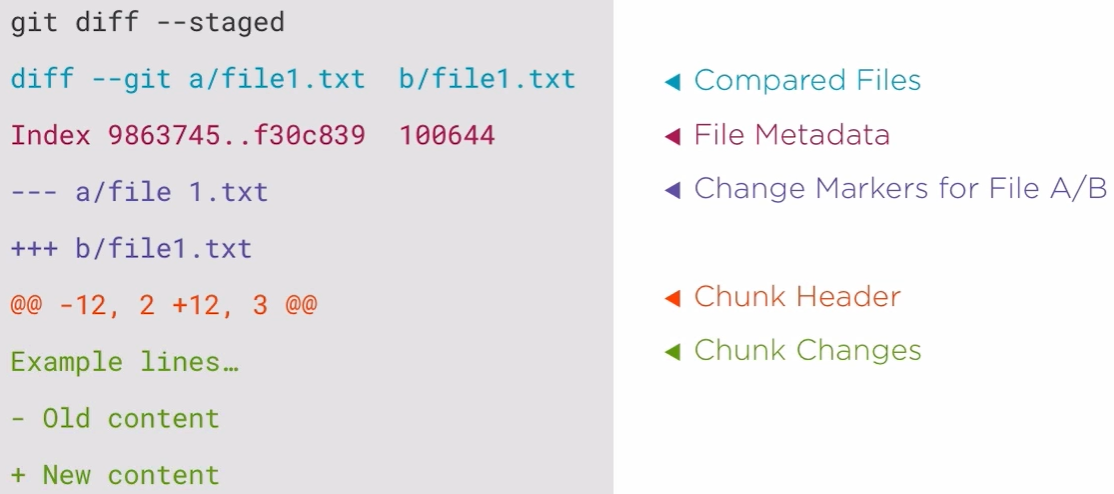
**#File Stages**



**#Git diff explained**



* git diff --staged



**#Check Commit History**

* git log

**#limited number of history**

* git log -1, got log -2

**#simplified history of list of commits**

* git log --oneline

**#detailed hostory**

* git log --stat

**#more detailed history (similar to git diff)**

* git log –patch

**#Remove and Move Files**

**#Remove files**

* git rm f <file-name>

**#TO delete hidden .git directory in the working directory**

* rm -fr .git

**#To delete sub directory**

* rm -fr <sub directory name>

**#untrack files**

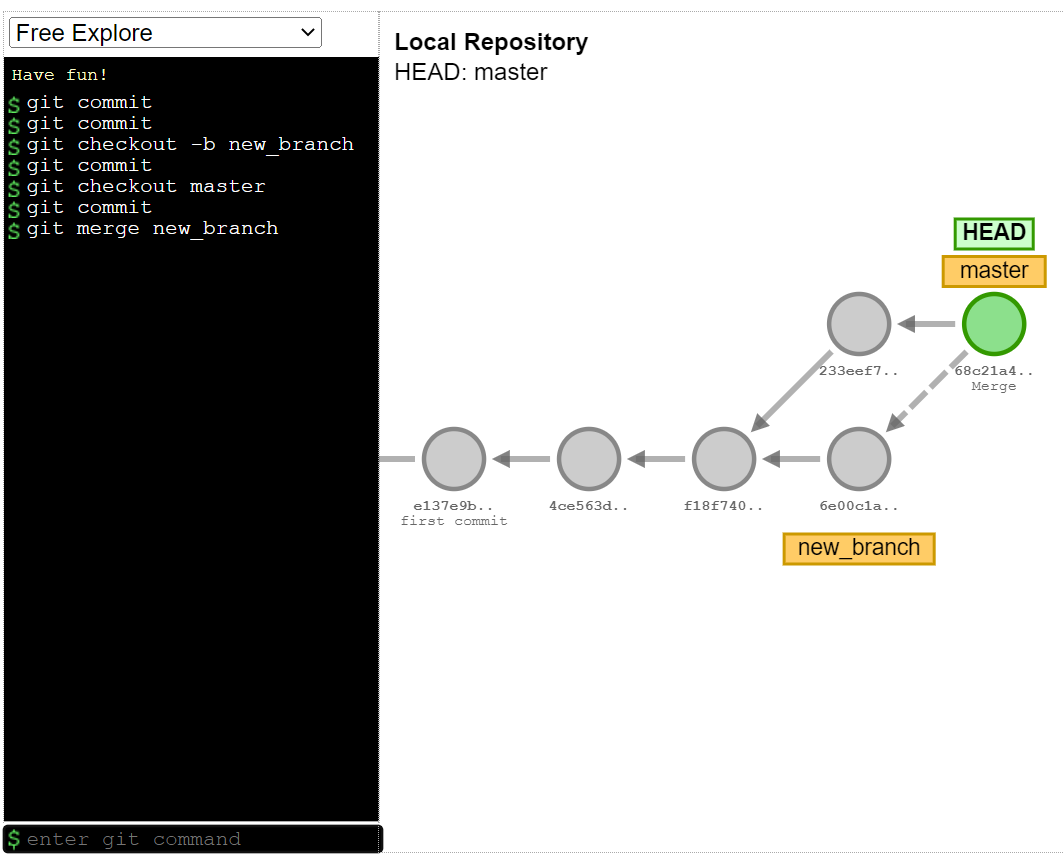
* git rm –cached <file-name>

#Rename files

* git mv filename <new-file-name>

**Introduction to Branches**

open <http://git-school.github.io/visualizing-git>



**#Creating new branch**

* git checkout -b <new-branch-name>

**#rolling back to master branch**

* git checkout master

**#list of all branches in the repository**

* git branch -a

**#without committing we can stash it so that the working directory will be clean**

* git stash

**#provides list of work-in-progress changes that we’ve stashed along with branch and some commit data**

* git stash list

**#to see a better detailed view of what file was stashed in a high-level overview Of the changes**

* git stash show

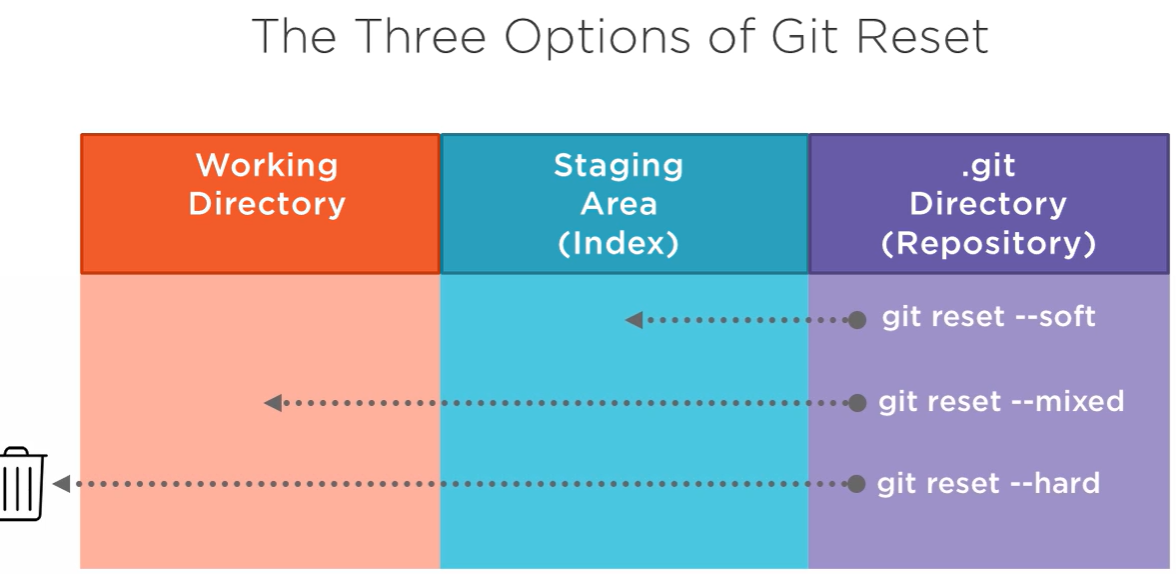
**#to get this change out of stash**

* git stash pop

**#merge branch**

* git merge <branch name>

#reset



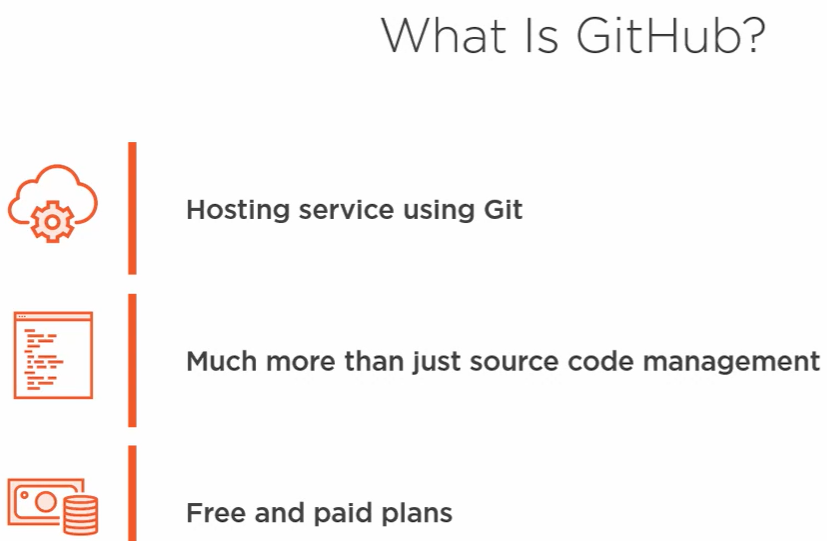
**#Pull from remote Origin**

* git clone <http link>

**#To update the git local repository**

* git pull origin <branch-name>

**GitHub Fundamentals**

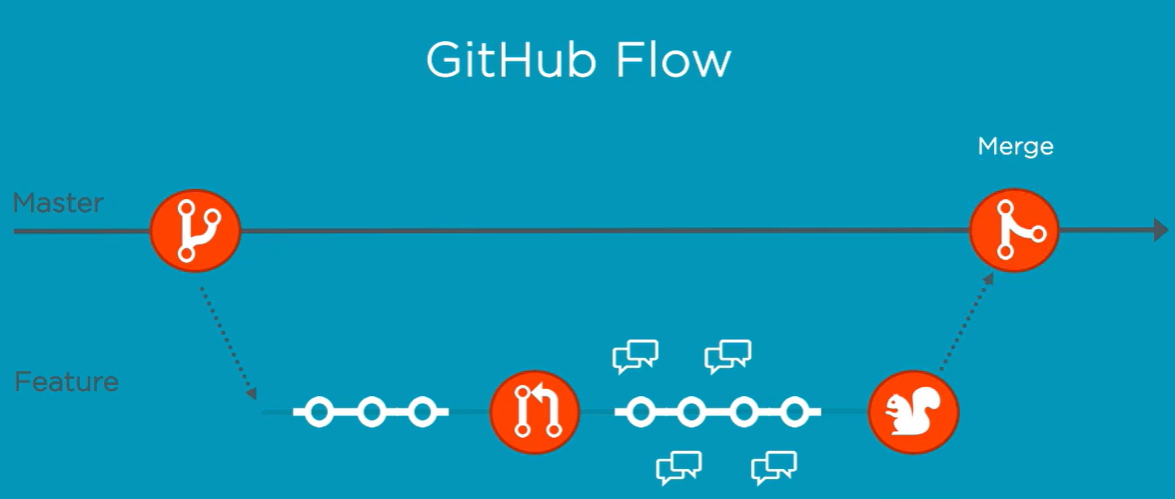
****

Imp point :but with forking, it won’t bring over any branches other than the master copy

**GitHub Actions**

Automation is key for streamlining your work processes and GitHub Actions is the best way to supercharge your GitHub workflow.

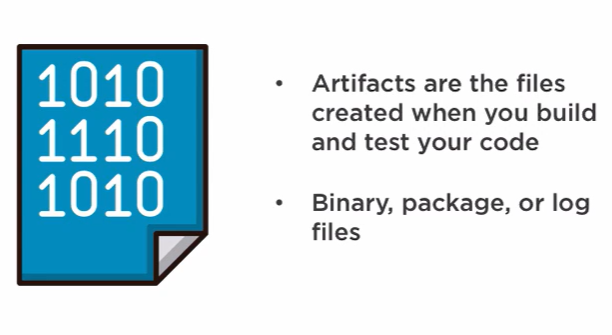
**Basic GitHub Flow**

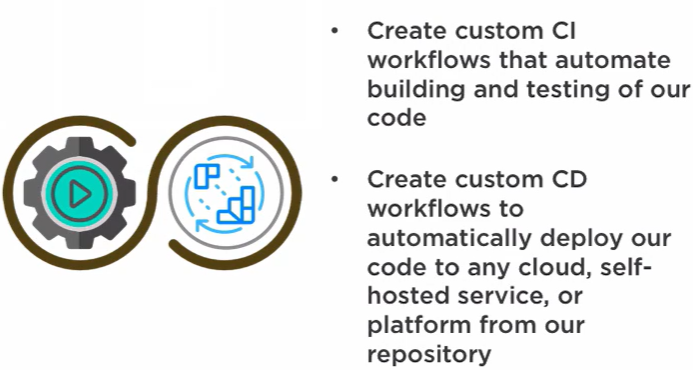


**GitHub Actions Core Concepts Overview**

****

****

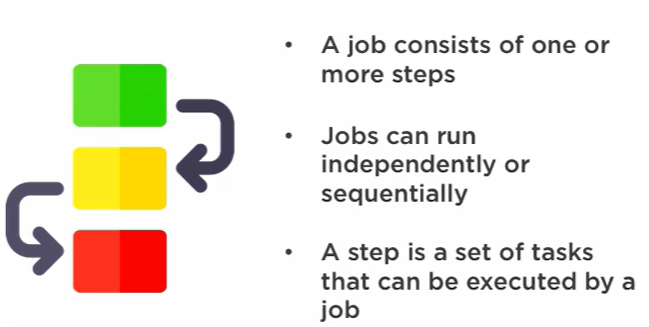
****

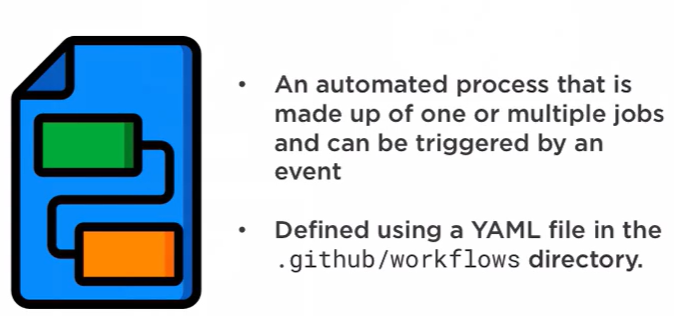
****

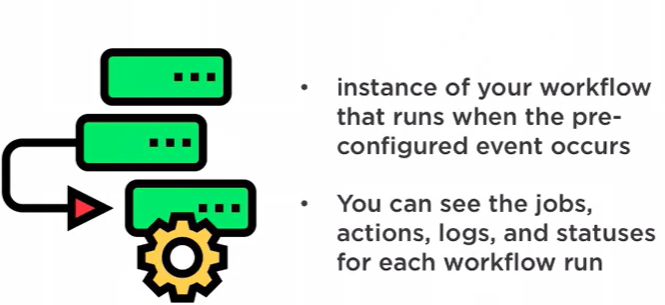
Next is continuous integration and continuous deployment. With GitHub Actions, we can create custom CI workflows that automate building and testing of our code, and we can view the status of our code changes and detailed logs for each action in our workflow. Continuous deployment builds upon continuous integration. When new code is committed and passes any CI tests, the code can be automatically deployed to a production environment. With GitHub Actions, we can create custom CD workflows to automatically deploy our code to any cloud self‑hosted service or platform for our repository. This is an area where GitHub Actions really shines

****

****

****

****

****

Links

https://github-actions-hero.now.sh

Hello git

I am harsha

Checking !...