# Version control concepts & GIT Basics

# About Me

- LinkedIn: https://www.linkedin.com/in/profileatingupta/
- Provides On Job Remote Support to working professionals.
- Providing trainings in various technologies from the last 9 years.
- Worked as a freelance consultant and trainer for various companies:
  - Accenture, HP, IBM, Infosys, Global Logic, HCL, Reckitt Benckiser, R Systems, Hero Motors, ACS Xeros, Subros and many other.
- 18+ years of Total IT experience in multiple technologies:
  - Node.JS
  - Blockchain, Solidity
  - Python, R Programming
  - Machine Learning, NLP
  - Multiple Web development frameworks,
  - C, C++, Linux Administration and Shell Scripting,
  - IBM Lotus Domino
  - Etc……

# Introduction to version control

- Enables multiple people to simultaneously work on a single project.
- Enables one person to use multiple computers to work on a project.
- Integrates work done simultaneously by different team members.
- Gives access to historical versions of project so that if one makes a mistake, s/he can roll back to a previous version.

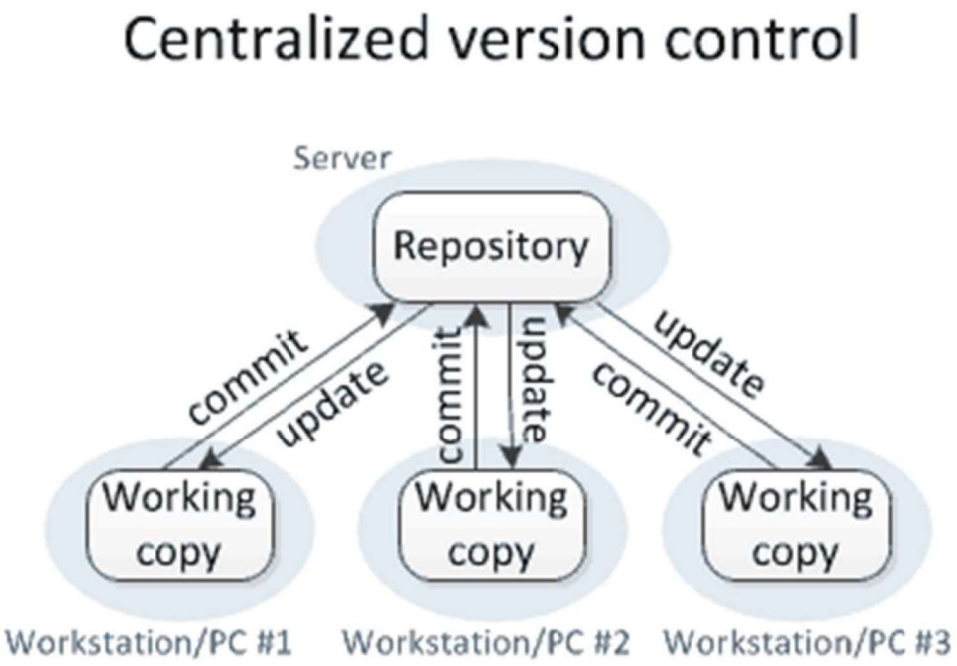
# Why Do We Need A Version Control System (VCS)?

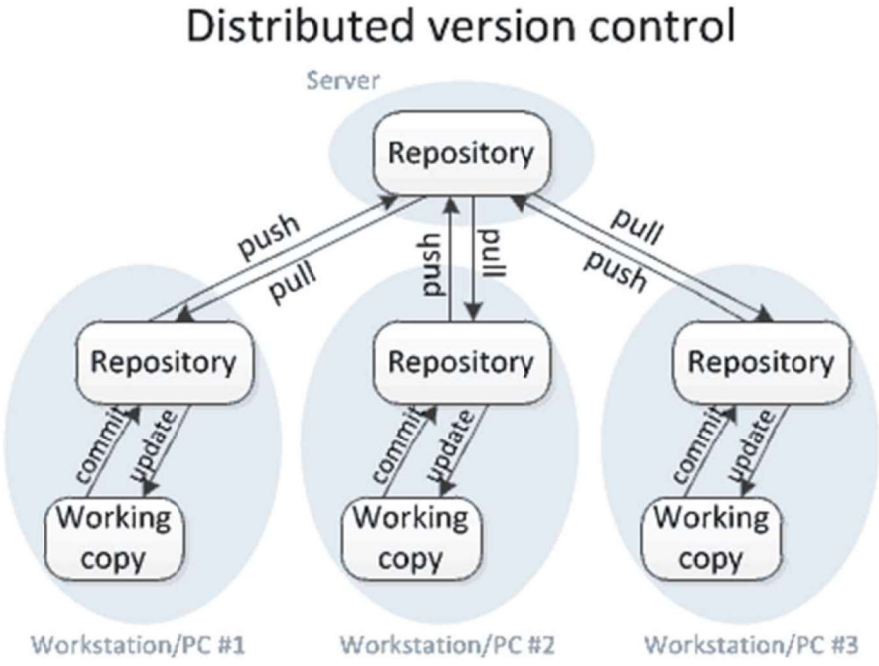
- Backup and Restore
- Synchronization
- Undo
- Track Changes
- Track Ownership
- Sandboxing
- Branching and merging

# Repositories and working copies

- Version control uses a repository and a working copy where we do our work.
- Working copy is personal copy of all the files.
- We changes this copy, without affecting our teammates.
- When we are happy with edits, commit changes to repo.
- A repo/repository is a database of all the edits and historical versions (snapshots) of project.

# Distributed and centralized version control



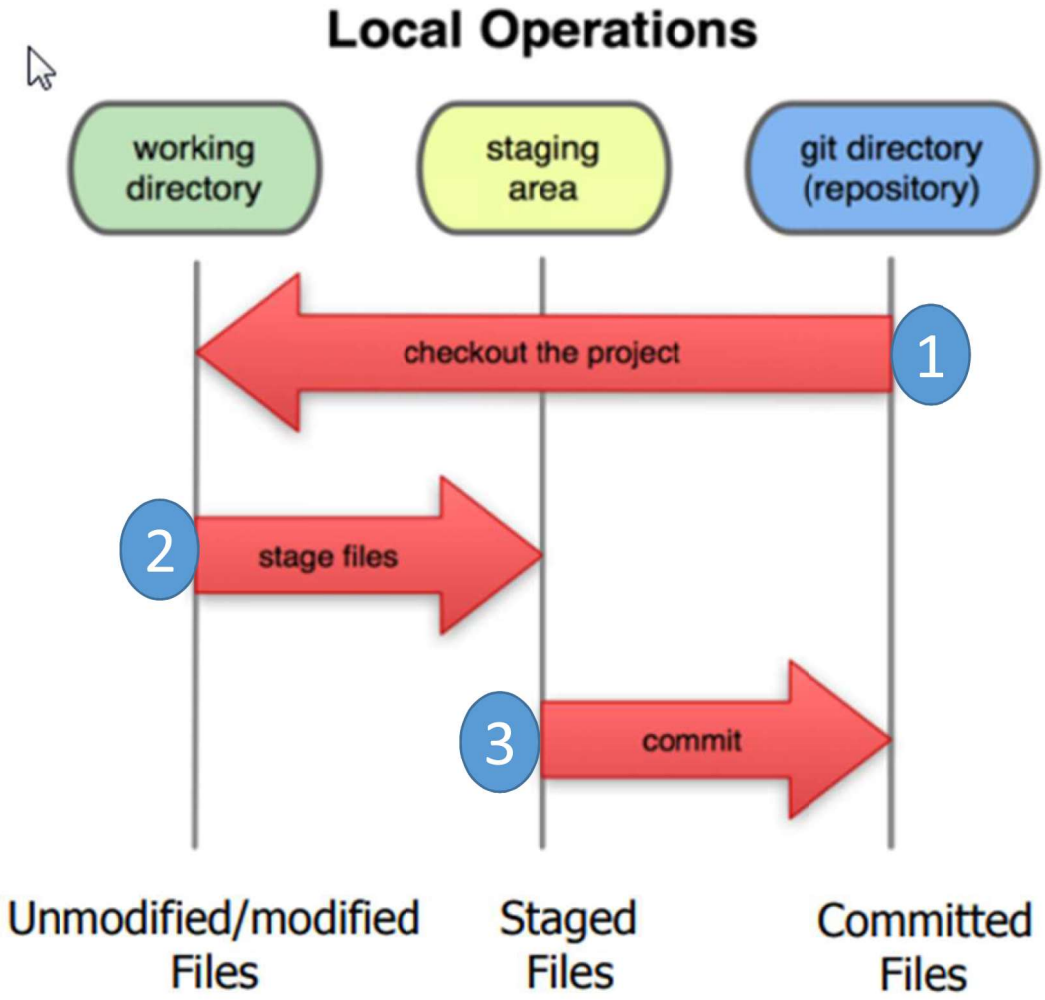

- Subversion/CVS - Just one repository
- Central server is must.
- Spoke and hub structure
- Local changes are not versioned
- Need to communicate with server at each check in/ checkout.

- GIT/ Mercurial - Multiple repositories
- Can be used Offline
- Full history of repository lives on every user's machine
- Peer to Peer model
- Many operations are local

# About Git

- Created by Linus Torvalds - creator of Linux, in 2005
- Goals of Git:
  - Speed
  - Support for non-linear development
    - (thousands of parallel branches)
  - Fully distributed
  - Able to handle large projects efficiently
- • Git website: http://git-scm.com/

# Local git operations



# Untracked, tracked, unstaged and staged changes

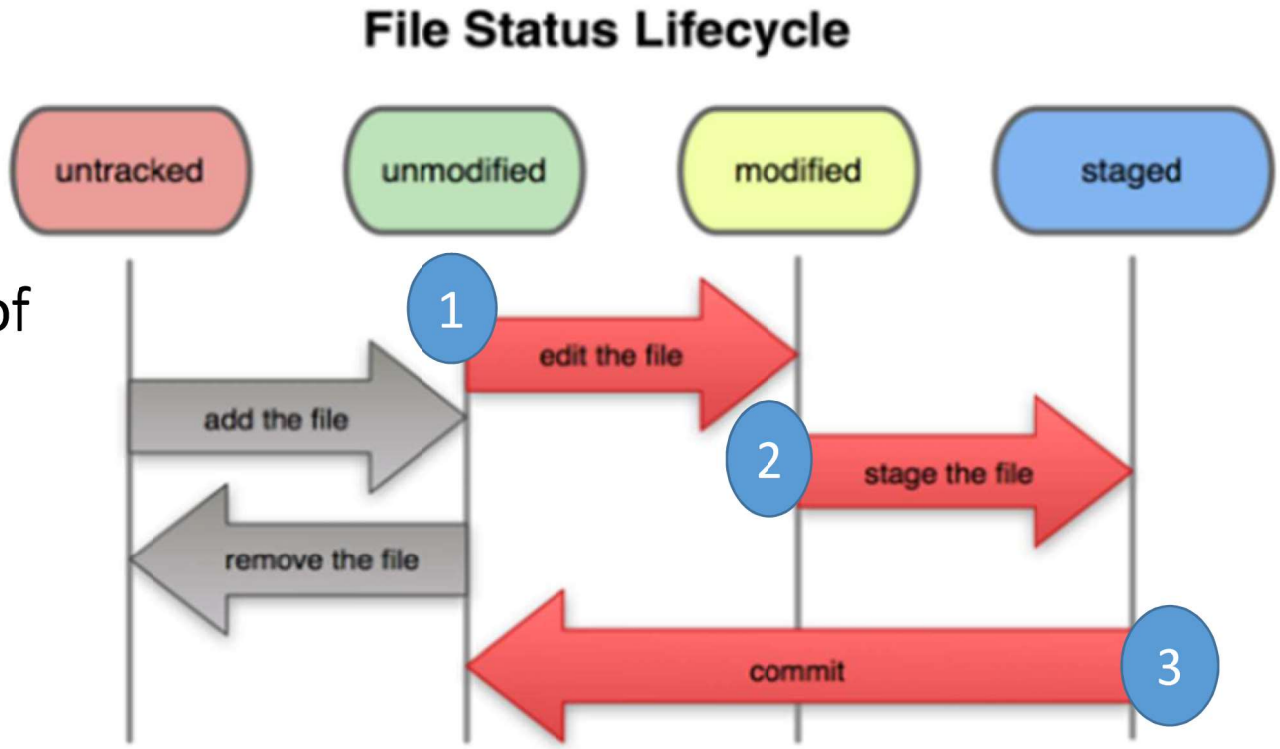
- Untracked changes are not in Git.
  - This file exists locally, but isn't a part of the Git repository.
  - The file's change history will not be recorded and it will not be pushed to remote copies of repository.
  - Untracked files will show up when viewing your Git status.
  - A tracked file becomes untracked when running git rm --cached [file]. This will remove the file from Git while preserving local copy.
- Tracked files are in git
  - Git tracks the file's change history and it will be pushed to remote copies.
  - These files will show up in your Git status report.
  - An untracked file becomes a tracked file when it is added using git add [file]

# Untracked, unstaged and staged changes

- Unstaged changes are in Git but not marked for commit.
  - Unstaged changes exist in working directory, but Git hasn't recorded them into its version history yet.
  - We usually want to stage them using git add <file> command
- Staged changes are in Git and marked for commit.
  - Staged changes are a lot like unstaged changes, except that they've been marked to be committed.
  - Upon next commit, staged changes become part of your Git history.
  - To stage the changes: git add <file>
- Have a last look at all staged changes before I commit
  - git diff --staged

# Basic Git workflow

1. Modify files in the working directory.
2. Stage files, adding snapshots of them to staging area.
3. Commit - takes the files in the staging area and stores to Git directory.



# Terminology

- Repository (repo): The database storing the files.
- Server: The computer storing the repo.
- Client: The computer connecting to the repo.
- Working Copy: Our local directory of files, where we make changes.
- master - the repository's main branch.
- clone - copies an existing git repository, normally from some remote location to local environment.
- commit - submitting files to the repository (the local one); in other VCS it is often referred to as "checkin"

# Terminology

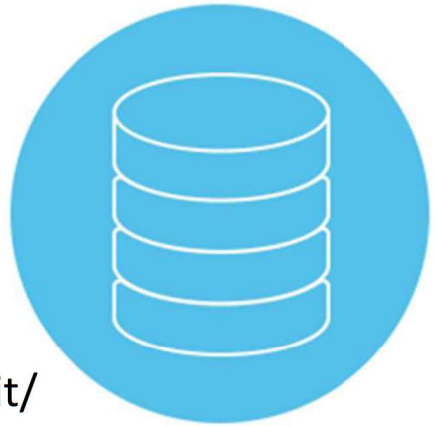
- fetch or pull - is like “update” or “get latest” in other VCS.
- push - is used to submit the code to a remote repository
- remote - these are “remote” locations of repository, normally on some central server.
- SHA - every commit or node in the Git tree is identified by a unique SHA key.
- head - is a reference to the node to which our working space of the repository currently points.
- branch - a particular label on a given node.

# Workstation Setup

- To setup Git simply go to git-scm.com/downloads.
- More detailed information can be found here as well:
  - http://git-scm.com/book/en/Getting-Started-Installing-Git
- After everything is set, the first thing we have to do is to configure git with our name and email:
  - git config --global user.name “My Name"
  - git config --global user.email myemail@gmail.com

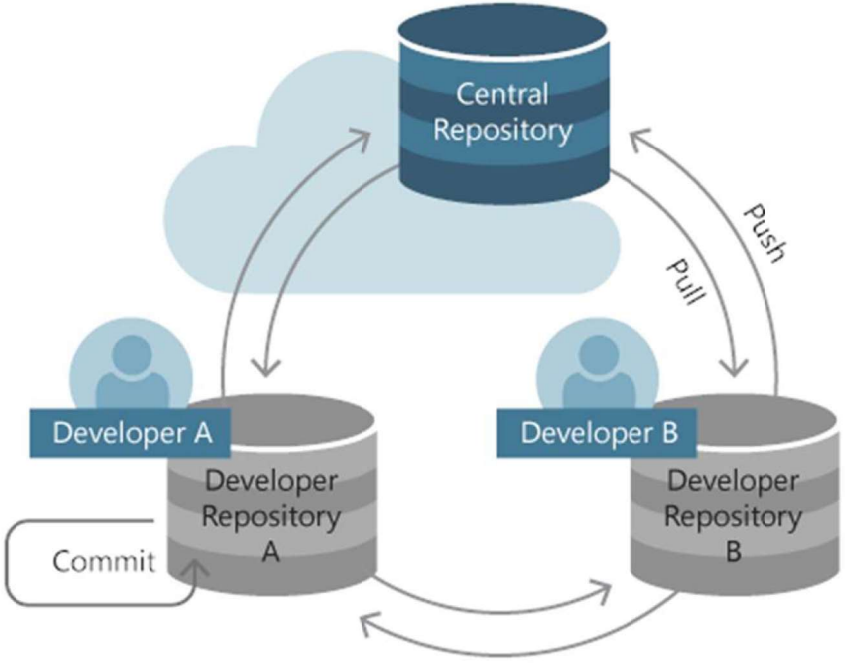
# Lets get started: Create a new Git Repository

- Before starting, lets create a new directory where the git repository will live and cd into it:
  - mkdir mygitrepo
  - cd mygitrepo
- Now we're ready to initialize a brand new git repository.
  - git init
    - Initialized empty Git repository in c:/projects/mystuff/temprepos/mygitrepo/.git/
- We can check for the current status of the git repository by using
  - git status
- Create and commit a new file
  - $ touch hello.txt
  - $ echo Hello, world! > hello.txt



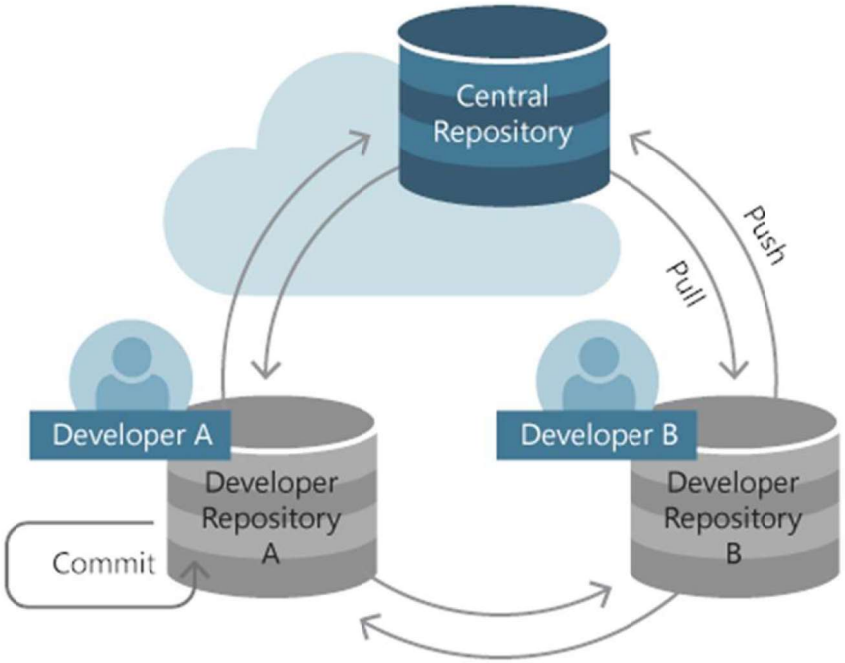
# Lets get started: Create a new Git Repository

- To “register” the file for committing we need to add it to git using
  - $ git add hello.txt
- Checking for the status now indicates that the file is ready to be committed:
  - $ git status
  - # On branch master
  - #
  - # Initial commit
  - #
  - # Changes to be committed:
  - #   (use "git rm --cached <file>..." to unstage)
  - #
  - #       new file:   hallo.txt
  - #



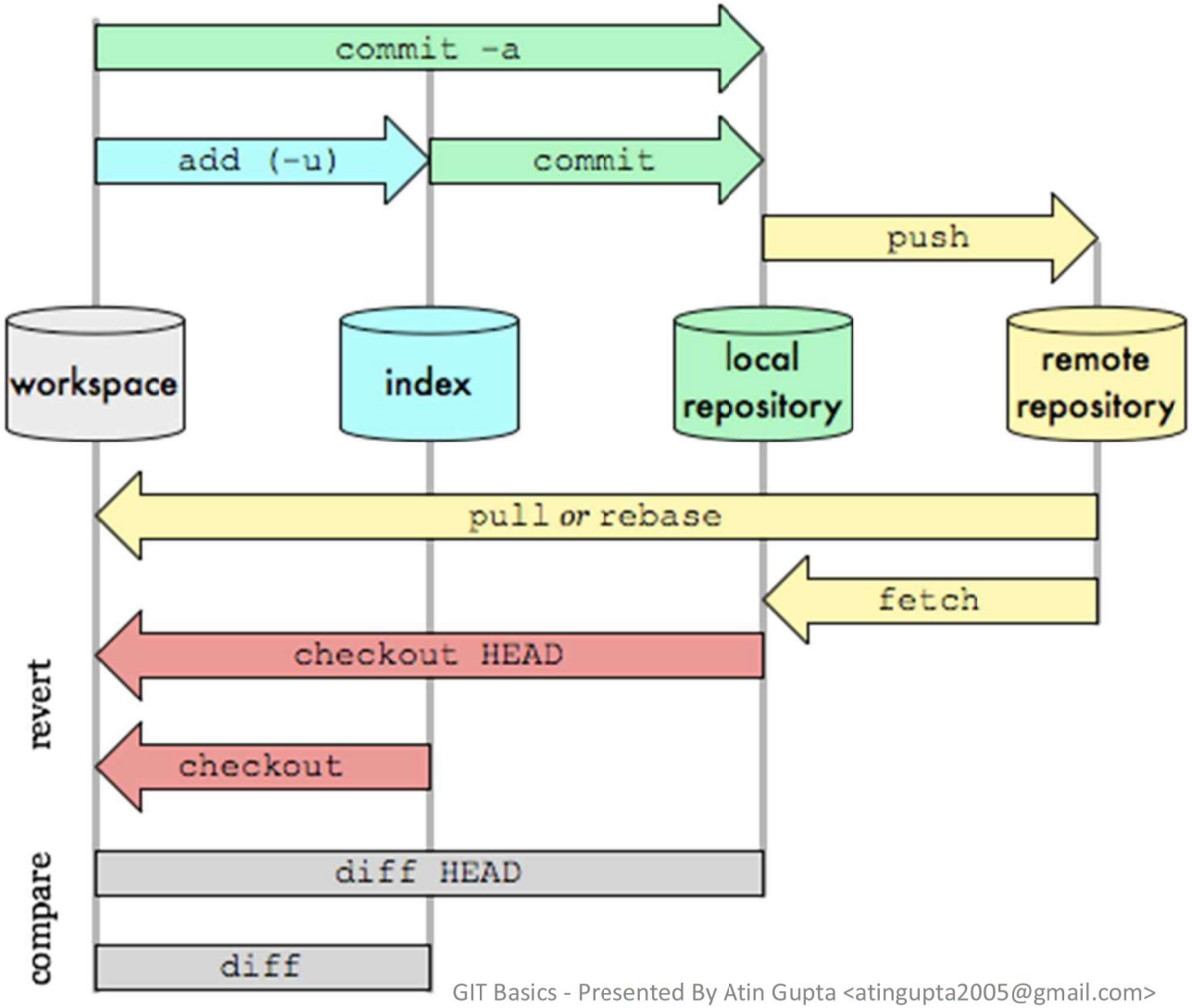
# Lets get started: Create a new Git Repository

- We can now commit it to the repository
  - $ git commit -m "Add my first file"
  - 1 file changed, 1 insertion(+)
  - create mode 100644 hallo.txt



# Git Data Transport Commands

http://osteele.com



# Git commands

| Command | Description |
| --- | --- |
| git clone url [dir] | Copy a Git repository so we can add to it |
| git add file | Adds file contents to the staging area |
| git commit | Records a snapshot of the staging area |
| git status | View the status of our files in the working directory and staging area |
| git diff | Shows diff of what is staged and what is modified but unstaged |
| git help [command] | Get help info about a particular command |
| git pull | Fetch from a remote repo and try to merge into the current branch |
| git push | Push our new branches and data to a remote repository |

# CLONING EXISTING PROJECTS


- The syntax to pull down a local copy of an existing repo is:
  - git clone http://github.com/matthewmccullough/hellogitworld.git
- The clone command performs several subtasks:
  - Sets up a remote (a Git repository address bookmark) named origin that points to the location http://github.com/matthewmccullough/hellogitworld.git
  - Asks this location for the contents of its entire repository.
  - Git copies those objects to the requestor's local disk.
  - Switches to a branch named master
- The local copy of this repo is now ready to have edits made, branches created, and commits issued – all while online or offline.

# DIFF

- A patch-style view of the difference between the currently edited and committed files
  - git diff

# LOG

- The full list of changes since the beginning of time:
  - git log
  - git log --since=yesterday
  - git log --since=2weeks

# BLAME

- To discover why, when and by whom a certain line was added
  - git blame <filename>

# ABORTING

- To abort current uncommitted changes and restore the working copy to the last committed state:
- Discards all of currently uncommitted (unstaged or staged) changes:
  - git reset --hard

# ADDING (STAGING) 

- When ready to put files into the next commit, they must first be staged with the add command.
  - git add .
  - git add *.java

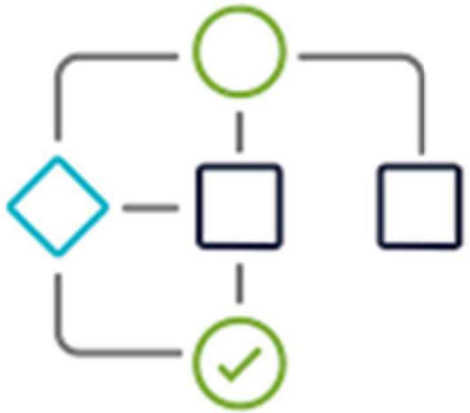
# COMMITTING

- Once all desired files are staged, a commit command saves the pending additions to the local repository.
  - git commit
- To supply the commit message directly at the command prompt:
  - git commit –m “<commit message>”
- To view the statistics and facts about the last commit:
  - git show

# THE REMOTE WORKFLOW

- Working with remote repositories is one of the primary features of Git.
- We can push or pull with colleagues.

- REMOTES
  - A remote called origin is automatically created if we cloned a remote repository. The full address of that remote can be viewed with:
    - git remote v
  - To add a new remote name:
    - git remote add <remote name> <remote address>



# PUSH / PULL

- To put changes from local repo in the remote repo
  - git push origin master

- From remote repo to get most recent changes.
  - git pull
  - git pull <remote name>
  - git pull <remote name> <branch name>


# GitHub

- GitHub.com is a site for online storage of Git repositories.
  - We can create a remote repo there and push code to it.
  - We can get free space for open source projects
- Its not mandatory to use Github to use Git.
  - We can use Git locally for our own purposes.
  - We can also set up a server to share files.
  - We can share a repo with users.


thanks