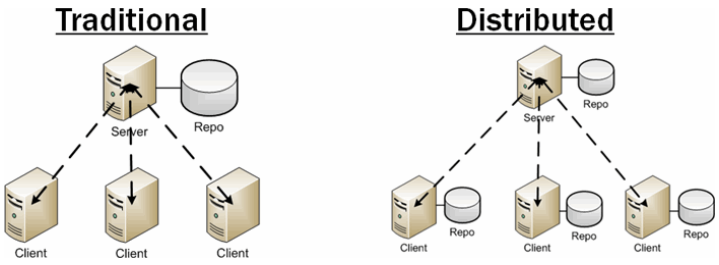
Version Control System

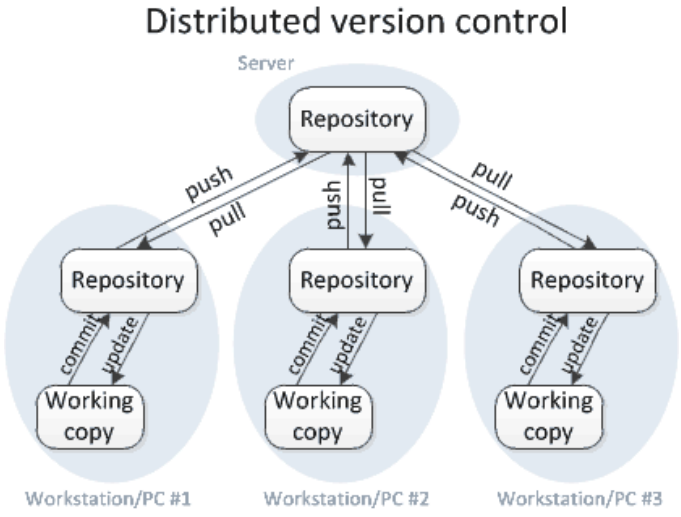
Raghuram PSS | QShore

A version control system (VCS) allows you to track the history of a collection of files. It supports creating different versi ons of this collection. Each version captures a snapshot of the files at a certain point in time and the VCS allows you to switch between these versions. These versions are stored in a specific place, typically called a repository.

	• Manage changes to source code over time • keeps track of every modification to the code in a special kind of database • If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake whi le minimizing disruption to all team members.
Localized Version Control System	• A localized version control system keeps local copies of the files. This approach can be as simple as creating a manual copy of the relevant files. • A centralized version control system provides a server software component which stores and manages the different versions of the files. A developer can copy (checkout) a certain version from the central sever onto their individual computer.
Distributed Version Control	In a distributed version control system each user has a complete local copy of a repository on his individual computer. The user can copy an existing repository. This copying process is typically called cloning and the resulting repository can be referred to as a clone. Every clone contains the full history of the collection of files and a cloned repository has the same functionality as the original repository.



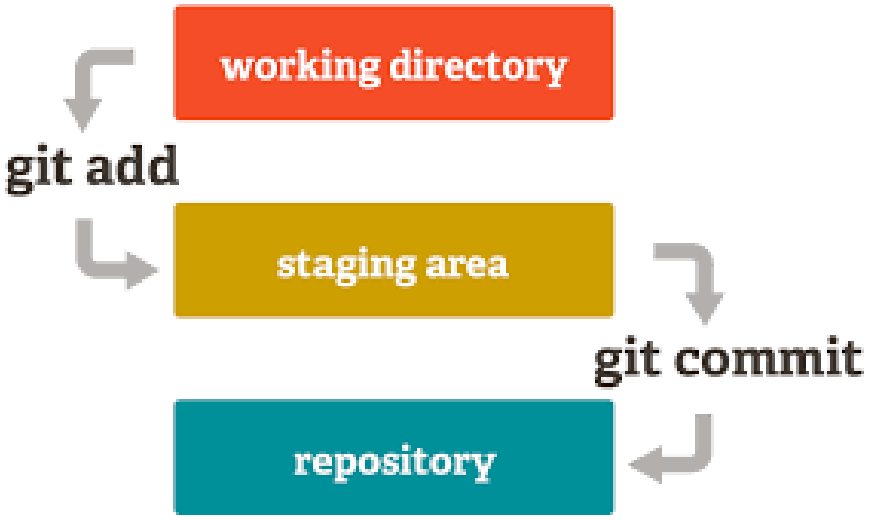
Git	Distributed
Mercurial	Distributed
Apache Subversion (SVN)	No local copy for developers
CVS (Concurrent Versions System)	Centralized



Git Overview

Git is a free and open source distributed VCS and SCM system.
Github is the core hosting platform; central repository

- Git allows and encourages you to have multiple local branches that can be entirely independent of each other.
- **Feature Based Workflow**. Create new branches for each new feature you're working on so you can seamlessly switch back and forth between them, then delete each branch when that feature gets merged into your main line.
- **Git is fas**t. With Git, nearly all operations are performed locally
- **Distributed**: Instead of doing a "checkout" of the current tip of the source code, you do a "clone" of the entire repository.
- **Multiple Backups**: This means that even if you're using a centralized workflow, every user essentially has a full backup of the main server. Each of these copies could be pushed up to replace the main server in the event of a crash or corruption
- **Data Assurance**: Every file and commit is checksummed and retrieved by its checksum when checked back out. It's impossible to get anything out of Git other than the exact bits you put in.
- **Staging Area:** Git has something called the "staging area" or "index". This is an intermediate area where commits can be formatted and reviewed before completing the commit.



Repository	A repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.
Branch	way to work on different versions of a repository at one time.

Git BASH

Git for Windows provides a BASH emulation used to run Git from the command line. *NIX users should feel right at home, as the BASH emulation behaves just like the "git" command in **LINUX and UNIX** environments.

Advantages

Fast and small	As most of the operations are performed locally, it gives a huge benefit in terms of speed. The core part of Git is written in C, which avoids runtime overheads associated with other high-level languages.
Compatible	Can access other repositories like GIT-SVN; Can have GIT as local and SVN as central repository
Non-linear with Branching	A tree structure
Lightweight	Data compression on client side repository
Speed	Local repository helps you fetch the data faster. Every file and commit is check-summed and retrieved by its checksum at the time of checkout
Open Source	
Reliable	Backup is always available in local repositories
Secure	Uses SHA1 to name objects within its database
Implicit backup	Data present on any client side mirrors the repository
Blobs	Blob stands for Binary Large Object. Each version of a file is represented by blob. A blob holds the file data but doesn't contain any metadata about the file. It is a binary file, and in Git database, it is named as SHA1 hash of that file. In Git, files are not addressed by names. Everything is content-addressed
Trees	Tree is an object, which represents a directory. It holds blobs as well as other sub-directories. A tree is a binary file that stores references to blobs and trees which are also named as SHA1 hash of the tree object.
Commits	Commit holds the current state of the repository. A commit is also named by SHA1 hash.
HEAD	HEAD is a pointer, which always points to the latest commit in the branch. Whenever you make a commit, HEAD is updated with the latest commit. The heads of the branches are stored in .git/refs/heads/ directory.

Github - SSH

Pre-requisite: Ensure that you have installed Git for Windows from URL https://git-scm.com/download/win

Connecting to Github from local system using Git Bash

Step #	Description	Command	Response
	Steps 1 to 5 have to be performed in your local machine from GitBash		
1	This creates a new ssh key, using the provided email as a label.	ssh-keygen -t rsa -b 4096 -C "pss.raghu@gmail.com" *Please replace with your emailId*	Generating public/private rsa key pair
2	When you're prompted to "Enter a file in which to save the key," press Enter. This accepts the default file location.	Enter a file in which to save the key (/c/Users/you/.ssh/id_rsa):[Press enter]	
3	At the prompt, type a secure passphrase or press ENTER	DON'T give any paraphrase. Just Press **ENTER**	
	start the ssh-agent in the background	eval "$(ssh-agent -s)"	Agent pid 9048
4	Add your SSH private key to the ssh-agent. If you created your key with a different name, or if you are adding an existing key that has a different name, replace id_rsa in the command with the name of your private key file.	ssh-add ~/.ssh/id_rsa	
5	Copy the SSH key to your clipboard.	clip < ~/.ssh/id_rsa.pub	copies the contents of the id_rsa.pub file to your clipboard
	Step 6 has to be performed in your Github account. **Make sure that you have created a github (https://github.com) account before running the below steps.**		
6	• Login to your github account • In the user settings sidebar (Pull down the dropdown on the top right of the screen and select "Settings" option.) • click SSH and GPG keys. • Click New SSH key or Add SSH key. • In the "Title" field, add a descriptive label for the new key. E.g. **raghu_windows_client** • Paste your key into the "Key" field. • Click Add SSH key. • If prompted, confirm your GitHub password.		
7		ssh -T -ai ~/.ssh/id_rsa git@github.com ssh -T -p 443 git@ssh.github.com	You've successfully authenticated, but GitHub does not provide shel l access.

Git Commands

Git task	Notes	Git commands
Tell Git who you are	Configure the author name and email address to be used with your commits. Note that Git strips some characters (for example trailing periods) from user.name.	git config --global user.name "Raghu" git config --global user.email pss.raghu@gmail.com
Config. List	Lists the entire configuration details	git config --list
To see help	git help <verb> git <verb> --help	git help config git config --help
Create a new local repository		git init
Check out a repository	Create a working copy of a local repository:	git clone /path/to/repository
	For a remote server, use:	git clone username@host:/<path>
Add files	Add one or more files to staging (index):	git add <filename> git add *
Commit	Commit changes to head (but not yet to the remote repository):	git commit -m "Commit message"
	Commit any files you've added with git add, and also commit any files you've changed since then:	git commit -a
Push	Send changes to the master branch of your remote repository:	git push origin master
Status	List the files you've changed and those you still need to add or commit:	git status
Connect to a remote repository	If you haven't connected your local repository to a remote server, add the server to be able to push to it:	git remote add origin <server>
	List all currently configured remote repositories:	git remote -v
Branches	Create a new branch and switch to it:	git checkout -b <branchname>
	Switch from one branch to another:	git checkout <branchname>
	List all the branches in your repo, and also tell you what branch you're currently in:	git branch
	Delete the feature branch:	git branch -d <branchname>
	Push the branch to your remote repository, so others can use it:	git push origin <branchname>
	Push all branches to your remote repository:	git push --all origin
	Delete a branch on your local repository	git branch -d <branchname>
	Delete a branch on your remote repository	git push origin -d <branchname>
Update from the remote repository	Fetch and merge changes on the remote server to your working directory:	git pull
	To merge a different branch into your active branch:	git merge <branchname>
	View all the merge conflicts: View the conflicts against the base file: Preview changes, before merging:	git diff git diff --base <filename> git diff <sourcebranch> <targetbranch>
	After you have manually resolved any conflicts, you mark the changed file:	git add <filename>
	To see all files in the current branch	git ls-files

Create a VM instance (e.g. git-server)	
Assign a public IP (by default a dynamic IP gets assigned, we need to change it to a static public IP) **This is an optional step - NOT MANDATORY**	
Log into the remote server using PUTTY/Git Bash	
sudo apt-get update	downloads the package lists from the repositories and "updates" them to get information on the newest versions of packages and their dependencies
apt-get install git	Installs git
Making changes to the SSH Config in git-serve	
sudo nano /etc/ssh/sshd_config	Edit ssh config in nano editor
PasswordAuthentication no to PasswordAuthentication yes	Enables you to connect to remote server using password
sudo systemctl restart sshd	Restart daemon (ssh server)
Create a user group, user and a bare git repository	
groupadd dev	Creates a group "dev"
useradd -G dev -d /home/git -m -s /bin/bash git	Adds a new user "git" and the new user gets added to the group "dev"
Passwd git	Set a password to git and please remember it
su git	Change the user to git
cd	Changes to the logged in user's (git) folder
pwd	Shows the current directory
mkdir project.git	Creates a directory "project.git"
cd project.git	Changes to the new directory
git --bare init	Installs git in the server (without workspace)

Git Client setup

Create a VM instance in GCP or AWS (e.g. Development)	
Login into client machine using PUTTY/Git Bash	
sudo apt-get install git	Install git server (becomes local git server)
sudo ssh-keygen	ls -al ~/.ssh # Lists the files in your .ssh directory, if they exist Installs SSH private and public key; Go with the default folder and don't give any pass phrase
ssh-copy-id -i git@git-server	Replace "git-server" with public IP of your git server Copies the public key from your client machine to git server. NOTE: This may create some issues; You will have to change the SSH Config file in git-server. Please follow the steps given below in the section
git clone git@git-server:project.git	Replace "git-server" with public IP of your git server; This command clones the repository that we have created in the remote git server.