



Welcome to Bootcamp Trainings

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

Trainer Introduction

- 20+ Years of Experience
- Educational Background : MCA
- Experience with Software Development and Training on different technologies.
- As SDET2 : VS 2008, IE8, Bing, MS Dynamics
- MCTS certified
- Key Skills : UI Technologies, MEAN/MERN, DotNet Fullstack
- Given valued added trainings for corporate clients.
- Recent Clients : WTW, Capgemini, Deloitte, Salesforce, Aggne, Josys, Sunlife,, Wipro, Prolific, Ness Tech, Cognizant, Godrej, Pega Systems, FIS Global, Aptean, CitusTech etc..

Index (Version Control : Git Essentials)

1. Git basics
2. Branching
3. Merging- merge, conflict, rebase,
4. Pull, push,
5. Commit
6. Code review
7. Practical Assignments
8. Q & A



git



1. What is Version Control and Why It Matters

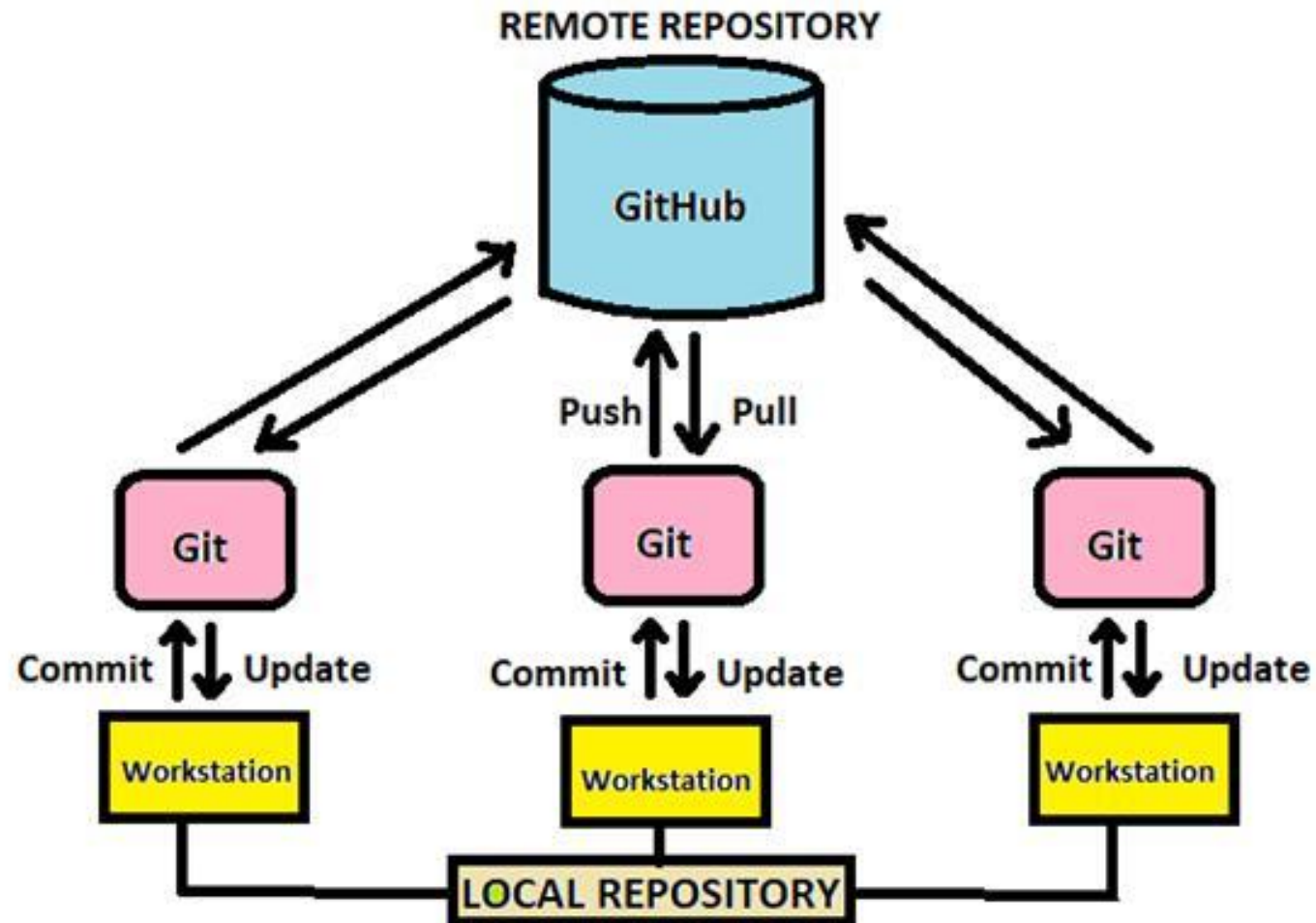
- **Definition:** A system that records changes to files over time, enabling collaboration and tracking.
- **Importance:**
 - Keeps a history of changes (rollback anytime).
 - Enables collaboration across teams.
 - Prevents conflicts and overwriting.
 - Supports parallel development with branching.
 - Ensures code stability with controlled releases.

2. Git Basics

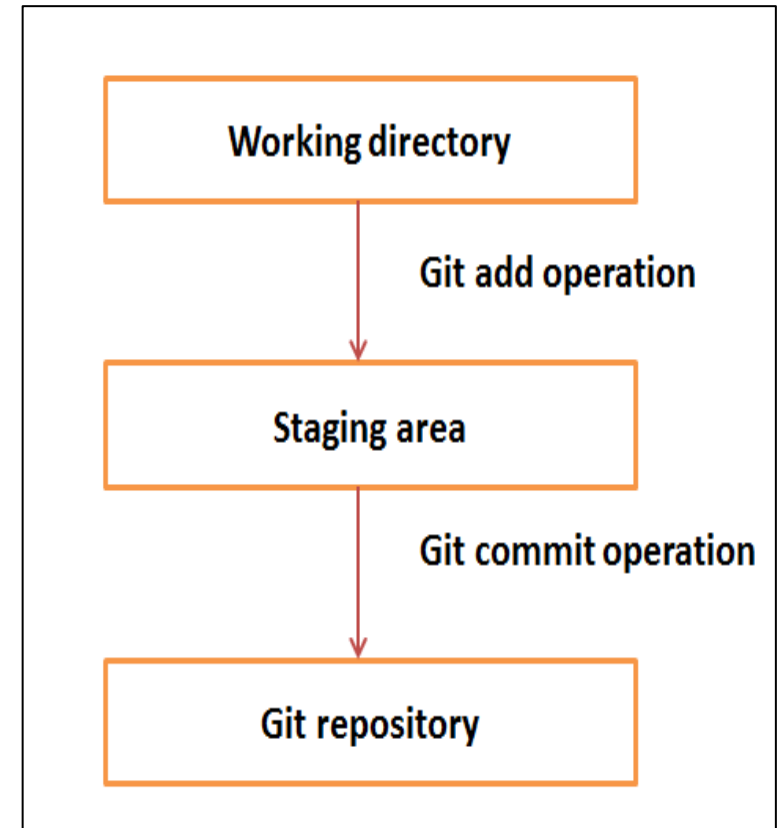
Common Git Commands

- `git init` → Initializes a new Git repository in a folder.
- `git add <file>` → Stages files for the next commit.
- `git commit -m "message"` → Records staged changes with a message.
- `git status` → Shows the current state of the working directory.
- `git push` → Sends local commits to a remote repository (e.g., GitHub).
- `git pull` → Fetches and merges changes from the remote repository.
- `git clone <url>` → Copies an existing remote repo to your machine.

Git Workflow



- Step 1** – You modify a file from the working directory.
- Step 2** – You add these files to the staging area.
- Step 3** – You perform commit operation that moves the files from the staging area. After push operation, it stores the changes permanently to the Git repository.



3. Working with GitHub

- Creating a Repository:
 - On GitHub → *New Repo* → Add name & description → Initialize with README.
- Cloning a Repository:
 - `git clone <repo-url>` → Downloads repo locally.
- Setting Remotes:
 - `git remote add origin <url>` → Links local repo to GitHub.
 - `git remote -v` → View configured remotes.

4. Branching and Merging

- **Branching:**

- `git branch <branch-name>` → Create new branch.
- `git checkout <branch-name>` → Switch branches.
- `git checkout -b <branch-name>` → Create & switch in one step.

- **Merging:**

- `git merge <branch>` → Combine changes from another branch into current branch.

- **Stash:**

- `git stash` → Temporarily saves uncommitted changes.
- `git stash pop` → Restores stashed changes.

- **Rebase:**

- `git rebase <branch>` → Moves commits to base of another branch (cleaner history).

- **Resolving Merge Conflicts:**

- Occurs when changes overlap.
- Git marks conflicts in files → developer edits manually → `git add` → `git commit`.

5. GitHub Flow

- **Forking:**
 - Create your own copy of another repo on GitHub.
- **Pull Requests (PRs):**
 - Propose changes from your branch/fork into main repo.
- **PR Reviews:**
 - Team members review, comment, approve, or request changes.
 - After approval → merge into main branch.

6. Integrating Git with Visual Studio

- **Setup:**
 - Install Git locally.
 - Connect Visual Studio to GitHub account.
- **Common Actions in Visual Studio:**
 - **Clone Repo** → *File* → *Clone Repository*.
 - **Commit & Push** → Use *Team Explorer* / *Git Changes* window.
 - **Branching** → Create/switch branches inside IDE.
 - **Pull Requests** → View, create, and review PRs within Visual Studio.
 - **Conflict Resolution** → Visual Studio provides GUI tools to merge conflicts.

Summary / Key Takeaways

- **Version control** is essential for collaboration, tracking, and safe experimentation.
- **Git basics** (init, add, commit, push, pull, clone) form the foundation.
- **GitHub** adds remote collaboration features (repos, forks, PRs, reviews).
- **Branching, merging, and rebasing** enable parallel development.
- **GitHub flow** (fork → branch → commit → PR → review → merge) is standard for collaboration.
- **Visual Studio** integrates Git seamlessly for developers who prefer a GUI.

Q & A

Narasimha Rao T

tnrao.trainer@gmail.com