Basic Git & GitHub for DevOps Engineers:

TABLE OF CONTENTS

What is Git?

GIT is an open source Distributed Version Control System (DVCS) which records changes made in your project or a set of files laying emphasis on speed, data integrity and distributed, non-linear workflows.

Git is one of the best version control tools available in the present market. This emerging star was first developed by Linus Torvalds, the creator of the Linux kernel. There is no singular centralized code base that the code can be pulled from, and different branches are responsible for hosting different areas of the code.

Git is fast and efficient and is used by system administrators and open-source projects to power their repositories.

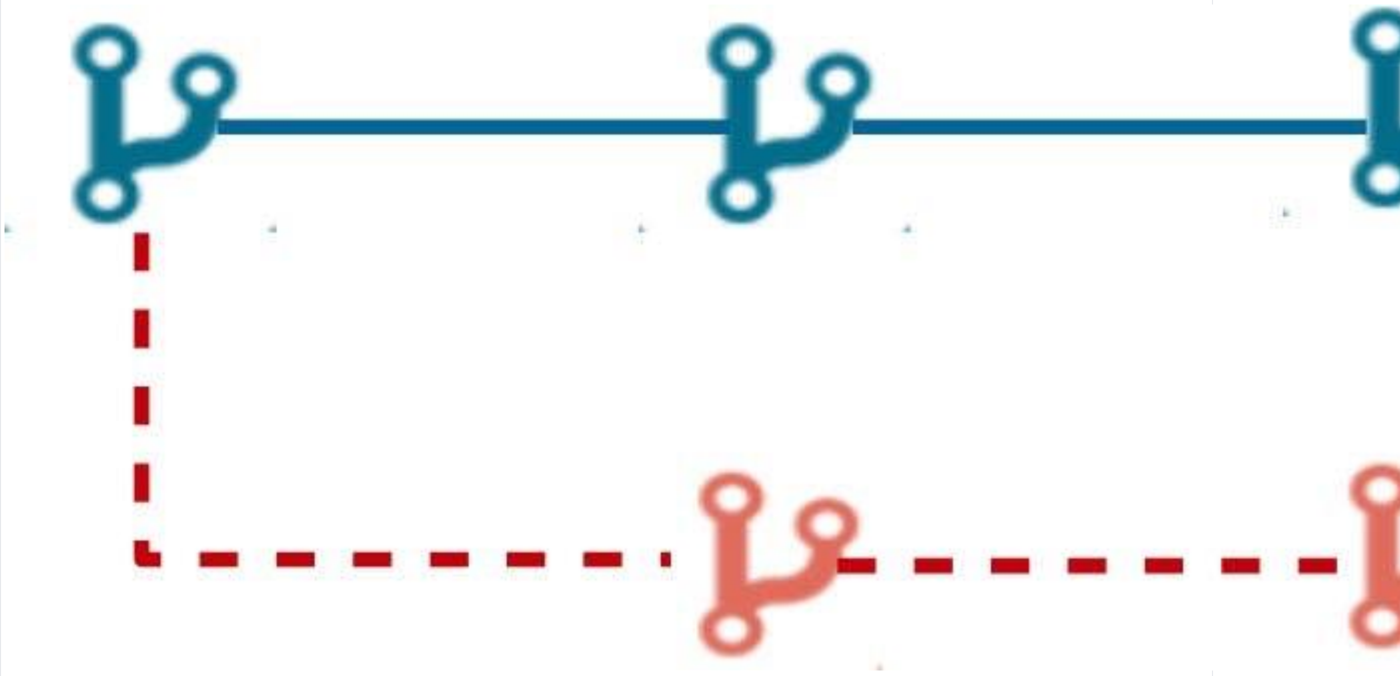
Goals of Git:

- **Speed**
- **Support for non-linear development**
- **Fully distributed**
- **Able to handle large projects efficiently**

Each update in the code is recorded in Git's history, which is stored in a data structure called a Git repository. This repository is the core of Git.

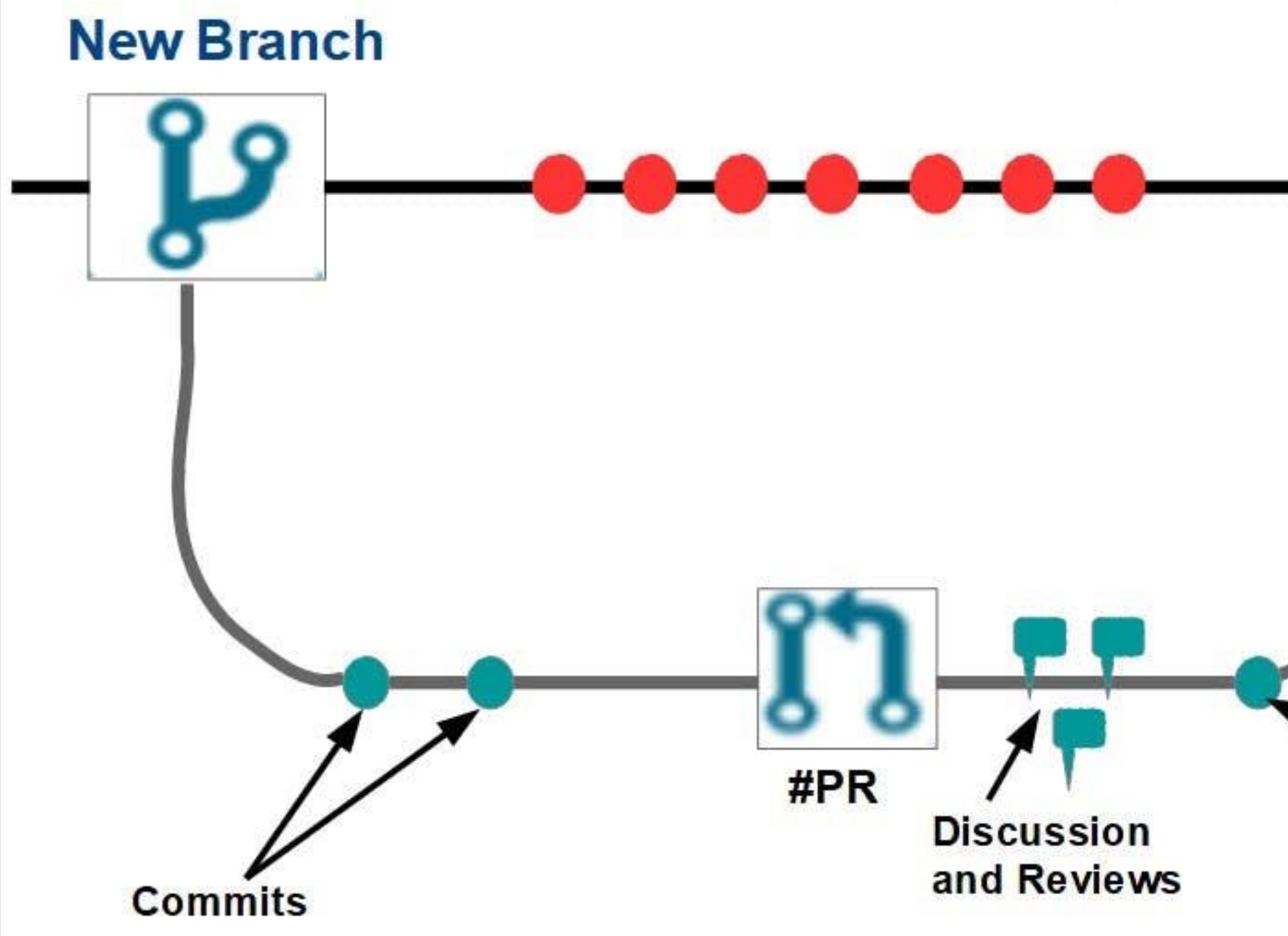
Why Git?

Feature branch workflow



Each branch function in an isolated environment while updates are made in the codebase. This way, its ensured that the master branch always has the production-quality code

Pull Requests



- A developer calls a pull request to ask another developer to merge one of his/her branches into the other's code repository
- Makes it easy for project leaders to monitor and track code changes

Community

- Very popular, widely used, and accepted as standard version control system by the vast majority within the developer's community.
- Large number of Git users make it easy to resolve issues and seek outside help using online forums.

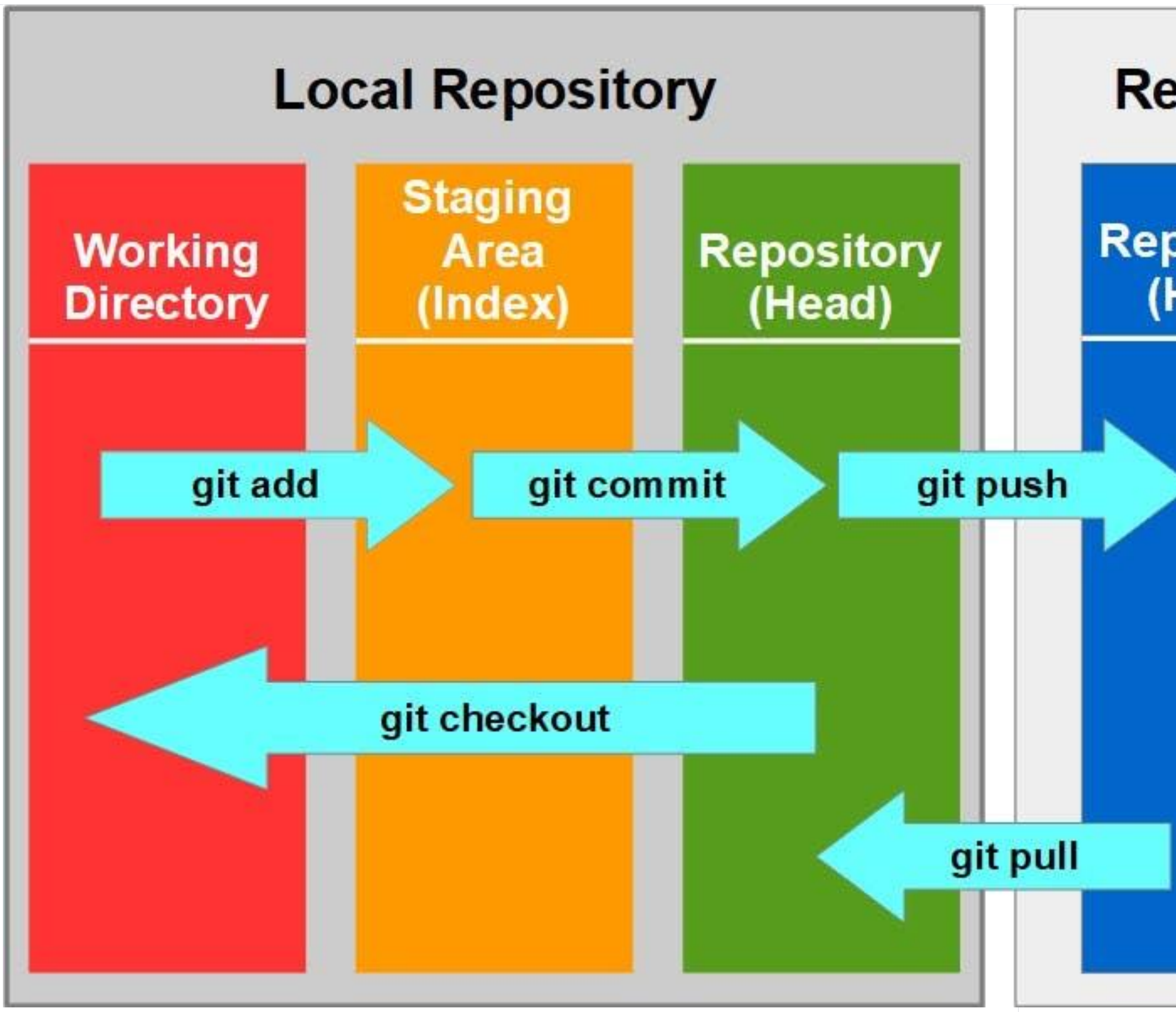
Git Basics

File States

There are three main states that your Git files can reside in:

- **In your local repo (Committed)**
- **HEAD – last commit snapshot, next parent**
- **Checked-out and modified but not yet committed (working directory)**
- **Sandbox – contents unpacked into actual files**
- **Or, in-between, in a “staging” area**
- **Staged files are ready to be committed**
- **A commit saves a snapshot of the current state of staging area**
- **Index – proposed next commit snapshot**

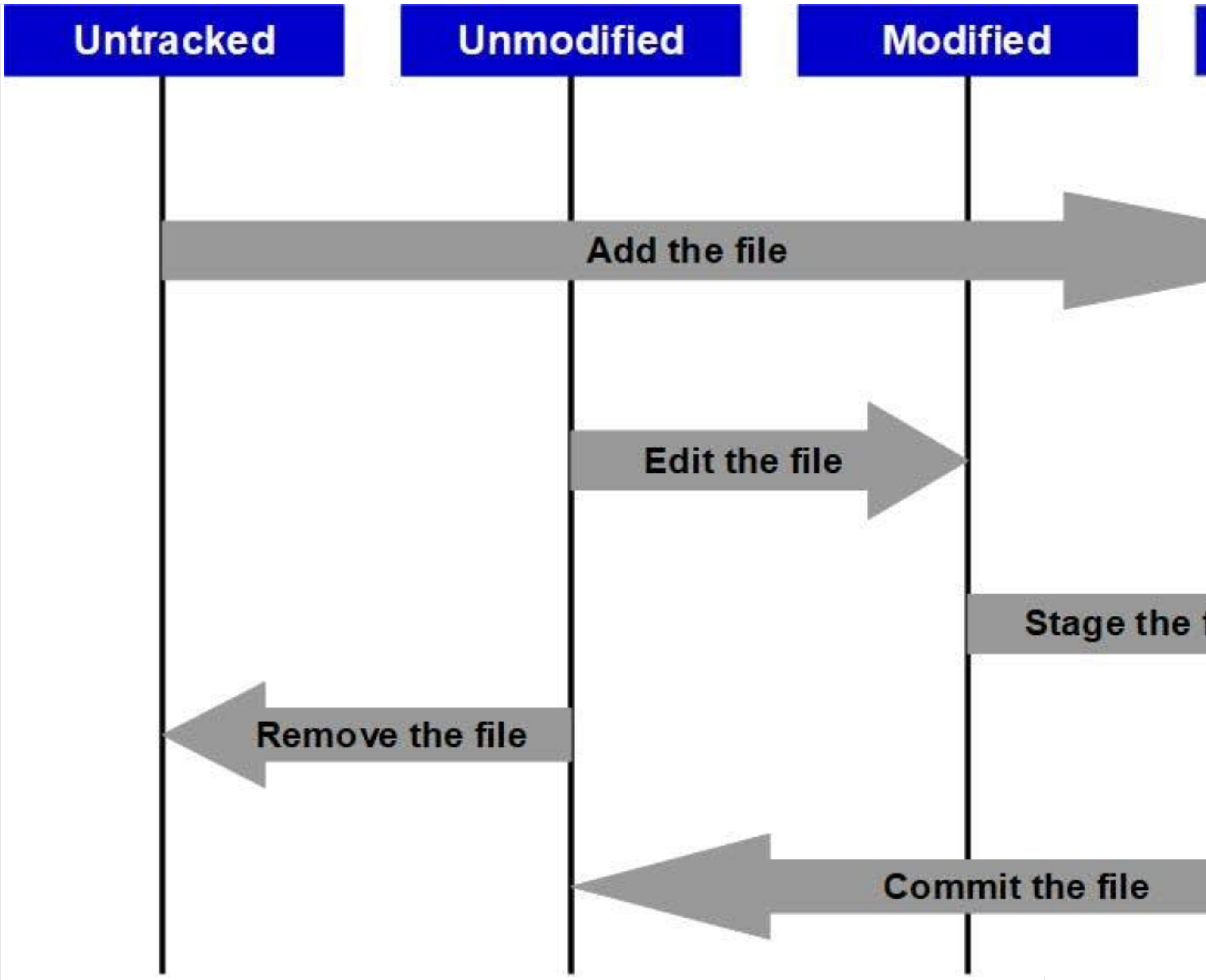
The remote repository (git directory) is the server where all the collaborators upload changes made to the file, it is where the work from all developers gets integrated.



Basic Git Workflow

- **Modify the files in your working directory.**
- **Move the files to the staging area by staging them – adding snapshots of them to your staging area.**
- **Commit, which takes the files in the staging area and stores that snapshot permanently to your Git directory.**

- **Git generates a unique ID which is a 40 character string of hex digits for each commit and refers to the commits by this ID rather than the version number**



Branching & Merging

- **A branch in Git is lightweight and the pointer always points to one of the commits.**
- **The default branch name in Git is “master”.**
- **Each time a commit is made, the master branch pointer moves forward automatically.**

Features Of Git

- **Free and open source:**
  Git is released under GPL (General Public License) open source license. You don’t need to purchase Git. It is absolutely free. As its open source, the source code can be modified as per requirements.

- **Speed:**
  Since there is no dependency on a network connection, the performance is really fast for all operations. Performance tests done by Mozilla have proved that its way faster than other version control systems. Fetching version history from a locally stored repository is much faster than fetching it from the remote server.

- **Scalable:**
  Git is quite scalable. Git can easily handle if the number of collaborators increase. The data stored on the client’s side is very small as Git compresses the huge data through a lossless compression technique.

- **Reliable:**
  Since every contributor has a local repository, if the system crashes, the lost data can be recovered from any of the local repository. There is always a backup of all files. The Git history is stored in a way that the version ID of a particular version (a commit in Git terms) depends upon the complete development history leading up to that commit.

- **Non-linear:**
  Git includes specific tools for visualizing and navigating a non-linear development history and supports rapid branching and merging. A change will be merged more often that it is written, when its passed around various reviewers, is what Git assumes. Branches in Git are very lightweight. A branch in Git is just a reference to a particular commit. With the knowledge about the parental commits, the full branch structure can be constructed.

- **Branching:**
  Branching is a very powerful feature of git. Branch management with Git is very simple. It takes just a few seconds to create, delete or

merge branches. For every change to the codebase, feature branches provide an isolated environment. Whenever a developer wants to work on something, no matter what the size, he will always create a new branch. So, this way it is ensured that the master branch always contains production-quality code.

- **Distributed:**
  Git provides each developer a local copy of the complete development history, and changes are copied from one such repository to another. These changes are imported as additional development branches, and can be merged in the same manner as a local branch.

- **Greater collaboration:**
  A major benefit of the Git workflow is increased collaboration. Git helps teams to talk to each other early and more frequently, enabling them to share feedback and integrate suggestions. The ease with which the Git repositories can be accessed, makes it simple for the users across the organization to collaborate and spot errors quickly, resulting in better and more stable code.

Git: A Giant Leap Toward DevOps

DevOps is about completely removing process bottlenecks, quicker feedback, shortening development cycle and improving overall software lifecycle.

A sensible way to start on the DevOps path is to learn how to use a version control system such as Git. Git is an integral part of DevOps.

Git plays an important role in managing the code that the collaborators commit to the shared repository. This code is then extracted to perform continuous integration, to create a build and test it on the test server and finally deploy it on the production.

Git enables communication between the development and operations team, commit messages play a vital role in information exchange. The deployable bits and pieces all lie in the Version Control system like Git.

Git alone might not be able to overcome the barriers between Dev and Ops but it surely provides some remarkable capabilities that make a big difference to the IT organizations, giving substantial benefits, like :

- **They are about twice as likely to exceed profitability, market share and productivity goals.**
- **Their employees reported far higher levels of job satisfaction.**
- **They realize 50 percent higher market capitalization growth over a three-year period.**

WRITTEN BY Biswaraj Sahoo --AWS Community Builder | DevOps Engineer | Docker | Linux | Jenkins | AWS | Git | Terraform | Docker | kubernetes

Empowering communities via open source and education. Connect with me over linktree: linktr.ee/biswaraj333