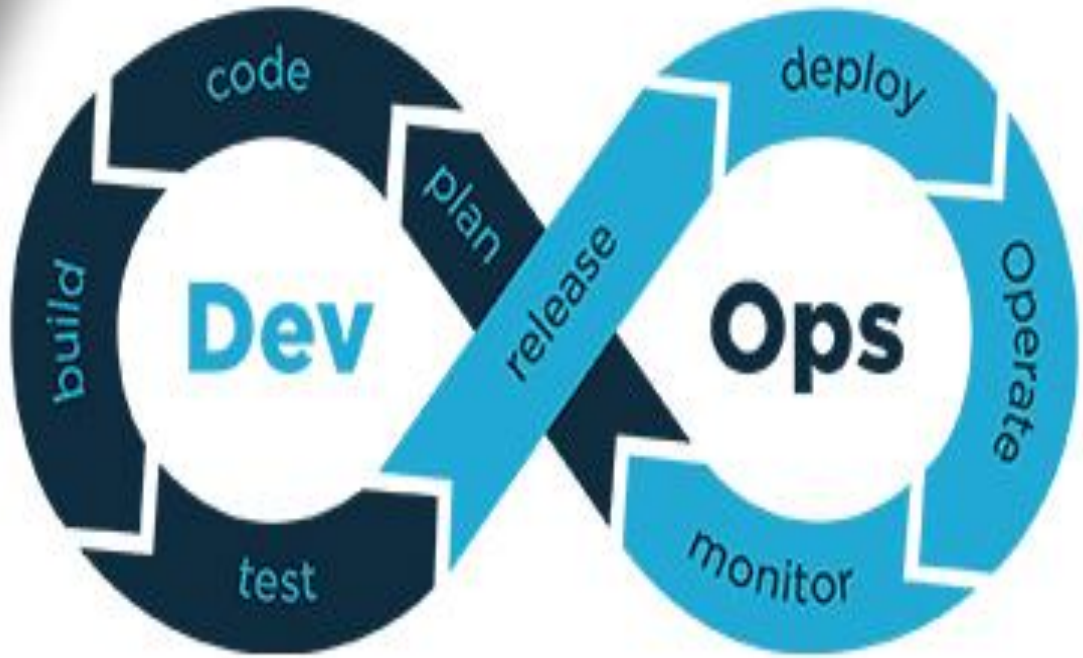
DevOps

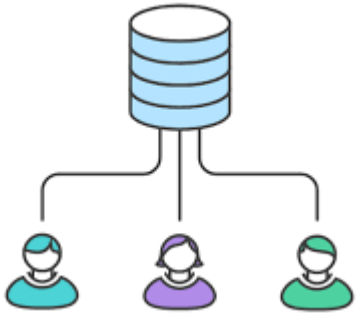
PREPARED BY
ARAVIND KUMAR G.K
Senior DevOps Engineer

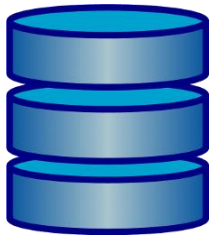


Training
Repository Management

AGENDA

- What is Repository?
- Branch , Trunk and Tags
- Version Control System
- Types of VCS
  - Centralized version control system (CVCS)
  - Distributed/Decentralized version control system (DVCS).
- GIT
- SVN
- Bit Bucket





What is Repository?

- Repository is a central file storage location. It is used by version control systems to store multiple versions of files.
- A repository provides a structured way for programmers to store development files.
- By committing changes to a repository, developers can quickly revert to a previous version of a program if a recent update causes bugs or other problems.
- when a repository is stored on a server, users can "check out" files for editing, which prevents files from being edited by more than one user at a time.

Branch , Trunk and Tags

- A repository contains three primary elements — a trunk, branches, and tags.
- Trunk →The trunk contains the current version of a software project.
- Branch →Branches are used to store new versions of the program. A developer may create a new branch whenever he makes substantial revisions to the program.
- Tags → Tags are used to save versions of a project, but are not meant for active development.

Version Control System

- **Version Control System (VCS)** is a software that helps software developers to work together and maintain a complete history of their work.

Listed below are the functions of a VCS –

- Allows developers to work simultaneously.
- Does not allow overwriting each other’s changes.
- Maintains a history of every version.

Types of VCS

- Centralized version control system (CVCS).
- Distributed/Decentralized version control system (DVCS).

Centralized version control system (CVCS)

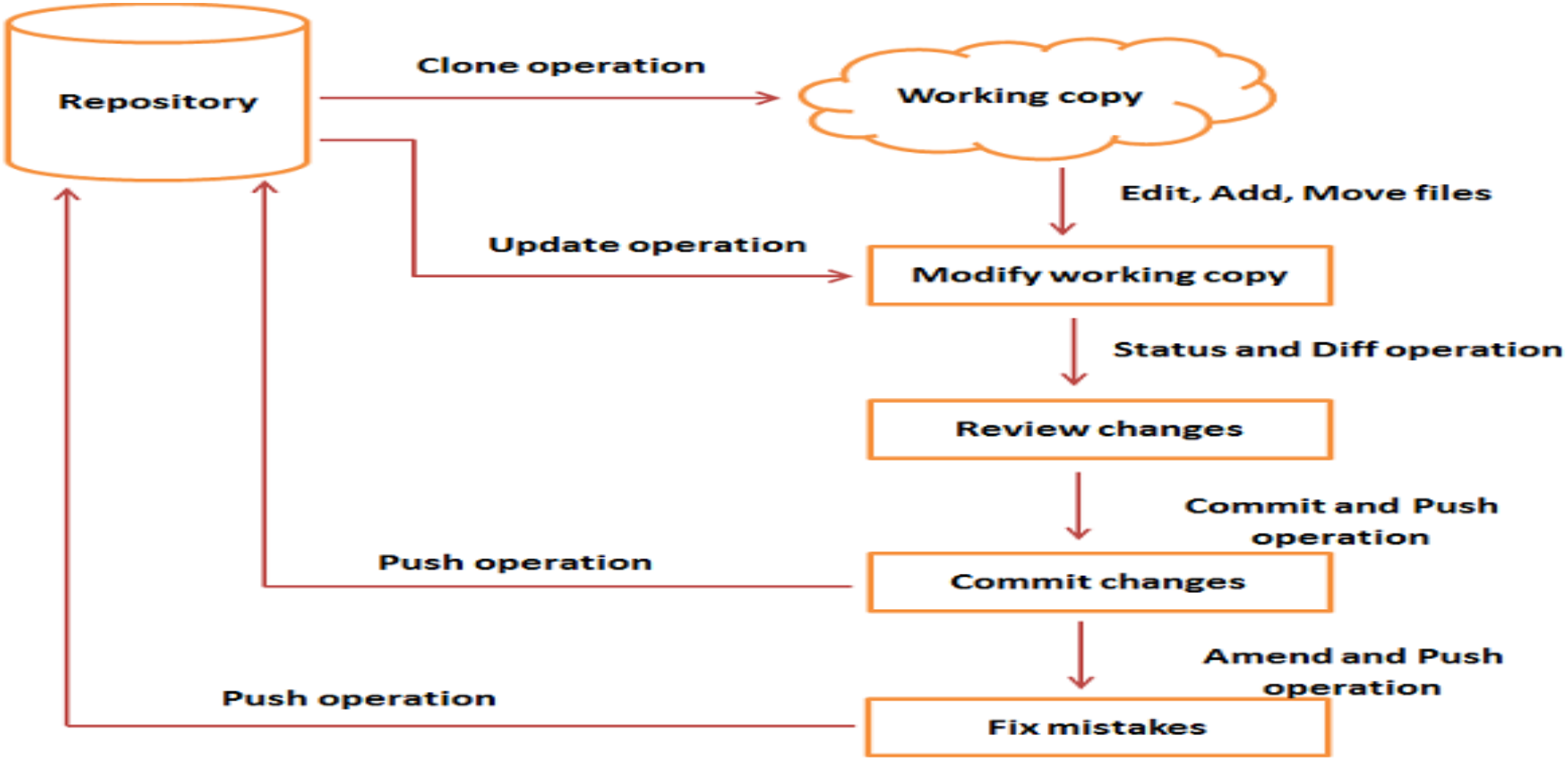
- Centralized version control system (CVCS) uses a central server to store all files and enables team collaboration.
- But the major drawback of CVCS is its single point of failure, i.e., failure of the central server. Unfortunately, if the central server goes down for an hour, then during that hour, no one can collaborate at all.
- And even in a worst case, if the disk of the central server gets corrupted and proper backup has not been taken, then you will lose the entire history of the project.

Distributed/Decentralized version control system (DVCS).

- DVCS clients not only check out the latest snapshot of the directory but they also fully mirror the repository.
- If the server goes down, then the repository from any client can be copied back to the server to restore it.
- Every checkout is a full backup of the repository. Git does not rely on the central server and that is why you can perform many operations when you are offline.
- You can commit changes, create branches, view logs, and perform other operations when you are offline. You require network connection only to publish your changes and take the latest changes.

- Git is a distributed revision control and source code management system with an emphasis on speed.
- Git was initially designed and developed by Linus Torvalds for Linux kernel development.
- Git is a free software distributed under the terms of the GNU General Public License version 2.

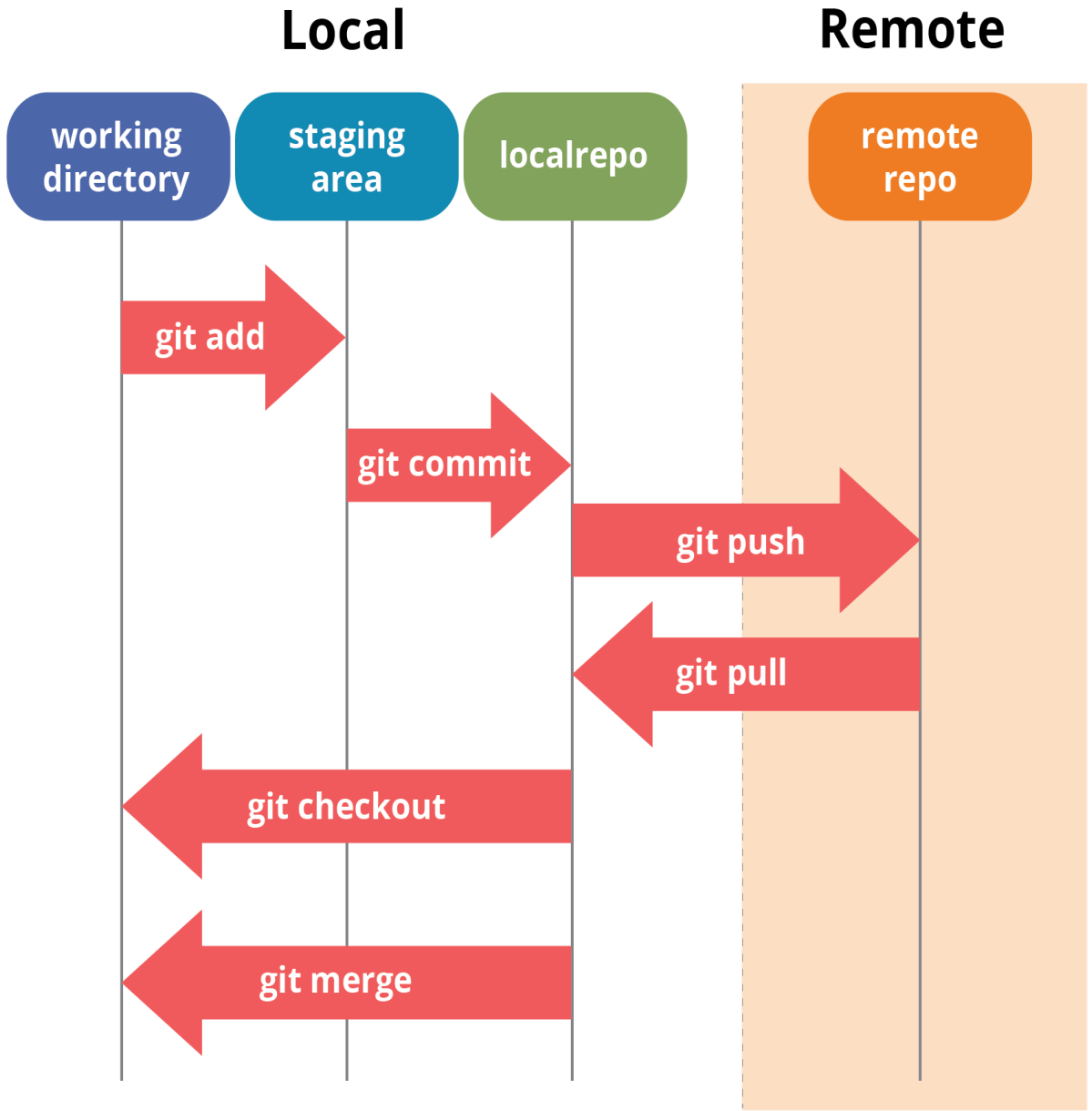
Git - Life Cycle



Common Git Commands

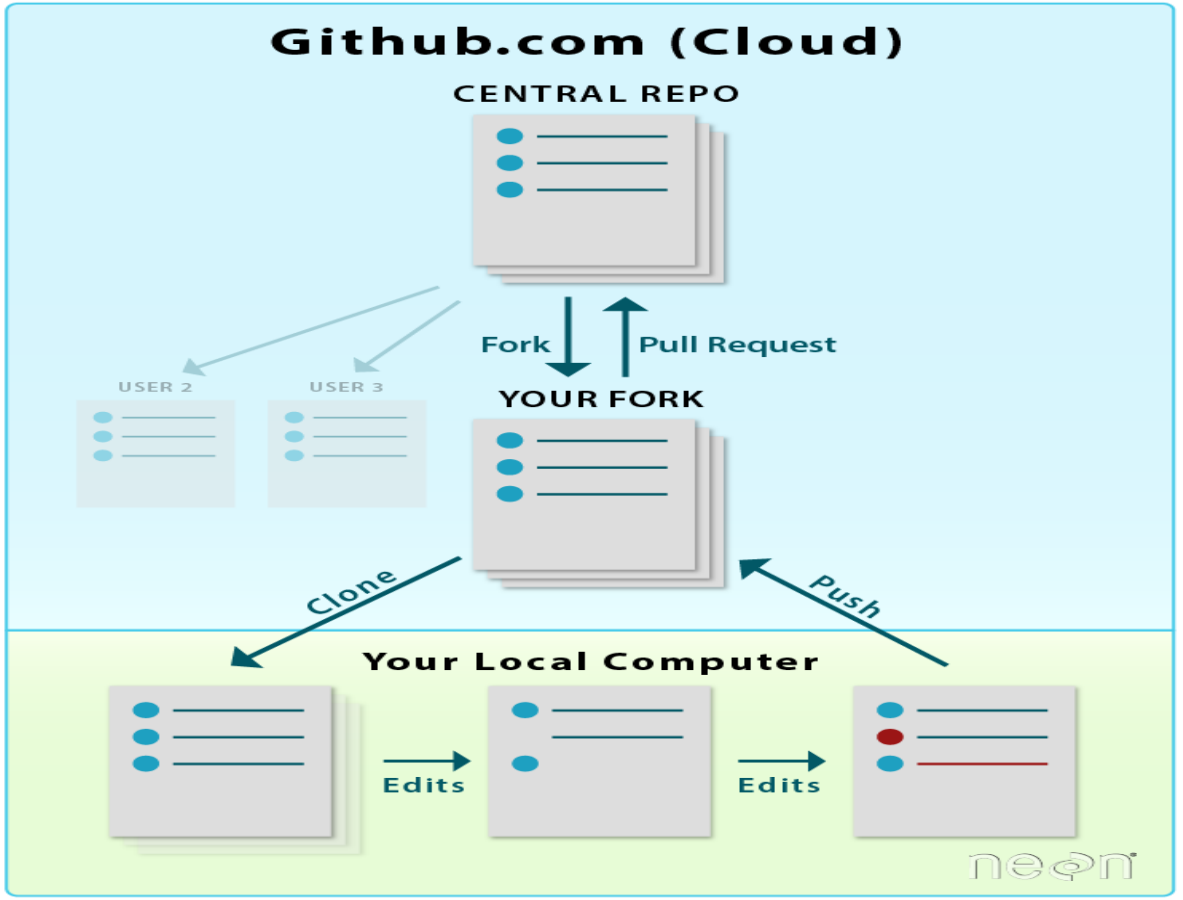
- $git config
- $git init
- $git clone <path>
- $git add <file_name>
- $git commit
- $git status
- $git remote
- $git checkout <branch_name>
- $git branch
- $git push
- $git pull
- $git merge <branch_name>
- $git diff
- $git reset
- $git revert
- $git tag
- $git log

```
git --help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   reset      Reset current HEAD to the specified state
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   branch     List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit     Record changes to the repository
   diff       Show changes between commits, commit and working tree, etc
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
```



Git Fork

- A **fork** is a copy of a repository.
- Forking a repository allows you to freely experiment with changes without affecting the original project.
- Most commonly, **forks** are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.



HANDS ON LABS

- Create a Online GIT Repo
- Add ssh keys of local mechine to GIT HUB
- Clone GIT repository
- Add Source code to Master
- Create a Branches DEV , QA and Prod
- Edit the source code , Add , Commit and Push to Repo.
- Try Various Git commands.