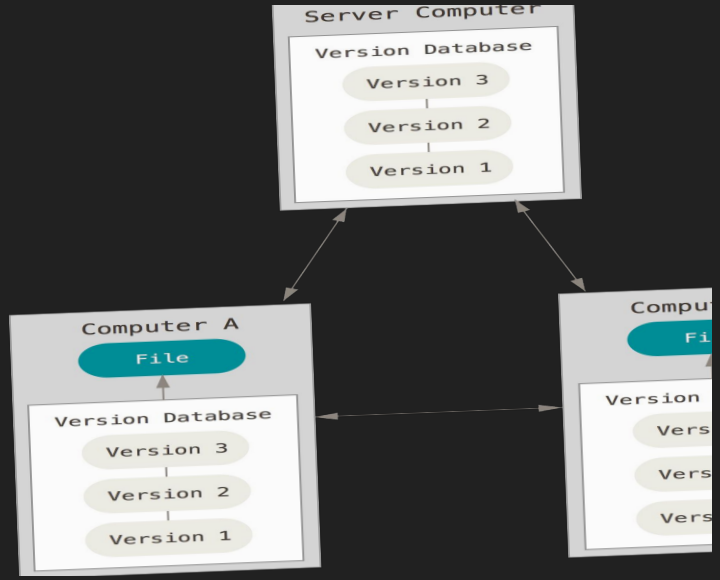
Git

Distributed Version Control System

Version Control System

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

Distributed Version Control



Git Hosting Providers

- GitHub
- GitLab
- Bitbucket
- Beanstalk

Terminology

Init

Clone

Checkout

Branch

Merge

Conflict

Pull, Push

Commit

Init

mkdir project.git

cd project.git

git init

This will create a new repository in the project.git directory

Clone

Cloning means creating a repository containing the revisions from another repository.

git clone <remote-repository-url>

Ex. git clone git://github.com/git/git.git

This will clone (download and create a local git repository) the remote repository.

Checkout

To checkout is to create a local working copy from the repository.

A user may specify a specific revision or obtain the latest.

git checkout <branch-name>

git checkout -b <new-branch-name> <existing-branch-name>

git checkout <file-name>

Branch

A set of files under version control may be branched or forked at a point in time so that, from that time forward, two copies of those files may develop at different speeds or in different ways independently of each other.

git branch

git branch -d <branch-name>

Ex. git branch -d local-branch

Merge

A merge is an operation in which two sets of changes are applied to a file or set of files

git merge <branch-name>

Ex. git merge master

Conflict

A conflict occurs when different parties make changes to the same document, and the system is unable to reconcile the changes.

A user must resolve the conflict by combining the changes, or by selecting one change in favour of the other.

Example Conflict

⇒ git merge master

CONFLICT (content): Merge conflict in _layouts/default.html

Auto-merged index.html

```
<<<<<<< HEAD
        nine
=======
        eight
>>>>>>> master
```

Pull and Push

Copy revisions from one repository into another.

Download changes from remote repository

git pull [options] <repository> <branch-name>

Ex - git pull origin master

Upload changes to remote repository

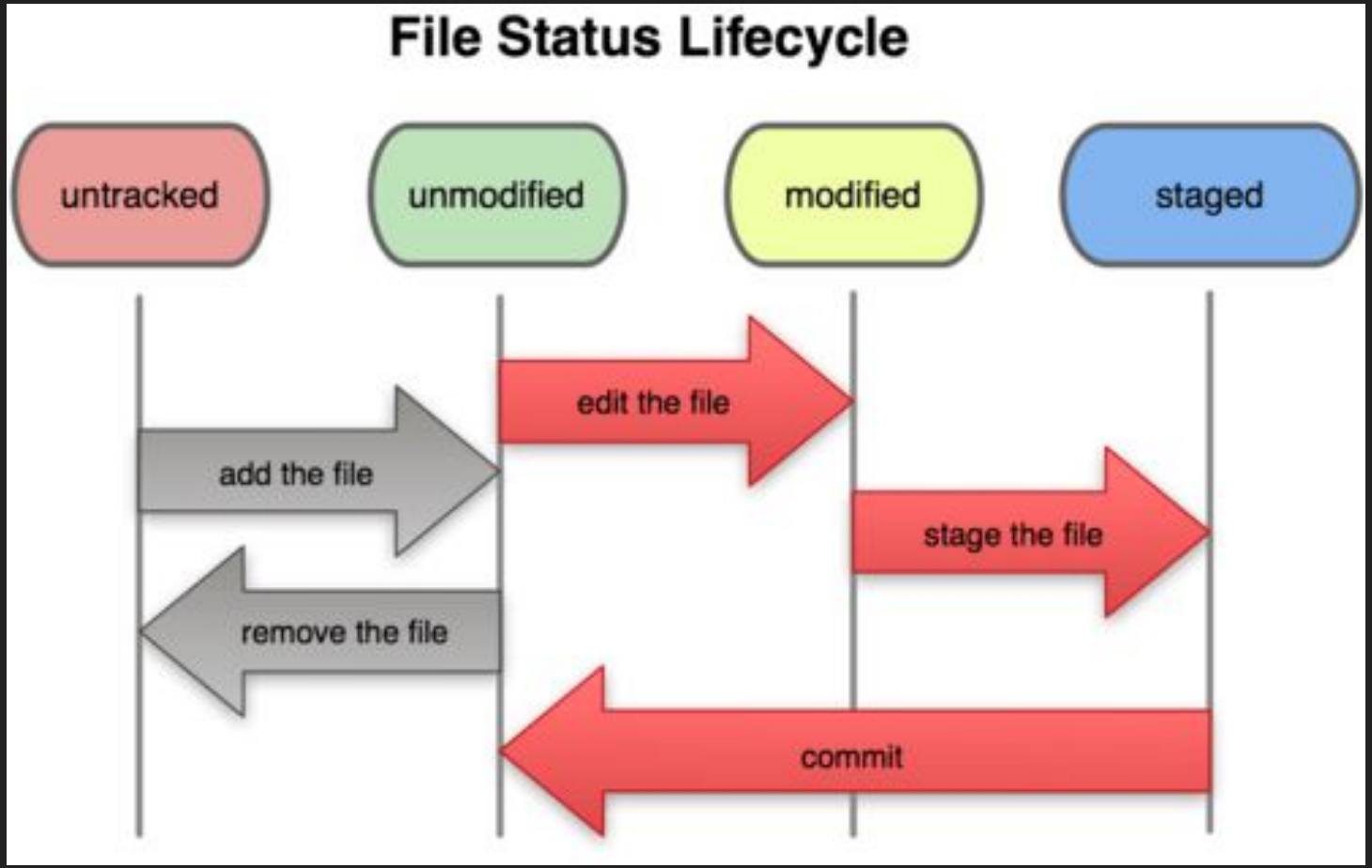
git push [options] <repository> <branch-name>

Ex - git push origin master

Commit

To commit (check in) is to write or merge the changes made in the working copy back to the repository.

- Verify changes
- Add the new or modified files
- Commit the added files


File Status Lifecycle
untracked
unmodified
modified
staged
edit the file
add the file
stage the file
remove the file
commit

Current File Status

⇒ git status

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

modified: index.html

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

modified: home.html

Untracked files:

(use "git add <file>..." to include in what will be committed)

dashboard.html

Verify File Changes

git diff <file-name>

It will show the changes made in the specified file

Adding files

git add <file-name>

git add -- .

This will add the new or modified file to the staged state

Commit

git commit -m “Commit message”

This command will commit the staged files

Commit History

git log

Will list down the commits made in the working branch

Tag

A tag or label refers to an important snapshot in time, consistent across many files.

These files at that point may all be tagged with a user-friendly, meaningful name or revision number.

Git Flow

- git pull origin master
- git checkout -b <your-branch-name> origin/master
- git add <file1> <file2>
- git commit -m “Commit message”
- git checkout master
- git pull origin master
- git checkout <your-branch-name>
- git rebase master
- git checkout master
- git merge <your-branch-name>
- git push origin master

Thank you