Git and GitHub Training

Sunitha C S

Joint Director

STDC, CDAC, Kochi

Project Teams- Past Experiences

When you were an engineering student you are probably used to programming courses that focus on solving relatively small problems, often (but not always) in isolation. Perform a binary search, traverse a graph, implement stacks and queues, etc..

- There will only be one user at a time.
- The code will only be seen by the group members and trashed (be forgotten) after the project is done.
- The project will run only when the instructor has to evaluate it or for testing, probably on the machine of the team member.
- Continuous integration means “continuously integrate new code, compile and test until things work”.
- Testing means “make sure it doesn’t throw a logical or segmentation fault when you do that thing that caused a logical or segmentation fault last time”.

Developing an application in a team

Team Work

- If you work on a project you'll likely be part of a team.
- Different modules of the project may be assigned to each member. Task with the members may be different also.
- As a group effort, the complete development can be done.

Issues developers face when working in a team

- Consider multiple developers work on the same project. Sharing the same stack might yield unexpected results.
- Imagine Alice and Bob start working on the same state, the one on top of the stack. Now imagine they both change the same set of files but in different ways.

- But it be nice if every developer could work on a separate stack and, only once a feature is complete or a bug is fixed, “merge” that stack with the main one. This technique is called “branching”. Because of its popularity and relatively ease of use we’ll take a look at the VCS git.

Introduction to Code Versioning System (CVS)

- A version control system records the changes made to our files over time, in a special database called Repository.
- The name says it all: It is a way of keeping versions of files stored in a repository. In our case these files happen to be text representing the source code of a program, nonetheless a VCS (Version Control System) can be used for all sorts of files.
- That is, imagine that a set of files and directory represents a state in time. Without a VCS you end up overwriting those files and, therefore, rewriting the state. With a VCS, you keep the states as a stack. At any point in time you can retrieve an older state or add a new one on top of the stack.

Why a Version Control System

- Real life projects generally have **multiple developers working in parallel**. So a version control system like Git is needed to ensure there are **no code conflicts** between the developers.
- Additionally, the **requirements in such projects change often.** So a version control system allows developers to revert and go back to an older version of the code.

History of code versioning system

- Different tools available for versioning

SDLC – Software Development Life Cycle

- Software development workflow

Version Control System : Benefits

Collaboration:
The team can work on any file at any instance and later allows to merge all the changes into a common version.

Storage:
Acknowledges that there is only one project whereas all the past versions and variants are neatly packed up inside the VCS

Backup:
Distributed VCS like Git act as a backup

Version Control System : Benefits

In short, a version control system like Git makes it easy to:

- Keep track of code history
- Collaborate on code as a team
- See who made which changes
- Deploy code to staging or production

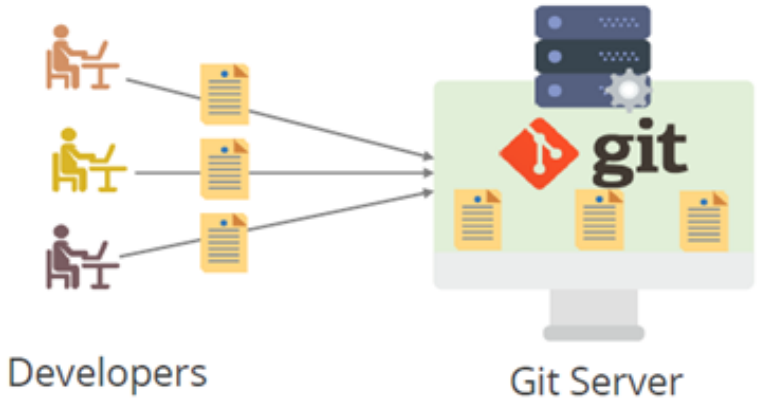
Version Control System : Tools

Software	Repository Model	Concurrency Model	Platforms Supported
CVS	Client-server	Merge	Unix-like, Windows, OS X
Git	Distributed	Merge	POSIX, Windows, OS X
SVN	Client-server	Merge or lock	Unix-like, Windows OS X
Mercurial	Distributed	Merge	Unix-like, Windows, OS X
Monotone	Distributed	Merge	Unix-like, Windows, OS X

Introduction to git

- Git is an **Open Source Distributed Version Control System**.

Git is a Version Control System for tracking changes in computer files. It is generally used for source code management in software development.



Tracks changes in the source code

Uses distributed version control tool for source code management

Allows multiple developers to work together

Supports non-linear development because of its several parallel branches

Git History

Linus Torvalds
(Creator of Linux)

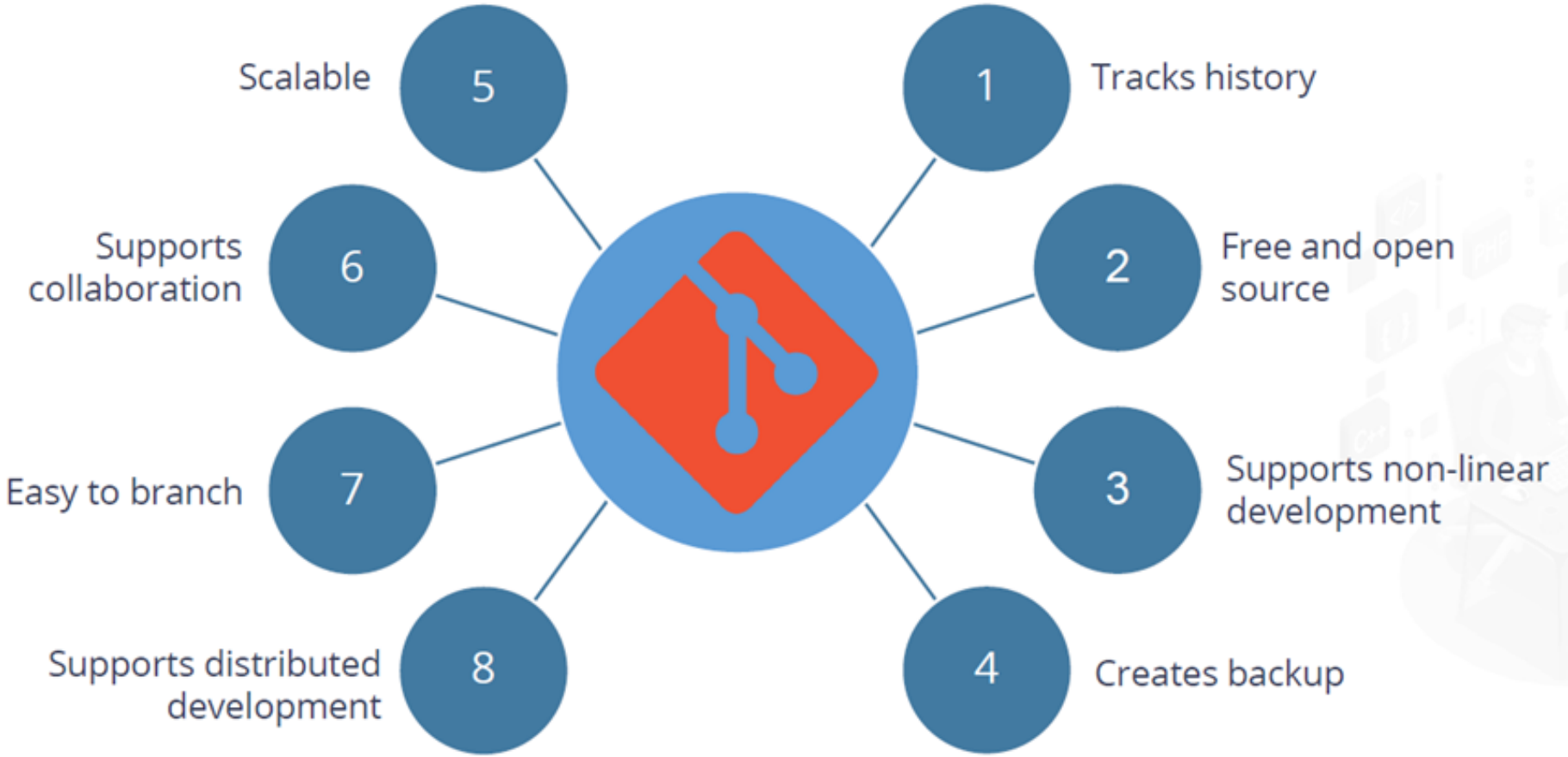
Initially, it was developed to manage the Linux development community

Originally, it was written in C language and then re-implemented in other languages

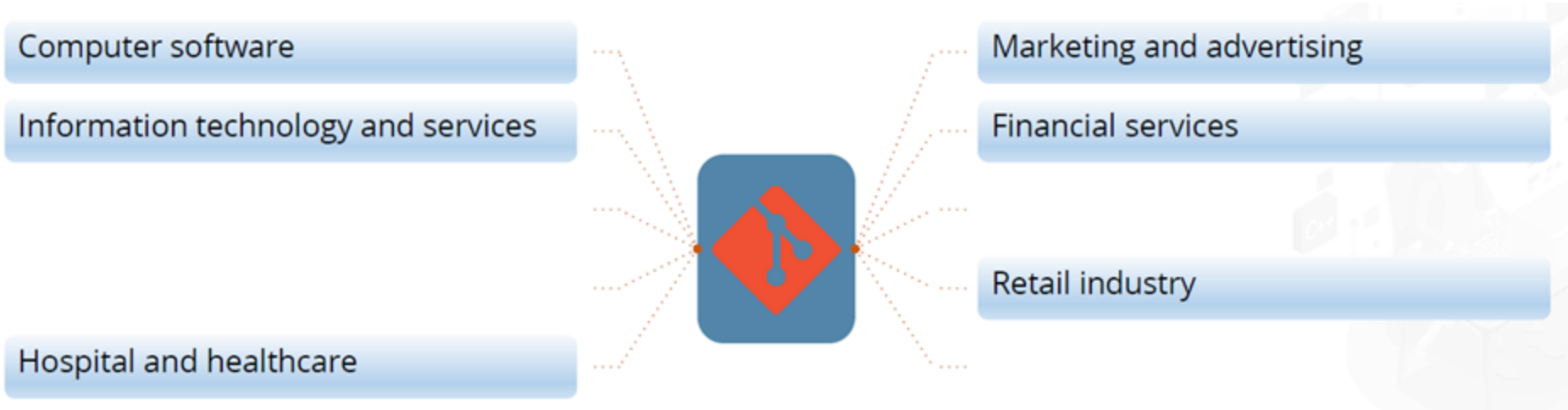
What is Git?

Git is a version control system for tracking changes in computer files. It is generally used for source code management in software development.

Features of Git



Who Uses Git?


Computer software
Information technology and services
Hospital and healthcare
Marketing and advertising
Financial services
Retail industry

Git Basics


#include <iostream>
#include <cstdlib>
using namespace std;
int main()
{
cout << "Hello, world!" << endl;
return 0;
}

Version Control Systems : Details

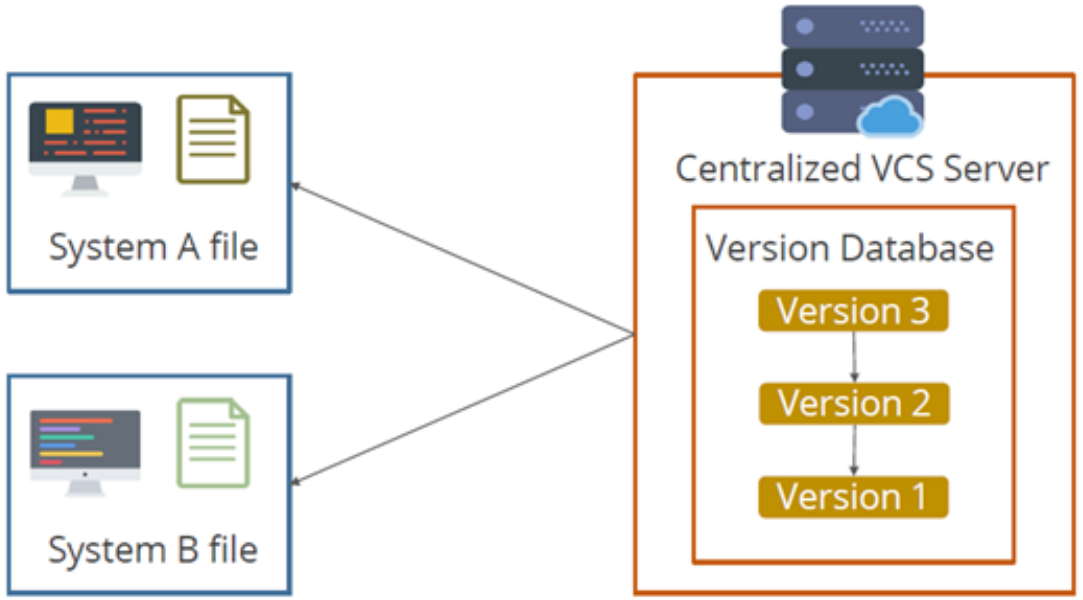
- Version control is a system that records changes to a set of files over a period of time to recall specific versions.
- Version Control System (VCS) can be used to store every version of an image or layout.

For example:

Version Control Systems : Details

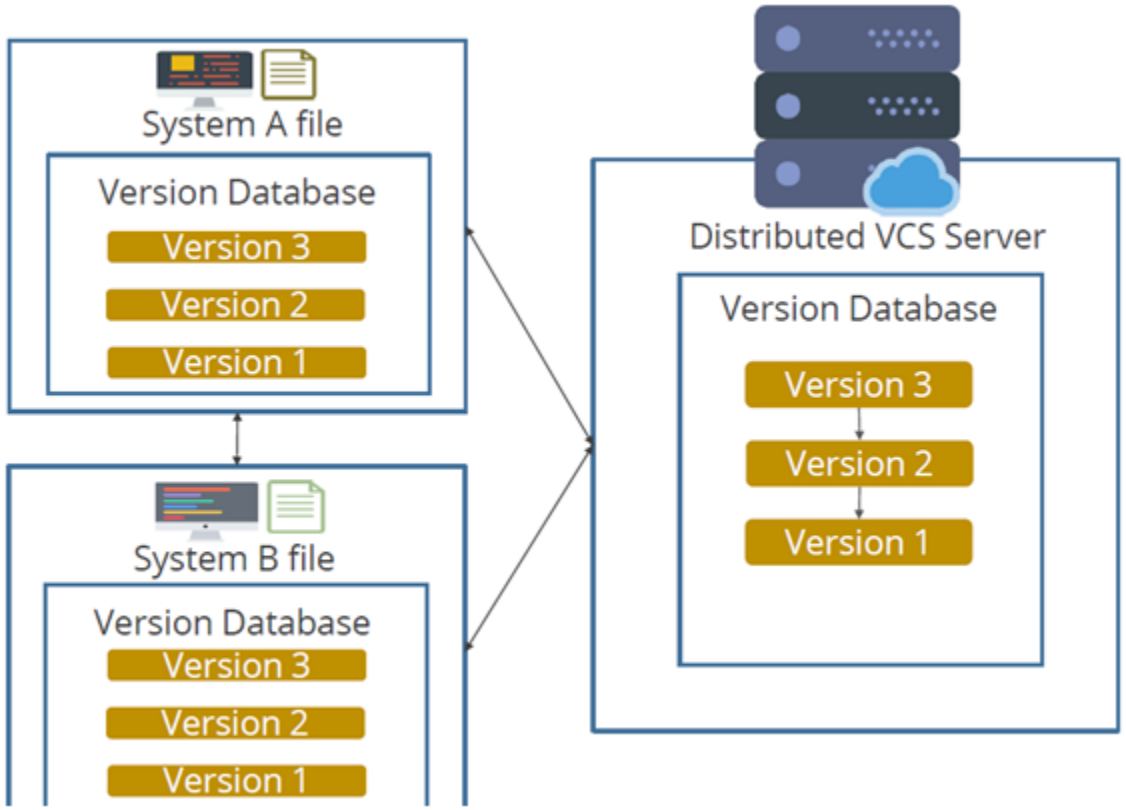
- Centralized VCS
- Distributed VCS

Centralized Version Control System



- Uses a central server to store all the files.
- Performs every operation directly on the repository.
- Stores file versions on the central VCS server.
- For example: Tortoise SVN

Distributed Version Control System



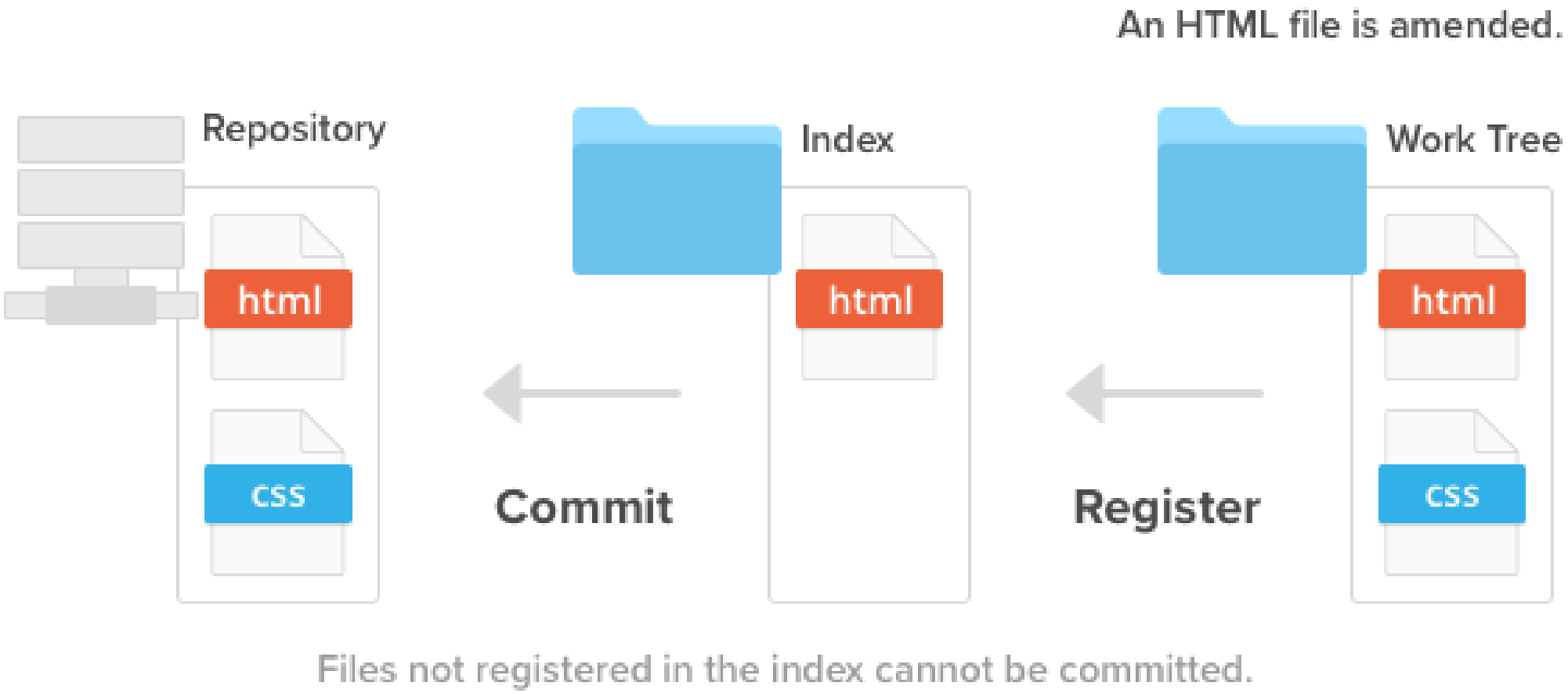
- Moves from the client-server approach to peer-to-peer approach.
- Updates the local repositories with new data from the central server. The changes get reflected to the main repository.
- For example: Git

Git workflow

There are three main components of a Git project:

- The **repository**, or repo, is the “container” that tracks the changes to your project files. It holds all of the commits — a snapshot of all your files at a point in time — that have been made. You can access the commit history with the Git log.
- The **working tree**, or working directory, consists of files that you are currently working on. You can think of a working tree as a file system where you can view and modify files.
- The **index**, or staging area, is where commits are prepared. The index compares the files in the working tree to the files in the repo. When you make a change in the working tree, the index marks the file as modified before it is committed.

Git workflow



Git workflow

You can think of this as your basic Git workflow:

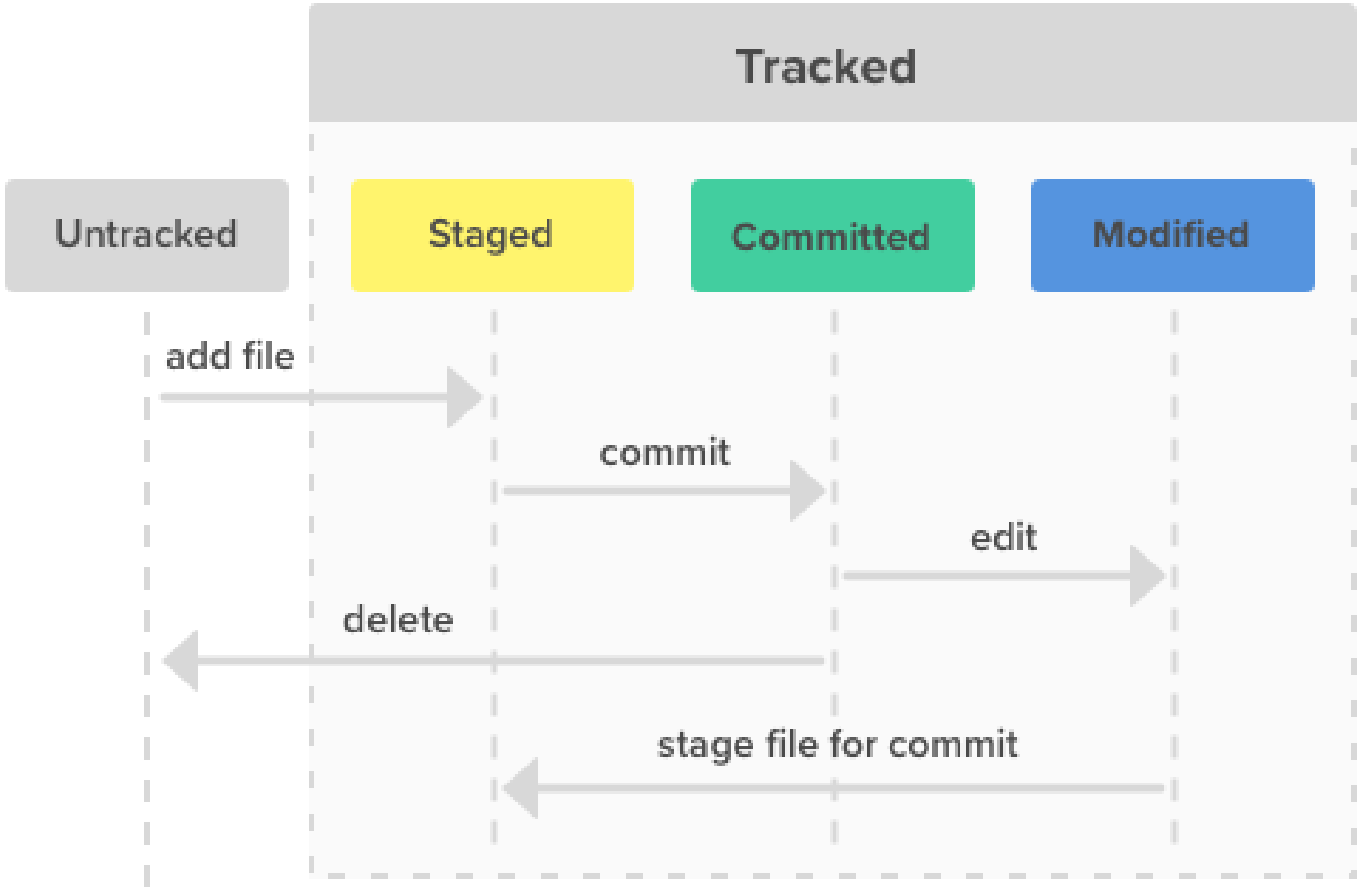
- Modify your files in the working tree.
- Stage the changes you want to include in the next commit.
- Commit your changes. (Committing will take the files from the index and store them as a snapshot in the repository.)

Three states of Git files

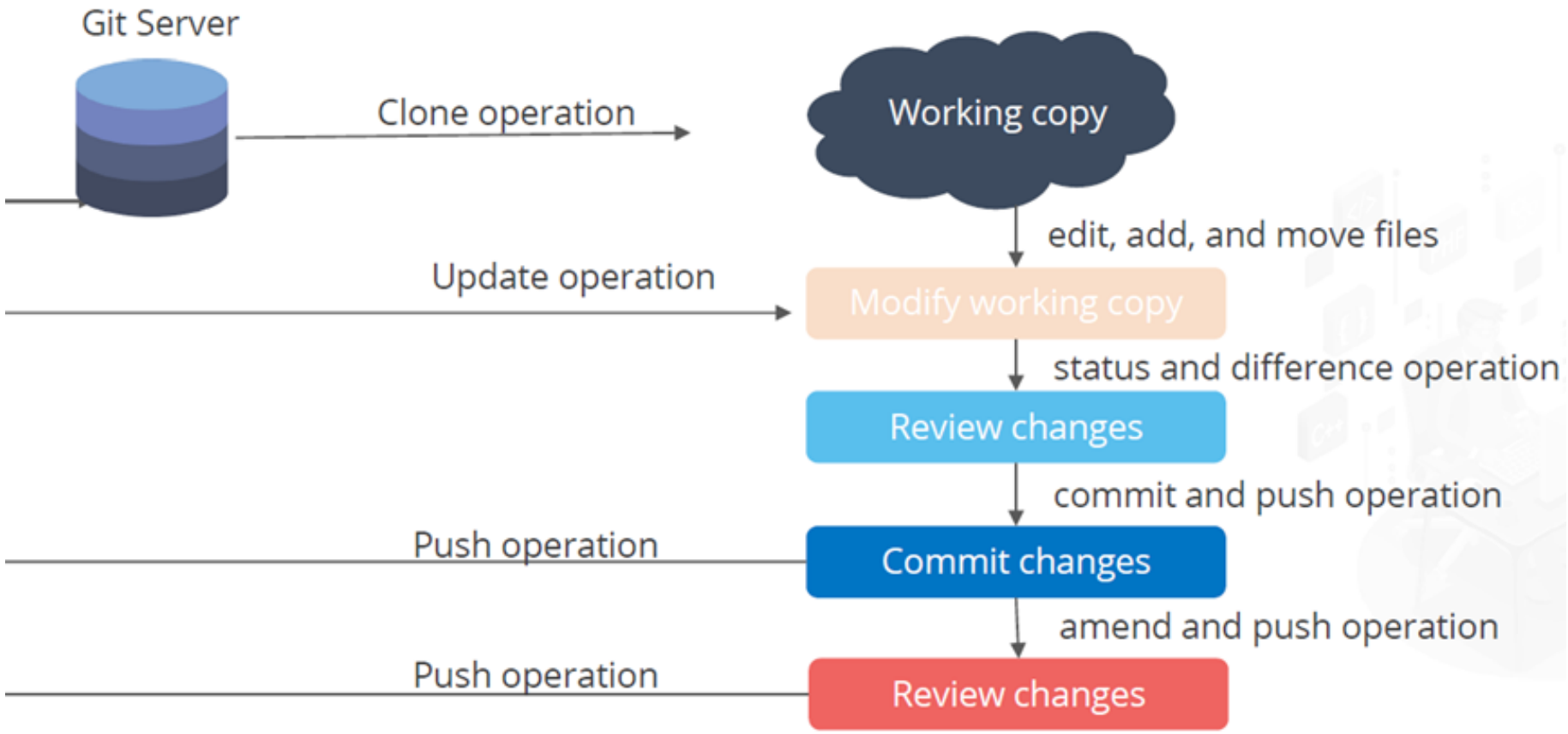
As you can probably guess from the Git workflow, files can be in one of three states:

- Modified
- Staged
- Committed
- When you modify a file, the change will only be found in the working tree. You must then stage the changes if you want to include them in your next commit. Once you finish staging all files, you can commit them and add a message describing what you changed. The modified files will then be safely stored in the repo.

Three states of Git files



Git : Life Cycle



Git vs GitHub

Git	GitHub
It is installed and maintained on the local system.	It is hosted on the web.
It is a command line tool.	It is a graphical interface.
It is a tool to manage different versions of the file in a git repository.	It is a space to upload a copy of the git repository.

Lab 1 : Install Git and verify

Lab 2 : Configure Git username and email

Lab 1 and 2

- Verify git by checking git version

  git --version

- Configure user and email

  git config -- global user.name “Sunitha”

  git config -- global user.email “sunics@gmail.com”

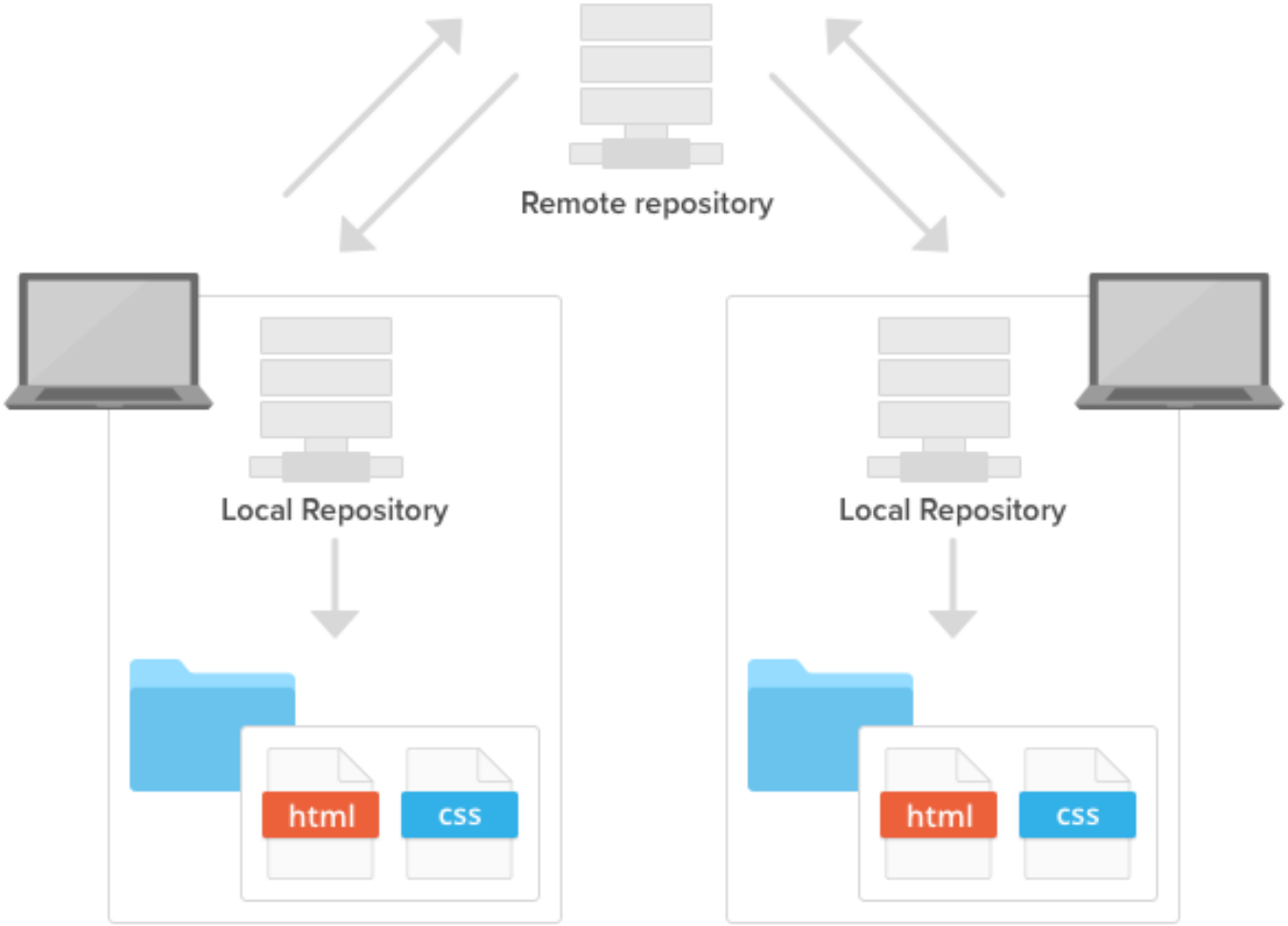
Creating a Git Repository

Remote repositories and local repositories

There are two types of Git repositories: remote and local.

- **A remote repository** is hosted on a remote, or off-site, server that is shared among multiple team members.
- **A local repository** is hosted on a local machine for an individual user.

Remote repositories and local repositories



Ways to create a repository

- There are two ways to create a local repository on your machine: you can create a new repository from scratch using a file folder on your computer, or you can clone an existing repository.
- **Git init -** You can create a new repo from scratch using the **git init** command. It can be used to introduce Git into an existing, unversioned project so that you can start tracking changes.
- **Git clone -** You can copy a remote repository onto your local machine using the **git clone** command. By default, **git clone** will automatically set up a local master branch that tracks the remote master branch it was cloned from.

Steps to Create a Git Repository

Create an empty directory	Command: **$ mkdir <directory-name>**
Convert the directory into a repository	Command: **$ git init**
Run git status	Command: **$ git status**

```
hp@DESKTOP-01792DU MINGW64 /d/dev/lgit/MyRepo (master)
$ git status
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
```

NOTE

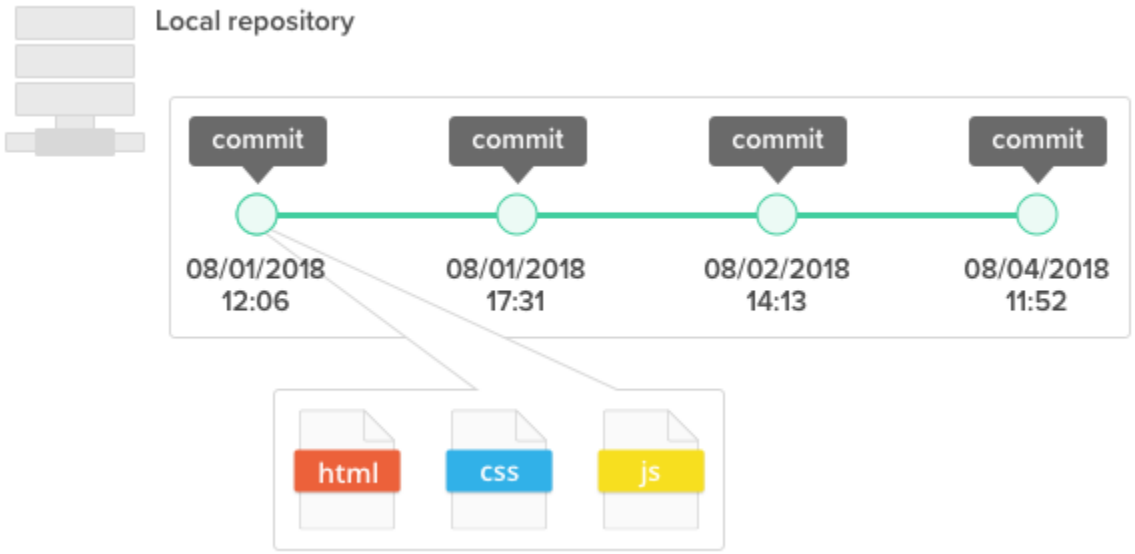
$ git init command creates a hidden directory that stores all the metadata required for it to function.

Recording changes

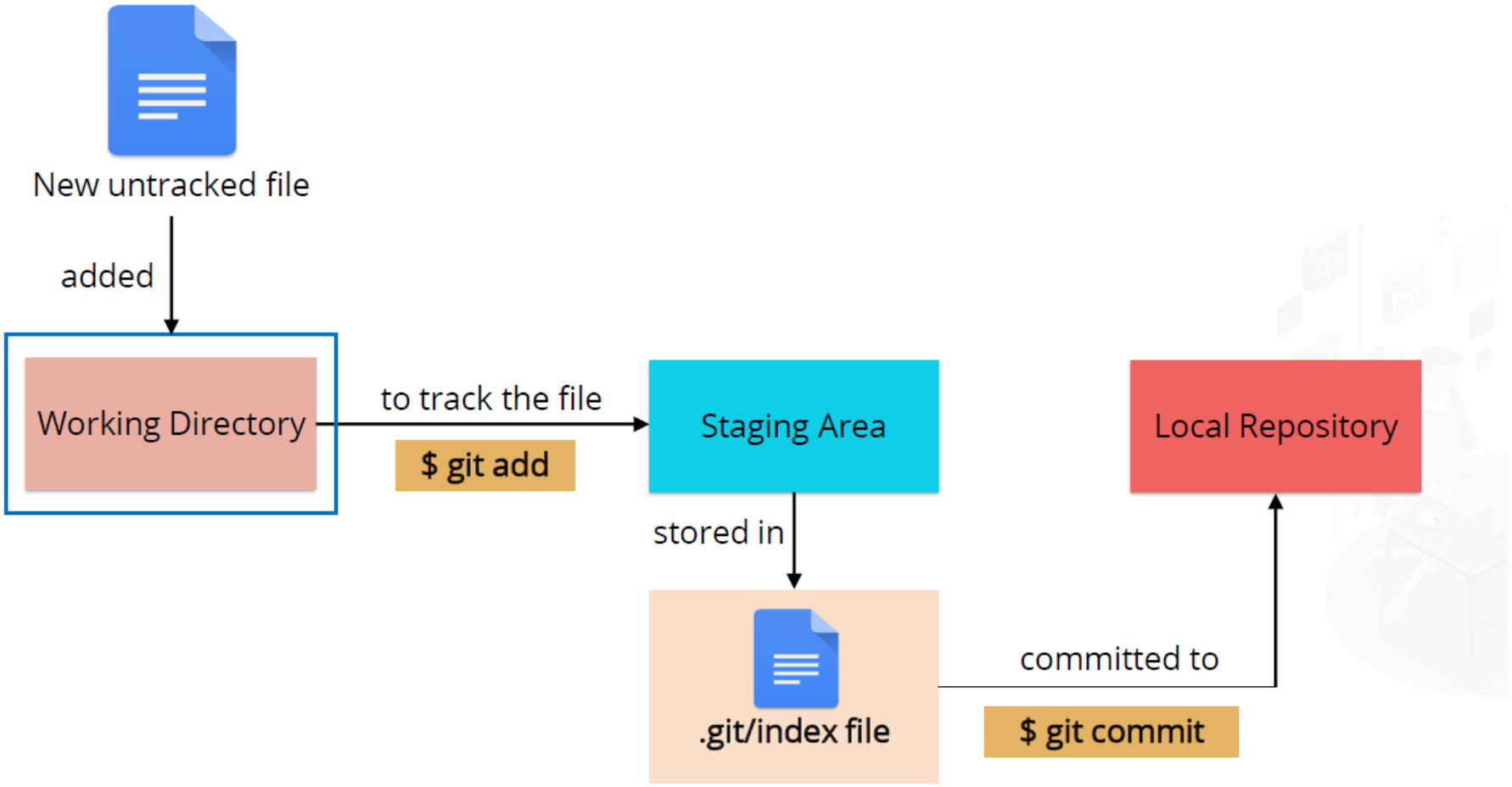
- Git does not automatically save every change you make. You must tell Git which changes you want to be recorded by staging those changes. After staging, you can then commit the changes so that they are recorded in the repo.
- **Making changes -** The working tree is where you make changes. There you can edit files, add new files, and remove files that are no longer needed. Once a file has been changed in the working tree, it is noted as modified in the index (e.g., the staging area where new commits are prepared) where it sits between the repository and the working tree.
- Changes made in the working tree are not saved directly to the repository. All changes are first **staged in the index** and then saved in the repo. Only the files in the index are committed to the repo.

Git commit

- The **git commit** command lets you record file changes in the repository's Git history.
- Every change you commit will be viewable in the respective file or directory in chronological order.



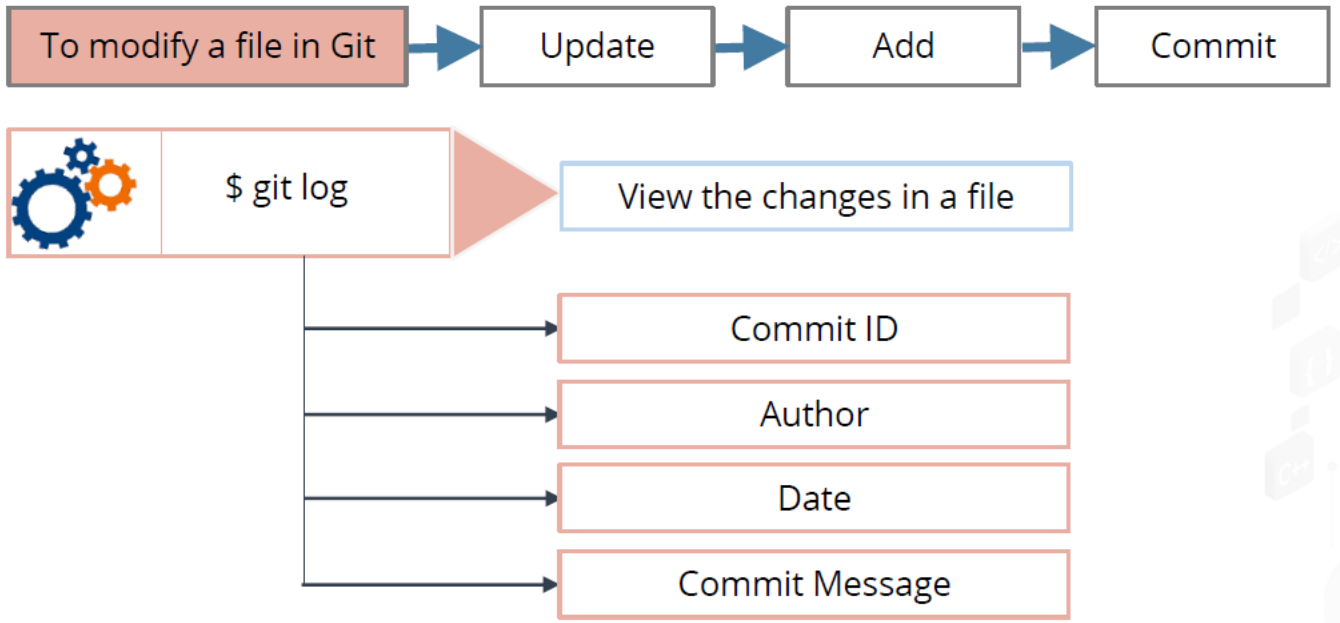
Git's Three-Stage Workflow



Git commit

- A 40-character checksum hash uniquely identifies each commit. You can use the checksum hash to retrieve the status or changes that were made on the given commit in your repository.
- When committing your changes, you are required to enter a commit message. The commit message should accurately describe the changes you’re making.
- Make commit messages easy to understand for all your team members.

Managing and Viewing Changes



NOTE

To restrict the output to one-line, execute: **$ git log --oneliner**, this will display the information in a single line.

Lab 3 : Create a Git Repository

Problem Statement: To create a Git repository to share the project files with your coworkers.

Steps to Perform:

1. Create a repository
2. Add files (README, project files) to the repository
3. Commit changes

Undoing changes

- One of the most valuable features of Git is the ability to undo mistakes. When you make a new commit, Git stores a snapshot of your project so that you can go back to an earlier version when you need to.
- There are two ways to undo changes: **git revert** and **git reset**.

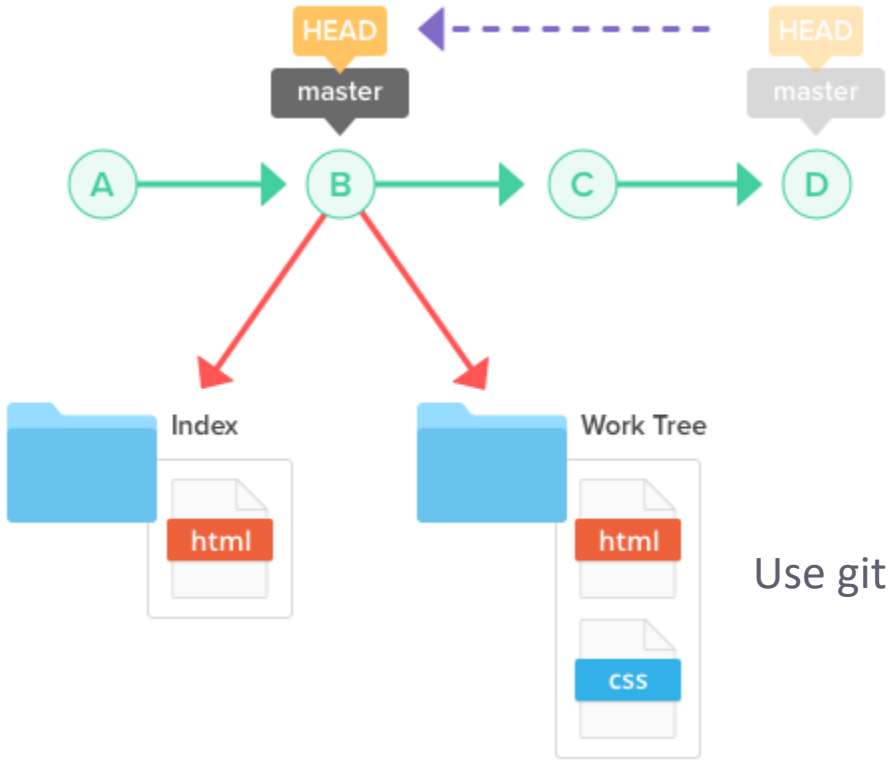
Git revert

- You can use the **git revert** command to safely undo a commit that has already been pushed.
- While you can also delete a previous commit from the history using **git reset** or **git rebase -i**, it is generally not a good idea because it causes the remote repository to become desynchronized with the local repositories of other members.

git revert is the safest method of undoing changes.

Git reset

- You can discard commits that you no longer need using the git reset command. You can specify the scope for the reset command by going into reset mode.



Use git reset to remove unnecessary commits.

Lab 4 : Undoing changes

- Create a user registration and login features in a java project
- Commit the project
- Add a function to reset the password
- Commit the changes
- Undo the last commit

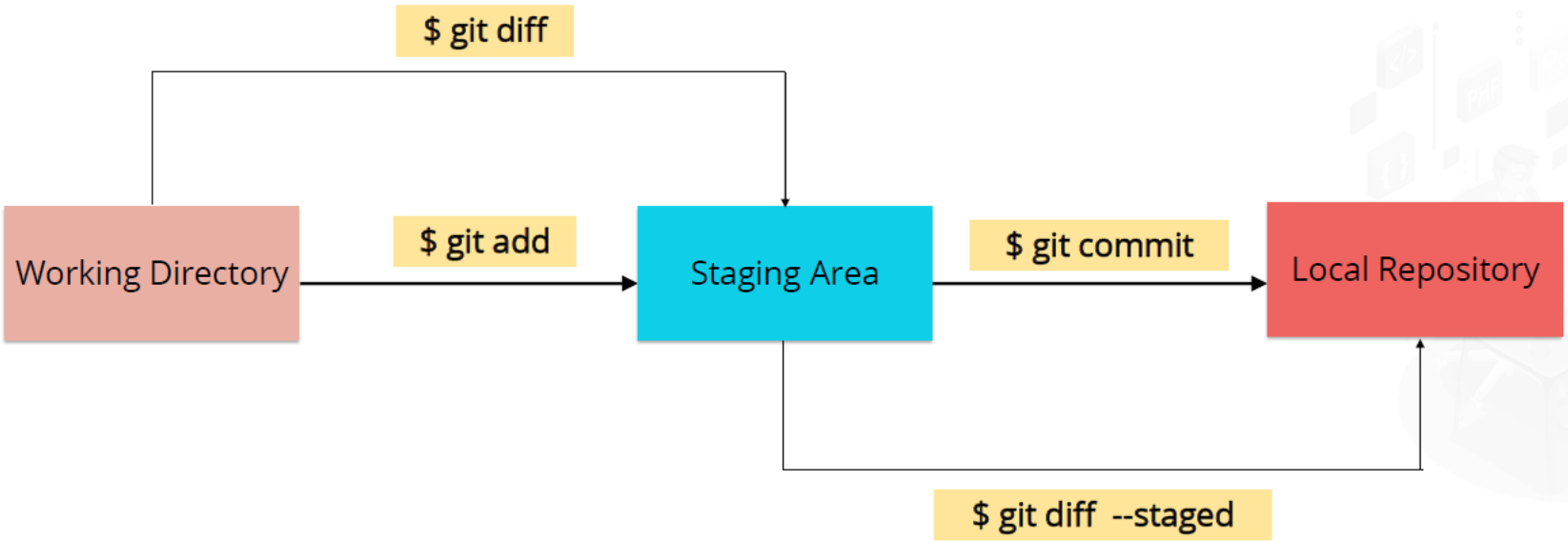
Git commands

$ git diff Shows unstaged file differences
$ git diff --staged Shows file differences between staging and the last file version
$ git branch Lists all branches in the current local repository
$ git branch [branch-name] Creates a new branch

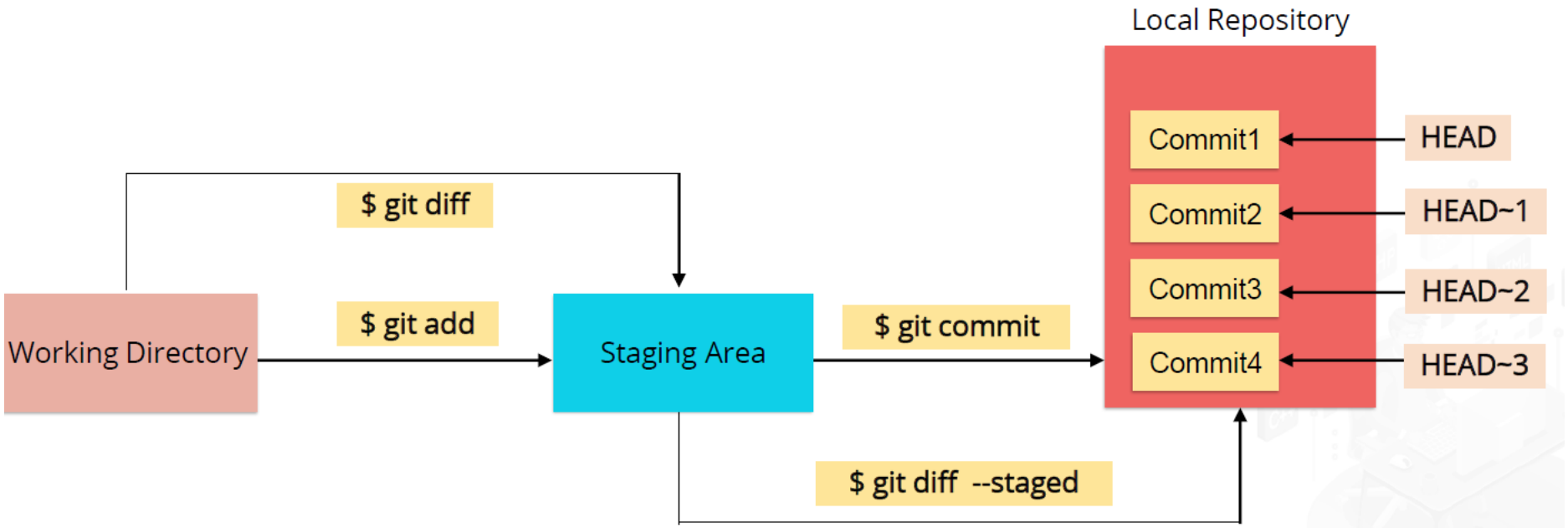
- git branch feature1

Track Changes between Stages

Tracking is the ability to identify changes between the different versions of the same file, spread across various repositories.



Track Changes between Stages



NOTE

1. Command "**git diff –staged**" is similar to "**git diff HEAD**"
2. If the number of commits increase beyond a certain count, use the hashcode command. Example: **$ git diff <commit hashcode>**

Git commands contd..

$ git checkout [branch-name] Switches to the specified branch and updates the working directory
$ git merge [branch-name] Combines the specified branch's history into the current branch
$ git branch -d [branch-name] Deletes the specified branch
$ git rm [file] Deletes the file from the working directory and the staging area

- git checkout feature1

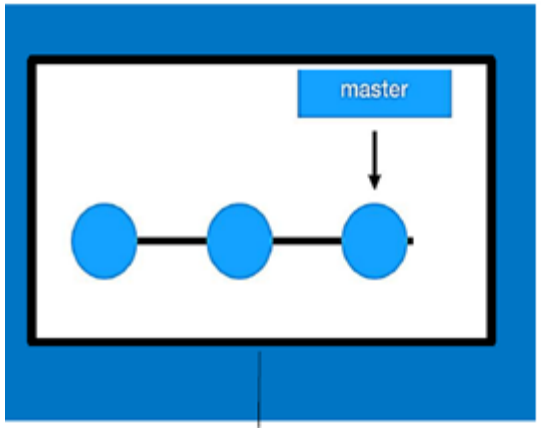
Lab 5 : Tracking File Changes

Problem Statement: Being a Project Lead, demonstrate git's ability to compare different commits and file changes.

Steps to Perform:

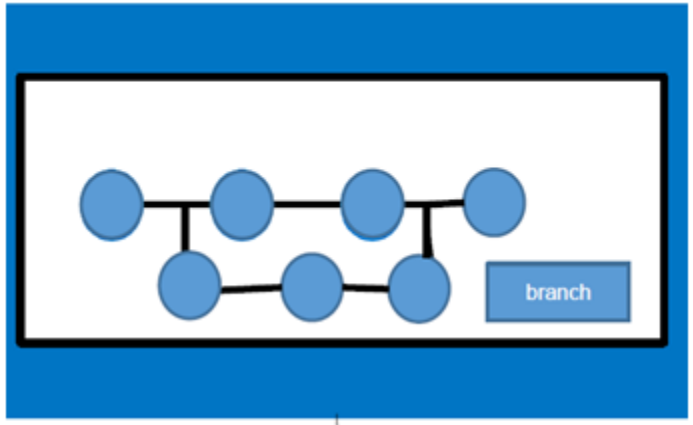
1. Check the git status
2. Open a file from your local repository and make some changes
3. Execute **git diff index.html** command to compare the file
4. Add the file to the staging area
5. Execute **git log --oneline** to check the recent log of commits
6. Execute the **git diff --staged HEAD**
7. Execute **git diff --staged HEAD~1** to refer to the commit prior to that.
8. Check the git status and commit the changes.

Git Buzzwords : master



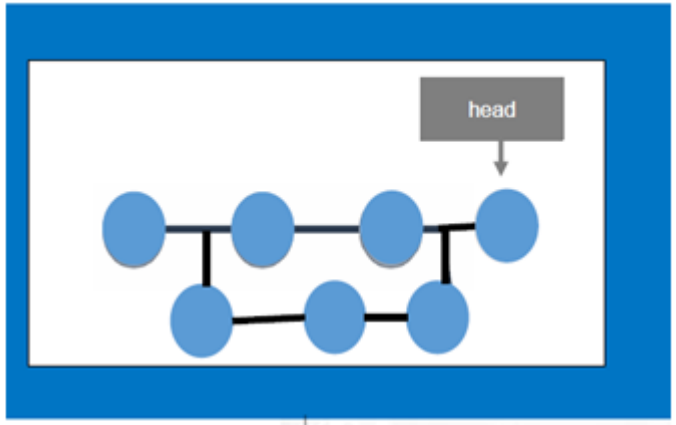
- It is a default branch.
- It is used by CI tools for build and deployment.
- It is followed by the other repositories.

Git Buzzwords : branch



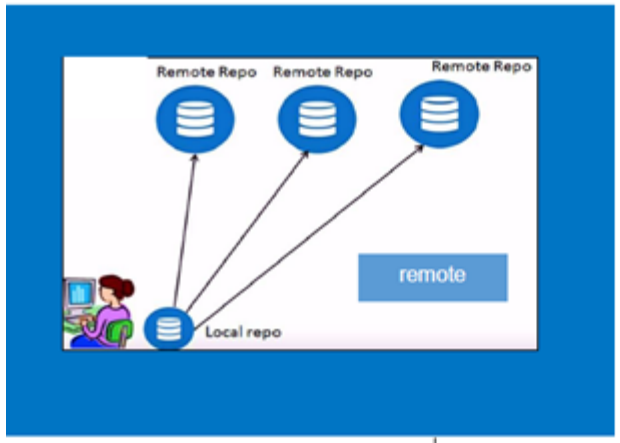
- It is a light weight working copy.
- It has a staging area.
- It works without impacting the master branch.

Git buzzword : head



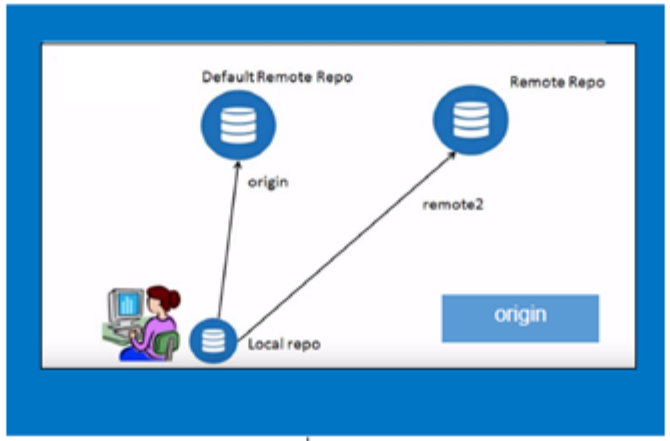
- It is a pointer to the latest commit of the working branch.
- It is present on every repository.
- It will point to the latest commit during branch switch.

remote



- It is a git repository on a network outside the local machine.
- It can have more than one remote repositories pointing from the local repository.
- It can be managed and referenced with short names.

origin



- It associates remote repository with names.
- It is a local name set for the default repository.
- It is useful to point the default repository when executing git commands.

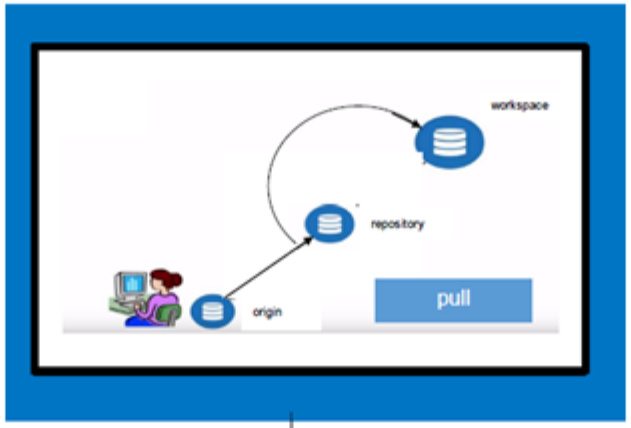
clone

- It copies the existing repository from a remote repository.
- It will get the complete repo, whereas checkout will only fetch the working copy.
- It helps to replicate the repo on the local machine.

push

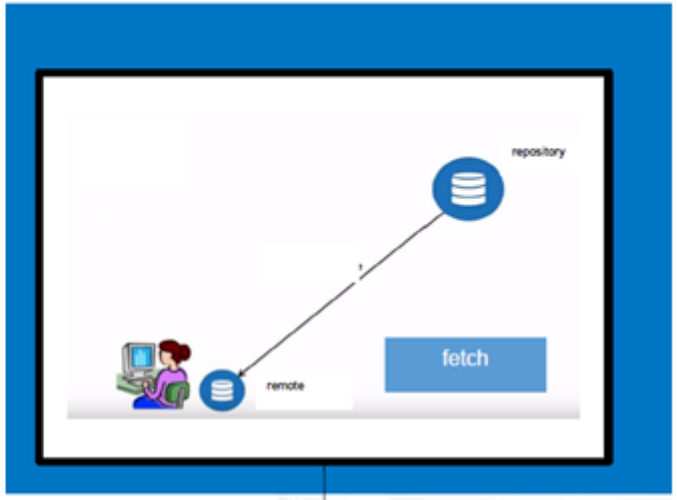
- It pushes changes from the local to the remote repository.
- It is performed after committing the changes to the local repository.
- It syncs the changes with the local and remote repository.

pull



- It transfers the updates from the local to the remote repository.
- It syncs the changes from remote to the local repository.
- It takes current code from remote repository and merges the change with the local repository.

fetch



- It will not merge the changes with a local repository.
- It gives updates from remote to the local repository.
- It syncs the changes from remote to the local repository.

Upstream and Downstream

- When the data is flowing between repository A and B, repository A is upstream and B is downstream making B pull data from repository A.

Deleting Files in Git

Deleting Files from Staging Area and Working Directory

$ git rm <filename>

Deletes the file from staging area and working directory

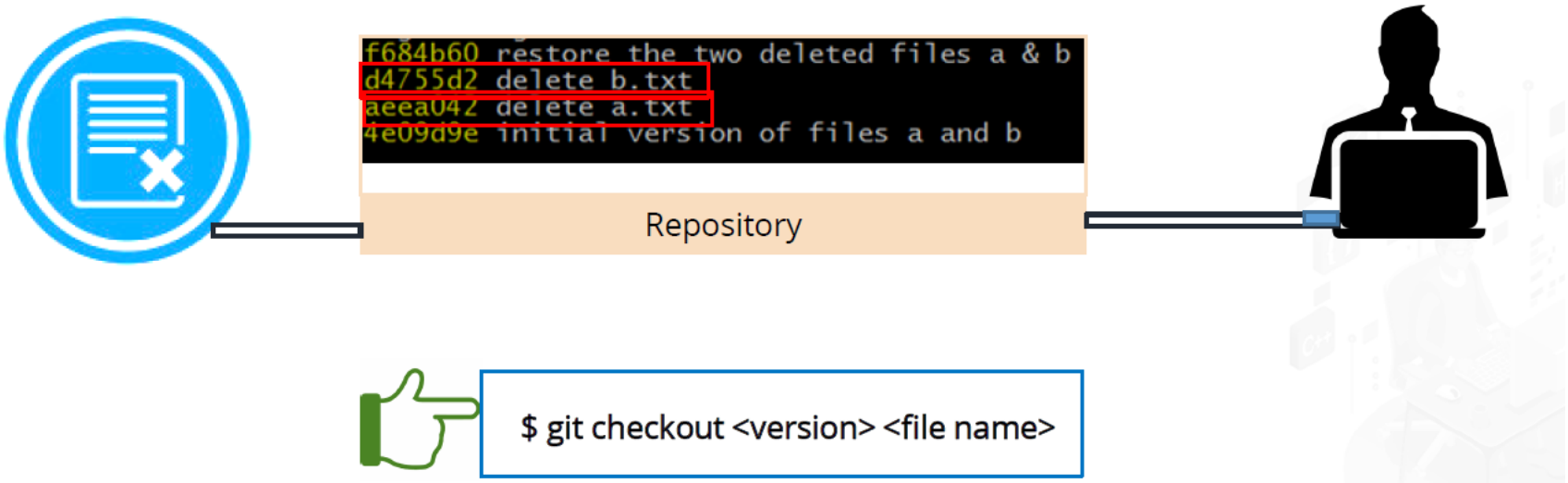
Delete a file only from the staging area: $ git rm --cached <filename>

$ git status

NOTE

This file will be untracked as it is removed from the staging area.

Restoring Deleted Files



$ git checkout <version> <file name>

Remote Repositories

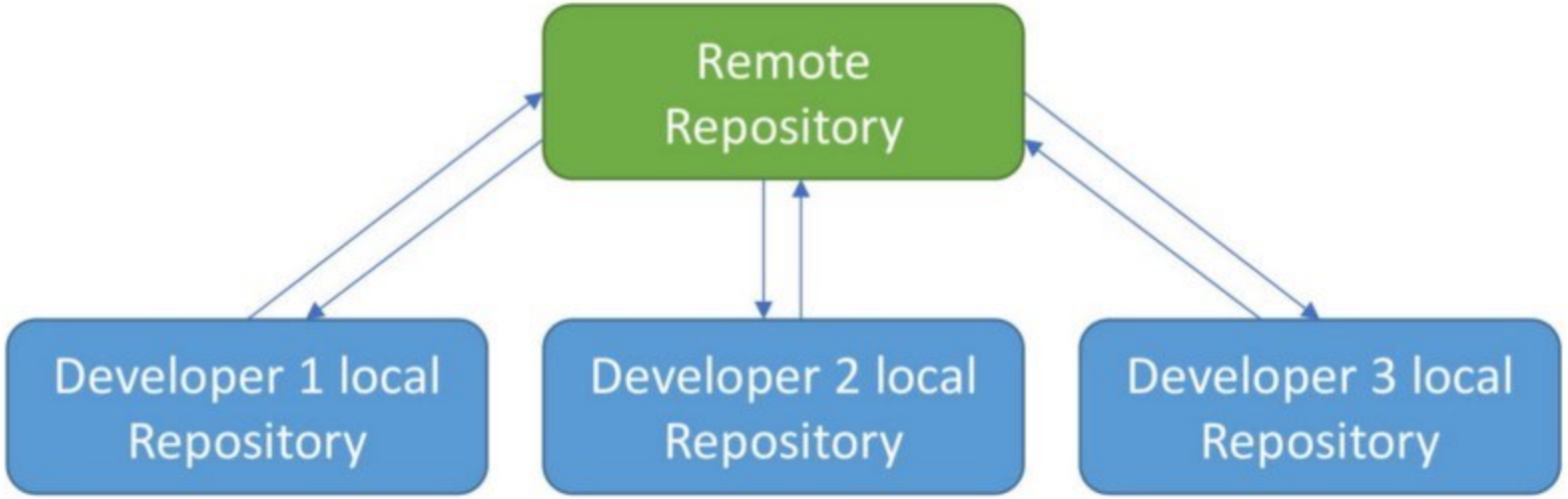
GitHub: Basics

- Transparent
- Publish Project
- Collaboration

Web-based Git repository hosting service

Web interface to upload files

Each developer will work in their local repository but eventually, they will push the code into a remote repository. Once the code is in the remote repository, other developers can see and modify that code.



Syncing repositories

- Remote repositories allow us to share our changes with other members of the team. They can be on a private server, on a different computer than yours, or hosted using a service like Backlog. Wherever yours is hosted, you'll need to be able to sync your local repository with the remote repository frequently. You'll do this using three commands: git push, git pull, and git merge.

Git push

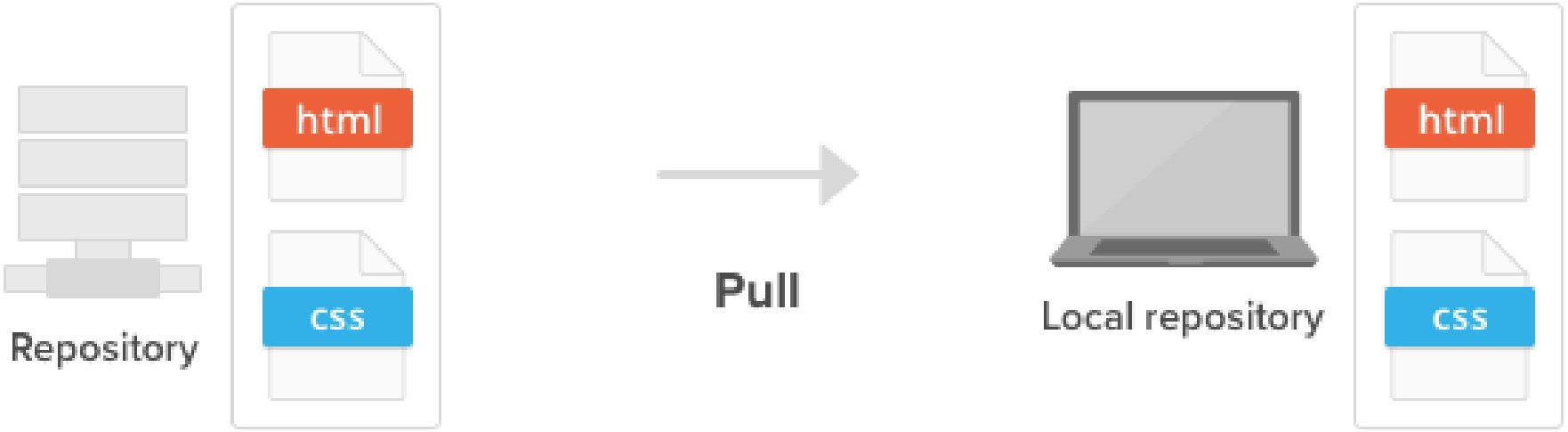
- In order to start sharing changes with others, you have to push them to a remote repository using the “push” command. This will cause the remote repository to update and synchronize with your local repository.

Git pull

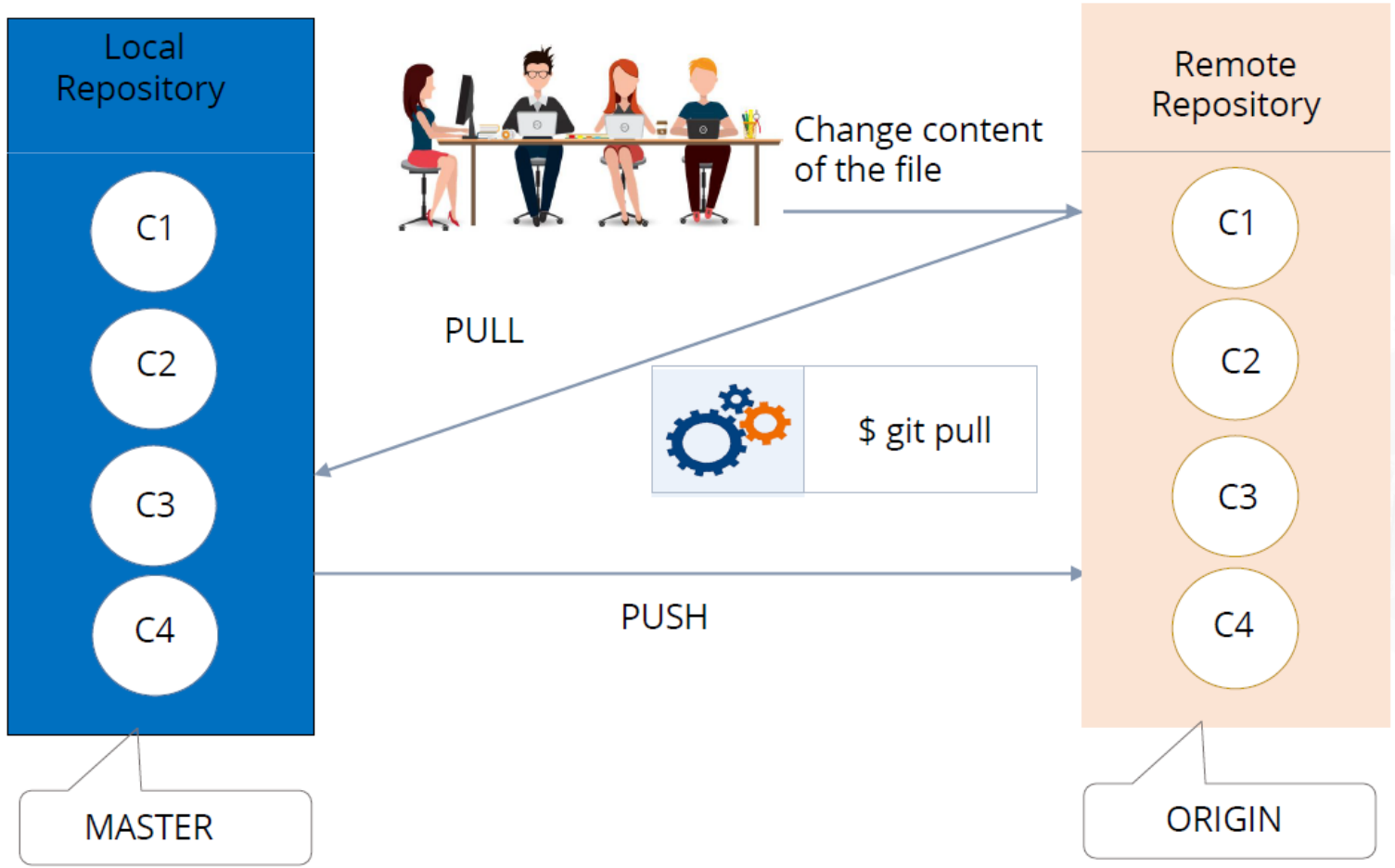
- Whenever somebody pushes their changes to the shared remote repository, your local repository becomes out of date. To re-synchronize your local repository with the newly updated remote repository, simply run the git pull operation.
- When the pull is executed, the latest revision history will download from the remote repository and import to your local repository.

Git pull

Pull = Download!

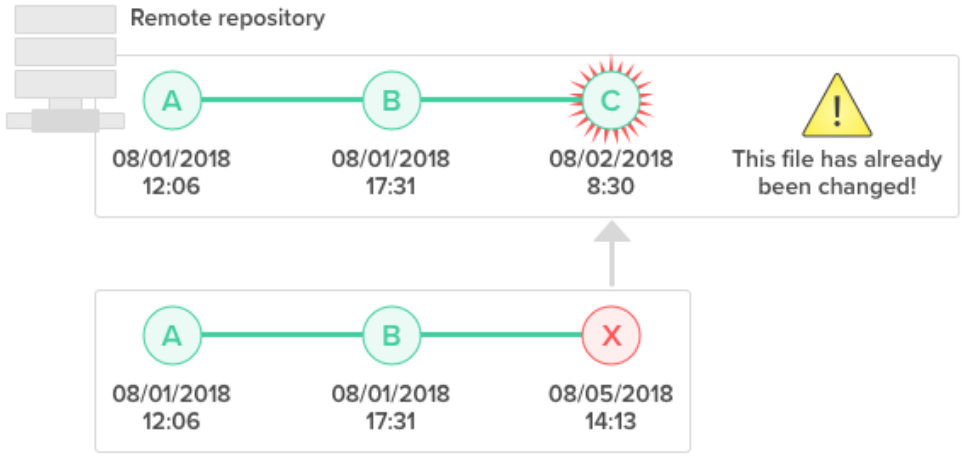


GitHub: Pulling Commits

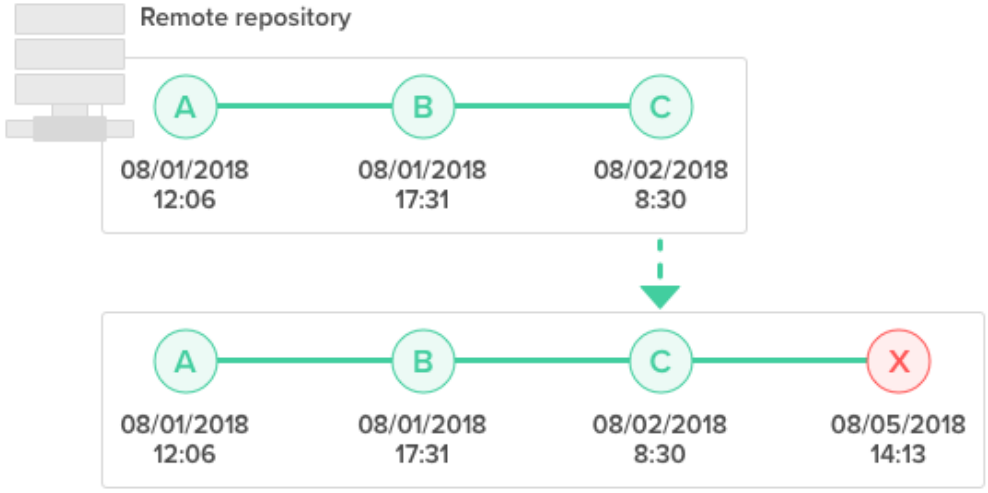


Git merge

- Your push to the remote repository will be rejected if your local repository is out of date, possibly because there are some updates on the remote repository that you do not have locally yet.



You are unable to push to the remote repository if your local repo is out of date.



You must merge the latest changes before pushing.

Git commands contd..

$ git push [alias] [branch]

Uploads all local branch commits to remote repository

$ git pull

Downloads from remote repository and incorporates changes

$ git clone [repository-url]

Clones an existing repository

$ git rebase [branch]

Rebases your current HEAD onto [branch]

Git commands contd..

- git remote add origin url/to server/repo
- git clone /path/to/repo
- git clone url/to/repo
- git push origin master
- git push origin branchName
- git pull
- git merge branchName

Lab 6 : Create Repository in GitHub

Problem Statement: Create public repositories for an open source project. When creating your public repository, make sure to use a credential helper so Git will remember your GitHub username and password every time it talks to GitHub

Steps to Perform:

1. Execute $ git status to fetch the status
2. Execute $ ls -lrt
3. Execute $ git log --online
4. Create a repository
5. Add the GitHub repository
6. Execute $ git remote -v
7. Push the origin master
8. Execute $ git status to fetch the status
9. Execute $ cat
10. Check the commits
11. Match the committed ID

Lab 7 : Clone a Git Repository

Problem Statement: Clone the GitHub repository shared by your coworker to access the project files.

Steps to Perform:

1. Clone the repository
2. Add files in the repository
3. Commit the changes
4. Push the commits to the primary repository

GitHub vs GitLab vs Bitbucket

	GitHub	GitLab	Bitbucket
Open Source	✓	✓	✗
CI pipeline	✓	✓	✗
Own APIs	✓	✓	✗
Git platform	✓	✓	✓
Active Bug tracking	✓	✓	✗
Supports	✓	✓	✓

Lab 8 : Centralized Git Workflow

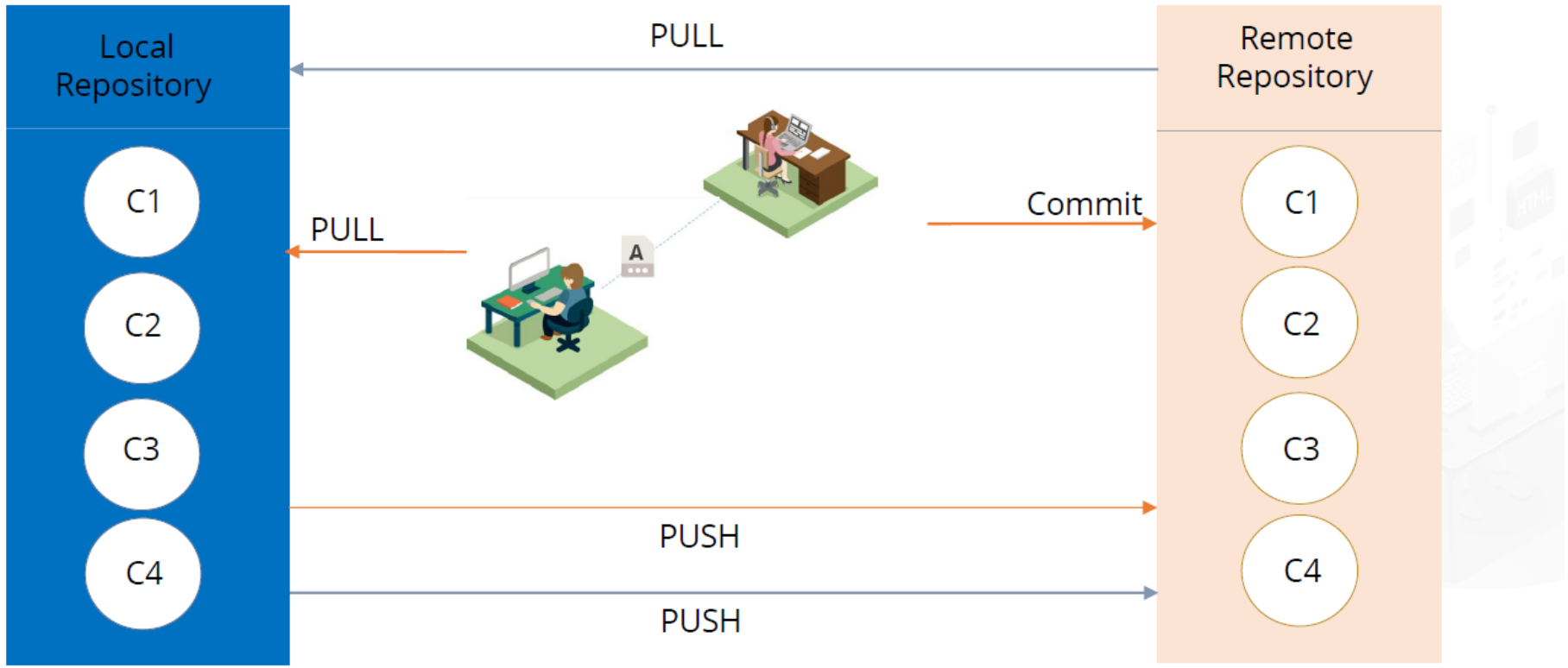
Problem Statement: You are a team lead in an organization that uses centralized git workflow. Demonstrate the workflow to your coworkers to help them share the project files among themselves.

Steps to Perform:

1. Create a file in the local repository
2. Check the status
3. Use **git add** to add the file
4. Commit the file
5. Check the status again
6. Update the file
7. Add the file
8. Check the status and commit the file again

Collaboration Between a Local and Remote Repository

How to Collaborate?



Lab 9 : Collaboration between Local and Remote Repositories

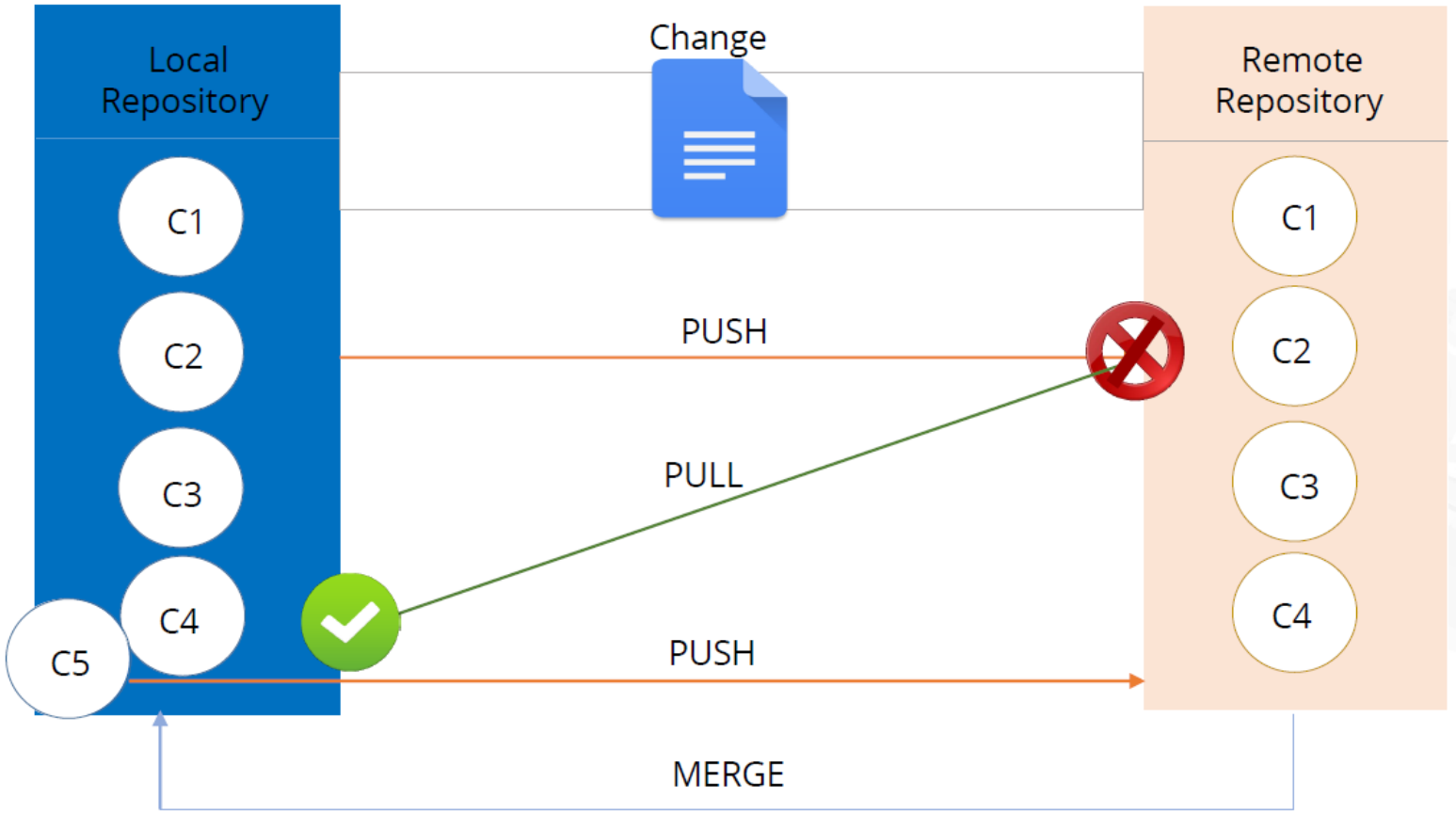
Problem Statement: The team wants the make a copy of someone else's GitHub repo in their GitHub account.

Steps to Perform:

1. Make changes on index.html
2. Execute $ git status
3. Execute $ git log –online
4. Pull the origin master
5. Execute $ cat
6. Execute $ vi index.html
7. Execute $ git status
8. Add $ git add and $ git commit -m
9. Add $ git commit -m
10. Add $ git status
11. Add $ git log --online
12. Push the last commit
13. Push the origin master

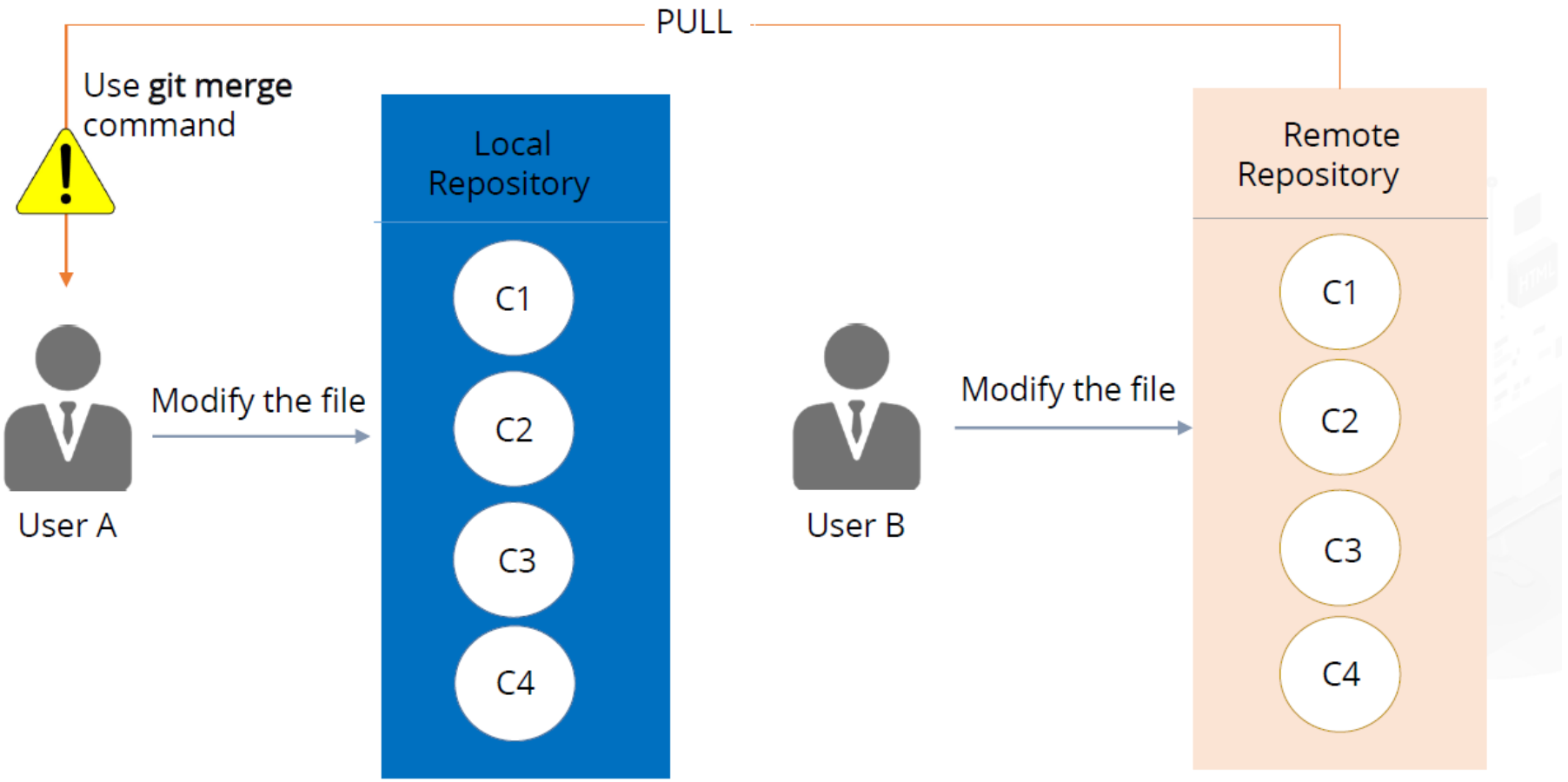
Managing Multiple Commits in Git

How does Git manage changes?



Merging File Changes in Git

Scenario of Merging File Changes in Git



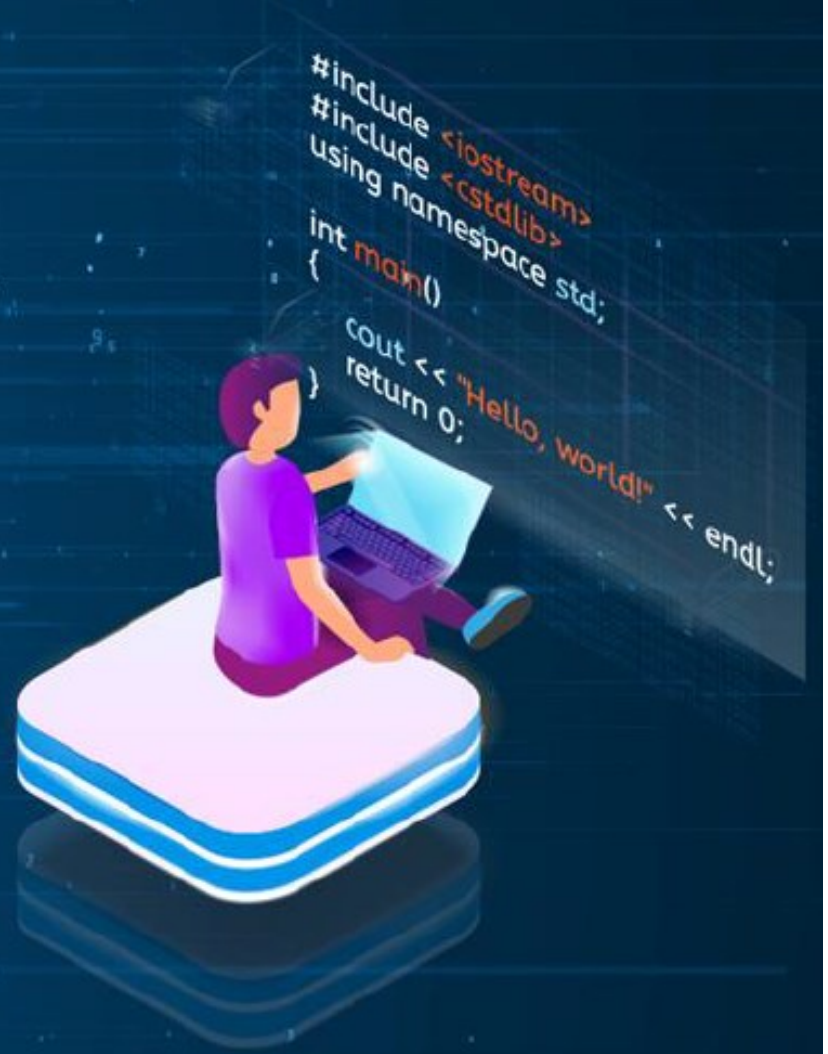
Lab 10 : Merging file changes in Git

Problem Statement: You have assigned a task to combine separate changes to an original in Git.

Steps to Perform:

1. Execute $ git log --online
2. Edit the index file
3. Make changes to index file
4. Execute $ git status
5. Execute git add and git commit
6. Execute $ git log –online
7. Push and pull the origin master
8. Edit the index file
9. Execute $ git status
10. Execute git add and commit
11. Execute $ git log --online
12. Push the origin master

Git Plugin with IDE



What Is EGit?

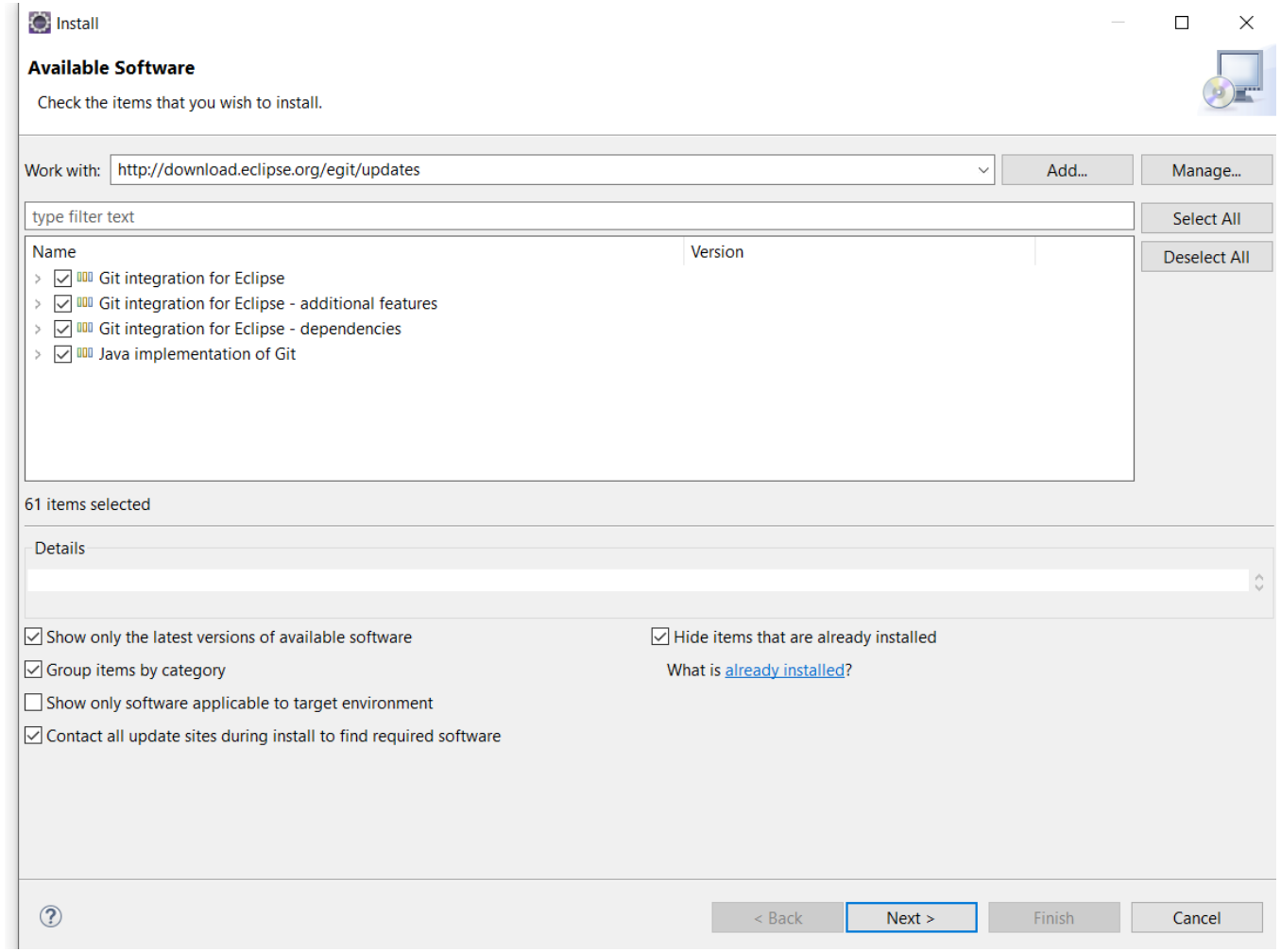
EGit is an Eclipse plug-in which allows you to use the distributed version control system “Git” directly within the Eclipse IDE.

NOTE

EGit is based on the JGit library. JGit is a library which implements the Git functionality in Java.

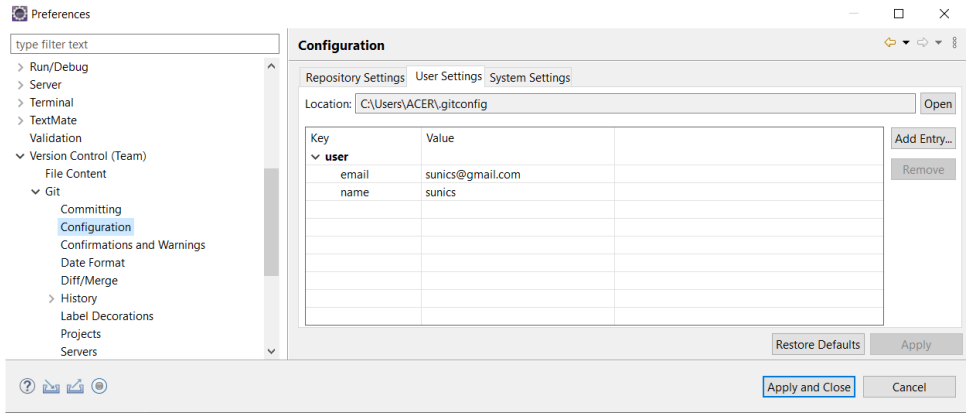
Updating Eclipse

- If the Git tooling is not available, you can install it via the Eclipse installation manager. Select the **Help --> Install new Software…** menu entry. Enter one of the following update site URLs:
- http://download.eclipse.org/egit/updates # Use this update site to get the latest release
- http://download.eclipse.org/egit/updates-nightly/ # use this update site to get the night build

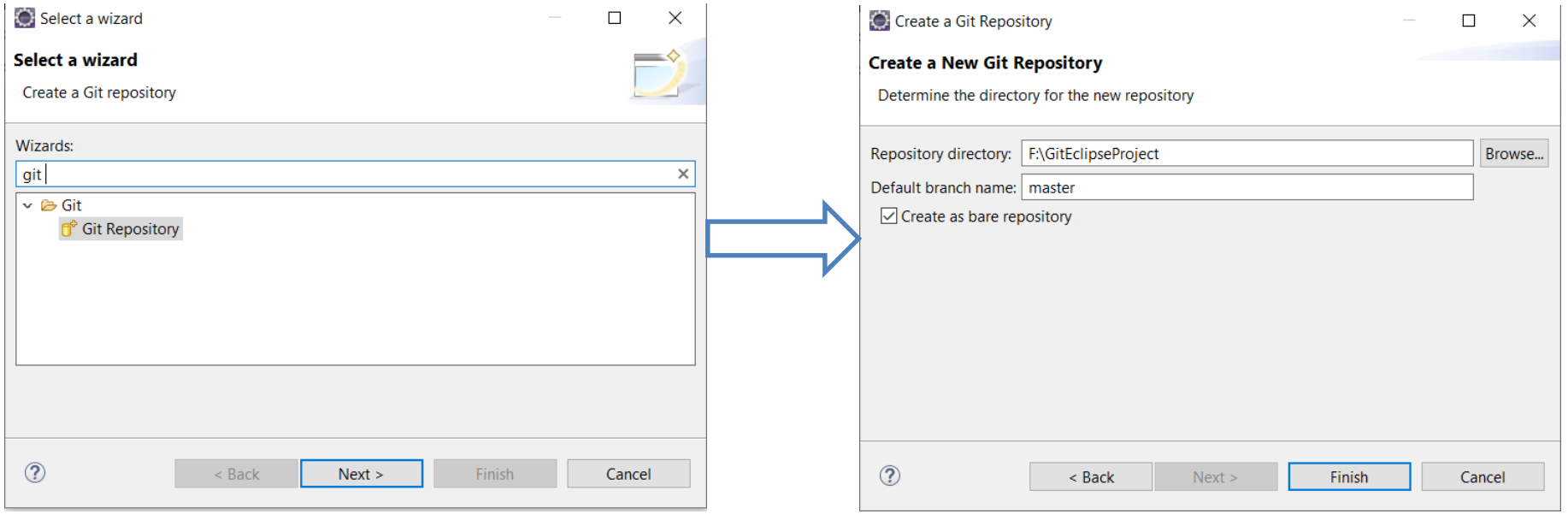

Install
Available Software
Check the items that you wish to install.
Work with:
http://download.eclipse.org/egit/updates
Add...
Manage...
type filter text
Select All
Deselect All
Name
Version
Git integration for Eclipse
Git integration for Eclipse - additional features
Git integration for Eclipse - dependencies
Java implementation of Git
61 items selected
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

Configuration for Git usage via the Eclipse IDE

- In your Eclipse IDE, select the **Window → Preferences → Version Control (Team) → Git → Configuration** entry. Configure your full name and email in the user settings. As the Eclipse IDE uses the same settings as the Git command line, this might already be done.
- If these keys are not available press the Add Entry… button and add them.



Create a new Git repository via Eclipse

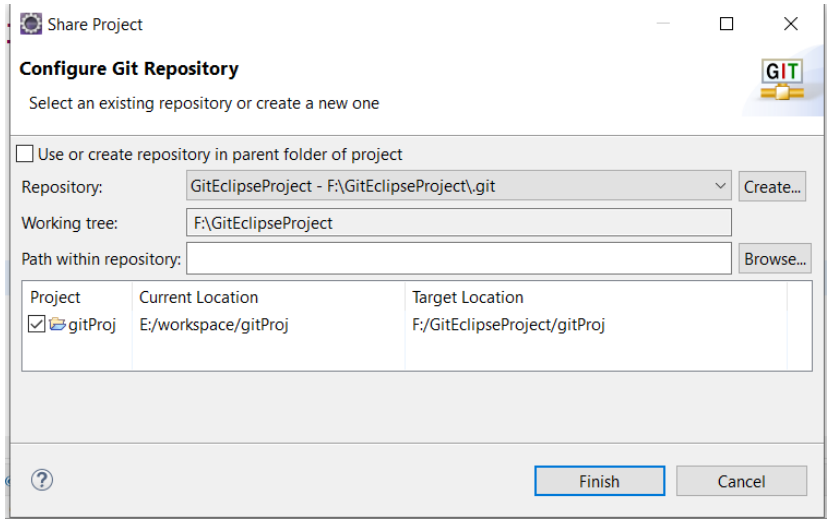


Select menu File → new → other and select

- Git Repository

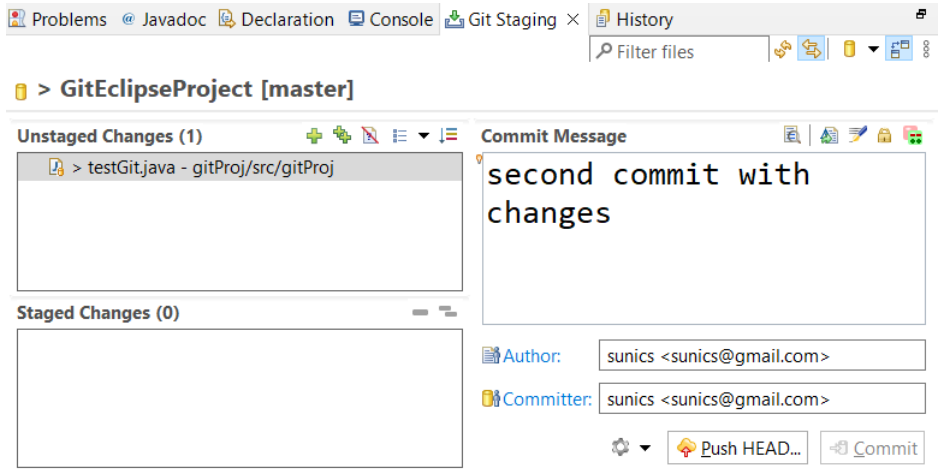
Put project under Git version control

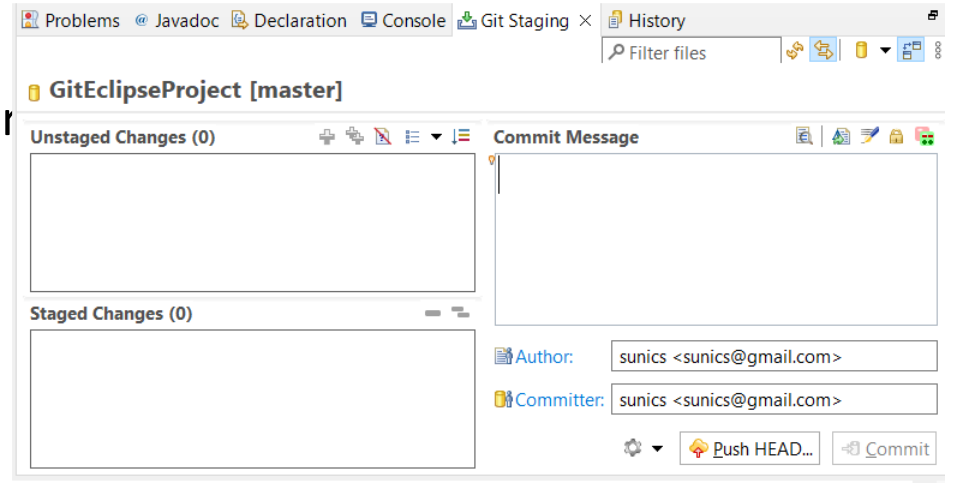
- Add java files as needed
- To put your new project under version control with Git, right-click on your project, select **Team → Share Project**. If another version control system is installed you have to select that you want to use Git as the version control system.
- Select your existing Git repository from the drop-down list.



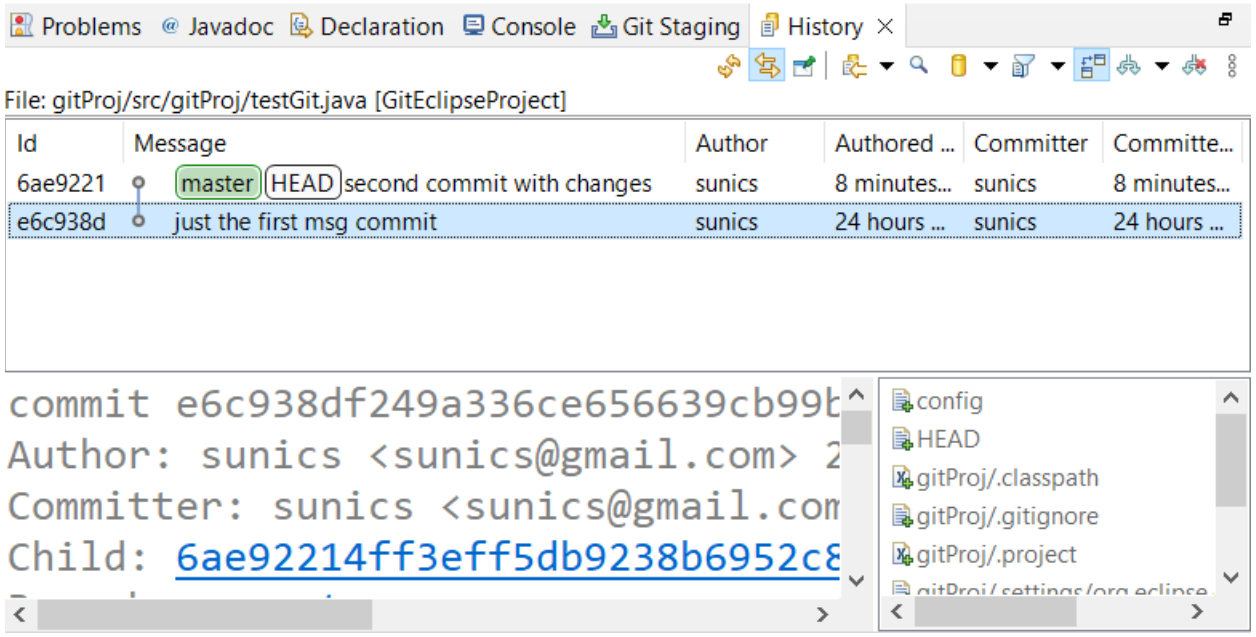
Using the Git Staging view for the initial commit

- Window → other → git staging area
- In this view use the + button to stage
  all files.
- After adding to staging, write a meaningful commit message and pr
the Commit button.

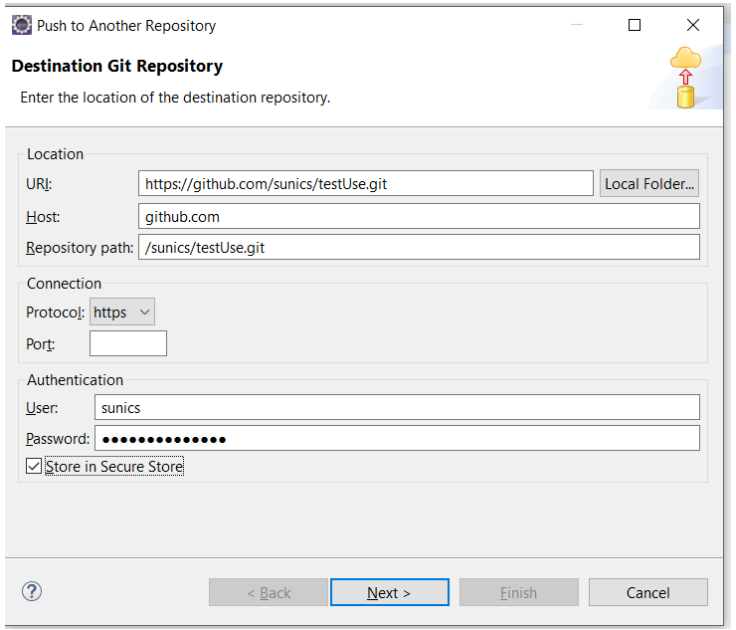


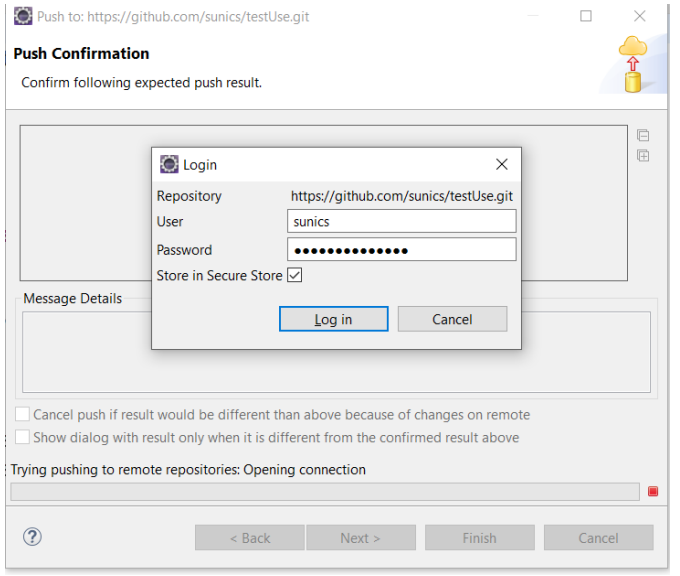


Open an older version with the current version of a file via the History view

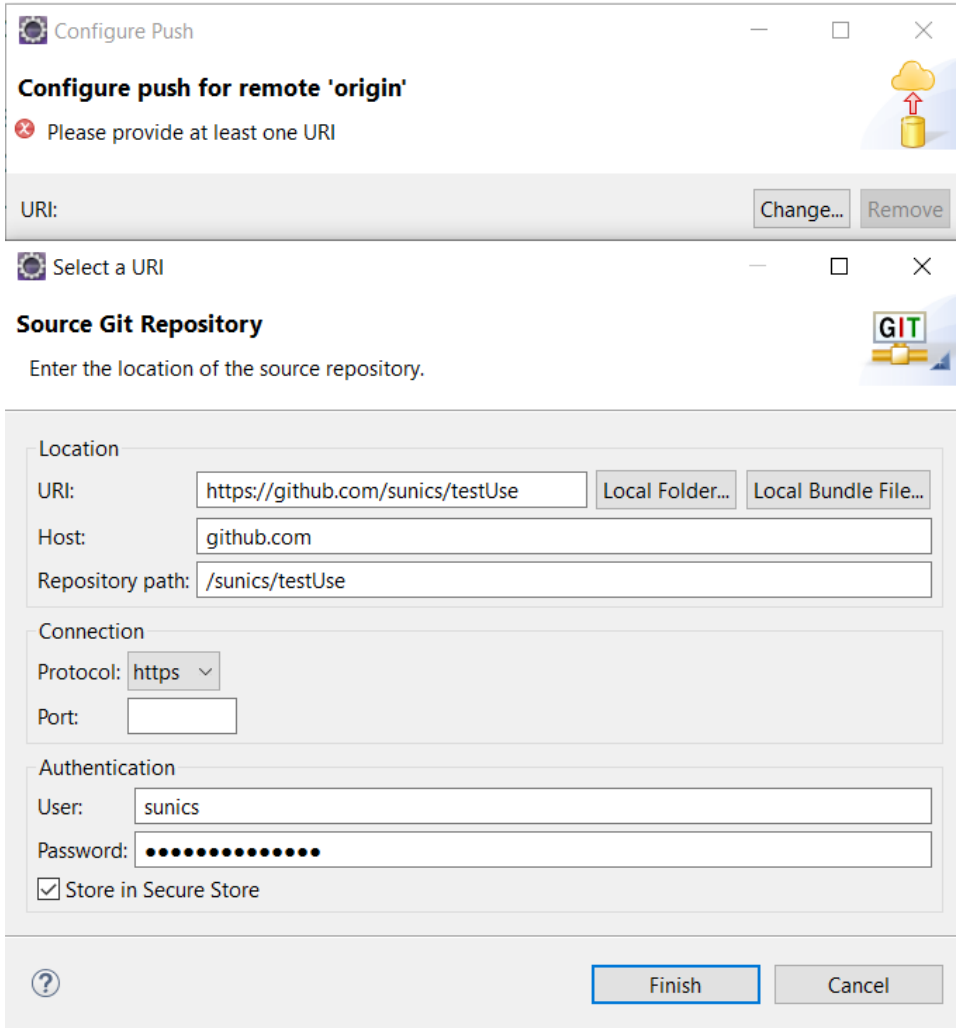


Push to remote

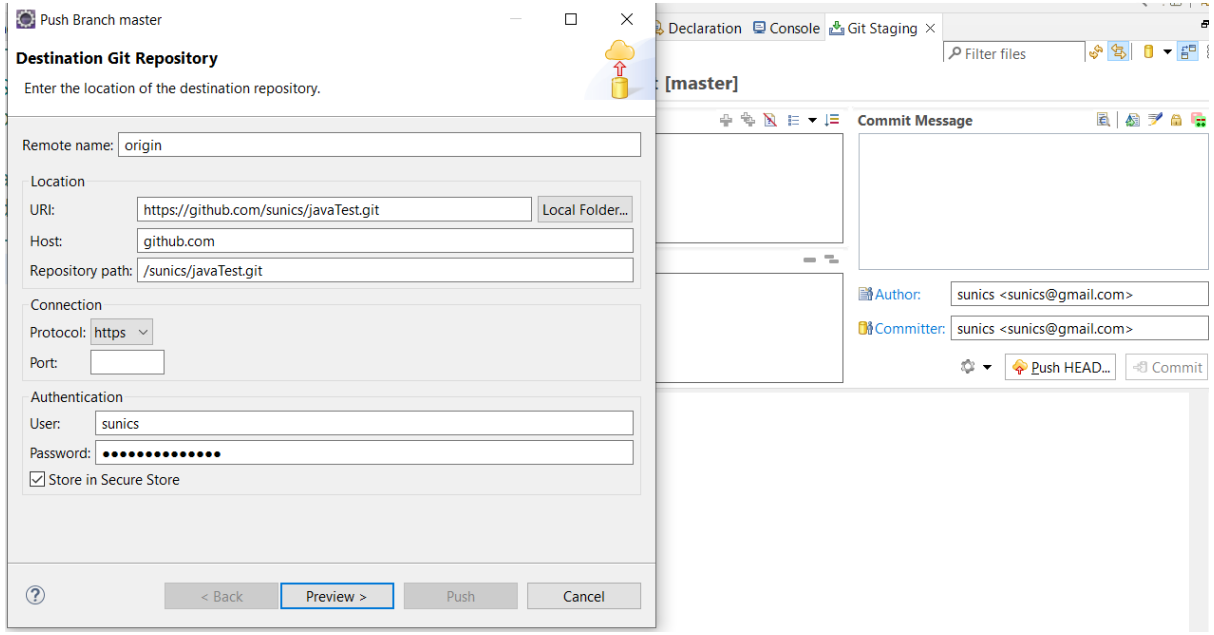




Configuring remote repository

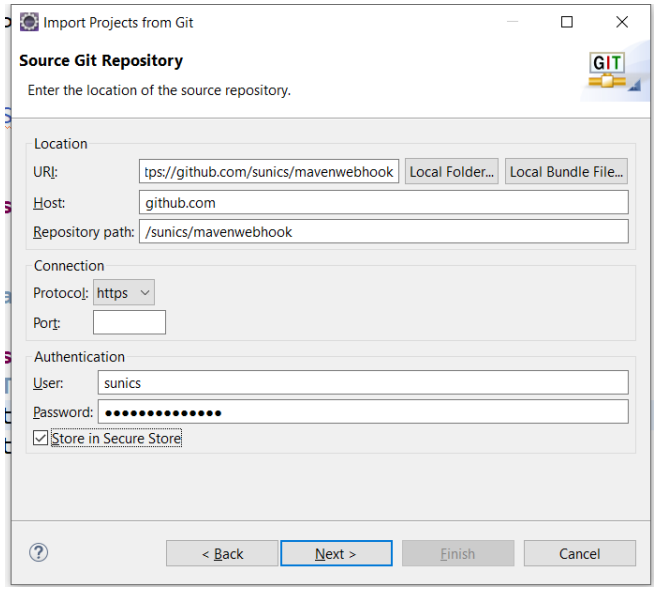


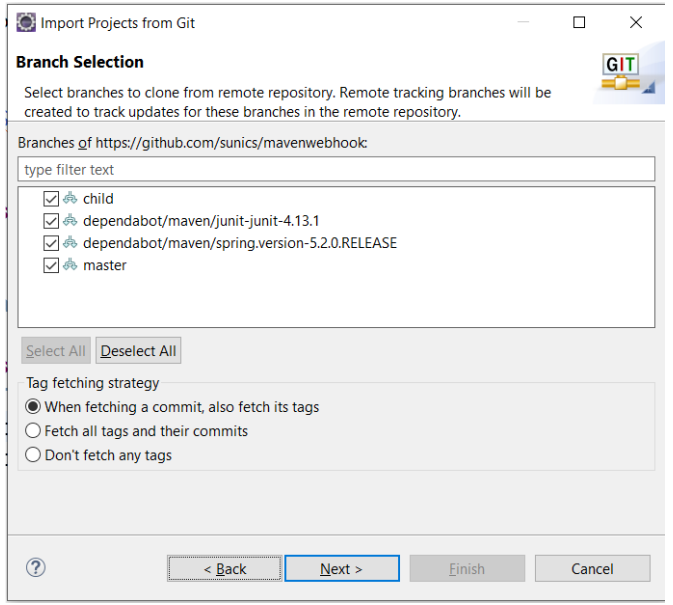
Git Push to remote



Import project from GitHub

- For this, select **File → Import → Git → Projects from Git → Clone URI**





Questions????


प्रगत संगणन विकास केंद्र
सी डैक
CDAC
॥ ज्ञानादेव तु कैवल्यम् ॥
CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING