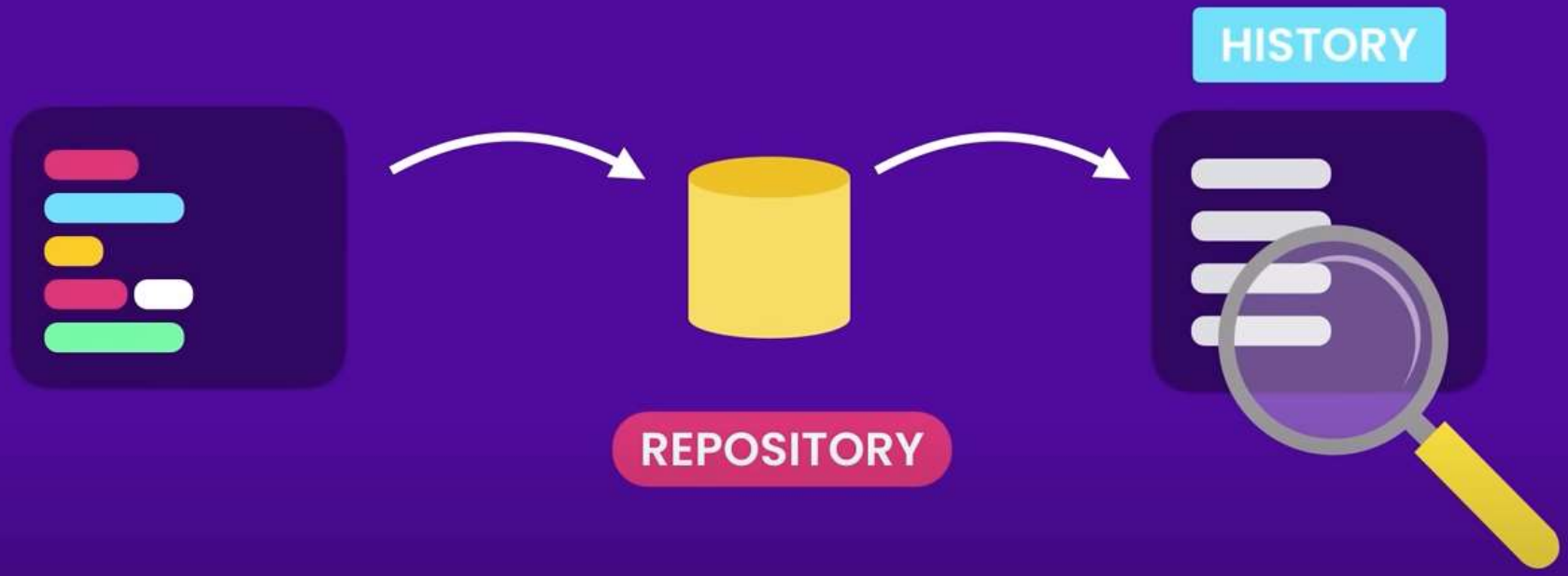
GIT

Lokesh Kumar

What is GIT?

- Git is Version Control System (VCS).
- VCS is a tool that helps to track changes in code
- Use cases:
  - Track history
  - Collaborate



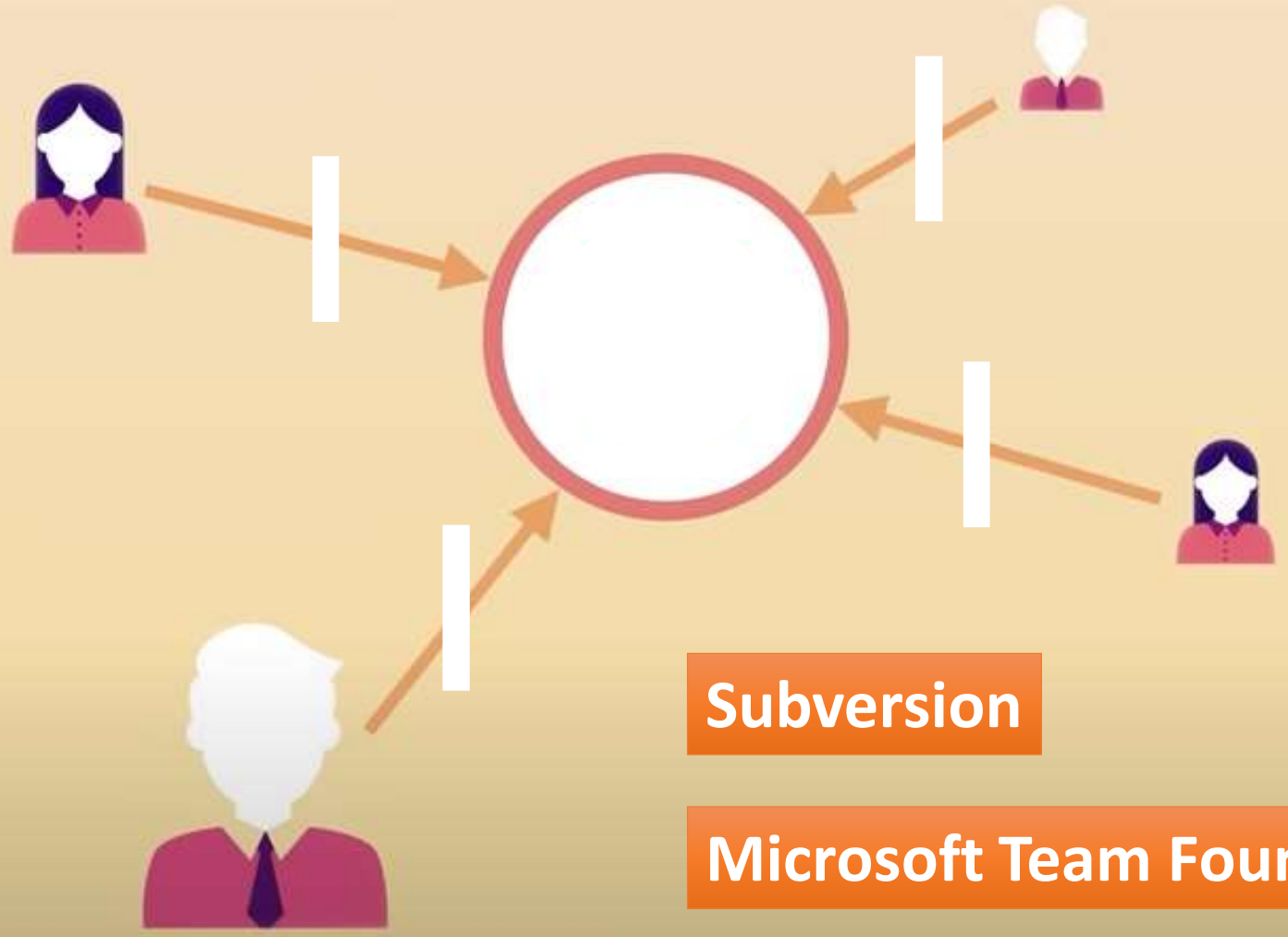
GIT (contd).

Version Control System

Centralized

Distributed

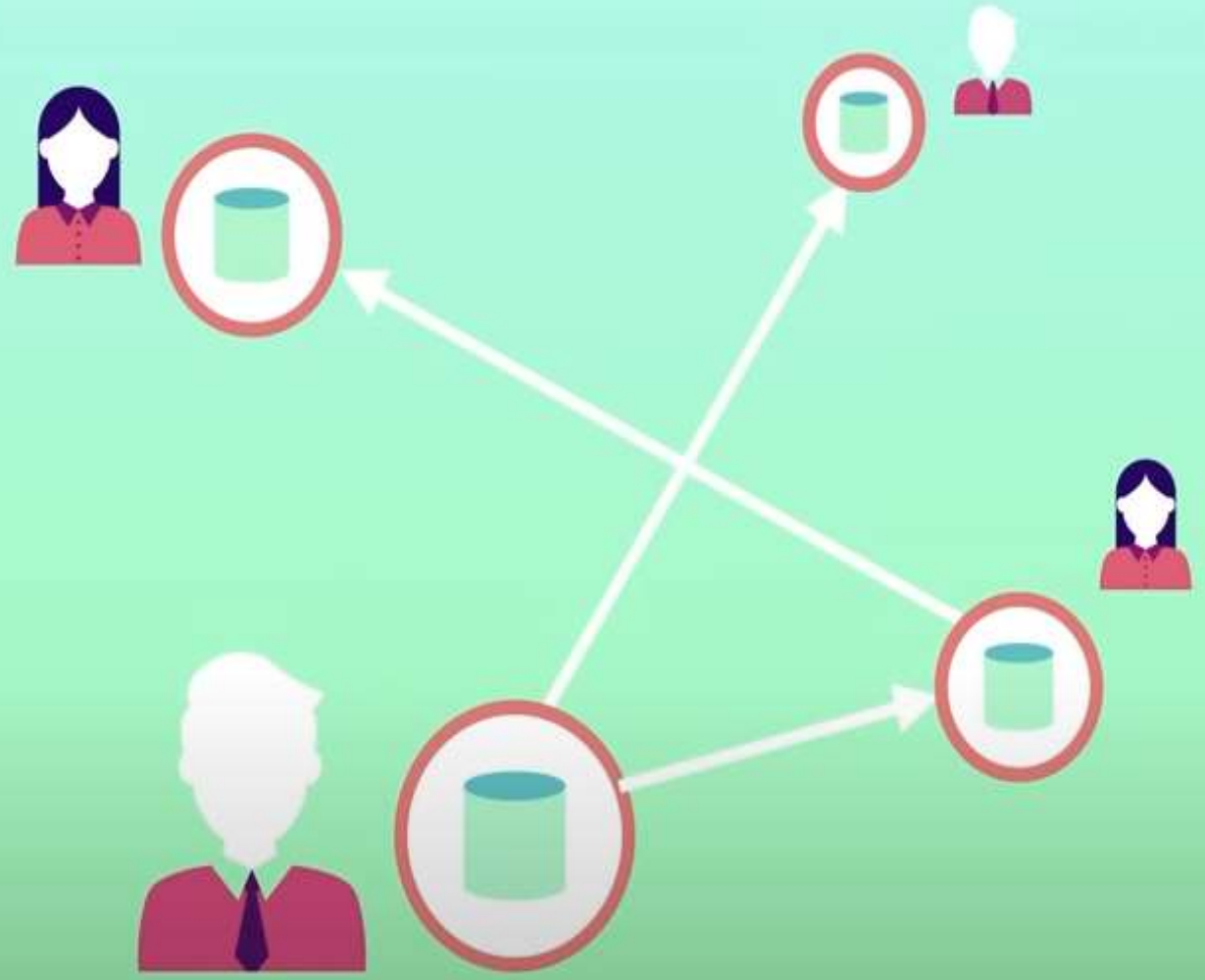
CENTRALIZED



Subversion

Microsoft Team Foundation Server

DISTRIBUTED



Git

Mercurial

USING GIT

- The command line way (fastest and easiest)
- Most Modern IDE have basic Version Control Features
  - GitLens for VSCode
- GUIs
  - GitKraken (Windows, MAC, Linux)
  - SourceTree (Windows and Mac)

Installing Git

Configuring Git

Settings

Name

Email

Default Editor

Line Ending

All users

All repositories of the current user

The current repository

Setting Up

```
C:\Users\LOKESH>git config --global user.name "Lokesh Kumar"
```

```
C:\Users\LOKESH>git config --global user.email lokesh.kumar@woxsen.edu.in
```

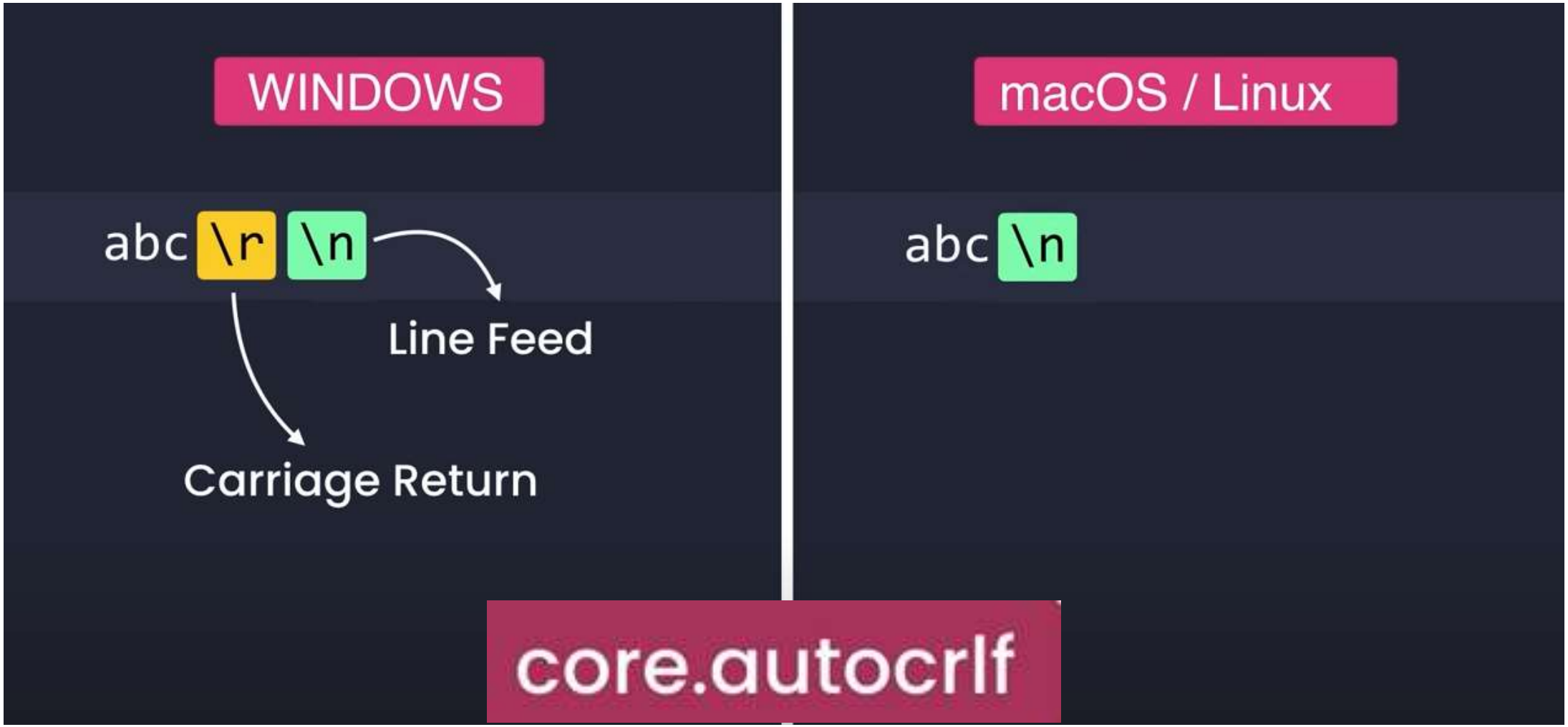
Next, If you don’t set the editor, in MAC default is VIM, In Linux Vi Editor and In Windows Notepad

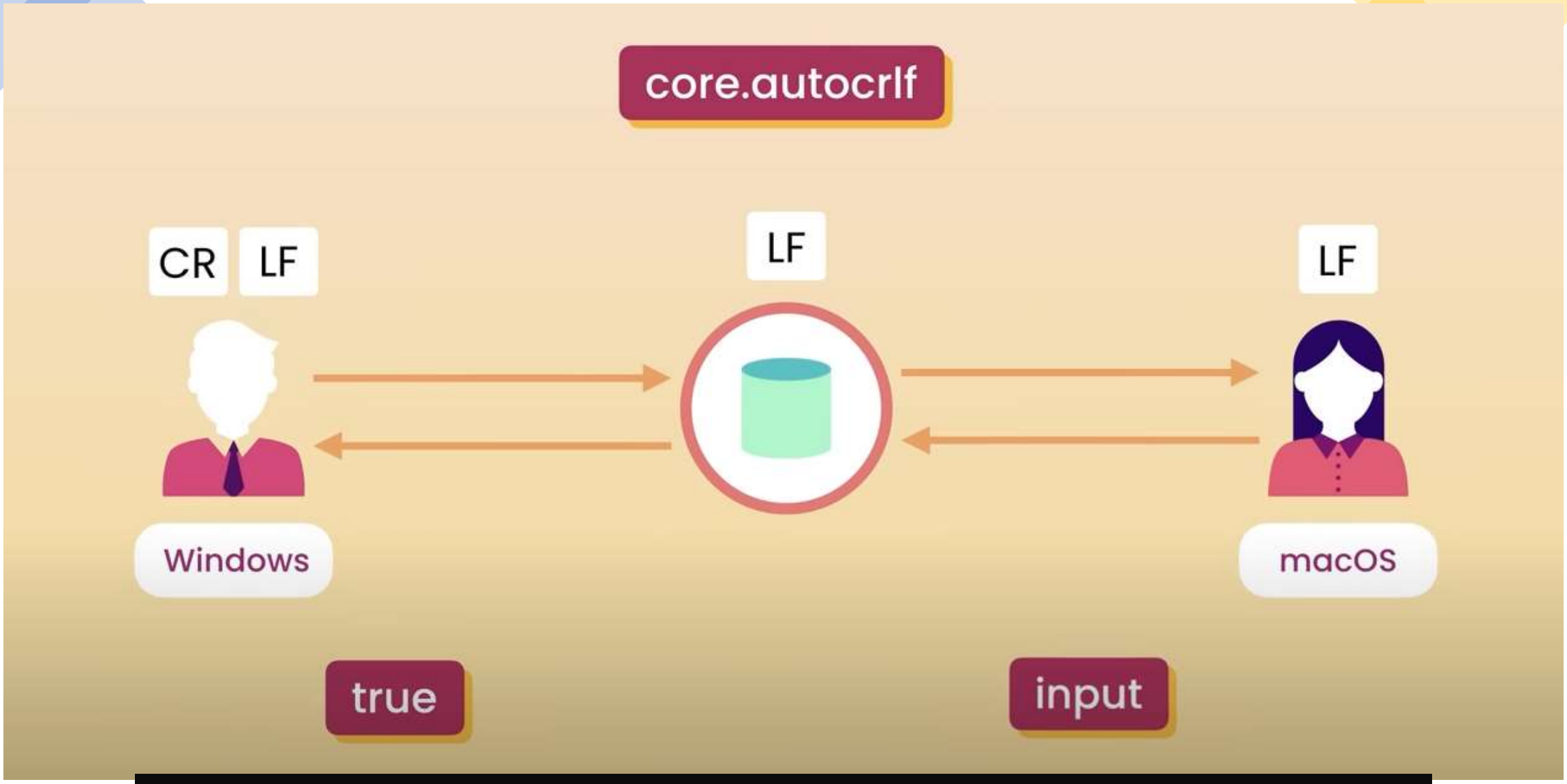
```
C:\Users\LOKESH>git config --global core.editor "code --wait"
```

```
C:\Users\LOKESH>git config --global -e
```

Open Default Editor with global settings

Line Feed


WINDOWS
macOS / Linux
abc \r \n
Line Feed
Carriage Return
abc \n
core.autocrlf




```
C:\Users\LOKESH>git config --global core.autocrlf true
```

Initializing a Repo

Initializing Repo

- Assume you have working/current directory of a project

```
C:\Users\LOKESH>mkdir learning_git
```

```
C:\Users\LOKESH>cd learning_git
```

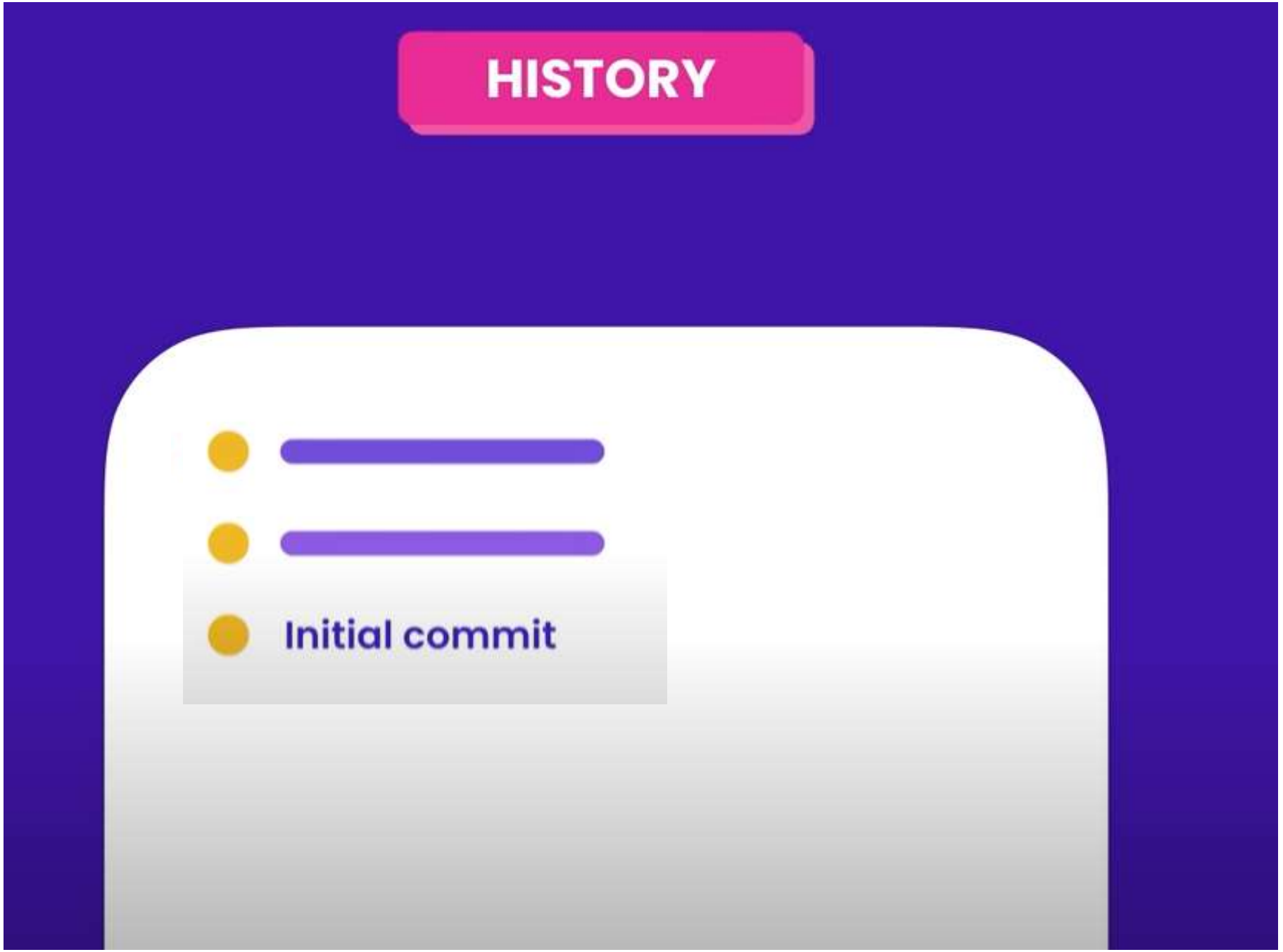
```
C:\Users\LOKESH\learning_git>git init
```

Initialized empty Git repository in C:/Users/LOKESH/learning_git/.git/

Git Workflow

Staging Area / Index allows us to review our work before committing/recording a snapshot

Example


HISTORY
Initial commit

WORKFLOW

```
git commit -m “Removed unused code”
```

Commit

ID

Message

Date/time

Author

Complete snapshot

Skipping Staging Area

Skipping Staging Area

- Assume you have modified a file.
- Now instead of putting it into staging area first and then committing. You can directly commit.
  - Git commit –a –m “bug fixed”
  - OR
  - Git commit –am “bug fixed”

Removing Files

- Let’s assume you don’t need one file now.
- Delete File
- Check Status
- Then add to staging area
- Then commit changes
- NOTE: To remove a file, you have to remove it from both places i.e your working directory and staging area
- Git rm file.txt *.txt //Shortcut to remove file from both places (working directory and staging area) in single command

Renaming/Moving Files

- Let’s say you want to rename the file.
- Say file1.txt → main.js (mv file1.txt main.js)
- NOTE: mv command is to move/rename file
- Run git status and note down message
- You will see both files file1 and main are untracked. As git doesn’t track files which are not in staging area
- Run git add file1.txt main.js
- Now run git status again
- This time git will recognize that you have renamed the file. However file is in staging area, so commit it
- Try git mv main.js file.js and then check status

Ignoring Files

- In large projects, we might need to tell git to ignore certain files/directories like api_key file or config files or binary or log files.
- Lets assume there is log directory within the main project directory.
- Create a log file with in that directory and check status
- You will get message that there is untracked directory. But we don't want it to add to staging area, as we don't want git to track it.
- To do that we have to create a special file called .gitignore
- Echo logs/ > .gitignore
- NOTE that .gitignore must be in root directory
- Now check status again, now you won't get message about untracked log directory or sub-files. However, .gitignore needs to be added to staging area
- NOTE: This only works if you have not already included a file/directory in your git. i.e. suppose you have already a file git repo then you add it to gitignore, it won't work

- To deal with that issue
- Remove that directory/file which you already added in git-repo
- To check files in staging area, use command
- Git ls-files
- So now we have to remove the file/directory from staging area.
- Lets look help, so that we remove directory/file from staging area only not from working directory
- Git rm –h //to open help
- Git rm –cached –r bin/
- Run status again , commit if required

Short Status

- Status command is very comprehensive
- However, there is option to get brief description
- Git status –s
- Notice the output:
  - it has two columns,
  - left column refers to staging area
  - Right column refers to working directory

Review Staged/Un-Staged Changes

- Always review code, before committing as you don't want bad code to be committed.
- Git diff –staged //to see content of each file which are in staging area
- Comparing old copy (committed one) with staging area (new one)
- To check for changes in working directory which are not staged yet
- Git diff
- Try making some changes in a file inside working directory
- Run status
- Then try again diff

Visual Diff Tools

- Kdiff
- P4merge
- WinMerge (Windows only)
- VSCode

- To use VSCode as diff tool, set setting:
- Git config –global diff.tool vscode

Telling Git How to Launch VSCode in Diff mode

- Git config --global difftool.vscode.cmd “code –wait –diff $LOCAL $REMOTE”

- Verify it
- Git config –global –e //to see global configuration setting in editor
- Then try:
  - git difftool, to compare changes in working directory or
  - git difftool --staged, to compare changes in staging area

View History

- Git Log //to see all the commits

- You might see something like HEAD -> master, head is reference to current branch
- To check short summary

- Git log --oneline
- Git log –online --reverse

View a commit

- To check what are changes have been done during a commit
- Two ways
- Git show #commit_id //no need to type entire id, few characters are enough as long multiple commits have same characters
- Git show HEAD~N //to go back N steps from HEAD

- Try: git ls-tree HEAD~N //Notice output
- Starting from right, we have file_name / directory name, then unique identifier based on the content of the file
- Blob is used for files and tree is used for directory

Unstaging Files

- git restore –staged filename or . Or *.js //to restore staging area or a file in staging area
- Try git status –s
- Basically, restore takes the copy of the file or diorectory from next environment (last commit)
- To restore working directory and undo local changes
- Git restore filename/./*.js
- Git clean //check help for arguments
- We commonly use –fd argument

Restoring file to an earlier version

- Lets say you want to restore a file to previous commit, instead of current one
- Git restore //use help to try figure out possible options
- Git restore –source=HEAD~1 filename

Github

- Website that allows developers to store and manage their code.
- https://github.com

Push to Remote Repo

- We will use GitHub for that:
- But first let’s generate ssh keys: ssh-keygen –o
- Add generate public keys to git, in settings, follows github steps
- OR You can use HTTPS, but need to login every time
- Add tag: you can assign a tag/version number to each/group of commit.
- Try: git tag
- git tag -a v1.0 -m "1st release“
- However, you need to push tags
- Git push origin v1.0