Git

- Version control : software tools that enable the management of changes to source code
- Maintaining version history
- Several version control tools: CVS ,SVN,Git etc
- Distributed version control system
- Developed by Linus Torvalds for managing Linux kernel development
- Widely adopted not by several projects
- The Node ecosystem thrives on it

Exercise

- Setting up git on your machine
- Using Git
- Using online Git repositories

Download Git
https://git-scm.com/downloads

Documentation
https://git-scm.com/docs

To check current version of Git
Git –version

Git config –global user.name “ram kumar”
Git config –global user.email ram@gmail.com
Git config –list

Exercise (Instructions): Setting up Git

Objectives and Outcomes

In this exercise you will learn to install Git on your computer. Git is required for using all the remaining Node.js and Node based tools that we encounter in the rest of the course. At the end of this exercise, you would be able to:

- Install Git on your computer
- Ensure that Git can be used from the command-line or command-prompt on your computer
- Set up some of the basic global configuration for Git

Downloading and Installing Git

- To install Git on your computer, go to https://git-scm.com/downloads to download the Git installer for your specific computing platform.
- Then, follow the installation steps as you install Git using the installer.
- You can find more details about installing Git at https://git-scm.com/book/en/v2/Getting-Started-Installing-Git. This document lists several ways of installing Git on various platforms.
- Installing some of the GUI tools like GitHub Desktop will also install Git on your computer.
- On a Mac, setting up XCode command-line tools also will set up Git on your computer.
- You can choose any of the methods that is most convenient for you.

Some Global Configuration for Git

- Open a cmd window or terminal on your computer.
- Check to make sure that Git is installed and available on the command line, by typing the following at the command prompt:

```
git --version
```

- To configure your user name to be used by Git, type the following at the prompt:

```
git config --global user.name "Your Name"
```

- To configure your email to be used by Git, type the following at the prompt:

```
git config --global user.email <your email address>
```

- You can check your default Git global configuration, you can type the following at the prompt:

```
1
git config --list
```

Git is a version control system that is widely used for software development and other version control tasks. It allows multiple developers to work on the same codebase simultaneously, and it keeps track of every change made to the code.

Here is an example of how Git might be used for version control:

1. A developer creates a new project and initializes a Git repository for it.
2. The developer writes some code and commits it to the repository, along with a message describing the changes.
3. Another developer makes additional changes to the code and commits them as well.
4. The first developer reviews the changes and decides to merge them into the main branch of the repository.
5. The project's codebase now includes the changes made by both developers, and the repository's commit history reflects the full history of all changes made to the code.

Git is a powerful tool that is used by many developers and organizations around the world. It can help teams collaborate more effectively, and it makes it easy to track and revert changes as needed.

git example

Here is an example of how to use Git for version control:

1. Install Git on your computer.
2. Open a terminal and navigate to the directory where you want to create your Git repository.
3. Run the following command to initialize the repository: `git init`

4. Create some files and add them to the repository. For example: `echo "My project" > README.md` `git add README.md`
5. Commit the files to the repository with a message describing the changes: `git commit -m "Initial commit"`
6. If you want to push your changes to a remote repository (such as on GitHub), you will need to add a remote and push your changes: `git remote add origin <remote repository URL>` `git push -u origin master`

This is just a basic example of how to use Git for version control. There are many other commands and features available in Git that you can use to manage your repository and collaborate with others.

git step by step

Here is a step-by-step guide for using Git for version control:

1. Install Git on your computer. You can download it from the official website (https://git-scm.com/) or install it using a package manager (such as apt on Linux or Homebrew on Mac).
2. Open a terminal and navigate to the directory where you want to create your Git repository. This can be an existing project or a new one.
3. Initialize the repository by running the following command: `git init`
4. Add your files to the repository. You can use the `git add` command to add individual files, or you can use `git add .` to add all the files in the current directory and its subdirectories.
5. Commit the changes to the repository with a message describing the changes: `git commit -m "My commit message"`
6. If you want to push your changes to a remote repository (such as on GitHub), you will need to add a remote and push your changes: `git remote add origin <remote repository URL>` `git push -u origin master`

That's the basic workflow for using Git for version control. There are many other commands and features available in Git that you can use to manage your repository and collaborate with others.

git command

Here is a list of some common Git commands:

- `git init`: Initialize a new Git repository.
- `git clone`: Clone an existing repository from a remote location.
- `git add`: Add one or more files to the staging area.
- `git commit`: Commit the staged changes to the repository with a message describing the changes.
- `git push`: Push the local commits to a remote repository.
- `git pull`: Fetch and merge changes from a remote repository.
- `git branch`: List, create, or delete branches.
- `git checkout`: Check out a branch or specific commit.
- `git merge`: Merge one branch into another.

- `git stash`: Stash changes that have not been committed.
- `git diff`: Show the differences between two commits or files.
- `git log`: Show the commit history for a repository.

These are just a few of the many commands available in Git. There are many others that you can use to manage your repository and collaborate with others.

Git repository and collaborate commands

Here are some Git commands that you can use to manage your repository and collaborate with others:

- `git init`: Initialize a new Git repository.
- `git clone`: Clone an existing repository from a remote location.
- `git add`: Add one or more files to the staging area.
- `git commit`: Commit the staged changes to the repository with a message describing the changes.
- `git push`: Push the local commits to a remote repository.
- `git pull`: Fetch and merge changes from a remote repository.
- `git branch`: List, create, or delete branches.
- `git checkout`: Check out a branch or specific commit.
- `git merge`: Merge one branch into another.
- `git stash`: Stash changes that have not been committed.
- `git diff`: Show the differences between two commits or files.
- `git log`: Show the commit history for a repository.

Using these commands, you can manage your repository, track and revert changes, and collaborate with others on your codebase. You can also use tools like GitHub or GitLab to manage your repository and collaborate with others online.