CHOCOLATELY IN WINDOWS

☑ What is Chocolatey in Windows?

🔧 What does it do?

Chocolatey helps you:

- Install software using the command line
- Update apps with one command
- Remove programs cleanly
- Manage dependencies and automation for development setups

📦 Example:

powershell

choco install googlechrome

This command installs Google Chrome on your PC automatically!

💻 How to install Chocolatey?

You can ask to ChatGPT for it.

🍫 Popular apps you can install with Chocolatey:

- choco install vscode
- choco install python
- choco install git
- choco install nodejs

HOW TO USE GIT & GITHUB

Local Repository: **The project folder on your own computer managed by Git.**

Remote Repository: **A copy of your project on GitHub (or any remote server)**
(GitHub Repo) **that you can share with others.**

- **What are the git directory and the working tree?**
- The git directory contains all the changes and their history and the working tree contains the current versions of the files.

Basic to know:

- **Commit** = Change/Update
- **Initial Commit** = First change in project
- **git --version** → Git version check
- **ls** → Show files
- **pwd** → Current directory (c/user/Asus)
- **clear** → Clear terminal

- **Configuring Git:** When you **commit** something in Git, Git records **who** made That change. So, you must tell Git:
  - **Your Name**
  - **Your Email**

One-Time Setup:

- ***git config --global user.name "Your Name"***
- ***git config --global user.email "you@example.com"***
- ***git config --list # Show config***

MAKE CLONE OF ALREADY EXISTS REPO:

1. clone: cloning a repository on our local machine.

- ***git clone <repo-link>***
- ***cd <folder-name>*** **# Enter folder**
- ***cd ..*** **# Go back (Exit from folder)**
- ***ls -a or ls -Hidden*** # Show hidden files

[**Note:** **.git** means it's a Git-tracked folder.]

2. Status, Add, Commit & Push:

Status: The git status command tells you the current state of your files in the repo.

- **How to Read git status Output:**

✅ **Clean Working Directory:**

[On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean]

⚠️ **Modified Files:**

[Changes not staged for commit:

modified: index.html]

 # Fix this with "git add index.html"

🆕 **Untracked Files:**

[Untracked files:

newfile.py]

 # Fix this with “git add newfile.py”

🟢 **Staged Changes (Ready to Commit):**

[Changes to be committed:

new file: newfile.py

modified: index.html]

 # Fix this git commit -m "Added new file and updated index"

🧠 **Work Flow:**

git clone <repo link>

git cd <folder name>

 # if Changed(Modified) | New file(Untracked)

git add <file name> or git add . **(**Add all changes in the current directory**)**

git commit -m “massage”

git push origin “branch name”

- ***git add -p*** # allows a user to interactively review patches before adding to the current commit.

  - **Meaning of Options:**

Key	Action
y	Yes — stage this hunk
n	No — skip this hunk
q	Quit — cancel the rest
a	Stage this and all remaining hunks
d	Don't stage this and all remaining hunks
s	Split hunk into smaller parts (if possible)
e	Edit hunk manually before staging
?	Help — show all options again

- ***git commit -a*** # a shortcut to stage any change to tracked files and commit them in one step.

  And you can use like this:

  git commit -a -m "your commit message"

- ***git commit --amend*** # lets you modify your most recent commit without creating a new one.
  You can use only with local repo.

MAKE REPO FROM SCRATCH:

3. Git init command:

- **What is Git init?**
  - Git init starts a new local Git repository in your project folder

🛠️ **What It Does:**

- Creates a hidden folder called .git
- Tells Git to **start tracking changes** in that folder

STEP 1: Create a folder and initialize Git:

- ***mkdir my-project*** # Step 1: Create a folder
- ***cd my-project*** # Step 2: Move into the folder
- ***git init*** # Step 3: Initialize Git in this folder

➕ **Extra Notes:**

- "mkdir" create a new folder.
- You can create multiple folders at once:

  mkdir folder1 folder2 folder3

- You can also create nested folders:

STEP 2: Check if Git is initialized:

- ***ls -a or ls -Hidden***

STEP 3: Make an initial commit:

- ***touch index.html*** # Create a new file
- ***git add .*** # Stage all files
- ***git commit -m "Initial commit"*** # First commit

STEP 4: (Optional) Connect to GitHub:

- **git remote add origin <repo-link>** # Link to remote GitHub repo
- ***git remote -v*** # to verify remote
- ***git push origin main*** # Push local code to GitHub

  git push -u origin main # Push local code to GitHub

🧠 **Breakdown:**

- **git push → Push your changes to GitHub**
- **origin → The default name for your GitHub remote**
- **main → The branch name (usually "main")**
- **-u → Set upstream link, so next time you can just do: git push**

🧠 **Work Flow:**

mkdir <folder name>

cd <folder name>

git init

git add .

git commit -m “massage”

git remote add origin <repo link>

git push origin <branch name>

GIT DIFF:

Shows the changes made in the working directory that haven't staged.

Area	Command
See unstaged changes	*git diff*
See staged changes	*git diff -- staged*
See all local changes	*git diff HEAD*
Compare two commits	*git diff <c1> <c2>*
Compare two branches	*git diff <main> <dev>*
Summary of file differences	*git diff --stat*

🛠 **Helpful Options:**

Option	Description
git diff --name-only	Shows only the names of changed files

Option	Description
git diff --stat	Summary of changes (files + line counts)
git diff <file>	Only see changes in a specific file

DELETE, RENAME, IGNOR & DIFF

git rm <file name>: is a Git command used to remove files from your working directory and stage the removal for the next commit.

git mv <old filename> <new filename>: is used to move or rename files in a Git repository, and automatically stages the change for the next commit.

. *gitignore:* put file into .gitignore which you want to ignore like this:

STEP 1: ***create .gitignore file***

gitignore tells Git **which files or folders to ignore** (i.e., not track or commit) in your project.

STEP 2:

📁 **Example .gitignore file:**

Gitignore:

Ignore all .log files

****.log***

Ignore the venv folder

venv/

Ignore a specific file

secrets.txt

Ignore everything in temp/ folder

temp/