## **Objectives**

* Explain git ignore
* Explain how to ignore unwanted files using git ignore

In this hands-on lab, you will learn how to:

* Implement git ignore command to ignore unwanted files and folders

## **Prerequisites**

The following are the pre-requisites to complete this hands-on lab:

* Setting up Git environment
* Integrate notepad++ as a default editor
* A Git repository in the local system and a remote repository in GitLab

Notes\*:

Please follow the below steps for creating a free account in GitHub.

Do not use cognizant credentials to login to GitHub.

Estimated time to complete this lab: **20 minutes.**

Create a **“.log”** file and a **log folder** in the working directory of Git. Update the **.gitignore** file in such a way that on committing, these files (.log extensions and log folders) are ignored.

Verify if the git status reflects the same about working directory, local repository and git repository.

**ANSWER**

### **1. What is .gitignore?**

* .gitignore is a plain text file that tells Git which files or folders it should **not** track or commit to the repository.
* When a file or folder is listed in .gitignore, Git will completely ignore it when you run commands like git status, git add, or git commit.
* This is useful for keeping your repository clean and free from unnecessary or sensitive files.

### **2. Why do we use .gitignore?**

Here are some common reasons:

* **Avoid committing temporary files** (like log files, cache files, or build artifacts) that are generated during development.
* **Ignore personal or system-specific files** that only exist on your computer (e.g., editor settings, OS hidden files).
* **Prevent sensitive data leaks**, such as passwords or API keys, from being uploaded to GitHub or GitLab.
* **Reduce clutter** in the repository, making it easier for developers to focus on important code.

### **3. How does .gitignore work?**

* You create a file named .gitignore in your Git repository.
* Inside it, you list patterns (rules) that match the files/folders you want to ignore.
* Git checks this file and **skips tracking** any files that match the patterns.

### **Step 1 – Create .log file and log folder**

$ touch error.log  
$ mkdir log  
$ echo "Test log data" > log/test1.log

### **Step 2 – Create .gitignore file**

$ notepad++ .gitignore

# Ignore all .log files

\*.log

# Ignore log folder

Log/

### **Step 3 – Check status before commit**

$ git status  
On branch master  
Untracked files:  
 (use "git add <file>..." to include in what will be committed)  
 .gitignore  
  
nothing added to commit but untracked files present (use "git add" to track)

### **Step 4 – Commit .gitignore**

$ git add .gitignore  
$ git commit -m "Added .gitignore to ignore .log files and log folder"  
[master 1a2b3c4] Added .gitignore to ignore .log files and log folder  
 1 file changed, 2 insertions(+)  
 create mode 100644 .gitignore

### **Step 5 – Final status check**

$ git status  
On branch master  
nothing to commit, working tree clean