**GIT**

**INTRODUCTION:**

**Git** is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

**Git HUB:**

GitHub is a hosting site where developers and programmers can upload the code they create and work collaboratively to improve it. A defining feature of GitHub is its robust version control system.

**How Git HUB works?**

Of the many features offered by GitHub, three of the most consequential include forking, pull requests, and merging. Forking a project creates a copy that allows you to experiment freely without affecting the original project.

**Creating a Repository:**

* In the upper-right corner of any page, use the drop-down menu, and select New repository.
* Type a short, memorable name for your repository.
* Optionally, add a description of your repository.
* Choose a repository visibility.
* Select Initialize this repository with a README.
* Click Create repository.

**Git Commands:**

1. **git:**

Moves changes from the working directory to the staging area. This gives you the opportunity

to prepare a snapshot before committing it to the official history.

1. **git branch:**

This command is your general-purpose branch administration tool. It lets you create isolated development environments within a single repository.

1. **git checkout:**

In addition to checking out old commits and old file revisions, git checkout is also the means to navigate existing branches. Combined with the basic Git commands, it’s a way to work on a particular line of development.

1. **git clean:**

Removes untracked files from the working directory. This is the logical counterpart to git reset, which (typically) only operates on tracked files.

1. **git clone:**

Creates a copy of an existing Git repository. Cloning is the most common way for developers to obtain a working copy of a central repository.

1. **git commit:**

Takes the staged snapshot and commits it to the project history. Combined with git add, this defines the basic workflow for all Git users.

1. **git config:**

A convenient way to set configuration options for your Git installation. You’ll typically only need to use this immediately after installing Git on a new development machine.

1. **git init:**

Initializes a new Git repository. If you want to place a project under revision control, this is the first command you need to learn.

1. **git log:**

Lets you explore the previous revisions of a project. It provides several formatting options for displaying committed snapshots.

1. **git merge:**

A powerful way to integrate changes from divergent branches. After forking the project history with git branch, git merge lets you put it back together again.

1. **git pull:**

Pulling is the automated version of git fetch. It downloads a branch from a remote repository, then immediately merges it into the current branch. This is the Git equivalent of svn update.

1. **git push:**

Pushing is the opposite of fetching (with a few caveats). It lets you move a local branch to another repository, which serves as a convenient way to publish contributions. This is like svn commit, but it sends a series of commits instead of a single changeset.

1. **git status:**

Displays the state of the working directory and the staged snapshot. You’ll want to run this in conjunction with git add and git commit to see exactly what’s being included in the next snapshot.