GIT Services Presentation

What is Git?

* Git is a free and open-source distributed version control system.
* Its purpose is to keep tracks of projects and files as they change over time as other users can contribute to them with their knowledge.

**Key Benefits**

* **Complete Project History:**
  + Track every change, decision, and progression.
  + Provides full context, making it easy for anyone to understand the project.
* **Seamless Team Collaboration:**
  + Enables efficient teamwork across all locations and time zones.
  + Maintains source code integrity while multiple developers work concurrently.
* **Safe & Flexible Development:**
  + Use branching to safely propose and test new features or fixes.
  + Experiment without impacting the stable, main codebase.
* **Enhanced Business Alignment:**
  + Breaks down communication barriers between diverse teams.
  + Facilitates expert collaboration on critical projects, driving efficiency.

Git Repositories:

* Repository (Repo): Git stores information about the project on a repository.
* Commits: Snapshots or milestones, it records changes to the project at a given point of time.
* Heads: References (pointers) to the latest commits on various branches, defining the current state of your project's history.
* **.git Folder:** All repository information (commits, heads, configuration) is stored in this hidden sub-folder within your project directory.

Basic Git Commands: Your Project's Toolkit

**Setting Up Your Project**

* **git init** 
  + **Start a New Project Here:** Turns a regular folder on your computer into a Git-managed project.
  + Creates a hidden folder (.git) to store all project history.
* **git clone**
  + **Get an Existing Project:** Downloads a complete copy of a project from GitHub (or another online location) to your computer.
  + Includes all its files, history, and different versions.

**Tracking & Saving Your Work**

* **git status**
  + **What's Changed?:** Shows you which files you've changed, which are new, or which are ready to be saved.
  + Your project's current checklist.
* **git add**
  + **Ready to Save:** Selects the specific changes you want to include in your next "save point."
  + It's like picking items for a photo before taking the picture.
* **git commit**
  + **Save Your Progress:** Takes a permanent "snapshot" of your selected changes and adds it to the project's history.
  + Always includes a short message explaining what you did.

**Managing Different Versions (Branches)**

* **git branch**
  + **See Your Work Paths:** Shows you all the different parallel versions (branches) you're currently working on.
  + Helps keep separate features organized.
* **git merge**
  + **Combine Work:** Blends changes from one branch into another.
  + Often used to bring a completed feature into the main project.

**Syncing with Others (or Online)**

* **git pull**
  + **Get Latest from Online:** Downloads the newest changes from the online project (e.g., GitHub) to your computer.
  + Keeps your local work up-to-date with your team.
* **git push**
  + **Send Your Work Online:** Uploads your saved changes (commits) from your computer to the online project (e.g., GitHub).
  + Shares your contributions with others.

**The GitHub Flow**

* A **lightweight, branch-based workflow** using core Git commands.
* Designed for **team collaboration** and continuous development.

**Six Steps of the GitHub Flow**

**Create a Branch**

* **Goal:** Work on new features or fixes without affecting the main project.
* Create **"topic branches"** from your main deployment branch (usually main or master).
* **Benefit:** Allows parallel work, keeping teams focused and enabling quick releases.

**Add Commits**

* **Goal:** Regularly save your progress on your branch.
* Create **snapshots** of your development efforts.
* **Benefit:** Provides safe, revertible points in your project's history, acting like "save points."

**Open a Pull Request (PR)**

* **Goal:** Announce your work and start the review process.
* Publicizes your ongoing changes and opens them for discussion.
* **Benefit:** Sets the stage for a transparent and collaborative development process.

**Discuss and Review Code**

* **Goal:** Get feedback and improve your code.
* Teams comment, test, and review the code in the open Pull Request.
* **Benefit:** Fosters an open, participatory culture and improves code quality.

**Merge**

* **Goal:** Integrate your changes into the main project.
* Once approved, clicking "Merge" on GitHub automatically combines your branch into the main deployment branch.
* **Benefit:** Keeps the entire development history of your branch visible within the merged PR.

**Deploy**

* **Goal:** Release your changes to users.
* Teams can choose their preferred release cycles or use automated deployment tools.
* **Benefit:** Provides assurance that code on the deployment branch has gone through a robust review and workflow.

**What is "Fork"?**

* **Definition:** "Forking" is a way to copy an existing repository (project) from someone else's GitHub account to **your own** GitHub account.
* It creates a **personal online copy** of their project under your username.

**Fork vs. Clone**

Understanding the Difference

* **Fork:**
  + Creates a **copy on GitHub** (your online account).
  + This copy stays linked to the original project.
  + *Think of it as getting your own editable version of a public document online.*
* **Clone:**
  + Downloads a copy of a repository from **GitHub to your local computer**.
  + This is what you work on directly.
  + *Think of it as downloading that editable document to your computer.*
* **Key Point:** You often **Fork first**, then **Clone** your forked copy to your computer to start working.

**How It All Works: The Forking Workflow**

1. **Fork the Original:**
   * You make a copy of the original project to *your* GitHub account.
2. **Clone Your Fork:**
   * You then download *your* copy (the fork) from GitHub to your local computer.
3. **Make Your Changes:**
   * You work on the project files on your computer, making edits or adding new features.
4. **Push to Your Fork:**
   * You upload your local changes back to *your* forked copy on GitHub.
5. **Open a Pull Request:**
   * You ask the original project's owner to review your changes and consider adding them to *their* original project.
   * *(This is how you contribute back!)*