Class Notes: Distributed Version Control - Session 1 (Git & GitHub - TL1)

Session Objectives

By the end of this session, students will be able to:

- Outline the necessity for version control.
- Provide an overview of Git.
- Describe the basics of Git.
- List key Git terminology.
- Set up Git on their systems.

1. Necessity for Version Control

Why Version Control?
Version control systems (VCS) track changes to files, enabling collaboration, history tracking, and error recovery. Without VCS, teams face issues like:

- Overwriting changes in collaborative projects.
- Losing previous file versions.
- Difficulty tracking who made what changes and when.

Benefits of Version Control:

- **Collaboration**: Multiple developers can work simultaneously without conflicts.
- **History Tracking**: Revert to previous versions if errors occur.
- **Accountability**: Track changes with author and timestamp.
- **Branching**: Experiment with new features without affecting the main project.

Example Scenario:
Imagine a team editing a shared document. Without version control, one person's changes might overwrite another's. VCS ensures changes are merged systematically.

2. Introduction to Git

What is Git?
Git is a distributed version control system designed for speed, efficiency, and reliability. It allows every developer to have a full copy of the project's history, enabling offline work and robust collaboration.

Key Features of Git:

- **Distributed**: Every user has a complete repository copy.
- **Fast**: Operations like committing and branching are quick.
- **Flexible**: Supports non-linear development via branches.
- **Open-Source**: Free and widely used in industry.

Git in Action:
A developer can commit changes locally, push them to a remote repository (e.g., GitHub), and pull updates from teammates, all while maintaining a complete project history.

3. Git Basics

Core Concepts:

- **Repository (Repo)**: A storage location for a project's files and history.
- **Commit**: A snapshot of changes made to files.
- **Branch**: A parallel version of the repository for isolated work.
- **Clone**: A copy of an existing repository.
- **Working Directory**: The local folder containing project files.

How Git Works:

1. Initialize or clone a repository.
2. Make changes to files in the working directory.
3. Stage changes (select what to commit).
4. Commit changes to save them in the repository's history.
5. Push commits to a remote repository for sharing.

Example:

```
# Initialize a new Git repository
git init my-project
cd my-project
# Create a file
echo "Hello, Git!" > README.md
# Stage the file
git add README.md
# Commit the changes
git commit -m "Initial commit"
```

4. Git Terminology

Key terms to understand:

- **Repository**: The project's storage, including files and history.
- **Commit**: A recorded change with a message, author, and timestamp.
- **Branch**: A separate line of development (default: `main` or `master`).
- **Stage**: The process of selecting changes to include in the next commit.
- **Remote**: A repository hosted elsewhere (e.g., GitHub).
- **Clone**: Copying a repository to a local machine.
- **Pull**: Fetching and merging changes from a remote repository.
- **Push**: Sending local commits to a remote repository.

Example:

- Creating a commit: `git commit -m "Add feature X"`.
- Checking repository status: `git status`.

5. Setting Up Git

Installing Git:

- **Windows**: Download from git-scm.com and follow the installer.
- **Mac**: Install via Homebrew (`brew install git`) or Xcode tools.
- **Linux**: Use package manager (e.g., `sudo apt install git` for Ubuntu).

Configuring Git:
Set your name and email to identify commits:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Verifying Installation:
Check Git version to confirm installation:

```
git --version
```

Expected output: `git version 2.x.x` (version may vary).

Command-Line Basics:
Git commands are run in a terminal (e.g., Command Prompt, Terminal, or Bash).

- Navigate directories: `cd path/to/folder`.
- List files: `ls` (Linux/Mac) or `dir` (Windows).
- Create a file: `echo "content" > filename.txt`.

Classwork: Setting Up Git and Initializing a Repository

Objective: Install Git, configure it, and create a simple repository.

Steps:

1. **Install Git**:
   - Download and install Git from git-scm.com.
   - Verify installation: `git --version`.
2. **Configure Git**:
   - Set your name: `git config --global user.name "Your Name"`.
   - Set your email: `git config --global user.email "your.email@example.com"`.
   - Check configuration: `git config --list`.
3. **Create a Repository**:
   - Create a folder: `mkdir my-first-repo && cd my-first-repo`.

- Initialize Git: `git init`.
- Create a file: `echo "# My First Repo" > README.md`.
- Stage the file: `git add README.md`.
- Commit: `git commit -m "Initial commit"`.
- Check status: `git status`.

4. **Explore History**:
   - View commit history: `git log`.

Deliverable:
Submit a screenshot of your terminal showing the output of `git --version`, `git config --list`, and `git log`.

Session Test

Instructions: Answer the following questions or complete the tasks. Submit answers via OnlineVarsity.

Multiple-Choice Questions:

1. What is a primary benefit of version control?
   a) Increases file size
   b) Tracks changes and enables collaboration
   c) Deletes old files
   d) Prevents code editing
   Answer: b
2. What does the `git init` command do?
   a) Deletes a repository
   b) Initializes a new Git repository
   c) Commits changes
   d) Clones a remote repository
   Answer: b

Practical Task:

1. Install Git on your system.
2. Configure your name and email.
3. Create a new repository named `test-repo`.

4. Add a file `hello.txt` with the content "Hello, Git World!".
5. Stage and commit the file with the message "Add hello.txt".
6. Submit the output of `git log` and `git status`.

Expected Output for Task:

- `git log`: Shows one commit with message "Add hello.txt".
- `git status`: Shows a clean working directory.

Self-Study (S1-S2)

- **Read**: Refer to Session 1 of "Version Control with Git and GitHub" on OnlineVarsity.
- **Practice**: Complete the "Practice 4 Me" exercises for Session 1.
- **Assignment**: Solve scenario-based assignments on OnlineVarsity (e.g., simulate creating a repository for a team project).
- **Explore**: Check the Glossary and References on OnlineVarsity for additional Git terms and resources.

References

- "Version Control with Git" by Jon Loeliger, Matthew McCullough (Library Reference).
- OnlineVarsity: Learner’s Guide (eBook), Glossary, FAQ, and References for Session 1.