Class Notes: Distributed Version Control - Session 2 (Git & GitHub - TL2)

Session Objectives

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

- Describe how to create a new Git repository.
- Explain how to set up a project folder for Git.
- Define Git configuration and its levels.
- Describe how to clone an existing Git repository.
- Outline the process of staging and committing changes.

1. Getting a Git Repository

What is a Git Repository?
A Git repository is a directory that contains your project files and a `.git` folder, which stores the history and configuration of the project.

Creating a New Repository:

- Use `git init` to initialize a new repository in a folder.
- This creates a `.git` subdirectory to track changes.

Example:

```
# Create a new directory and initialize a repository
mkdir my-project
cd my-project
git init
```

Output: `Initialized empty Git repository in /path/to/my-project/.git/`

2. Setting Up the Project Folder

Project Folder Structure:

- The project folder (working directory) contains all your files (e.g., code, documentation).
- The `.git` folder is automatically created by `git init` and should not be modified manually.

Steps to Set Up:

1. Create a folder for your project: `mkdir project-name`.
2. Navigate to the folder: `cd project-name`.
3. Initialize Git: `git init`.
4. Add initial files (e.g., `README.md`, source code).
5. Stage and commit files to start tracking.

Example:

```
# Set up a project folder
mkdir blog-project
cd blog-project
git init
echo "# Blog Project" > README.md
git add README.md
git commit -m "Initial commit with README"
```

3. Git Configuration

What is Git Configuration?
Git configuration sets user-specific settings, such as name, email, and default editor, to identify commits and customize behavior.

Basic Configuration Commands:

- Set user details (required for commits):

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

- Set default editor (e.g., for commit messages):
```
git config --global core.editor "nano"
```

Checking Configuration:

- View all settings: `git config --list`.
- View specific setting: `git config user.name`.

Example:

```
# Configure Git
git config --global user.name "Jane Doe"
git config --global user.email "jane.doe@example.com"
# Verify settings
git config --list
```

4. Levels of Configuration

Git supports three levels of configuration, applied in order of precedence:

- **Local**: Specific to a single repository (stored in `.git/config`).
```
git config --local user.email "project.email@example.com"
```

- **Global**: Applies to all repositories for the user (stored in `~/.gitconfig`).
```
git config --global user.name "Jane Doe"
```

- **System**: Applies to all users on the system (stored in `/etc/gitconfig`).
```
git config --system core.editor "vim"
```

Precedence: Local overrides Global, which overrides System.

Example:

```
# Set local email for a specific project
cd my-project
git config --local user.email "work.email@example.com"
# Check local config
git config --local --list
```

5. Git Clone

What is Cloning?

Cloning creates a local copy of a remote repository, including its files and history.

Command:

```
git clone <repository-url> [directory-name]
```

Example:

```
# Clone a repository from GitHub
git clone https://github.com/example/sample-repo.git
cd sample-repo
# View repository contents
ls
```

Notes:

- The repository URL is typically from platforms like GitHub, GitLab, or Bitbucket.
- Cloning automatically sets up a remote connection named `origin`.

6. Staging Files

What is Staging?

Staging selects files or changes to include in the next commit. The staging area (index) acts as a snapshot of changes to be committed.

Commands:

- Stage a specific file: `git add filename`.
- Stage all changes: `git add .`.
- Check status: `git status`.

Example:

```
# Create and stage a file
echo "Initial content" > index.html
git add index.html
git status
```

Output:

```
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html
```

7. Committing Files

What is a Commit?

A commit is a snapshot of staged changes, saved with a message describing the changes.

Command:

```
git commit -m "Descriptive commit message"
```

Best Practices:

- Write clear, concise commit messages (e.g., "Add login page").
- Commit related changes together (e.g., don’t mix unrelated features).

Example:

```
# Stage and commit changes
echo "Update content" >> index.html
git add index.html
git commit -m "Update index.html with new content"
# View commit history
git log --oneline
```

Output:

```
a1b2c3d Update index.html with new content
e4f5g6h Initial commit with README
```

Classwork: Creating and Managing a Git Repository

Objective: Create a repository, configure Git, clone a sample repo, and practice staging and committing.

Steps:

1. **Configure Git Locally**:
   - Set a local email for a new project:
     ```
     mkdir my-app
     cd my-app
     git init
     git config --local user.email "project.email@example.com"
     ```
   - Verify: `git config --local --list`.
2. **Set Up a Project Folder**:
   - Create a file `app.js` with content: `console.log("Hello, Git!");`.
   - Stage and commit:
     ```
     git add app.js
     git commit -m "Add initial app.js"
     ```

3. **Clone a Repository**:
   - Clone a sample repository (use a public repo or instructor-provided URL, e.g., `https://github.com/example/sample-repo.git`):
     ```
     git clone https://github.com/example/sample-repo.git sample-repo
     cd sample-repo
     ```
   - List files: `ls`.
4. **Stage and Commit Changes**:
   - In `my-app`, modify `app.js` to add: `console.log("Updated!");`.
   - Stage and commit:
     ```
     git add app.js
     git commit -m "Update app.js with new message"
     ```
   - Check history: `git log --oneline`.

Deliverable:
Submit screenshots of:

- `git config --local --list` from `my-app`.
- `git log --oneline` from `my-app`.
- `ls` output from the cloned `sample-repo`.

Session Test

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

Multiple-Choice Questions:

1. What does `git clone` do?
   a) Deletes a repository
   b) Creates a local copy of a remote repository
   c) Commits changes
   d) Initializes a new repository
   Answer: b
2. Which command stages all modified files?

a) `git commit -m "message"`
b) `git add .`
c) `git init`
d) `git status`
Answer: b

Practical Task:

1. Create a new repository named `test-app`.
2. Configure a local user name: `git config --local user.name "Test User"`.
3. Create a file `main.py` with content: `print("Git Test")`.
4. Stage and commit the file with message: "Add main.py".
5. Modify `main.py` to add: `print("Updated")`.
6. Stage and commit with message: "Update main.py".
7. Submit the output of:
   - `git config --local --list`
   - `git log --oneline`
   - `cat main.py`

Expected Output:

- `git log --oneline`: Shows two commits.
- `main.py` content:

```
print("Git Test")
print("Updated")
```

Self-Study (S1-S2)

- **Read**: Refer to Session 2 of "Version Control with Git and GitHub" on OnlineVarsity.
- **Practice**: Complete "Practice 4 Me" exercises for Session 2.
- **Assignment**: Solve scenario-based assignments on OnlineVarsity (e.g., set up a repository for a collaborative project).
- **Explore**: Review Glossary and References on OnlineVarsity for terms like "staging" and "commit".

References

- "Version Control with Git" by Jon Loeliger, Matthew McCullough (Library Reference).
- OnlineVarsity: Learner’s Guide (eBook), Glossary, FAQ, and References for Session 2.