Git Class Notes

Introduction to Git

- **What is Git?**
    - A distributed version control system.
    - Keeps track of changes to files and coordinates work on those files among multiple people.
- **Why Use Git?**
    - Version control.
    - Collaboration.
    - Branching and merging.
    - Backup and restore.

Setting Up Git

- **Installation**
    - Windows: Download from git-scm.com.
    - macOS: Install via Homebrew (`brew install git`).
    - Linux: Use the package manager (`sudo apt-get install git`).
- **Configuration**
    - Set up your username: `git config --global user.name "Your Name"`
    - Set up your email: `git config --global user.email "your.email@example.com"`
    - Check configuration: `git config --list`

Basic Git Commands

- **Initializing a Repository**
    - `git init`: Create a new Git repository.
- **Cloning a Repository**
    - `git clone <repository_url>`: Clone an existing repository.
- **Checking Repository Status**
    - `git status`: Show the working tree status.
- **Adding Changes**
    - `git add <file>`: Add a specific file.
    - `git add .`: Add all changes.
- **Committing Changes**

    - `git commit -m "commit message"`: Commit changes with a message.

Working with Branches

- **Creating a Branch**
    - `git branch <branch_name>`: Create a new branch.
- **Switching Branches**
    - `git checkout <branch_name>`: Switch to a specific branch.
- **Merging Branches**
    - `git merge <branch_name>`: Merge a branch into the current branch.

Working with Remote Repositories

- **Adding a Remote Repository**
    - `git remote add origin <repository_url>`: Add a remote repository.
- **Fetching Changes**
    - `git fetch origin`: Fetch changes from the remote repository.
- **Pushing Changes**
    - `git push origin <branch_name>`: Push changes to a remote branch.
- **Pulling Changes**
    - `git pull origin <branch_name>`: Pull changes from a remote branch.

Resolving Conflicts

- **View Conflicts**
    - `git status`: See conflicted files.
- **Resolve Conflicts**
    - Edit the conflicted files manually.
    - Mark as resolved: `git add <resolved_file>`
    - Commit the resolution: `git commit -m "resolved conflict"`

Useful Git Commands

- **Viewing Commit History**
    - `git log`: Show commit logs.
- **Stashing Changes**
    - `git stash`: Stash changes temporarily.
    - `git stash apply`: Apply stashed changes.
- **Viewing Differences**
    - `git diff`: Show changes between commits, commit and working tree, etc.

Best Practices

- Commit often with meaningful messages.
- Use branches for new features and bug fixes.
- Regularly pull changes from the remote repository to stay up to date.
- Resolve conflicts promptly.