SOEN 6441
Advanced Programming Practice

Repositories and Continuous Integration

What is Git and GitHub ?

Git

Git is an **open source, distributed version control system** commonly used in software development to track changes in source code and collaborate on software projects.

Think of Git as a series of **snapshots (commits)** of your code.

GitHub

GitHub is a web-based platform and service that provides hosting for software development and **version control using the Git system**. It serves as a central hub for **code collaboration, project management, and code hosting**.

On GitHub you can **remotely publish** your Git repositories and **collaborate with other people**.

How to use Git and GitHub

How to create a new repository on GitHub

- **Navigate to Your Profile:** Click on your profile picture in the top right corner of the GitHub homepage to access your profile.
- **Access Your Repositories:** In your profile, click on the "Repositories" tab to view your existing repositories.
- **Create a New Repository:** Click on the green "New" button on the right side of the "Repositories" page to create a new repository.
- **Fill in Repository Details:** Repository Name, Description (Optional), Visibility, Initialize this repository with - You can choose to initialize your repository with a README file, and a license. (Optional)
- **Create Repository:** After filling in the details, click the green "Create repository" button at the bottom of the page.

How to use Git and GitHub

How to download Git

- **Visit the Git Website:** https://git-scm.com/.
- **Choose Your Operating System:** On the Git website's homepage, you'll find download links for various operating systems. Click on the link that corresponds to your operating system.
- **Run the Installer:** After downloading the installer, run it on your system.
- **Follow Installation Steps:** Follow the on-screen instructions to complete the Git installation.
- **Verify Installation:** After the installation, open a command prompt or terminal and type git --version to confirm the correct installation of Git, which should display the installed version.

How to use Git and GitHub

- **Link GitHub account to Git and set up SSH keys for secure authentication**
- Basic workflow of Git
- Collaborating with others by using GitHub

Link GitHub account to Git and set up SSH keys for secure authentication, follow these steps:

- Generate SSH Key Pair
- Save the SSH Key
- Add SSH Key to SSH Agent (Optional but recommended)
- Add SSH Key to Your GitHub Account
- Test Your SSH Connection
- Configure Git with Your GitHub Account

Generate SSH Key Pair

- SSH keys are used to securely authenticate your computer with your GitHub account. You can generate an SSH key pair using the following command:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Save the SSH Key

- When prompted, choose a location to save your SSH key pair (usually the default location, like ~/.ssh/id_ed25519), and optionally protect it with a passphrase.

Add SSH Key to SSH Agent (Optional but recommended)

- If you've protected your SSH key with a passphrase, you can add it to an SSH agent to avoid entering the passphrase every time you use the key. Run the following commands:

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

Add SSH Key to Your GitHub Account

- Copy the public key (~/.ssh/id_ed25519.pub) to your clipboard.
- Go to your GitHub account settings by clicking on your profile picture in the top right corner and selecting "Settings.”
- In the left sidebar, click on "SSH and GPG keys.”
- Click the "New SSH key" button.
- Paste your public key into the "Key" field, give it a title (e.g., "My SSH Key"), and click "Add SSH key.”

Test Your SSH Connection

- To test if your SSH key is properly set up, run the following command:

```
ssh -T git@github.com
```

- You should receive a message confirming your successful connection.

Configure Git with Your GitHub Account

- Set your Git username and email to match your GitHub account

```
git config --global user.name "Your GitHub Username"
git config --global user.email "your_email@example.com"
```

How to use Git and GitHub

- Link GitHub account to Git and set up SSH keys for secure authentication
- **Basic workflow of Git**
- Collaborating with others by using GitHub

Basic Workflow

- **Configure User Profile:** Begin by setting up your user profile in Git, which includes providing your name and email address. This step ensures that your contributions are properly attributed.
- **Clone the Repository:** Next, obtain a copy of the Git repository you want to work on by using the "git clone" command. This creates a local copy on your computer that you can edit.
- **Make Changes:** Work on the files within the local repository, making any necessary edits or additions to the code or content.
- **Commit Changes:** After making changes, use the "git commit" command to save your modifications locally, along with a descriptive commit message explaining the purpose of the changes.

Configure User Profile

- Before you start using Git, it's essential to configure your user profile. This information will be associated with your commits.
- Set your name and email address using the following commands:

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

Clone the Repository

- To work on an existing Git repository, you can clone it to create a local copy on your computer.

```
git clone https://github.com/username/repository.git
```

To initialize a new Git repository in a directory, follow these steps:

```
# Create a new directory for your project
   mkdir my_project
# Move into the project directory
   cd my_project
# Initialize a new Git repository
   git init
```

Make Changes

- navigate to the project directory and make the necessary changes to the files. Example: Edit a file (e.g., index.html) using a text editor or IDE.

Commit Changes

- Once you've made changes, you need to commit them to your local repository with a meaningful commit message. Example: Stage and commit your changes.

```
git add .
git commit -m "Updated index.html with new content"
```

Understanding Git Pull: Fetching and Merging Changes from a Remote Repository

- ‘git pull’ is a Git command used to fetch changes from a remote repository and merge them into the current branch. It essentially combines two other Git commands: git fetch and git merge.

Here's how git pull works:

- **Fetching Changes (Equivalent to git fetch):** When running git pull, Git contacts the default remote repository for your current branch, usually "origin", and retrieves new commits or changes. These changes are downloaded to your local machine but not automatically merged.
- **Merging Changes (Equivalent to git merge):** git pull automatically merges those changes into your current local branch. If there are any conflicts between your local changes and the changes from the remote, Git will prompt you to resolve those conflicts before merging.

Here's the basic syntax of git pull:

- git pull [<remote>] [<branch>]

<remote> specifies the remote repository to pull changes from (e.g., "origin" by default).

<branch> specifies the branch from the remote repository to pull changes from (e.g., "main" or "master" by default).

Understanding Git Push: Uploading Local Commits to a Remote Repository

- The git push command in Git is used to upload your local branch and its associated commits to a remote repository. It is an essential part of collaborating with others and keeping remote repositories up to date.

Here's a breakdown of how git push works:

- **Committing Changes Locally:** To push changes to GitHub, you must make and commit changes locally by modifying files in your working directory, staging them using ‘git add’, and committing them using ‘git commit’.
- **Using ‘git push’:** To share your local changes with others or update a remote repository, you use git push.

Understanding Git Push: Uploading Local Commits to a Remote Repository

- **Upload Process:** ‘git push’ compares your local branch with the corresponding remote branch, identifying and sending only the missing commits to the remote repository.
- **Updating the Remote Branch:** After a successful git push, your commits are integrated into the remote branch, making your changes accessible to other users who can access the remote repository.

Here's the basic syntax of ‘git push’:

- git push <remote> <branch>

<remote> specifies the name of the remote repository, which is usually named "origin" by default.

<branch> specifies the branch you want to push to the remote repository.

GitHub Actions

- GitHub Actions is a powerful and versatile automation and continuous integration/continuous deployment (CI/CD) platform provided by GitHub. It enables you to automate various tasks, workflows, and processes in your software development projects.
- Using GitHub Actions involves defining workflows in your GitHub repository's configuration files (YAML files) to automate various tasks.

Here are the steps to get started with GitHub Actions:

- **Access the Repository:** Log in to your GitHub account and access the repository where you want to set up GitHub Actions.
- **Define a Workflow:** Create a YAML file in the .github/workflows directory to define your workflow. You can name it whatever you like, but it must have a .yml or .yaml extension. In this YAML file, define the workflow including its name, triggers (events that initiate the workflow, e.g., push, pull_request, etc.), jobs, and steps.

Here's a simple example of a GitHub Actions workflow that runs when code is pushed to the "main" branch and performs a build and test:

```yaml
name: Build and Test

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v2

    - name: Set up Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'

    - name: Install dependencies
      run: npm install

    - name: Run tests
      run: npm test
```

Here are the steps to get started with GitHub Actions:

- **Commit and Push:** Commit the workflow configuration file to your repository. Push this commit to the repository on GitHub.
- **GitHub Actions Execution:** GitHub Actions will automatically detect the new workflow configuration file and start running the defined workflow when the specified trigger conditions are met (e.g., when a push event to the specified branch occurs).
- **Monitor and Review:** You can monitor the progress and results of your workflows in the "Actions" tab of your repository on GitHub. This tab provides details about each workflow run, including logs and status.

Stay here if you have any questions