Introduction to GitHub and GitLab

A Guide for Beginners

GitHub vs. GitLab

DigiTalk

https://digitalksystems.com/

Reach us at digitalk.fmw@gmail.com
DigiTalk Channel: https://www.youtube.com/channel/UCCGTnI9vvF_ETMhGUXGdFWw
Playlists: https://www.youtube.com/@digitalk.middleware/playlists
Weblogic Server Architecture: https://youtu.be/gNqeIfLjUqw

DigiTalk Udemy Courses and Coupon Code

SOA Suite Administration
https://www.udemy.com/course/mastering-oracle-soa-suite-12c-administration/?couponCode=739A60915F86847014EB
Coupon Code: 739A60915F86847014EB

JBoss 8 Administration
https://www.udemy.com/course/mastering-jboss-eap-8-administration-from-intro-to-advanced/?couponCode=BF65EB008CFE16686BD2
Coupon Code:BF65EB008CFE16686BD2

OHS Administration
https://www.udemy.com/course/mastering-oracle-ohs-http-12c-web-server-administration/?couponCode=8E990556B21AF3E1A316
Coupon Code: 8E990556B21AF3E1A316

Weblogic Server Administration
https://www.udemy.com/course/oracle-weblogic-server-12c-and-14c-administration/?couponCode=87BC1314AC7690FD5294
Coupon Code:87BC1314AC7690FD5294

You can write us on digitalk.fmw@gmail.com if coupon code expired.

Introduction to GitHub

GitHub is a web-based platform that uses Git, a version control system, to help developers collaborate on software projects. It provides a repository hosting service where developers can store their code, track changes, and collaborate with others. GitHub facilitates version control, code review, and project management, making it a popular choice for both open-source and private projects.

Important Components of GitHub

Repositories:

Description: A repository (or repo) is a storage space for your project. It contains all the project's files, including code, documentation, and configuration files, along with the revision history.
Example: For a project like a personal website, the repository would include HTML, CSS, JavaScript files, and any images.

Command: To clone a repository:

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

Branches:

Description: Branches allow you to work on different features or fixes simultaneously without affecting the main codebase. The main (or master) branch is the default branch where the stable code resides.

Example: If you're adding a new feature to a website, you might create a feature/new-design branch to work on the changes separately.

Command: To create and switch to a new branch:

```
git checkout -b feature/new-design
```

Commits:

Description: Commits are snapshots of your project at a particular point in time. Each commit has a unique ID and includes a message describing the changes made.

Example: A commit message like "Fixed bug in user login feature" documents the specific changes made to the code.

Command: To create a commit:

```
git commit -m "Fixed bug in user login feature"
```

Pull Requests (PRs):

Description: Pull requests are a way to propose changes from one branch to another. They are used to review code, discuss changes, and merge them into the main codebase.

Example: After completing a new feature, you create a pull request to merge feature/new-design into the main branch, allowing team members to review and approve the changes.

Command: Pull requests are typically created via the GitHub web interface, but you can use the command line to push branches:

```
git push origin feature/new-design
```

Issues:

Description: Issues are used to track tasks, bugs, and feature requests. They provide a way to discuss and manage work items related to the project.

Example: An issue might be created to report a bug in the website's contact form or to request a new feature like a search bar.

Command: Issues are created and managed through the GitHub web interface, but you can use the GitHub API to interact with them programmatically.

Forks:

Description: Forking a repository creates a personal copy of someone else's repository. This allows you to freely experiment with changes without affecting the original project.

Example: If you want to contribute to an open-source project, you fork the repository, make changes, and then submit a pull request to the original repository.

Command: Forking is done through the GitHub web interface.

GitHub Actions:

Description: GitHub Actions is a CI/CD tool that automates workflows such as testing, building, and deploying code. It uses workflows defined in YAML files to automate various tasks.

Example: You can set up an action to automatically run tests whenever code is pushed to a branch.

Command: Workflow files are placed in .github/workflows/ directory and defined using YAML syntax.

GitHub Pages:

Description: GitHub Pages allows you to host static websites directly from a repository. It's commonly used for project documentation, personal blogs, and portfolios.

Example: You can host a personal portfolio or project documentation site using GitHub Pages.

Command: To deploy a site, you typically push content to the gh-pages branch:

```
git push origin gh-pages
```

Important Commands

Clone a repository:

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

Create and switch to a new branch:

git checkout -b branch-name

Add changes to staging:

git add filename

Commit changes:

git commit -m "Commit message"

Push changes to the remote repository:

git push origin branch-name

Pull changes from the remote repository:

git pull origin branch-name

Merge a branch:

git merge branch-name

GitHub Example

Imagine you and your team are developing a new recipe app.

Repository: You create a GitHub repository named recipe-app to store all the project files, including code for the app, images of recipes, and documentation.

Branches: You start with the main branch for the stable version of the app. When working on a new feature like a search function, you create a branch called feature/search.

Commits: As you add and test the search functionality, you make commits with messages like "Added search bar" and "Fixed search results filtering."

Pull Requests (PRs): Once the search feature is ready, you open a pull request to merge feature/search into main. Your team reviews the code and, after approval, the feature is merged into the main project.

Issues: During development, you create issues to track bugs and feature requests, such as "Fix crash on recipe image load" or "Add user login feature."

GitHub Actions: You set up GitHub Actions to automatically run tests and build the app whenever you push new code, ensuring everything works correctly before merging.

GitHub Pages: For documentation, you use GitHub Pages to host a website with user guides and developer notes about how to use and contribute to the recipe app.

This setup helps you manage the development process, collaborate with your team, and maintain a high-quality application.

Introduction to GitLab

GitLab is a web-based DevOps platform that provides source code management (SCM) with Git, CI/CD pipelines, and various project management features. It integrates all stages of the software development lifecycle, from planning and coding to building, testing, and deploying, into a single application.

Important Components of GitLab

Projects:

Description: A project in GitLab is a repository that stores your code, configuration files, and other assets. Each project includes features for managing code, issues, and CI/CD pipelines.

Example: A project could be a web application, a microservice, or a library, including files like index.html, style.css, and main.js.

Command: To clone a project:

git clone https://gitlab.com/username/project.git

Branches:

Description: Branches allow multiple developers to work on different features or fixes simultaneously. The main (or master) branch typically represents the stable version of the code.

Example: You might create a feature/new-ui branch to develop a new user interface without affecting the stable code in main.

Command: To create and switch to a new branch:

git checkout -b feature/new-ui

Commits:

Description: Commits are snapshots of your code changes. Each commit is accompanied by a unique ID and a message describing the changes made.

Example: A commit message like "Added responsive design for mobile view" records the changes made to the codebase.

Command: To create a commit:

git commit -m "Added responsive design for mobile view"

Merge Requests (MRs):

Description: Merge Requests are used to propose changes from one branch to another. They facilitate code review, discussion, and merging of code changes.

Example: After developing a feature on a branch, you open a merge request to merge feature/new-ui into main, allowing team members to review and approve the changes.

Command: Merge requests are typically created through the GitLab web interface, but you can push branches with:

git push origin feature/new-ui

Issues:

Description: Issues are used to track bugs, tasks, and feature requests. They allow teams to manage and discuss work items related to the project.

Example: You might create an issue to report a bug in the login system or request a new feature like multi-language support.

Command: Issues are managed through the GitLab web interface, but they can be interacted with using GitLab's API.

CI/CD Pipelines:

Description: GitLab CI/CD pipelines automate the processes of building, testing, and deploying code. Pipelines are defined using .gitlab-ci.yml files and consist of various stages and jobs.

Example: You can set up a pipeline to automatically run tests and deploy your application whenever you push code to the repository.

Command: Pipelines are defined in the .gitlab-ci.yml file, and jobs are triggered automatically based on this configuration.

GitLab Pages:

Description: GitLab Pages allows you to host static websites directly from a GitLab repository. It's useful for project documentation, personal blogs, and other static content.

Example: You can host a portfolio site or project documentation using GitLab Pages.

Command: To deploy a site, you typically push content to the main branch or a specific branch designated for Pages.

Runners:

Description: Runners are agents that execute jobs defined in CI/CD pipelines. They can be shared or specific to a project and run on various environments like Docker, Virtual Machines, or physical servers.

Example: You might use a GitLab Runner to build your application in a Docker container and run unit tests.

Command: Runners are configured through the GitLab web interface, but you can register a runner with:

gitlab-runner register

Important Commands

Clone a project:

git clone https://gitlab.com/username/project.git

Create and switch to a new branch:

git checkout -b branch-name

Add changes to staging:

git add filename

Create a commit:

git commit -m "Commit message"

Push changes to the remote repository:

git push origin branch-name

Pull changes from the remote repository:

git pull origin branch-name

Merge a branch:

git merge branch-name

Register a GitLab Runner:

gitlab-runner register

GitLab Example

Project: You create a GitLab project named recipe-app to manage your app's code, images, and documentation.

Branches: You work on the main branch for stable releases and create a feature/search branch to develop a new search feature.

Commits: As you work on the search feature, you make commits with messages like "Added search bar" and "Fixed search results filtering."

Merge Requests (MRs): When the search feature is complete, you open a merge request to merge feature/search into main. Your team reviews and approves the changes before merging.

Issues: You create issues to track tasks and bugs, such as "Fix crash on recipe image load" or "Implement user login."

CI/CD Pipelines: You configure GitLab CI/CD pipelines to automatically run tests and build your app whenever new code is pushed, ensuring everything works properly.

GitLab Pages: You use GitLab Pages to host a website with documentation for your recipe app, providing user guides and developer notes.

This example illustrates how GitLab's features support the development process and team collaboration similarly to GitHub.

GitHub vs. GitLab

1. Core Focus

GitHub: Primarily focuses on code hosting and collaboration. It has a strong emphasis on social coding, with features that support open-source projects and community engagement.

GitLab: Offers a broader DevOps lifecycle management solution, integrating not just version control but also continuous integration/continuous deployment (CI/CD), project management, and monitoring.

2. Repository Management

GitHub: Provides a straightforward repository management system. It's widely used for open-source projects and integrates well with various third-party tools.

GitLab: Includes advanced repository management features, such as built-in CI/CD and issue tracking, with a focus on providing an all-in-one DevOps solution.

3. CI/CD Integration

GitHub: Uses GitHub Actions for CI/CD. GitHub Actions is a flexible automation tool for building, testing, and deploying code. It allows you to create custom workflows.

GitLab: Has built-in CI/CD pipelines integrated directly into the platform. GitLab CI/CD is highly customizable and comes with pre-defined templates for various stages of the software development lifecycle.

4. Issue Tracking and Project Management

GitHub: Includes basic issue tracking and project boards, but more advanced project management features require integration with other tools or services.

GitLab: Offers comprehensive issue tracking, project management, and milestone features built into the platform. It supports Agile methodologies and provides features like burndown charts and roadmaps.

5. Code Review

GitHub: Uses Pull Requests (PRs) for code review. PRs can be reviewed, commented on, and approved before merging into the main branch.

GitLab: Uses Merge Requests (MRs) for code review, similar to PRs. MRs include integrated code review tools and can be linked with CI/CD pipelines for automated checks.

6. User Interface and Experience

GitHub: Known for its user-friendly and clean interface, which is highly intuitive for managing repositories and collaborating on code.

GitLab: Provides a more integrated interface that combines code management, CI/CD, and project management tools into a single application. It can be more complex due to its broader feature set.

7. Pricing and Plans

GitHub: Offers free and paid plans with various levels of access. The free plan includes unlimited public repositories and limited private repositories.

GitLab: Provides a free tier with comprehensive features, including unlimited private repositories and CI/CD. Paid plans add more advanced features and support.

8. Hosting Options

GitHub: Primarily offers cloud-hosted services through GitHub.com. GitHub Enterprise provides on-premises hosting options.

GitLab: Offers both cloud-hosted (GitLab.com) and self-hosted (GitLab Self-Managed) options, allowing for more control over the infrastructure and deployment.

9. Community and Ecosystem

GitHub: Has a large, active community, especially strong in open-source projects. Its ecosystem includes a vast number of integrations and third-party tools.

GitLab: Also has a growing community, with a focus on DevOps and integrated tools. It's known for its comprehensive feature set and all-in-one approach.

Summary

- **GitHub** excels in code collaboration, community engagement, and integration with third-party tools. It's ideal for open-source projects and social coding.
- **GitLab** provides a full DevOps platform with integrated CI/CD, project management, and monitoring. It's suited for teams looking for an all-in-one solution for the entire software development lifecycle.
- Both platforms are powerful, and the choice between them depends on your specific needs, whether it's code collaboration, DevOps features, or integration with other tools.

DISCLAIMER AND CONSENT

This document is being provided by DigiTalk as part of its effort to assist users in understanding and working with GitHub and GitLab. While every effort has been made to ensure the accuracy and reliability of the information presented in this document, there is a possibility of typographical errors or inaccuracies. DigiTalk does not guarantee the correctness or completeness of the content provided in this document.

Users of this document are encouraged to cross-reference the information presented here with official documentation available on their website or other authoritative sources. Any discrepancies or inaccuracies found in this document should be reported to us at digitalk.fmw@gmail.com.

By using this document, you acknowledge and consent to the following:

This document is not officially endorsed or verified by any other third party organization..

The Company makes no claims or guarantees about the accuracy or suitability of the information contained in this document.

Users are responsible for verifying and validating any information presented here for their specific use case.

DigiTalk disclaims any liability for any errors, omissions, or damages that may result from the use of this document.

If you discover any inaccuracies or errors in this document, please report them to digitalk.fmw@gmail.com, and the Company will endeavor to correct them as necessary.

This consent statement is provided to ensure transparency and understanding of the limitations of the information contained in this document. By using this document, you agree to abide by the terms and conditions outlined herein.