Capgemini

Git and Github-
MASTER CLASS: Participant Guide

Capgemini

Git and Github - Master Class

Document Revision History

Document Revision History Date	Revision No.	Author	Summary of Changes
31-05-2023	1.0	Alka Jhanwar	Version created for SME/BU Approval.

Capgemini

Table of Contents

Introduction:

By end of this workshop, the Participants will gain a clear understanding of version control concepts and the importance of using Git and GitHub for managing code changes and collaboration.

After completing this workshop, the participants will be able to:

1. Install and configure Git and GitHub.
2. Use basic and advanced Git commands.
3. Adding and committing files in git then pushing it to GitHub.
4. Creating, merging, and deleting branches.
5. Pushing and pulling changes from remote repositories.
6. Use GitHub as a collaborative platform for hosting Git repositories, creating branches, pushing changes, and creating and merging pull requests.
7. Resolving merge conflicts and creating issues.
8. How to automate build, test, and deployment processes for their projects using Github actions.

Implementing Version Control using Git and Github

Task 1: Setting up Git and GitHub

1. Install Git on your local machine if it's not already installed.
2. Create a GitHub account if you don't have one.
3. Set up Git credentials on your machine, including your name and email address.

Task 2: Initializing a Repository and Pushing to GitHub

1. Initialize a new Git repository on your local machine.
2. Add a new file to the repository and make some changes.
3. Stage and commit the changes.
4. Create a new repository on GitHub.
5. Link your local repository to the remote repository on GitHub.
6. Push your local repository to GitHub.

Task 3: Branching, Merging, and Pull Requests

1. Create a new branch in your local repository.
2. Make changes to the files on the new branch.
3. Stage and commit the changes.
4. Push the new branch to GitHub.
5. Create a pull request on GitHub to merge the new branch into the main branch.
6. Assign reviewers to the pull request.
7. Reviewers review the code changes and provide feedback.

8. Make necessary modifications based on the feedback.
9. Complete the pull request and merge the branch into the main branch.

Task 4: Collaborating with Team Members

1. Invite a team member as a collaborator to the GitHub repository.
2. Have the team member clone the repository to their local machine.
3. Create a new branch and make changes on the team member's local repository.
4. Stage, commit, and push the changes to GitHub.
5. Create a pull request from the team member's branch to the main branch.
6. Review, provide feedback, and merge the pull request.

Task 5: Resolving Conflicts and Managing Branches

1. Create a branch from the main branch in your local repository.
2. Make changes to the same file on both branches (main and new branch).
3. Stage, commit, and push the changes from both branches.
4. Encounter a merge conflict when merging one branch into the other.
5. Resolve the conflict by manually editing the conflicted file.
6. Commit the resolved changes and complete the merge process.
7. Delete the merged branch.

Task 6: Issue Tracking

1. Create an issue related to a bug or feature.
2. Link the issue to a branch or pull request.
3. Resolve the issue by implementing the necessary changes.
4. Close the issue once it's resolved.
5. Delete merged branches to maintain a clean repository.

Task 7- Set up a CI/CD workflow using GitHub Actions to automate the build, test, and deployment process for a web application.

1. Set up and configure GitHub Actions workflow
2. Define the steps for the CI workflow:
    - Checkout the repository code.
    - Install
    - Build the application.
    - Run tests to ensure code quality and functionality.
4. Extend the existing CI workflow to include a deployment step.
5. Define deployment actions based on your deployment environment (e.g., cloud hosting, server).
    - For example, if deploying to a cloud hosting platform, configure actions to deploy the application to the platform.

6.Configure the deployment workflow to trigger on successful builds.

7.Observe the CI workflow being triggered automatically.
8. Monitor the workflow execution to ensure it builds and tests the application successfully.
9. Once the CI workflow is successful, observe the deployment workflow being triggered automatically.
10. Monitor the deployment workflow execution to ensure the application is successfully deployed to the target environmet