1. **GitHub-** It is an online software development platform, it is usually used for storing, tracking and collaborating on software projects.

**Functions**

* It usually allows collaboration with developers from all over the world.
* It allows the user to create, store, change, merge the projects.
* It helps to create excellent documents.

**Features**

* Access controls and collaboration features like team management.
* It is easy to contribute to open source projects.
* There is easy project management.
* Easy code hosting.
* Increased code safety.

It allows the users to pull all the latest changes made by other collaborators from the remote repository.

1. **GitHub repository-** A repository is an element in the GitHub that is used to store the users code, files and each files revision history.

**How to create a new Git Repository.**

* Create a new folder for your project.
* Open the folder in GITBASH.
* Issue the git init command to create the new Git repo.
* Note the creation of the hidden. Git folder in the project.
* Add files and folders to your project.
* Routinely stage files and create commits.

**ESSENTIAL ELEMENTS TO BE INCLUDED WHEN CREATIONG NEW REPOSITORY**

* Repository
* Index
* Working tree

1. **Version Control-** It usually allows developers to track changes in their codebase overtime.

It helps in manage collaboration, maintain history and revert to previous states.

**HOW GIT ENHANCES DEVELOPERS**

* Developers create branches to work on specific features independently.
* Git allows merging, collaboration and branching.
* Git usually stores code history in a repository.

**BRANCHING**

**Creating a branch**

* To create a new branch and switch to it simultaneously

**Working on a branch**

* Developers work on their features or bug fixes within their branches.
* Commits on a branch move the branch pointer forward**.**

**MERGING**

**Merging changes**

* When a feature is complete, it is merged back into the main branch.
* Use git merge to combine all the changes from one branch to another**.**

1. **Branches-** Is a new version of the main repository that allows the user to develop features, fix bugs or experiment with new ideas in a contained area of your repository

**IMPORTANCE OF BRANCHES**

It is used to isolate development work without affecting other branches in the repository.

**PROCESS OF CREATING A BRANCH**

* Start by creating a new branch, you can use the command.
* Replace my feature with a descriptive name for your branch**.**

**MAKING CHANGES**

* Work on your feature within the newly created branch.
* Make code changes, add files, add files and commit your work**.**

**PUSHING CHANGES TO REMOTE**

* Push your branch to the remote repository e.g. GitHub.

**CREATING A PULL REQUEST**

* On GitHub, navigate to your repository and switch to the newly pushed branch.
* Click on “New pull request”
* Select the base branch and compare it with your feature branch.
* Write clear description of your changes and create the pu**ll** request.

**CODE REVIEWS**

* Team members review your code in the pull request.
* Address feedback, make necessary changes and continue the discussion.
* Once approved your changes are ready for merging**.**

**MERGING BACK TO MAIN BRANCH**

* Merge your changes into the main branch:
* Locally
* Remotely via the pull request.
* Click merge on the pull request.
* Confirm the merge.

1. **Pull request-** It is a proposal to merge a set of changes from one branch into another.

In pull request collaborators can review and discuss the proposed set of changes before they integrate the changes into the main codebase. Pull requests display the differences, or differences, between the content in the source branch and the content in the target branch.

**STEPS TO CREATE PULL REQUEST**

**CREATING A PULL REQUEST**

* Start by pushing your code changes to a branch in your repository.
* Navigate to your repository
* Click on the “pull request” button.
* Select the base branch and the branch containing your changes.
* Write descriptive title and comment explaining your changes.
* Click “Create pull request”

**REVIEWING A PULL REQUEST**

* Open the pull request you want to review.
* Click on the “files changed” tab to see the code effectiveness.
* Leave comments directly on a specific line of code.

**COMMENT**

Comment feedback without explicit approval or change requests.

**APPROVAL**

Feedback with approval for merging the changes.

**REQUEST CHANGES**

Feedback that must be addressed before merging.

Once viewed, click “Approve” or “Request changes”

1. **GITHUB ACTIONS-** is a continuous integration and continuous delivery(CI/CD) platform that allows you to automate your build, test and deployment pipeline. Workflows can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.
2. **VISUAL STUDIO-** It is an integrated development environment (IDE) it is usually used to develop computer programs including web apps, web services.

**FEATURES OF VS**

* It writes high-quality code with comprehensive testing tools.
* It has compilers.
* It has code completion tools.
* It provides full support for NET and its unified framework.

It differs from visual studio code since visual studio is a comprehensive intergrated development environment (IDE) tool for software development, while the visual studio code is an Extension-based code Editor.

1. **STEPS TO INTERGRATE A GITHUB REPOSITORY.**

* **OPEN YOUR PROJECT IN VISUAL STUDIO.**

Launch visual studio and open the project you want to add to GitHub.

Click “Add to source control” and select Git.

* **PROVIDE GIT AND REPOSITORY DETAILS.**

You will need to provide your GitHub account details.

Authenticate with your GitHub credentials.

Specify the repository where you want to push your code.

* **START COMMITING AND PUSHING.**

Browse your GitHub repositories within visual code.

Clone your repo to your local machine.

Make changes, commit them and push to GitHub.

**The integration enhances development workflow in several ways;**

* **Seamless integration**

Authenticate your GitHub account directly within visual control.

* **CLONING AND COMMITTING.**
* Browse GitHub repositories from within visual studio.
* Clone repos to your local machine and start committing changes.
* **BRANCHING AND MERGING**
* Create and switch branches directly from the status bar.
* View changes, stage files and make commits.
* Merge or rebase branches within visual studio.

1. **DEBUGGING TOOLS.**

* Node.js Debuggers.
* Edge and chrome Debuggers.
* C/C++ Debugging
* Launch configuration.

**DEBUGGING TOOLS CAN BE USED TO DO THE FOLLOWING;**

Debugging means to run a code step by step in a debugging tool.

* **CALL STACK:** View the call stack to understand how functions are nested and which function caused an exception.
* **INSPECT VARIABLES:** Hover over variables to see their current values. You can also set watches to track changes in variables.
* **BREAKPOINTS:** Set breakpoints to pause execution at specific lines of code. You can examine variable values, memory behavior and control flow.

1. **GITHUB AND VISUAL STUDIO WORK IN THE FOLLOWING WAYS;**

* **SEAMLESS INTERGRATION.**

Authenticate your GitHub account directly within visual studio.

Create repos and push commits.

* **CLONING AND COMMITTING.**

Browsing GitHub repositories from within Visual Studio.

Clone repos to your local machine and start committing changes.

* **BRANCHING AND MERGING**

Create and switch branches directly from the status bar.

View changes, stage files and make commits with ease.

* **PULL REQUESTS AND CONFLICT RESOLUTION.**

Create pull request from remote branches.

Resolve merge conflicts using the built-in merge editor.

* **INTERGRATED CI/CD WORKFLOWS:**

Set up GitHub Actions.

**REAL-WORLD EXAMPLE**

Imagine a startup building a web application. Their team uses Visual Studio for development and GitHub for version control. This is how they may benefit:

**COLLABORATION:** Developers cone the repo, create branches and collaborative on feartures.

**CODE REVIEW:** Pull request allow thorough code review before merging.

**CI/CD:** GitHub actions automate testing and deployment.