GitHub is a platform that allows developers and programmers to store and manage their code repositories using Git. Git is a version control system (VCS) that tracks changes made to files and enables multiple people to collaborate on projects without interfering with each other's work. GitHub builds on Git by providing a user-friendly interface and additional features for collaboration.

Here’s a breakdown of what GitHub is and how it works:

### Key Concepts:

1. \*\*Repositories (Repos)\*\*: A repository is like a project folder where all your files (code, documentation, etc.) are stored. You can have multiple repositories on GitHub for different projects.

2. \*\*Branches\*\*: Git allows developers to create different "branches" of a project. The default branch is usually called `main` or `master`. Branches are useful for experimenting with new features without affecting the main codebase.

3. \*\*Commits\*\*: A commit is a snapshot of your code at a certain point in time. When you make changes to a file and save them, you create a commit. Each commit has a unique identifier (SHA), and a commit message describing the changes made.

4. \*\*Pull Requests (PRs)\*\*: When you want to merge changes from one branch into another (typically from a feature branch into the main branch), you create a pull request. It allows others to review and discuss the changes before they are merged into the main codebase.

5. \*\*Forking\*\*: Forking a repository means creating a copy of someone else's repository in your own GitHub account. You can make changes to this forked version without affecting the original project. If you want, you can later propose these changes by opening a pull request to the original repository.

6. \*\*Cloning\*\*: Cloning is creating a local copy of a GitHub repository on your computer. You can then make changes locally and push them back to the GitHub repository.

7. \*\*Issues\*\*: GitHub also has an "Issues" section where you can report bugs, request features, or track tasks related to the project.

8. \*\*GitHub Actions\*\*: This is a feature for automating workflows, such as continuous integration (CI), deployment, and other repetitive tasks.

### Common GitHub Commands:

- \*\*git clone <repository-url>\*\*: Clone a repository to your local machine.

- \*\*git status\*\*: Check the status of your working directory and staging area.

- \*\*git add <file>\*\*: Stage a file for commit.

- \*\*git commit -m "message"\*\*: Commit staged changes with a message.

- \*\*git push\*\*: Push your commits to a remote repository (e.g., on GitHub).

- \*\*git pull\*\*: Fetch the latest changes from a remote repository and merge them into your local repository.

### Why Use GitHub?

- \*\*Collaboration\*\*: GitHub allows multiple developers to work on the same project simultaneously. By using features like branches and pull requests, you can manage contributions efficiently.

- \*\*Version Control\*\*: GitHub helps track and manage changes over time, making it easy to revert to previous versions or identify when and why changes were made.

- \*\*Open Source\*\*: Many open-source projects are hosted on GitHub, allowing anyone to contribute. You can fork projects, suggest improvements, or contribute directly.

- \*\*Documentation\*\*: You can store README files and wikis on GitHub, making it easy to document your project.

- \*\*Integration\*\*: GitHub integrates with many third-party tools and services, such as CI/CD tools, project management tools, and code analysis tools.

In short, GitHub is an essential platform for modern software development, offering version control, collaboration features, and a social aspect to share and contribute to code.

GitHub and IIS serve very different purposes, so it’s not quite accurate to say that one is "more useful" than the other for deployment. Rather, they are often used in conjunction with each other in a development and deployment workflow. Let me explain why:

### \*\*GitHub: Code Management and Collaboration\*\*

- \*\*Primary Role: \*\* GitHub is a platform for \*\*source code management\*\* (via Git), \*\*version control\*\*, and \*\*collaboration\*\*.

- \*\*Deployment Use: \*\* GitHub itself is \*\*not directly a deployment tool\*\* (except for hosting static websites via \*\*GitHub Pages\*\*). However, GitHub plays an important role in modern deployment workflows by storing code and providing an interface for managing versions, issues, and collaboration. GitHub can integrate with CI/CD (Continuous Integration/Continuous Deployment) tools, which can trigger deployments to various environments.

#### \*\*How GitHub Helps in Deployment: \*\*

1. \*\*Code Repository\*\*: Developers store the application code in GitHub repositories, ensuring that it is versioned and easily accessible.

2. \*\*Continuous Integration/Continuous Deployment (CI/CD) \*\*: GitHub integrates with tools like \*\*GitHub Actions\*\*, \*\*Jenkins\*\*, \*\*Travis CI\*\*, and \*\*Circles\*\* to automate the deployment process. For example, when code is pushed to a repository or a pull request is merged, these tools can automatically build, test, and deploy the code to different environments, such as production, staging, or testing environments.

3. \*\*Collaboration\*\*: Multiple developers can collaborate on the code, review pull requests, and discuss changes, making the process of managing and deploying code more structured and efficient.

### \*\*IIS (Internet Information Services): Hosting and Deployment\*\*

- \*\*Primary Role:\*\* IIS is a \*\*web server\*\* and \*\*application hosting platform\*\*, specifically for Windows-based systems.

- \*\*Deployment Use:\*\* IIS is \*\*directly involved in deploying and hosting\*\* web applications, especially \*\*ASP.NET\*\* and \*\*PHP\*\* applications. It handles HTTP/HTTPS requests from users and serves web content to browsers.

#### \*\*How IIS Helps in Deployment:\*\*

1. \*\*Web Hosting\*\*: IIS is used to \*\*host websites and web applications\*\*. Once your code is ready, you deploy it to an IIS server where it is served to users over the web.

2. \*\*App Pooling\*\*: IIS allows you to manage web applications using \*\*application pools\*\*, isolating and running them independently, improving performance and security.

3. \*\*Security and Authentication\*\*: IIS provides various features like \*\*SSL\*\*, \*\*authentication\*\*, and \*\*access control\*\* to ensure secure communication between clients and your server.

### \*\*GitHub vs IIS in Deployment: \*\*

- \*\*GitHub\*\* is more focused on \*\*code management\*\* and \*\*automation\*\*, while \*\*IIS\*\* is focused on \*\*serving web applications\*\* to users. They can both be part of a deployment pipeline but serve distinct roles.

- GitHub does not directly handle deployment; it facilitates the process through integration with CI/CD tools. \*\*IIS\*\*, on the other hand, directly hosts and runs your application in a live environment once it's deployed.

### \*\*How They Work Together: \*\*

A common deployment flow might look like this:

1. \*\*Development on GitHub\*\*:

- Developers push code to a \*\*GitHub repository\*\*.

- Code changes are tracked, and collaboration takes place on GitHub.

2. \*\*CI/CD Pipeline\*\*:

- After a change is committed to GitHub (like a new feature or a bug fix), a CI/CD tool is triggered.

- \*\*GitHub Actions\*\*, for example, could run tests and build the application automatically. It could then deploy the build to a \*\*server\*\* (e.g., a Windows server running IIS).

3. \*\*Deployment to IIS\*\*:

- Once the build is ready, the \*\*deployment process\*\* pushes the code to a \*\*server running IIS\*\*, where the web application is hosted and made publicly available.

- IIS then \*\*serves\*\* the application to users through HTTP requests.

### \*\*So, Which One is More Useful for Deployment?\*\*

- \*\*GitHub\*\* is indispensable for \*\*code management\*\*, collaboration, and automating deployment pipelines. However, it \*\*does not serve as a hosting solution\*\* for your applications. It is part of the process that helps in deploying code to environments.

- \*\*IIS\*\* is directly involved in \*\*hosting and running\*\* web applications and websites. If you are working in a \*\*Windows environment\*\* and need to serve dynamic web content (especially ASP.NET), IIS is a key part of your \*\*deployment infrastructure\*\*.

### \*\*Conclusion:\*\*

- \*\*GitHub is not a direct deployment tool\*\* — it is part of the development workflow, helping with code management, version control, and automation (via CI/CD).

- \*\*IIS, on the other hand, is essential for web hosting and deployment\*\* — it is where the application is actually \*\*deployed and served to users\*\*.

They work \*\*together\*\* to ensure that code is properly developed, tested, and deployed to a hosting environment. So, \*\*GitHub\*\* is more useful for \*\*managing and automating the deployment process\*\*, while \*\*IIS\*\* is more useful for \*\*hosting\*\* the actual web application once it has been deployed.

1)Create or clone Repository

2)Create Branch

3)commit/pull/push

------------------------------------------------------------------------------------------------------------------

1)Create a Repository:

A screenshot of a computer

AI-generated content may be incorrect.

2)clone the Repository to local system.

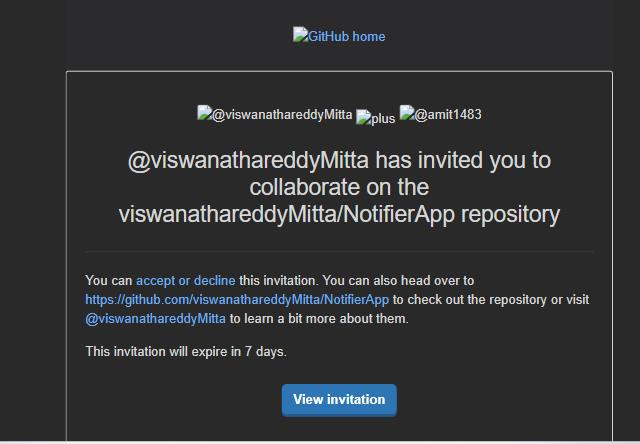
A screenshot of a computer

AI-generated content may be incorrect.

3)Shared the Repo to the team.

A screenshot of a computer

AI-generated content may be incorrect.



Created a Repo from local visual studio and pushed to the GitHub

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Placed the locally created code into the GitHub Repository.

A screenshot of a computer

AI-generated content may be incorrect.

Below we can see the count of new changes are 79 which are added to the local.

Branch is master branch and there is no pull and push changes.

A screenshot of a computer

AI-generated content may be incorrect.

Below we can the new changes in changes (local repository)

A screenshot of a computer

AI-generated content may be incorrect.

Below shows changes (local repository) which are moved to stagging (staging repository)

These changes we need to commit and push to the master branch.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

When we see here the is a change 1 which we need to Push.

A screenshot of a computer

AI-generated content may be incorrect.

After pushing change from local to GitHub repository we can see below.

A screenshot of a computer

AI-generated content may be incorrect.

Using below tabs we can create new branch dev. From the source branch aster to dev.

It means all the code will move from master to dev.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Below images shows how many ways we can take the code from GitHub repository.

A screenshot of a computer

AI-generated content may be incorrect.

Below image shows the we have created the branch in Remote repo then we get that branch in local.

A screenshot of a computer

AI-generated content may be incorrect.

Inside the newly created branch we done some changes in local repository like 3 .

A screenshot of a computer

AI-generated content may be incorrect.

Below pic shows those 3 changes moved to staging repository in local and added the commit description before commit the changes. Once commit is done, we can push our changes to local repository.

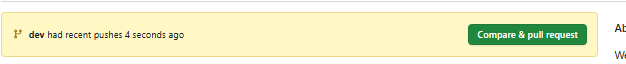
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

When we check in GitHub it indicates new changes in dev branch and we need to create pull request to move those changes to dev to master branch.



A screenshot of a computer program

AI-generated content may be incorrect.

Below image shows how we can merge our local dev branch code into the master using GitHub.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

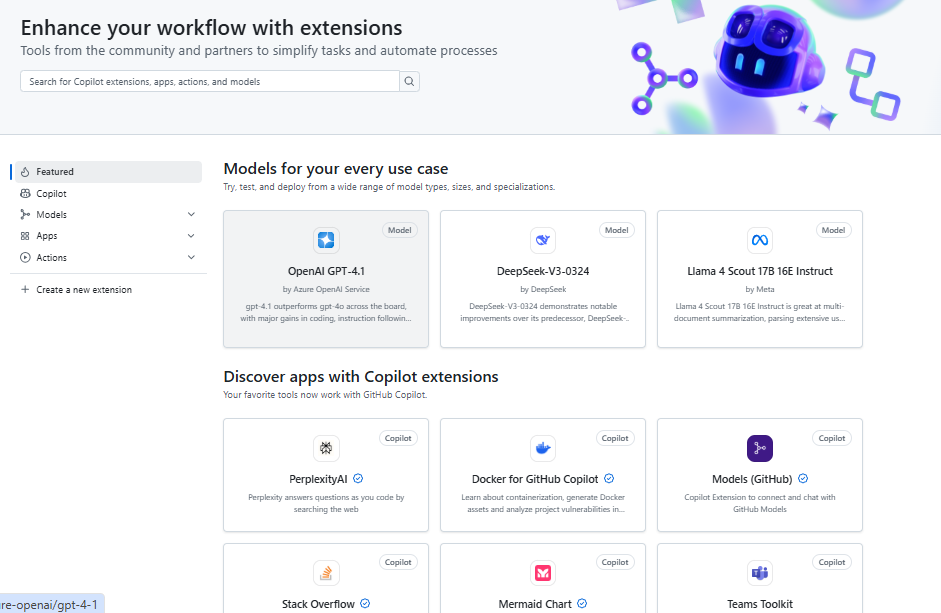
AI-generated content may be incorrect.

Once changes are merged from dev branch to master branch. All code will be available in master branch.

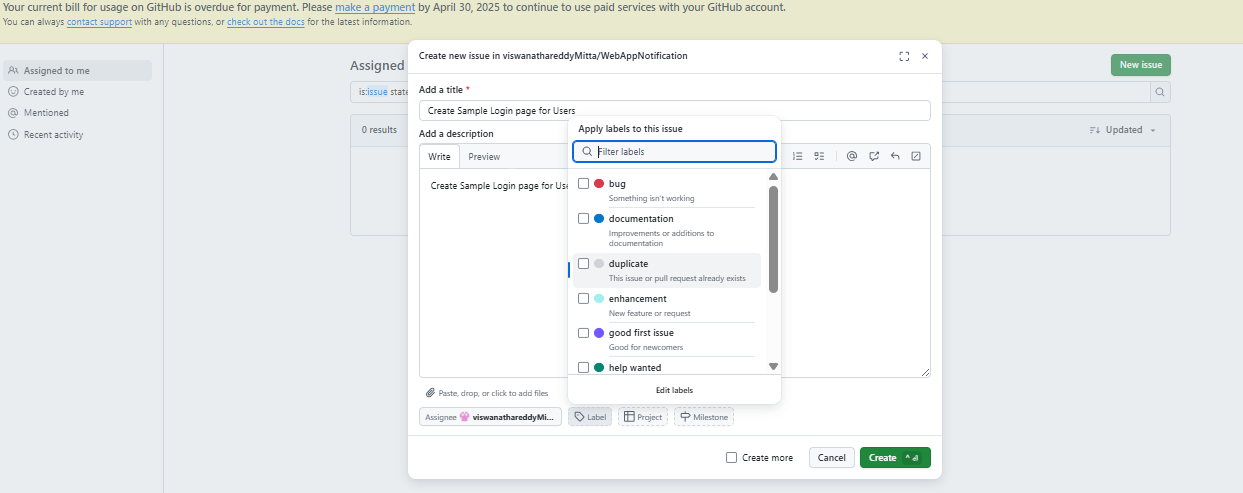
A screenshot of a computer

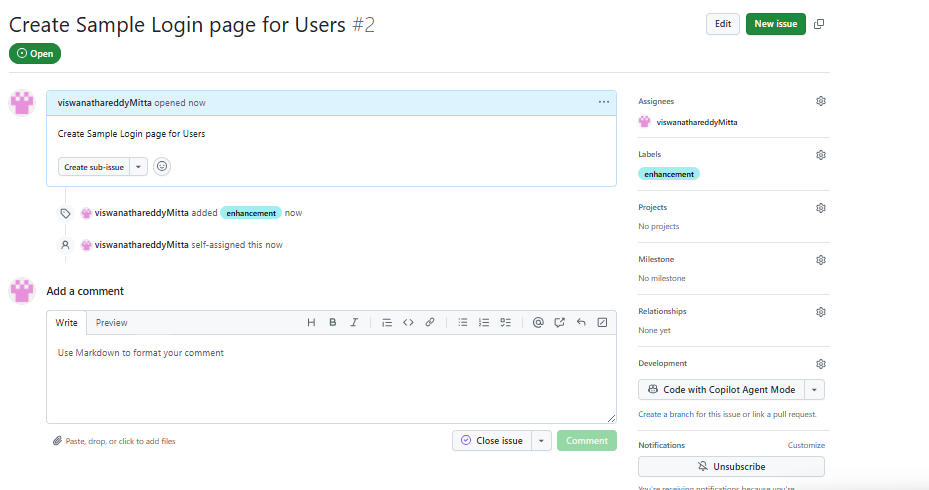
AI-generated content may be incorrect.

Below are the some of the additional features for GitHub which can add as a extensions.

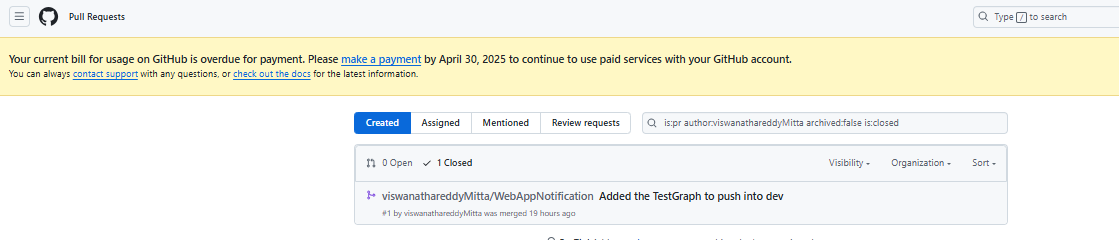


7. \*\*Issues\*\*: GitHub also has an "Issues" section where you can report bugs, request features, or track tasks related to the project.

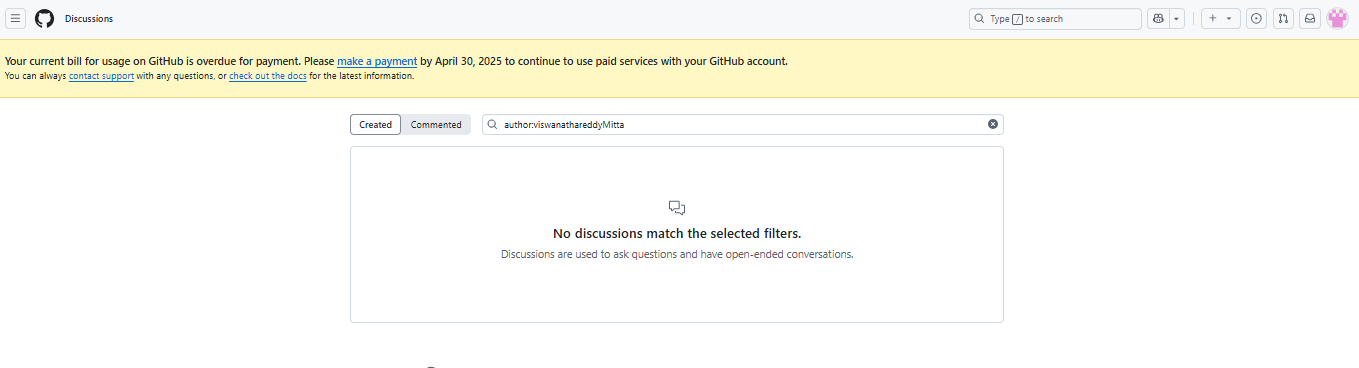




Pull Request:

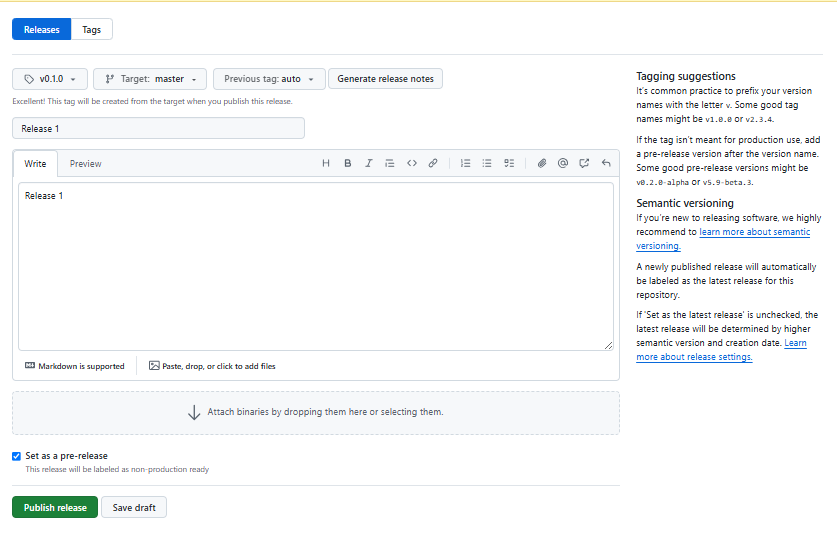


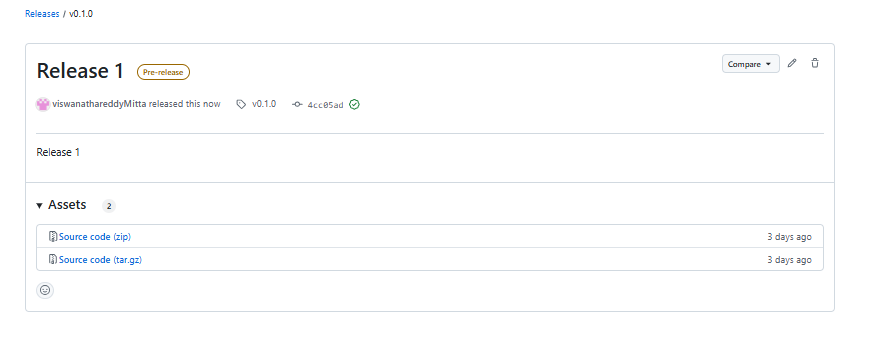
Discussions tab:



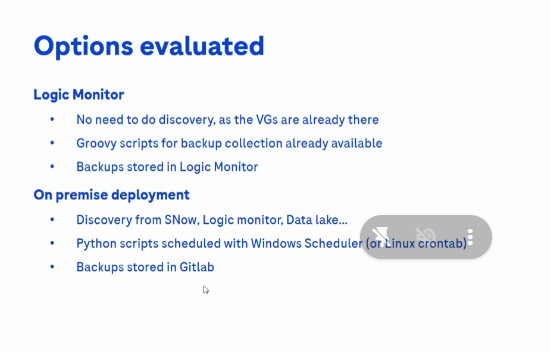
Semantic versioning (SemVer):

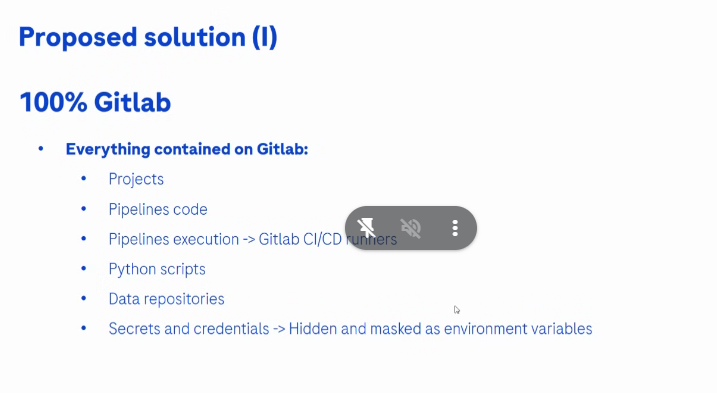
is a widely adopted versioning scheme that uses a three-part number (Major.Minor.Patch) to indicate the compatibility and stability of software releases. It helps developers and users understand how a software update might affect their application or project.



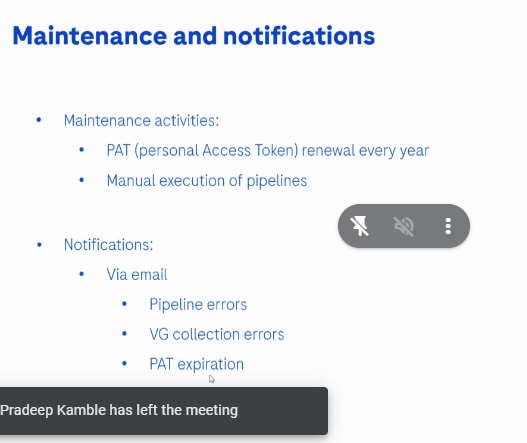


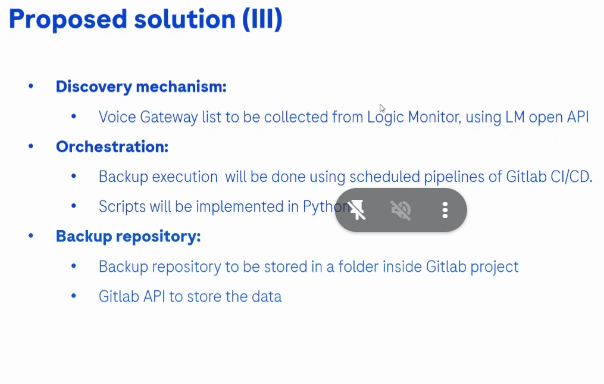
<https://www.bairesdev.com/blog/git-github-and-gitlab-whats-the-difference/>

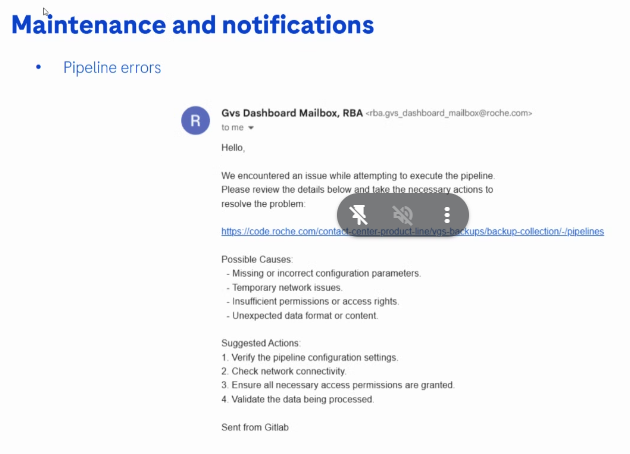


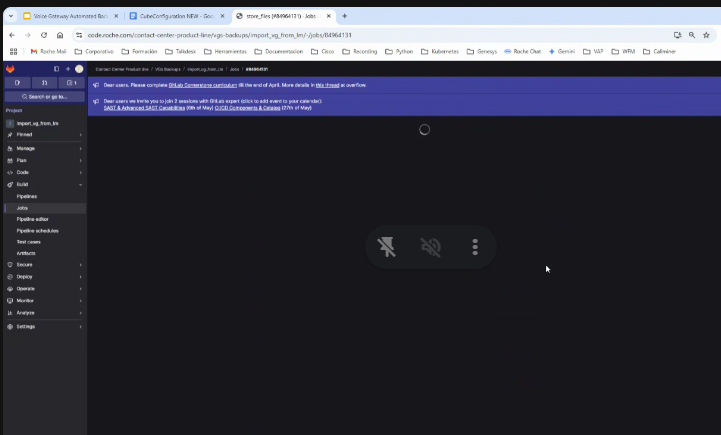


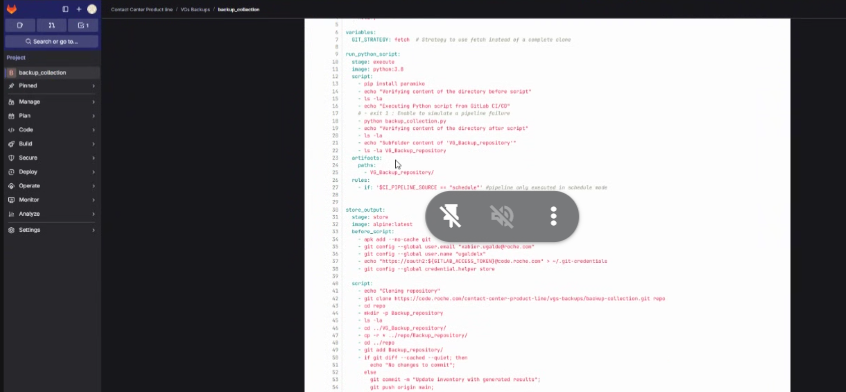


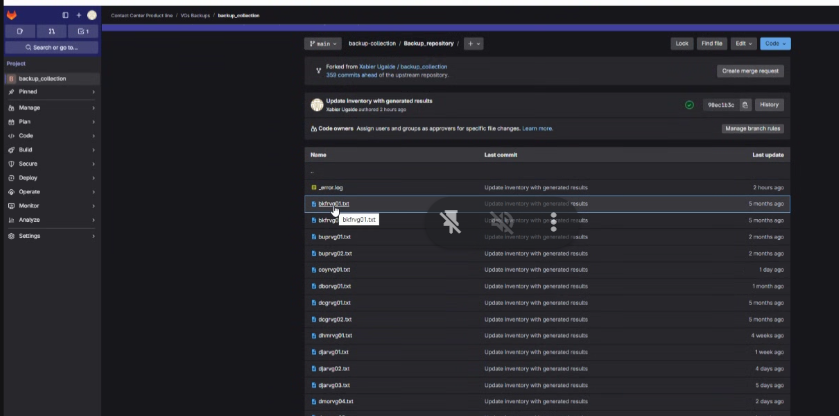


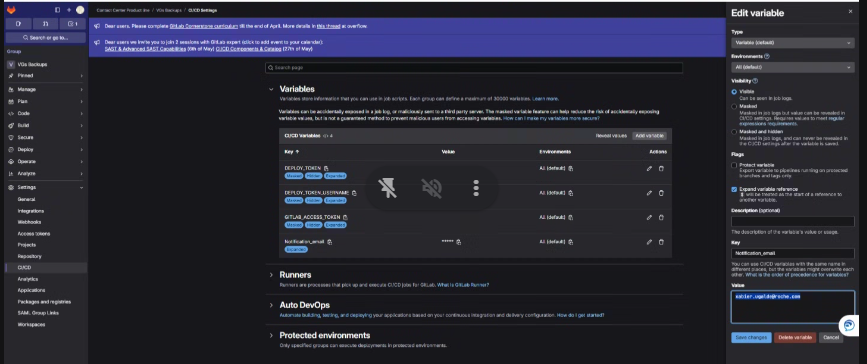










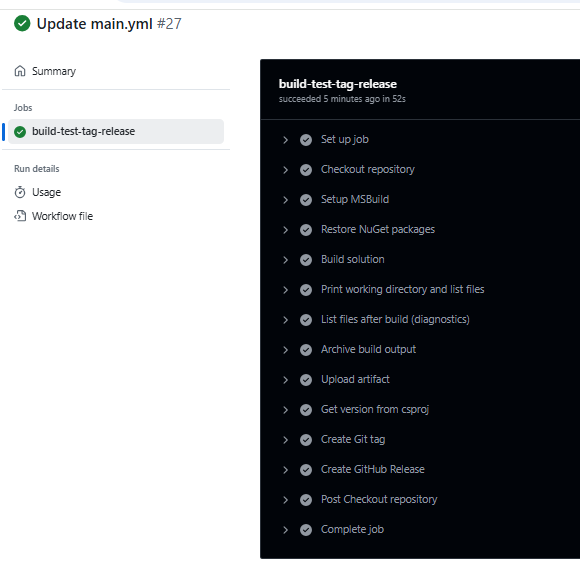


Git Tag:

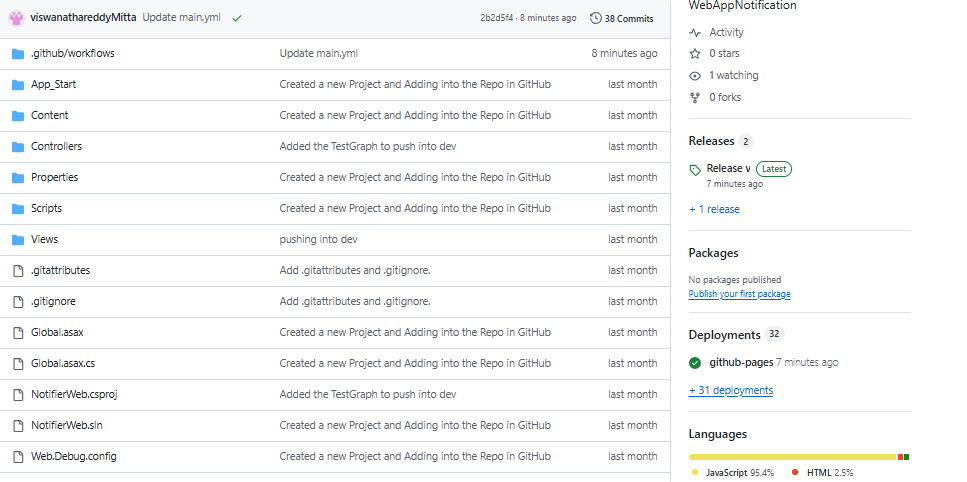
Github Actions:

Created the auto Workflow for Build, Test, Tag, and Release

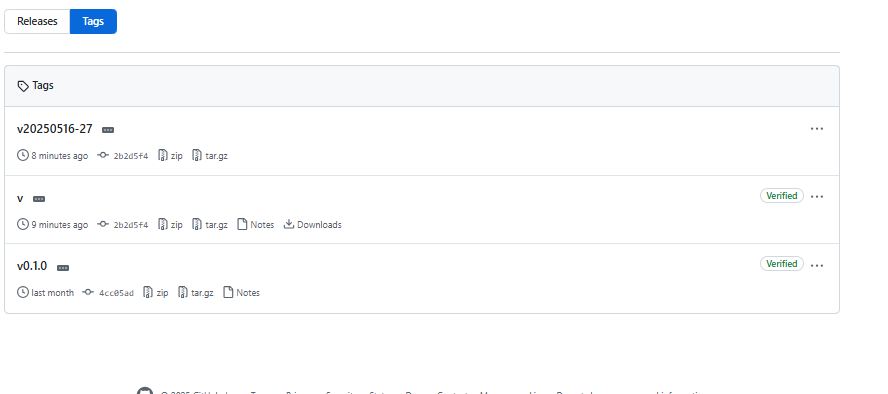




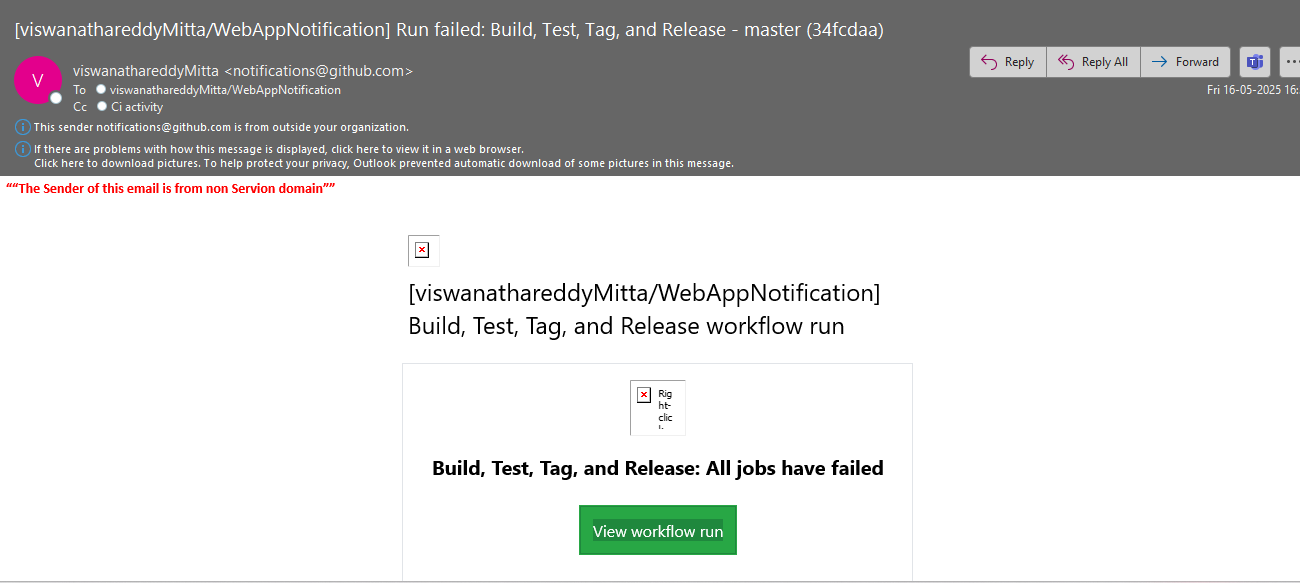
It can create tags and deployments.



It will create tags and release



If Work flow fails



If Workflow succussed

