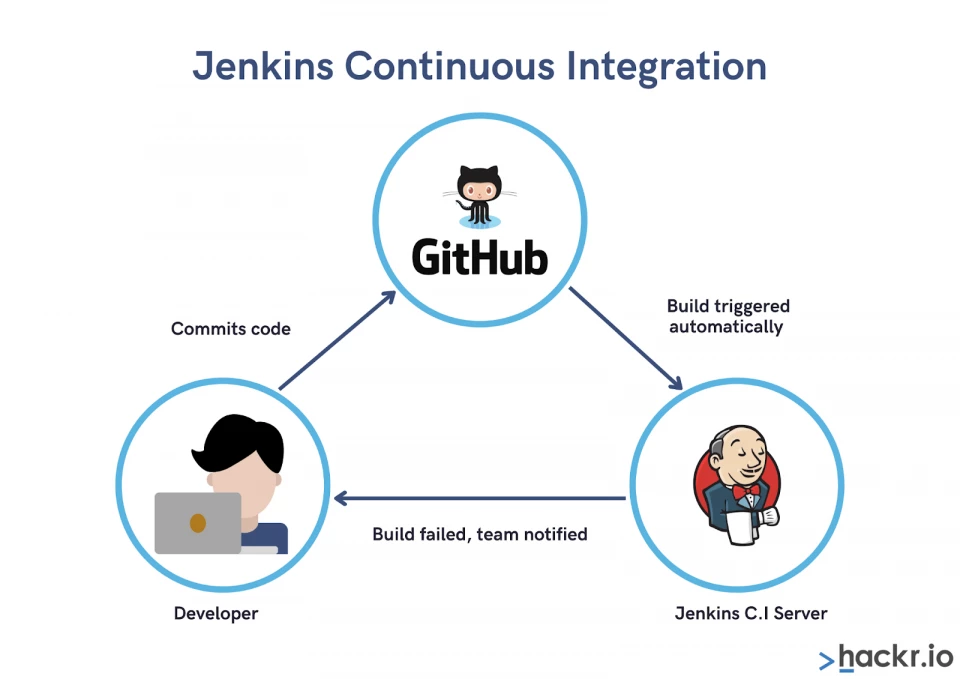
**Jenkins CI**

Jenkins-CI is like a helpful tool in the world of computer programs. It's open-source and used for something called continuous integration (CI). CI is a way of putting together different pieces of software smoothly. Jenkins-CI makes this happen automatically or with a little nudge, making it easier to build, test, and launch software parts. It is the most famous open-source CI/CD tool used to support DevOps with several other cloud-native tools.

Even though it started with the job of building software, Jenkins-CI is quite flexible. Jenkins-CI can perform more than 800 different tasks, and the credit goes to its additional add-ons and extensions. These extra features enhance its functionality and make it adaptable to various needs. In Jenkins-CI, we talk about "Projects," which are like a set of tasks to turn starting information into clear results. Projects have important parts like rules, triggers, and steps that can be adjusted or added to fit different needs.

Jenkins-CI isn't just for building software anymore; it can handle many different kinds of jobs, especially in science or research where you need to do a series of tasks in order. It's like having a handy assistant that can handle various tasks automatically and make things easier for you.



Development teams, often comprised of multiple members, face the challenge of handling potential bugs within numerous commits. Continuous Integration (CI) proves invaluable in identifying these bugs early in the development process, preventing flawed code from reaching production. The critical step in CI is the "build," where all pieces of code come together to form an executable form. If the build is successful (referred to as "green"), the code is ready for deployment. In case of a failed build, the team must address the bugs and restart the entire process.

Jenkins, as a leading CI tool, owes much of its strength to its vibrant community. This supportive community contributes significantly to the tool's success, evident in its exponential growth, millions of downloads, and active user base.

The continuous integration achieved by Jenkins relies on its plugin-based approach. Jenkins seamlessly integrates various DevOps stages through these plugins. Installation of the required plugins enables teams to set up and utilize Jenkins for their specific integration needs.

Now, delving into the practical use of Jenkins, its involvement in the development process unfolds as follows:

Developers regularly commit code to the repository, prompting Jenkins to check for changes.

Upon detecting code changes, Jenkins initiates the compilation process.

In the case of a failed build, developers receive timely notifications to address the issues.

For successful builds, Jenkins facilitates the deployment of code to production.

Jenkins's effectiveness is further enhanced by its diverse range of plugins, offering extensive customizability to adapt to various environments. Beyond its core CI capabilities, Jenkins supports scheduling tests based on specific events, allowing users to define the frequency of these tests without manual intervention. This flexibility and automation make Jenkins a versatile and indispensable tool in the realm of DevOps, playing a pivotal role in organizations ranging from startups to tech giants like Netflix.

(https://hackr.io/blog/what-is-jenkins)

**What are the Jenkins Features**

Jenkins offers many attractive features for developers:

**1.Easy Installation**

Setting up Jenkins is a breeze – it's a platform-flexible, self-contained Java-based program that's good to go. Whether you're on Windows, Mac OS, or a Unix-like system, Jenkins has you covered with tailored packages.

Plus, its user-friendly setup ensures you can start automating tasks without getting lost in technical details. It's like having a tech-savvy friend setting up your workspace effortlessly.

**2.Easy Configuration**

Configuring Jenkins is a straightforward process facilitated by its user-friendly web interface. This interface not only assists with setup but also offers error checks and a built-in help function. Think of it as having a supportive friend by your side, ensuring you navigate the configuration process with ease and confidence.

**3.Available Plugins**

Jenkins, with its extensive plugin library, opens up a world of possibilities. The Update Center houses hundreds of plugins that seamlessly integrate with every aspect of the continuous integration (CI) and continuous deployment (CD) toolchain. Think of it as a vast toolbox, providing endless customization options for your projects. By leveraging Jenkins plugins, you can tailor your experience, effortlessly connecting it to your favorite tools and enhancing your workflow with ease. The plugin ecosystem serves as your gateway to a feature-rich and personalized Jenkins experience

**4.Extensible**

Jenkins, with its plugin architecture, offers unparalleled flexibility. You can effortlessly enhance its functionality, unlocking a world of possibilities. Imagine having a customizable toolbox for your development needs – Jenkins’ extensibility empowers you to add and tailor features, ensuring they align seamlessly with your project’s unique requirements. Need additional tools for your software toolkit? Jenkins has you covered. Its extensibility opens doors to a vast selection of plugins, providing solutions for a multitude of development scenarios .

**5.Easy Distribution**

Jenkins takes collaboration to the next level by seamlessly distributing tasks across multiple machines. This cooperative approach accelerates your builds, tests, and deployments, ensuring an efficient project pipeline. Imagine Jenkins as a project conductor, harmoniously orchestrating tasks across different machines. It ensures that no single machine bears the entire load, optimizing overall performance. Need speed? Jenkins has you covered. Its easy distribution feature efficiently spreads the workload, reducing bottlenecks and enhancing project efficiency. Think of it as a well-coordinated team – Jenkins ensures everyone plays their part, contributing to faster and more reliable outcomes across various platforms. And remember, Jenkins’ easy distribution isn’t just about speed; it’s also about smart resource management. Tasks are strategically allocated, making the most of available resources and maintaining a balanced workload.

**6.Free Open Source**

Jenkins embodies the essence of development freedom! As an open-source resource, it not only equips you with powerful features but also comes without a price tag, making it accessible to everyone.

Join the community! Being open-source means that Jenkins is backed by a diverse and engaged community. You’re not merely using a tool; you’re part of a collaborative network that shares knowledge and supports one another.

Budget constraints? Not here! Jenkins, being free and open-source, ensures that even smaller teams or individual developers can harness its capabilities without worrying about licensing costs.

Think of it as having a helpful mentor – Jenkins’ open-source nature means you have a wealth of shared experiences and collective wisdom from the community at your disposal.

7. Continuous Monitoring:

Jenkins keeps an eye on your projects, like a vigilant guardian. It monitors changes and alerts you if something goes wrong, ensuring a smooth sailing development process.

No surprises! With continuous monitoring, Jenkins acts as a watchful companion, giving you a heads-up if anything needs attention. It's like having a reliable ally always by your side.

8. Security Integration:

Jenkins takes security seriously. With built-in security features and integrations, it ensures that your development environment remains protected, providing a secure foundation for your projects.

It's like having a vigilant security guard – Jenkins seamlessly integrates security measures, offering features like access controls and encryption to safeguard your code and sensitive information.

(https://www.simplilearn.com/tutorials/jenkins-tutorial/what-is-jenkins#what\_are\_the\_jenkins\_features)

**History & Origin of TeamCity**

TeamCity, developed by JetBrains, emerged as a build management and continuous integration server. Its inaugural release took place on October 2, 2006. TeamCity operates under a commercial software model and is licensed under a proprietary license. A freemium license allows for up to 100 build configurations, accompanied by three complimentary Build Agent licenses.

**What is TeamCity?**

TeamCity stands as a versatile Continuous Integration and Deployment server, offering built-in features for continuous unit testing, code quality analysis, and prompt reporting on build issues. With a straightforward installation process, deploying TeamCity is quick and easy, enabling you to enhance your release management practices within minutes. TeamCity seamlessly supports Java, .NET, and Ruby development and integrates harmoniously with leading IDEs, version control systems, and issue tracking systems.

**Why use TeamCity?**

TeamCity serves as an automated solution for building and testing software products. It ensures swift feedback on each code change, mitigates code integration issues, and fosters enhanced teamwork efficiency. Notably, TeamCity plays a pivotal role in the development of numerous popular games, websites, banking systems, and all products developed by JetBrains.

(https://www.devopsschool.com/blog/what-is-teamcity-and-how-it-works-an-overview-and-its-use-cases/#Features\_of\_TeamCity\_are)

Features Of TeamCity:

**Features of TeamCity (https://intellipaat.com/blog/what-is-teamcity/#features\_of\_teamcity)**

TeamCity comes with some helpful features for teams, which are shown below:

**1.Build Pipelines**:

Teams can seamlessly link multiple builds, establishing a coherent sequence. This functionality proves invaluable for automating intricate workflows, encompassing the comprehensive process of building, testing, and deploying software applications.

Streamlining complexity: Build Pipelines act as a navigational thread, simplifying intricate workflows and ensuring a smooth transition through the various stages of software development.

**2.Dependencies**:

TeamCity exhibits intelligent sequencing, automatically determining the order in which builds should run. This ensures a well-organized and synchronized execution of tasks, promoting a seamless workflow.

Streamlining execution: Dependencies in TeamCity contribute to an optimized workflow, eliminating guesswork and enhancing the efficiency of build runs by ensuring proper order and coordination.

**3.Notifications**:

TeamCity excels in keeping teams informed with timely notifications, signaling successes or hitches in builds. This proactive communication ensures that teams are consistently updated on the status of their builds.

Proactive awareness: Notifications in TeamCity act as a reliable messenger, empowering teams with real-time updates. This ensures quick responses and informed decision-making based on the latest build outcomes.

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

Features of Team City (<https://mindmajix.com/teamcity-tutorial#features>)

**4.Technology Awareness:**

TeamCity earns its title as an "Intelligent CI server" due to its comprehensive integration approach. When TeamCity claims support for a tool, it goes beyond surface-level compatibility. For instance, backing Visual Studio projects entails automatic detection of hardware versions, support for testing frameworks, code coverage analysis, static code analysis, and more. The noteworthy aspect is that you receive this support seamlessly, without the need for additional plugins or modifications to build scripts!

**5.Cloud Integrations:**

TeamCity maximizes the potential of cloud computing, dynamically expanding its build agents across platforms such as Amazon EC2, Microsoft Azure, and VMware vSphere.

Effortless scalability: Through Cloud Integrations, TeamCity ensures seamless scaling of build operations across diverse cloud platforms, providing flexibility and efficient resource utilization for optimized performance.

**6.Key Integrations:**

TeamCity recognizes the importance of essential tools in the CI and CD landscape, including version control, issue tracking, build tools, and package repositories. The platform prioritizes seamless integration to provide comprehensive support for these foundational elements.

Holistic support: TeamCity ensures a well-rounded CI and CD experience by seamlessly integrating with crucial tools. This commitment enhances collaboration and ensures a cohesive development environment by connecting version control, issue tracking, build tools, and package repositories.

**7.Configuration:**

Recognizing the programming principle of avoiding code duplication, TeamCity extends the same philosophy to build engineers by offering multiple avenues to prevent duplication of settings.

Streamlined configuration: TeamCity understands the importance of efficient configuration management. By providing diverse options for reusing settings, it empowers developers to create streamlined and consistent build configurations, minimizing redundancy in the process.

**8.Code Quality Tracking:**

TeamCity is dedicated to elevating code quality in your projects, and it goes the extra mile to achieve this goal. Bundled with renowned Intelligent IDEA and ReSharper code analysis, it provides comprehensive reviews for Java and .NET code. Additionally, essential coverage tools are included and supported right out of the box. TeamCity even allows you to use code quality as a condition for build failure.

Elevating standards: TeamCity doesn't just build projects; it nurtures a culture of superior code quality. With built-in code analysis and support for major coverage tools, it empowers developers to maintain high standards and utilize code quality as a key criterion in the build process.

https://jst.iuh.edu.vn/index.php/jst-iuh/article/view/762/369

Circle CI

Continuous Integration (CI) has become a widely adopted development practice in the software industry . In this approach, team members frequently integrate and merge their development work, such as code changes, multiple times a day. CI offers several benefits, including shorter and more frequent release cycles, improved software quality, and increased team productivity . The significance of continuous practices has led to a growing body of literature discussing various approaches, tools, practices, and challenges. This trend is evidenced by the existence of five secondary studies on CI, rapid release, Continuous Delivery (CDE), and Continuous Deployment (CD).

As a CI service, CircleCI is chosen for its advantages, offering 1500 free build minutes per month and supporting builds for projects hosted in private repositories. It's worth noting that CircleCI specifically integrates with GitHub and Bitbucket, providing a versatile solution for teams embracing CI principles.

https://circleci.com/docs/about-circleci/

CircleCI's mission is centered on facilitating change management to enable software teams to innovate at an accelerated pace. The platform empowers technology-driven organizations, fostering an environment for engineering teams to enhance productivity and drive innovation through effective change management. CircleCI delivers enterprise-class support and services, seamlessly operating across various environments, including Linux, macOS, Android, and Windows. It adapts to your preferred infrastructure, whether in the cloud or on your servers.

With CircleCI, teams can seamlessly build, test, and deploy their software through intelligent automation. The platform's focus on automation streamlines the development lifecycle, allowing teams to iterate quickly, maintain high-quality code, and achieve faster innovation cycles.

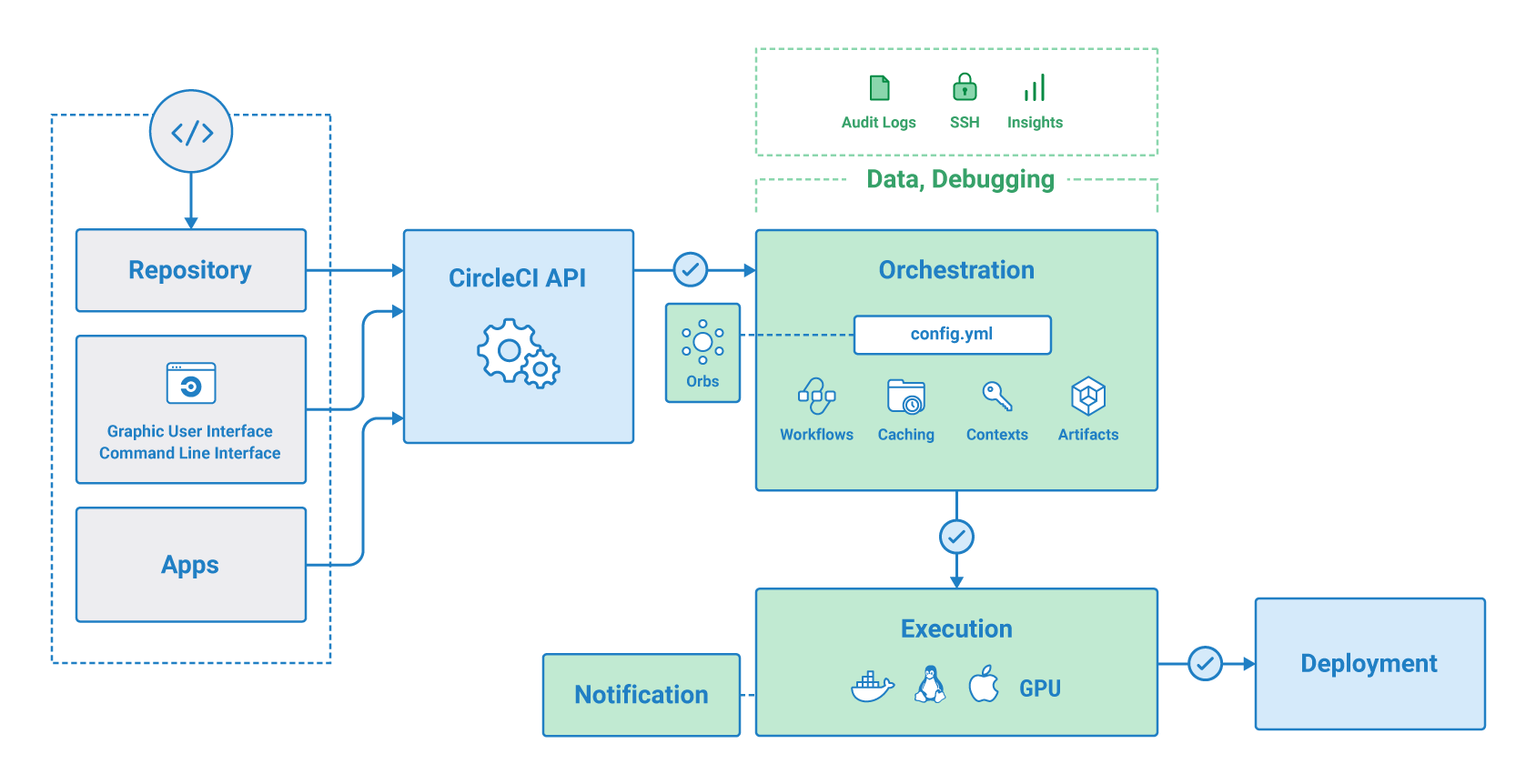


Figure 1

1.Artifacts

refer to files or directories that you want to persist and make available after a job has completed. These files could include build artifacts, test results, documentation, or any other outputs that you want to retain for further analysis or use in subsequent stages of your CI/CD pipeline.

CircleCI allows you to specify which files or directories should be treated as artifacts, and these artifacts can then be accessed, downloaded, or passed to other jobs within the same workflow. Artifacts provide a way to share and persist data between different parts of your CI/CD process, enhancing the flexibility and functionality of your build and deployment workflows.

**2.Caches:** Yes, CircleCI supports caching to store dependencies and build artifacts for faster subsequent builds.

**3.Workspaces**: Yes, CircleCI uses workspaces to share files and directories between jobs.

**4.Docker layer caching:** Yes, CircleCI integrates with Docker, and it supports caching Docker layers to speed up builds. (Chat GPT)

**5.Dynamic configuration:** CircleCI allows for dynamic configuration through the use of environment variables and other configuration options.

**6.Execution environments:** Yes, CircleCI provides environments for executing jobs, allowing you to define the runtime environment for each job.

**7.Jobs:** Yes, jobs are the individual units of work in a CircleCI configuration.

Jobs are the building blocks of your configuration. Jobs are collections of [steps](https://circleci.com/docs/concepts/#steps), which run commands/scripts as required. Each job must declare an executor that is either docker, machine, windows, or macos. For docker you must [specify an image](https://circleci.com/docs/executor-intro/#docker) to use for the primary container. For macos you must specify an [Xcode version](https://circleci.com/docs/executor-intro/#macos). For windows you must use the [Windows orb](https://circleci.com/docs/executor-intro/#windows).

8.Orbs: Yes, orbs are reusable packages of configuration that simplify the setup of common tools and integrations in your CircleCI configuration.

9.Parallelism: CircleCI allows for parallel execution of jobs, which can significantly speed up the build process.

10.Pipelines: Pipelines in CircleCI allow you to define the entire workflow, including dependencies and job sequences.

11.Resource class: Yes, CircleCI allows you to specify the resource class for your jobs to control the amount of CPU and RAM allocated.

**Features of GIT Lab**

Features that are generally available in GitLab:

1.Team Planning:

GitLab has features that facilitate team collaboration, such as issue tracking, boards, and milestones. These tools help teams plan and organize their work.

**Enable teams to effectively plan and execute work in a single application**

As GitLab aims to replace fragmented DevSecOps toolchains with a unified platform, the Plan stage strives to develop comprehensive planning tools seamlessly integrated into the entire DevSecOps lifecycle. Our objective is to empower teams to consistently deliver customer and business value, achieving the shortest cycle times possible.

## Stage Overview

The Plan Stage offers a range of tools for teams to effectively manage and streamline their work processes, track operational performance, and evaluate outcomes. As an all-encompassing DevSecOps platform, GitLab is in a unique position to provide a comprehensive planning suite. This suite empowers business leaders to drive their vision forward and enables DevSecOps teams to deliver value while enhancing their workflow efficiency. Moreover, GitLab's integration of the DevSecOps process enables seamless data linkage across every stage of development—from ideation and planning to implementation, deployment, and ultimately delivering value to customers.

-----------

2.Design Management:

GitLab includes design management capabilities, allowing teams to version control and collaborate on design assets within the platform.

Design Management at GitLab encompasses two broad areas of features:

1.Transient "Communication-based" Features: These features are temporary and typically have a clear beginning and end, similar to Issues. They facilitate communication and collaboration throughout the design process. Examples include:

Attaching Designs to Issues

Visual communication in comments on designs

Approvals of Designs

Links to prototypes in the description of issues

Developer handoff with specifications

Production Design Inspect to compare the code to the design

Closing Issues once Designs are "done"

These features support seamless communication and collaboration among team members during various stages of the design process.

2.Evergreen "Project-based" Features: These features are permanent and exist as standalone projects or assets. They serve as repositories for backing up native design files, connecting local design system files to live frontend assets, and deep linking assets to original repository files.

GitLab's unique offering encompasses both types of Design features, capitalizing on its support for issues, which fosters collaboration with engineering teams, and repositories, which host the actual code. This comprehensive approach significantly enhances collaboration and streamlines design management processes within the platform, ensuring a seamless experience for users.

3.Quality Management:

GitLab has features related to quality management, including built-in CI/CD for automated testing and code quality analysis tools.

#### 1.Using GitLab as a Quality Management System for ISO 13485:2016  GitLab's portfolio and project management (PPM) tools can potentially aid in achieving compliance with the requirements outlined in ISO 13485:2016. This international standard specifies the criteria for a quality management system (QMS) applicable to organizations participating in various stages of the medical device lifecycle, encompassing design through disposal, as well as suppliers and vendors associated with these organizations. By leveraging GitLab's PPM tools, businesses may streamline their processes and ensure adherence to ISO 13485:2016 standards.

#### Can GitLab be used as part of an ISO 13485:2016 QMS? GitLab can serve dual purposes within a QMS: as both an application integrated into the system and as a functional tool to facilitate faster development and deployment processes. By consolidating Portfolio and Project Management (PPM) within the same platform as development and deployment tools, organizations can effectively centralize and streamline their QMS, fostering collaboration among teams and breaking down silos.

4.Wiki:

GitLab includes a Wiki feature, providing a space for documentation and information sharing within a project.  
  
The Wiki serves as an integrated documentation system within every GitLab project. Users can create and manage Wiki pages using a user-friendly web interface or Git for more advanced users. GitLab maintains a complete history of all Wiki pages, enabling users to keep a comprehensive record of changes and updates over time.

5.Pages:

GitLab Pages allows users to create static websites for their projects, serving as an additional documentation or showcase platform.   
  
We're excited to announce that GitLab Pages, a feature previously exclusive to GitLab Enterprise Edition, is now available in the open-source Community Edition. GitLab Pages enables you to effortlessly host static websites directly from GitLab, supporting any static site generator of your choice.

This expansion comes in response to requests from the broader community to open-source GitLab Pages. We're thrilled to bring this capability to the Community Edition, allowing users to leverage GitLab Pages for hosting their static sites without any limitations. Whether it's personal websites, documentation sites, or any other static content, GitLab Pages offers a seamless hosting solution integrated within GitLab's ecosystem.

6.Code Review Workflow:

Available: GitLab supports a comprehensive code review workflow, including merge requests, comments, and collaborative code review tools.

Code Review stands as a critical aspect of software development, ensuring that contributions uphold and enhance code quality and security. It serves as a platform for mentorship and feedback among engineers, though it can sometimes be a time-consuming process.

GitLab adheres to the principle that code review enhances the overall product by refining code quality while prioritizing efficient delivery. The process commences with authors proposing changes through a change proposal. Subsequently, they solicit feedback from peers, including developers, designers, security and operations teams, and product managers, among others. Authors then address this feedback, leading to the eventual approval and merging of a merge request, thereby concluding the code review process for a particular changeset.

7.GitLab CLI:

Available: GitLab provides a command-line interface (CLI) for interacting with GitLab features and performing various actions from the terminal.  
  
  
The command line serves as a fundamental tool for software engineers, central to their workflow and processes. It's where they customize their environment, extend functionality through applications, and execute scripts critical to their work. GitLab fosters collaboration and software development within its application, yet developers spend a significant portion of their time locally, implementing tasks outlined in issues, addressing merge request feedback, and testing/debugging applications.

To support developers where they do their most meaningful work, GitLab aims to enhance efficiency closer to their local environment. This includes:

Configuration: The GitLab CLI aids users in efficiently configuring projects and managing project configurations, vital for instance management. Additionally, it facilitates managing and testing GitLab CI/CD configurations to ensure correct setup.

Contribution: For engineering personas involved in contributing directly to project code, the GitLab CLI simplifies actions such as viewing issue contents, checking merge request statuses, accessing feedback and pipeline output, and more.

Automation: Automating tasks is integral to the engineering process, and the GitLab CLI supports users in scripting repetitive tasks, whether interactively or non-interactively. It enables automation in GitLab CI and facilitates reporting for external systems.

Flexibility: Recognizing the need for diverse workflows and use cases, the GitLab CLI offers a set of predefined commands for interacting with issues, merge requests, and pipelines. Moreover, it provides flexibility by allowing users to alias any API interaction to tailor commands according to their workflow preferences.

8.Web IDE:

GitLab includes a Web IDE, allowing users to edit code directly in the web browser without needing a local development environment.

### Strategy and Themes

There are three critical workflows we aim to support in the Web IDE.

**Configuration**

Users who configure projects or GitLab need editing tools to help them be efficient at this process. Creating specialized configuration files for working with GitLab CI or other areas of GitLab benefit from feedback provided directly in the editor.

A code editor stands as a vital tool in a software developer's arsenal. With the Web IDE, our goal is to meet developers where they are, offering a mature, feature-rich editing experience directly in the browser. By transitioning to a browser-based instance of VS Code, the most popular code editor, we aim to empower developers to tackle more complex tasks and enhance efficiency within GitLab.

Software development involves iterative processes that entail requesting and responding to feedback from various stakeholders, including developers, designers, and product managers. While desktop editors are often optimized with extensions for specific languages or frameworks, they may not be ideal for the frequent context switching and rapid feedback cycles typical during the review phase. GitLab's Web IDE offers a familiar workflow to developers while remaining user-friendly for designers, product managers, and others. It seamlessly integrates editing capabilities into the context of their current tasks within GitLab, providing an editing experience tailored to their workflow.

Key Workflows Supported by the Web IDE:

Configuration:

Users configuring projects or GitLab need efficient editing tools for this process. Creating specialized configuration files, such as gitlab-ci.yml or CODEOWNERS, can be challenging as they must be both syntactically and semantically valid. The Web IDE offers features for validating syntax and semantics, ensuring correct configuration setup. Additionally, once GitLab CI configurations are validated, they may generate review applications or reports on code quality and vulnerabilities, which developers need to reconcile within their editor.

Contribution:

Developers contributing directly to project code must act on feedback received during the review process. Managing feedback often involves juggling multiple windows to view MR feedback while working in an editor. The Web IDE streamlines this process by providing easy access to reviewer feedback and CI job information directly within the editor, facilitating swift action on feedback and updates to merge requests.

* [**Team Planning**](https://about.gitlab.com/features/?stage=plan#team_planning)
* [**Portfolio Management**](https://about.gitlab.com/features/?stage=plan#portfolio_management)
* [**Design Management**](https://about.gitlab.com/features/?stage=plan#design_management)
* [**Requirements Management**](https://about.gitlab.com/features/?stage=plan#requirements_management)
* [**Quality Management**](https://about.gitlab.com/features/?stage=plan#quality_management)
* [**Wiki**](https://about.gitlab.com/features/?stage=plan#wiki)
* [**Pages**](https://about.gitlab.com/features/?stage=plan#pages)
* [**Value Stream Management**](https://about.gitlab.com/features/?stage=plan#value_stream_management)
* [**DORA Metrics**](https://about.gitlab.com/features/?stage=plan#dora_metrics)
* [**DevOps Reports**](https://about.gitlab.com/features/?stage=plan#devops_reports)
* [**Source Code Management**](https://about.gitlab.com/features/?stage=create#source_code_management)
* [**Code Review Workflow**](https://about.gitlab.com/features/?stage=create#code_review_workflow)
* [**GitLab CLI**](https://about.gitlab.com/features/?stage=create#gitlab_cli)
* [**Web IDE**](https://about.gitlab.com/features/?stage=create#web_ide)
* [**Remote Development**](https://about.gitlab.com/features/?stage=create#remote_development)
* [**Editor Extensions**](https://about.gitlab.com/features/?stage=create#editor_extensions)
* [**Code Suggestions**](https://about.gitlab.com/features/?stage=create#code_suggestions)
* [**Continuous Integration (CI)**](https://about.gitlab.com/features/?stage=verify#continuous_integration)
* [**Merge Trains**](https://about.gitlab.com/features/?stage=verify#merge_trains)
* [**Code Testing and Coverage**](https://about.gitlab.com/features/?stage=verify#code_testing)
* [**Review Apps**](https://about.gitlab.com/features/?stage=verify#review_apps)
* [**Pipeline Composition**](https://about.gitlab.com/features/?stage=verify#pipeline_composition)
* [**GitLab Runner Core**](https://about.gitlab.com/features/?stage=verify#runner)
* [**Fleet Visibility**](https://about.gitlab.com/features/?stage=verify#fleet_visibility)
* [**GitLab Runner SaaS**](https://about.gitlab.com/features/?stage=verify#runner_saas)
* [**Build Artifacts**](https://about.gitlab.com/features/?stage=verify#build_artifacts)
* [**Secrets Management**](https://about.gitlab.com/features/?stage=verify#secrets_management)

(https://about.gitlab.com/features/?stage=plan)

Here's an investigation of Jenkins, TeamCity, CircleCI, and GitLab CI based on the specified eight features in a tabular column:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

| **Feature** | **Jenkins** | **TeamCity** | **CircleCI** | **GitLab CI** |
| --- | --- | --- | --- | --- |
| **Flexibility** | Highly flexible due to open-source nature. Allows extensive customization through plugins and scripting. | Offers flexibility with advanced features and customization options. | Known for cloud-native flexibility. Provides platform-agnostic environment. | Integrated flexibility within GitLab platform. |
| **Community Support** | Benefits from large and active community. Extensive documentation and community-driven plugins. | Offers professional support services, supplemented by active community and documentation. | Provides support services and maintains active user community. | Offers support services as part of GitLab platform. |
| **Integration** | Seamlessly integrates with wide range of tools, including version control, build, test, and deployment. | Seamless integration with version control, build, IDEs, and other development tools. | Strong integration with popular version control systems and third-party services. | Integrated within GitLab platform, seamlessly integrates with code repositories, issue tracking, and other DevOps tools. |
| **Extensibility** | Highly extensible with wide range of plugins available for various use cases. | Extensible with plugins and custom build runners, enabling integration with third-party tools. | Supports automation of workflows through configuration files and custom scripts. | Part of GitLab platform, supports extensions and custom integrations through APIs. |
| **Build Distribution** | Can distribute builds across multiple machines. | Efficiently scales build agents on cloud platforms. | Dynamically scales builds using cloud platforms. | Integrated within GitLab platform, handles distributed builds efficiently. |
| **Adoption** | Widely adopted across diverse industries. | Popular in enterprise settings. | Well-received for cloud-native approach. | Gaining popularity as an integrated DevOps platform. |
| **Build Types** | Supports both freestyle and pipeline-based builds. | Offers advanced features for defining and executing build pipelines. | Supports creating sequences of builds. | Integrated CI/CD pipelines within GitLab platform. |
| **Distributed Builds** | Can distribute builds across multiple machines. | Efficiently scales builds on cloud platforms. | Distributes work across multiple machines. | Integrated within GitLab platform, handles distributed builds efficiently. |

This table provides a comparative overview of Jenkins, TeamCity, CircleCI, and GitLab CI based on the specified eight features. Each tool has its strengths and is suited to different use cases and preferences.

Jenkins

Jenkins, a prominent open-source continuous integration server, stands out due to its remarkable flexibility stemming from its open-source nature. Supported by a large and active community, Jenkins offers an abundance of plugins and resources. It seamlessly integrates with various tools, providing development teams with a versatile platform. Its extensibility is a key asset, allowing users to tailor it to specific needs using a wide range of plugins.

Jenkins stands out in the efficient distribution of builds across multiple machines, a feature that enhances scalability and accelerates the build processes. This capability proves invaluable for optimizing resource utilization, especially in large-scale projects or organizations. Jenkins' robust support for distributed builds is a key factor contributing to its widespread popularity and effectiveness across diverse development environments.

**Advantages and Disadvantages of Jenkins**

Advantages of Jenkins

Jenkins is not the only option for continuous integration in the market; however, there are some other tools also available to do the same. But the Jenkins is more important and most widely used because of its use and amazing features. It would help if you spend some time studying Jenkins, and definitely, it will give you a lot of amazing features beyond your imagination. It is also known as the Heart of the DevOps pipeline. There are a lot of benefits of using the Jenkins pipeline; a few of them are given below:

A black background with colorful rectangular objects

Description automatically generated with medium confidence

1. **Open Source and Free:** Developers don’t need to take tension about the money; it is free of cost. It is platform-independent.
2. **Plug-ins and Integration:** It is one of the most important features that make it most widely used. It has its type of plug-in, which helps the developer a lot in executing the jobs. Jenkins plug-ins can be developed by anyone and for anyone. Dashboard view plug-in, test analysis plug-in, build pipeline plug-in, and many more like this makes the developer familiar with the Jenkins tool.
3. **Hosting Option:** It is yet another important feature of the Jenkins, which can be installed on any operating system like Windows, MacOS, Linux, etc. You can also run Jenkins on the cloud by downloading and deploying it on a VM. You can also use a Docker container in it.
4. **Community Support:** Jenkins has great support from the developer community. You can assume its popularity and community support that it has more than 1000000 users all over the world, while it was officially published in 2011.
5. **Integration with other CI/CD platforms:** Jenkins supports many CI/CD platforms, not only the pipeline. It can make interaction with other tools also. Several plug-ins are available in it, which allows users to make connections with other CI/CD platforms.
6. **Keep your team in sync:** Jenkins focuses on a centralized way of working. All the members of the team move in sync.
7. **Easy to debug:** It is very easy to find out the errors in the Jenkins. The developer can easily check the bug and resolve it.
8. **Less time to deliver the project:** It happens because of its continuous integration feature.
9. **Flexible in creating the jobs:** It is very flexible in creating the jobs. It can create jobs both in freestyle and in the pipeline process very easily.
10. **Source Code Management (SCM):** Jenkins supports different types of source code repositories like SVN, Git, etc. The developer can set different trigger after making changes in the codes. He can do it every time.
11. It makes the process of converting in GUI from CLI very easy.
12. It provides accurate data support to project management.
13. It supports many languages, like Java, Python, etc.

### Disadvantages of Jenkins:

Following are some cons of the Jenkins tool:

1. All plug-ins are not compatible with the declarative pipeline syntax.
2. Jenkins has many plug-ins in its library, but it seems like they are not maintained by the developer team from time to time. This is when it becomes very important that whatever plug-ins you are going to use; are getting a regular update or not.
3. Lots of plug-ins have a problem with the updating process.
4. It is dependent on plug-ins; sometimes, you can't find even basic things without plug-ins.
5. Jenkins UI is not user friendly in comparison to current UI Technologies. It cannot be very clear for the first-time user.
6. Managing of Jenkins dashboard is hard when we have too many jobs to be executed.
7. Changes made by a developer will not be visible to another developer of the team; only the manager can have its access. It makes tracking difficult in large projects.
8. Although many plug-ins are available, it does not provide the analytical support in the end to finish the deployment cycle. It leads to the creation of difficulties in the tracking of the project deployment.
9. The error message is not clear sometimes. It becomes hard to find out what went wrong.
10. You have to maintain your infrastructure by yourself.
11. It is not easy to install and configuration the Jenkins tool.

(https://www.tutorialandexample.com/advantages-and-disadvantages-of-jenkins)

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

TeamCity

TeamCity has proven to be an essential asset for software development teams, offering streamlined workflows and heightened productivity, particularly within agile project management systems. Users leverage its seamless integration of git repositories with ticketing systems, facilitating the testing and release of code across environments. The software's automation testing capabilities are lauded for evaluating and integrating numerous projects, ensuring the quality and reliability of SaaS-based products. Noteworthy features include a user-friendly interface, praised for easy navigation and efficient build management. TeamCity's flexible templates, variables, and parameterization capabilities enable quick and efficient solutions, such as deploying MVC applications to IIS. The software seamlessly integrates with popular platforms like Git, Azure, AWS, and Visual Studio Team Services, enhancing CI/CD processes. While users appreciate its functionality, some express concerns about a lengthy and manual upgrade process, the absence of a SaaS-based solution, and perceived weaknesses in pipeline visualization. Despite these considerations, TeamCity remains a valuable tool, providing reliable performance, a user-friendly interface, and commendable customer support.

Pros:

Reliable Performance: Users consistently praise TeamCity for its reliable performance, meeting expectations without issues.

User-friendly Interface: Many customers appreciate the intuitive and easy-to-navigate interface, simplifying usage and understanding of features.

Great Customer Support: Numerous users express satisfaction with the excellent customer support, reporting positive experiences with helpful, knowledgeable, and prompt representatives.

Cons:

Long and Manual Upgrade Process: Users find the upgrade process time-consuming and cumbersome, requiring significant effort.

Lack of SaaS-based Solution: Some users express disappointment in the absence of a SaaS-based solution, necessitating the hosting and maintenance of a large server and multiple build agents.

Weak Pipeline Visualization: Users commonly raise concerns about the weakness of TeamCity's pipeline visualization feature, suggesting improvements for clarity and usability compared to other platform aspects.

(https://www.trustradius.com/products/teamcity/reviews?qs=pros-and-cons#reviews)

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

Circle CI

CircleCI is a highly regarded continuous integration and deployment tool widely used by engineering teams to streamline application development workflows. Its seamless integration with GitHub allows for effortless pull request checks, automatically running tests and linters for each pull request and merge. Users commend CircleCI for its reliability, responsiveness, and comparatively fewer issues than other CI tools. The platform's strong support is consistently praised, with ongoing improvements enhancing the user experience.

A key use case involves running tests on every GitHub commit and deploying to various environments based on the branch, ensuring bug-free code delivery and accelerating development through automated building, testing, and deployment. CircleCI excels in automating diverse build and test processes, supporting pre-deploy and post-deploy scripts, executing test suites, and integrating notifications via platforms like Slack.

CircleCI shines in handling the build process for different application types, including Android, iOS, and web applications. By outsourcing build server management to CircleCI, teams can focus on custom application development while relying on a robust CI/CD solution. Noteworthy features include CircleCI's concurrency, allowing efficient test suite distribution across multiple slices, reducing testing time. The tool is widely adopted for its cloud integration capabilities, enabling automated parallel tests in containers.

In summary, CircleCI stands out as a preferred choice for engineering teams across various industries, simplifying deployment processes, providing a customizable environment, and optimizing development workflows. Its tight GitHub integration, responsive support, and reported reliable performance contribute to its popularity in automating and enhancing the software development lifecycle.

Pros:

Continuous Improvement: Users appreciate CircleCI's commitment to continuous enhancements, ensuring a high-quality product. The ongoing improvements are well-received by many users, reflecting the company's dedication to refining CircleCI.

Responsive Support: The highly responsive and helpful support from CircleCI is valued by users when facing issues. The support team's responsiveness and assistance have garnered praise from numerous users, contributing to a positive experience.

Seamless GitHub Integration: Users commend the seamless integration between CircleCI and GitHub, emphasizing its reliability and efficiency in capturing all commits. The integration with GitHub is highlighted as a standout feature by many reviewers.

Cons:

Limited Communication on Updates: Some users express frustration with the lack of communication regarding updates and breaking changes. Delays in responses from account representatives have been reported, impacting workflow efficiency.

Confusing Configuration Options: Users criticize the organization of options in the config file, finding it arbitrary and unhelpful for effective configuration management. This confusion has resulted in challenges during project setup and maintenance on CircleCI.

Limited Customization Options: Several users note that they perceive CircleCI's customization options as somewhat limited compared to tools like Jenkins. They express a desire for more flexibility and control over build processes and workflows, feeling that CircleCI's current feature set doesn't fully meet these expectations.

(https://www.trustradius.com/products/circleci/reviews?qs=pros-and-cons#reviews)

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

GitLab

GitLab is a versatile platform that facilitates the complete automation of DevOps processes, covering planning, design, creation, building, verification, testing, deployment, and monitoring. Widely adopted for its extensive features and modular code availability, GitLab supports collaboration on code platforms, offering CI/CD pipelines, container registry, Kubernetes integration, and robust data portability. The platform allows projects to be both private and public, emphasizing open-source principles for widespread accessibility. GitLab is designed to store and manage committed objects efficiently, serving as a central hub for creating, storing, testing, sharing, and collaborating on web projects.

With GitLab, teams can leverage powerful continuous integration and seamlessly integrated continuous development pipelines. The platform supports various features, including public repositories for wikis, documentation, and issue tracking. GitLab promotes collaboration by enabling teams to work on code scripted in Go and Ruby. Its centralized hub efficiently organizes information and committed objects, making it suitable for beginners and developers alike.

Notable features of GitLab include a robust CI/CD pipeline, an integrated container registry, and Kubernetes support. GitLab offers project management tools such as group milestones, burnup and burndown charts, targeted dates for issues, issue movement between projects, and CSV exports. It facilitates the import and export of projects, surpassing GitHub's capabilities for large projects. GitLab's integrated monitoring gathers performance metrics for both applications and servers, defining the impact of combining branch and production systems.

The platform's importance lies in its comprehensive feature set, including milestone setting, issue tracking, multiple issue boards, labeling, issue assignment, merging, closing sequences, and time tracking. GitLab supports web-based IDEs, a continuously available service desk, fuzzy file finders, fast-forward merges, statistical graph representations, and more. It provides a rich environment for code management, collaboration, and project monitoring, making it a valuable tool for development teams.

Recommendation

Many users on Stackshare highlight several key features of Jenkins, including its ability to be hosted internally, its status as a free and open-source tool, its versatility in building, deploying, or launching applications asynchronously, its extensive collection of plugins with well-documented support, and its wide range of integrations. G2 Crowd reviews echo these sentiments, with users expressing satisfaction with Jenkins' powerful capabilities, easy setup, and straightforward configuration. Users also appreciate Jenkins' visualization of the data deployment process, its support for popular version control systems like GitHub and SVN, and its robust support for Maven and Java. Despite the initial setup time and adoption barriers, users find Jenkins to be a valuable asset in the development lifecycle, saving them significant time and effort. Once properly set up, Jenkins effectively meets users' requirements and needs.

**2nd tool Recommendation (TeamCity)**

Users often prefer TeamCity over Jenkins due to its added features, user-friendly interface, and strong integration capabilities. They highly recommend TeamCity to IT companies, emphasizing its value for enterprise-level continuous integration. Additionally, users suggest self-hosting TeamCity for smaller companies, noting its advantages over other services. Furthermore, they advise leveraging the free trial, exploring available plugins, and assessing new major versions and security concerns. Overall, users regard TeamCity as an excellent choice for build automation and continuous integration, considering it one of the top CI tools on the market.

**Chocolatey**

**Chocolatey** is a package manager designed for Windows operating systems. It streamlines the installation, upgrading, and removal of software. Built on the NuGet infrastructure, Chocolatey utilizes PowerShell to manage software packages. Here are its key features:

1. **Command-line Interface (CLI)**: Chocolatey is primarily accessed via its CLI, allowing users to perform tasks like searching, installing, updating, and uninstalling packages using simple commands.
2. **Package Management**: With an extensive repository of packages containing popular software titles, Chocolatey simplifies software installation and updates from the command line.
3. **Automated Installation**: Chocolatey automates the entire process of downloading, installing, and configuring software packages, saving time and effort.
4. **Integration**: It seamlessly integrates with existing infrastructure and configuration management tools, making it suitable for enterprise environments.
5. **Community and Pro Versions**: Chocolatey offers both a free community edition and a commercial Pro version. The Pro version includes additional features such as package internalization, package creation, and package caching.
6. **Deploy Anywhere:** Chocolatey can be deployed on various Windows environments, including on-premises, Azure, AWS, or any other cloud provider. It requires PowerShell v2+ and .NET Framework 4.x.
7. **Deploy with Everything:** Chocolatey can be integrated with various configuration management solutions like Ansible, Chef, PowerShell DSC, Puppet, or Salt.
8. **All Software Is a First-Class Citizen:** Chocolatey manages all software, not just installers, allowing for comprehensive reporting and inventory management.
9. **Packages are Independent and Portable:** Chocolatey packages are independent and portable, making it easy to migrate between systems.
10. **Test Your Deployments:** Chocolatey allows for testing of deployment scripts, providing more confidence before pushing changes.
11. **Completely Offline and Secure:** Chocolatey operates completely offline and has no call home feature, enhancing security.
12. **Create Your Own Deployment Packages:** Users can create their own deployment packages for free using Chocolatey.
13. **PowerShell Automation:** Chocolatey offers PowerShell automation for software deployments, streamlining the process.
14. **Manage Dependencies With Ease:** Dependency management is simplified with Chocolatey, making complex installations easier to handle.
15. **Open Source Licensing:** Chocolatey is licensed under the Apache v2 license, with no special restrictions. Commercial options come with a business-friendly software license agreement.
16. **Commercial Options Add Support + AMAZING Features:** Chocolatey for Business (C4B) edition includes over 20+ features, providing enhanced software management capabilities and support.
17. **Customers Help Define Our Work:** Chocolatey prioritizes its development based on customer feedback, working closely with users to improve the platform.
18. **Security Is Our Focus:** Chocolatey prioritizes simplicity, scalability, and security in its design and development.

**Pros**:

1. **Easy-to-Use CLI**: Chocolatey provides a straightforward command-line interface (CLI) for managing software packages.
2. **Extensive Package Repository**: Users can access a wide range of software packages from Chocolatey’s repository, making installations convenient.
3. **Automated Software Management**: Chocolatey streamlines software installation, upgrades, and removal, saving time and effort.
4. **Active Community Support**: A vibrant community contributes to Chocolatey, providing assistance and sharing knowledge.
5. **Suitable for Personal and Enterprise Use**: Chocolatey caters to both individual users and enterprise environments.

**Cons**:

1. **Outdated or Unsupported Packages**: Some packages within Chocolatey’s repository may be outdated or lack support.
2. **Limited GUI Interface**: Chocolatey primarily relies on the CLI, which might be less intuitive for users accustomed to graphical interfaces.
3. **Challenging Dependency Management**: Handling dependencies can sometimes be complex when using Chocolatey.

**NuGet**

Top of Form

**NuGet** is a package manager specifically designed for **.NET development**. It seamlessly integrates with **Microsoft’s Visual Studio IDE** and serves as a tool for managing dependencies within .NET projects. Here are its key features:

1. **Integration with Visual Studio**: NuGet is directly accessible from Visual Studio, allowing developers to manage packages within their development environment.
2. **Package Management**: It grants access to a vast repository of **.NET libraries and tools**, simplifying the process of adding functionality to projects.
3. **Versioning**: Developers can specify the desired package versions, ensuring consistency and compatibility across projects.
4. **Package Creation**: NuGet enables developers to create and share their own packages via the NuGet repository.
5. **Dependency Resolution**: When a package is added to a project, NuGet automatically resolves dependencies by downloading and installing required libraries.

**Pros**:

* **Seamless Integration with Visual Studio**
* **Extensive Repository of .NET Packages**
* **Versioning and Dependency Resolution Features**
* **Support for Package Creation and Publishing**
* **Widely Used in the .NET Development Community**

**Cons**:

* **Limited to .NET Development Projects**
* **Complexity in Dependency Management**
* **Variable Package Quality; Some May Be Outdated or Unsupported**

In summary, both **Chocolatey** and **NuGet** are robust package management tools for Windows, each catering to different needs. Chocolatey excels in managing system-level software across various applications, while NuGet focuses specifically on .NET libraries and tools within development projects. The choice depends on your specific requirements and use cases. 🚀

**ghp\_pe43AmZKDAGoQilgcI48Bj7Byp6FIt25FoEb**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated