A picture containing text, clipart

Description automatically generated

INFORMATION TECHNOLOGY PROGRAMME

**SOFT806 Continuous Integration and Continuous Deployment**

**Assessment 1**

**Individual Assignment**

**Trimester 1, 2024**

**Saju Sebastian**

**20230446**

**March 15, 2024**

Table of Contents

[**Task 1 - B** 2](#_Toc161399636)

[**Jenkins CI** 2](#_Toc161399637)

[**TeamCity** 6](#_Toc161399638)

[**Circle CI** 9](#_Toc161399639)

[**GIT Lab** 12](#_Toc161399640)

[Disadvantages of Jenkins: 20](#_Toc161399641)

[**Task 1 – C** 26](#_Toc161399642)

[**Task 2 – A** 28](#_Toc161399643)

[**Jenkins Installation:** 28](#_Toc161399644)

[**TeamCity Installation:** 36](#_Toc161399645)

[**Task 2 – B** 43](#_Toc161399646)

[**Jenkins Plugins & Configurations** 43](#_Toc161399647)

[**TeamCity Plugins & Configurations** 45](#_Toc161399648)

[**Task 2 – C** 45](#_Toc161399649)

[**Jenkins** 45](#_Toc161399650)

[**TeamCity** 45](#_Toc161399651)

[**Task 3 – A** 45](#_Toc161399652)

[**Task 3 – B** 48](#_Toc161399653)

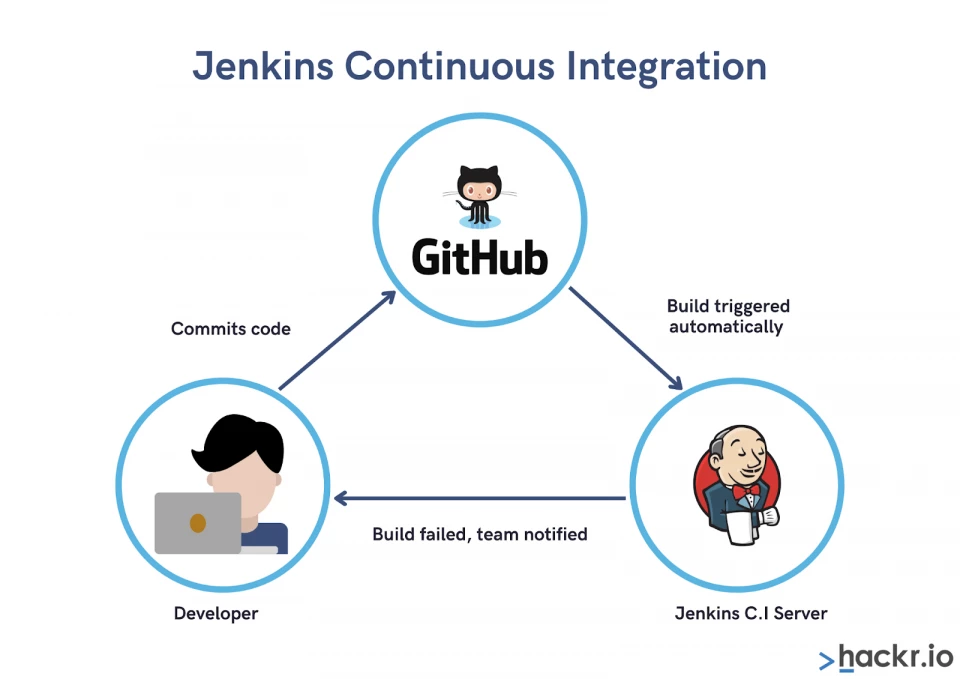
[**Task 3 – C** 49](#_Toc161399654)

# **Task 1 - B**

**For Southern Institute of Studies (SIS), for efficient code integration and delivery, the following four continuous integration tools can be selected:**

## **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.



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)

**Features of Jenkins:**

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 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.

1. **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 ensuring you navigate the configuration process with ease and confidence.

1. **Available Plugins**

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

1. **Extensible**

Jenkins, with its plugin architecture, offers unparalleled flexibility. You can effortlessly enhance its functionality, unlocking a world of possibilities. Jenkins’ extensibility empowers you to add and tailor features, ensuring they align seamlessly with your project’s unique requirements. Its extensibility opens doors to a vast selection of plugins, providing solutions for a multitude of development scenarios.

1. **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. It ensures that no single machine bears the entire load, optimizing overall performance. Its easy distribution feature efficiently spreads the workload, reducing bottlenecks and enhancing project efficiency. Jenkins ensures everyone plays their part, contributing to faster and more reliable outcomes across various platforms. Tasks are strategically allocated, making the most of available resources and maintaining a balanced workload.

1. **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. Being open source means that Jenkins is backed by a diverse and engaged community. You’re not merely using a tool; you are part of a collaborative network that shares knowledge and supports one another. Jenkins, being free and open source, ensures that even smaller teams or individual developers can harness its capabilities without worrying about licensing costs.

1. **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. 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.

1. **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>)

**TeamCity**

TeamCity was developed by JetBrains and was released on October 2, 2006. It 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. It seamlessly supports Java, .NET, and Ruby development and integrates harmoniously with leading IDEs, version control systems, and issue tracking systems. It also plays a pivotal role in the development of numerous popular games, websites, banking systems, and all products developed by JetBrains.

A logo with black text and blue and green circles

Description automatically generated

(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**](https://intellipaat.com/blog/what-is-teamcity/#features_of_teamcity)**)**

(<https://mindmajix.com/teamcity-tutorial#features>)

TeamCity comes with some helpful features for teams:

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.

1. **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.

1. **Notifications**:

TeamCity excels in keeping teams informed with timely notifications, signalling 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.

1. **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.

1. **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.

1. **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.

1. **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.

1. **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).

A black and white logo

Description automatically generated

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

Circle CI's mission is centred 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. Circle CI 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.

**Features of Circle CI:**

1. **Artifacts:**

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. Circle CI 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.

1. **Caches:**

Circle CI’s caching feature is a powerful tool that optimizes dependency management and accelerates build execution. By storing dependencies and build artifacts, Circle CI reduces the need to download them repeatedly from remote repositories. Custom cache keys allow tailored strategies, while robust invalidation mechanisms maintain cache integrity. Developers benefit from seamless integration, scalability, and cross-platform compatibility, enabling efficient software delivery and innovation.

1. **Workspaces:**

Workspaces in Circle CI are like shared folders where teams can put files and folders they need for different parts of their project. This makes it easy to move stuff around between different steps in the development process. It ensures everyone has access to the same files, like project settings or test results. You can control who sees what, and Circle CI cleans up after itself, so things stay tidy, and your computer doesn't get overloaded.

1. **Docker layer caching:**

In Circle CI, Docker layer caching speeds up the building of Docker images by storing previously built layers. This means that when you build a new Docker image, Circle CI only rebuilds the layers that have changed, instead of starting from scratch each time. Docker layer caching saves time and resources, making your builds faster and more efficient. It's especially useful for projects with complex Docker images or frequent builds.

1. **Dynamic configuration:**

Dynamic configuration in Circle CI allows for flexible and adaptable setup of your CI/CD pipeline using environment variables and other configuration options. This feature enables you to customize your pipeline based on varying conditions or requirements, making it easy to adjust settings as needed without modifying your code. With dynamic configuration, you can handle different environments, deployment strategies, or integration scenarios seamlessly. Circle CI empowers developers to create versatile pipelines that can adapt to changes in their projects or workflows, ultimately enhancing productivity and deployment efficiency.

1. **Execution environments:**

In Circle CI, execution environments play a crucial role in defining the runtime settings for executing jobs within your CI/CD pipeline. With execution environments, you can specify the platform, operating system, and dependencies needed for each job to run successfully. This flexibility enables you to tailor the execution environment to match your project's requirements, whether it's a specific programming language, framework, or third-party tool. Circle CI supports various environments, including Docker, machine, Windows, and MacOS, allowing you to choose the most suitable environment for your jobs.

1. **Jobs:**

In Circle CI, jobs are the fundamental units of work within a configuration, serving as the building blocks of your CI/CD pipeline. Each job comprises a collection of steps, which are executed sequentially to accomplish specific tasks such as building, testing, or deploying your application. Circle CI supports different executors including Docker, machine, Windows, and MacOS, allowing you to choose the appropriate environment for your job's requirements.

1. **Parallelism:**

In Circle CI, parallelism refers to the capability of executing multiple jobs concurrently within a workflow, thereby accelerating the overall build process. By leveraging parallel execution, Circle CI optimizes resource utilization and reduces build times, especially for projects with complex configurations or extensive test suites. Parallelism enables jobs that are independent of each other to run simultaneously, maximizing efficiency and throughput.

**GIT Lab**

GitLab is a web-based DevOps platform that provides a comprehensive set of tools for managing the software development lifecycle. It offers features for source code management, continuous integration, and delivery (CI/CD), issue tracking, collaboration, and more, all integrated into a single platform. GitLab is used by software development teams to streamline their workflows, improve collaboration among team members, track project progress, and automate the software delivery process. With GitLab, teams can efficiently manage code repositories, automate testing and deployment tasks, track issues and bugs, and foster collaboration through code reviews and discussions. Overall, GitLab serves as a centralized hub for all aspects of software development, helping teams deliver high-quality software products faster and more efficiently.

****

**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.

1. **Design Management:**

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

Design Management encompasses two broad areas of features:

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

1. **Quality Management:**

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

1. **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.

1. **Pages:**

GitLab Pages allows users to create static websites for their projects, serving as an additional documentation or showcase platform. This expansion comes in response to requests from the broader community to open-source GitLab Pages, 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.

1. **Code Review Workflow:**

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.

1. **GitLab CLI:**

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. 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.

1. **Web IDE:**

GitLab includes a Web IDE, allowing users to edit code directly in the web browser without needing a local development environment. The Web IDE offers features for validating syntax and semantics, ensuring correct configuration setup. It also streamlines the feedback 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.

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

Here's an investigation and evaluation of Jenkins, TeamCity, Circle CI, 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 flexibility with advanced features and customization options. | Known for cloud-native flexibility. Provides platform-agnostic environment. | Integrated flexibility within 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 builds on cloud platforms. | Supports automation of workflows through configuration files and custom scripts. | 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.

Investigation & Evaluation of 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

Advantages of Jenkins:

Open Source and Free: Jenkins is freely available and open-source, making it accessible to developers without any cost concerns. It is also platform-independent, allowing users to run it on various operating systems.

Plug-ins and Integration: Jenkins boasts a vast array of plug-ins that enhance its functionality and integration capabilities. Developers can easily extend Jenkins' features through plug-ins, such as dashboard view, test analysis, and build pipeline plug-ins, enabling seamless integration with other tools and platforms.

Hosting Options: Jenkins offers flexibility in hosting options, allowing installation on different operating systems including Windows, MacOS, and Linux. It can also be deployed on the cloud using virtual machines or Docker containers, providing scalability and adaptability to various environments.

Community Support: With over a million users worldwide, Jenkins benefits from strong community support. Developers can rely on a vast community of users for assistance, sharing knowledge, and troubleshooting, ensuring robustness and reliability of the tool.

Integration with Other CI/CD Platforms: Jenkins supports integration with various CI/CD platforms and tools, facilitating interoperability and collaboration across different development environments. Its extensive library of plug-ins enables seamless interaction with other tools and systems.

Disadvantages of Jenkins:

Compatibility Issues with Declarative Pipeline Syntax: Some plug-ins may not be fully compatible with Jenkins' declarative pipeline syntax, leading to compatibility issues and limitations in functionality.

Maintenance of Plug-ins: While Jenkins offers a wide range of plug-ins, not all of them are regularly maintained by the developer team. This lack of maintenance can lead to issues with updates and compatibility.

Problems with Plug-in Updates: Updating plug-ins in Jenkins can sometimes be problematic, with users facing issues such as compatibility issues or plugin dependency conflicts.

Dependency on Plug-ins: Jenkins heavily relies on plug-ins for its functionality, sometimes requiring users to install additional plug-ins even for basic features. This dependency can lead to complexity and overhead in managing plug-ins.

User Interface (UI) Challenges: Jenkins' user interface may not be as intuitive or user-friendly as modern UI technologies. For first-time users, navigating the UI may be challenging, impacting usability and adoption rates.

Overall, while Jenkins offers a robust and versatile platform for continuous integration and delivery, users should be aware of its limitations and challenges, particularly regarding plug-in management and user interface usability.

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

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

Investigation & Evaluation of 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)

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

Investigation & Evaluation of 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)

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

Investigation & Evaluation of 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.

# **Task 1 – C**

Based on the evaluation of four continuous integration tools (Jenkins, TeamCity, CircleCI, and GitLab), the following recommendations are made:

\*Tool Recommendation Plan A (First Choice): Jenkins\*

Justification:

1. \*Cost:\* Jenkins is open-source and free to use, making it a cost-effective option for SIS, especially considering its budget constraints as a newly established institute.

2. \*Efficiency:\* Jenkins offers a wide range of plugins and integrations, allowing for extensive customization and automation of CI/CD processes. Its flexibility and scalability make it efficient in handling the growing needs of SIS as it expands.

3. \*User Interface:\* Although Jenkins may have been criticized for its user interface compared to modern tools, it still offers a relatively straightforward interface once users become accustomed to it. With a bit of training, the team at SIS can effectively navigate and utilize Jenkins' features.

4. \*Community Support:\* Jenkins has a large and active community, providing ample resources, documentation, and support forums. This ensures that SIS can easily find assistance and solutions to any issues that may arise during implementation or usage.

Overall, Jenkins is recommended as Plan A due to its cost-effectiveness, efficiency, and strong community support, making it an ideal choice for SIS's continuous integration needs.

\* Tool Recommendation Plan B (Second Choice): TeamCity\*

Justification:

1. \*Reliable Performance:\* TeamCity is known for its reliable performance, meeting expectations without issues, which can contribute to a smooth CI/CD process for SIS.

2. \*User-friendly Interface:\* TeamCity offers a user-friendly interface, which can facilitate easier adoption and usage by the SIS team, potentially reducing the learning curve.

3. \*Excellent Customer Support:\* Users express satisfaction with TeamCity's customer support, indicating that SIS can rely on prompt assistance and guidance if needed during implementation or usage.

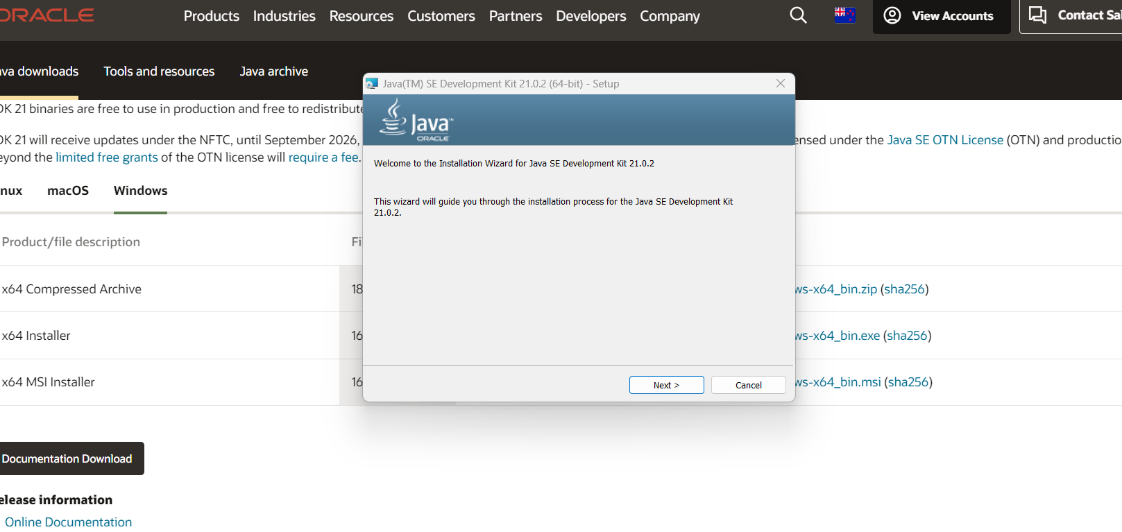
4. \*Strong Integration Capabilities:\* TeamCity's strong integration capabilities can ensure seamless integration with SIS's existing tools and workflows, enhancing overall efficiency.

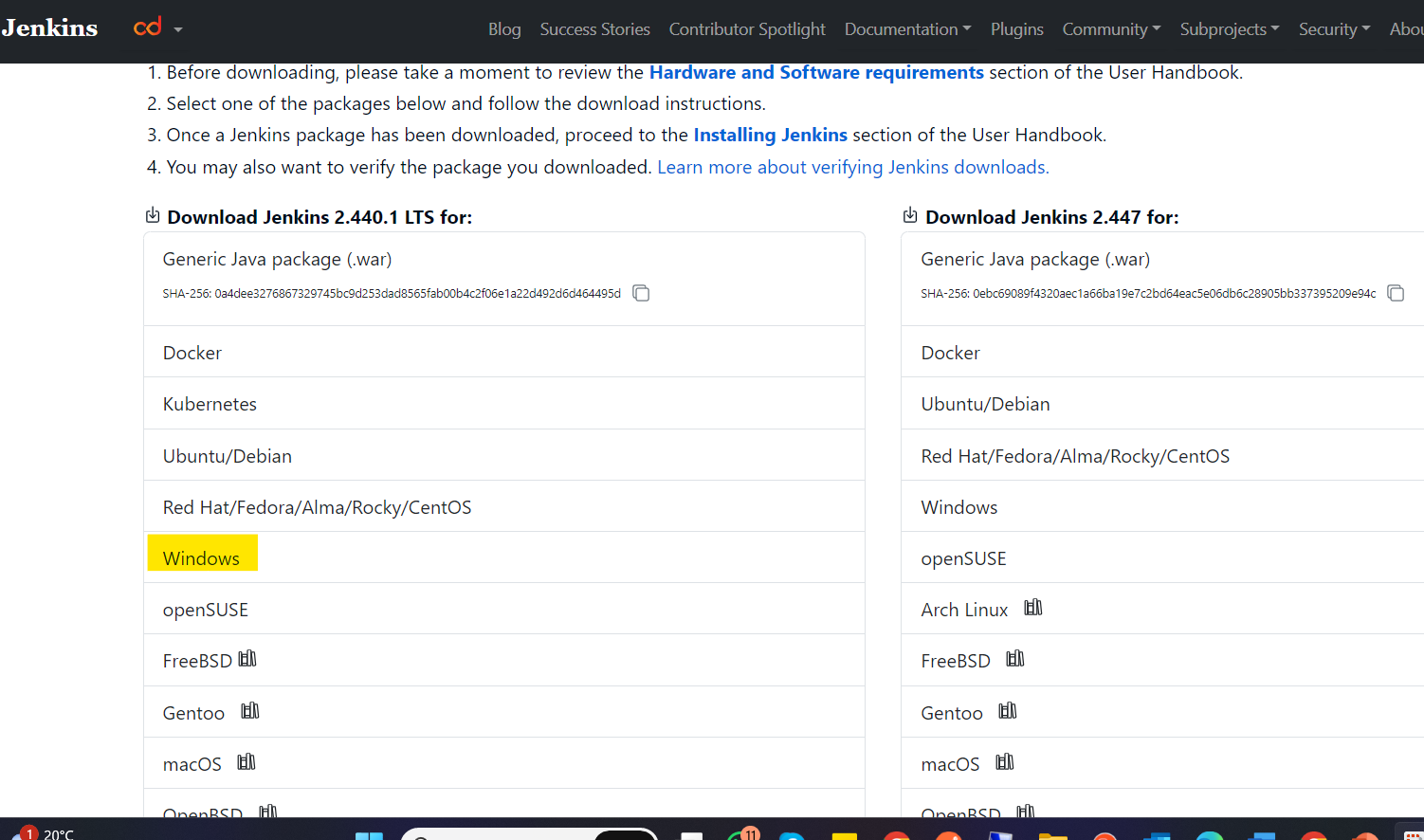
Although TeamCity may have some limitations such as a lengthy upgrade process and weaknesses in pipeline visualization, it still presents a solid option as Plan B due to its reliable performance, user-friendly interface, excellent customer support, and strong integration capabilities, offering an alternative to Jenkins for SIS's continuous integration needs.

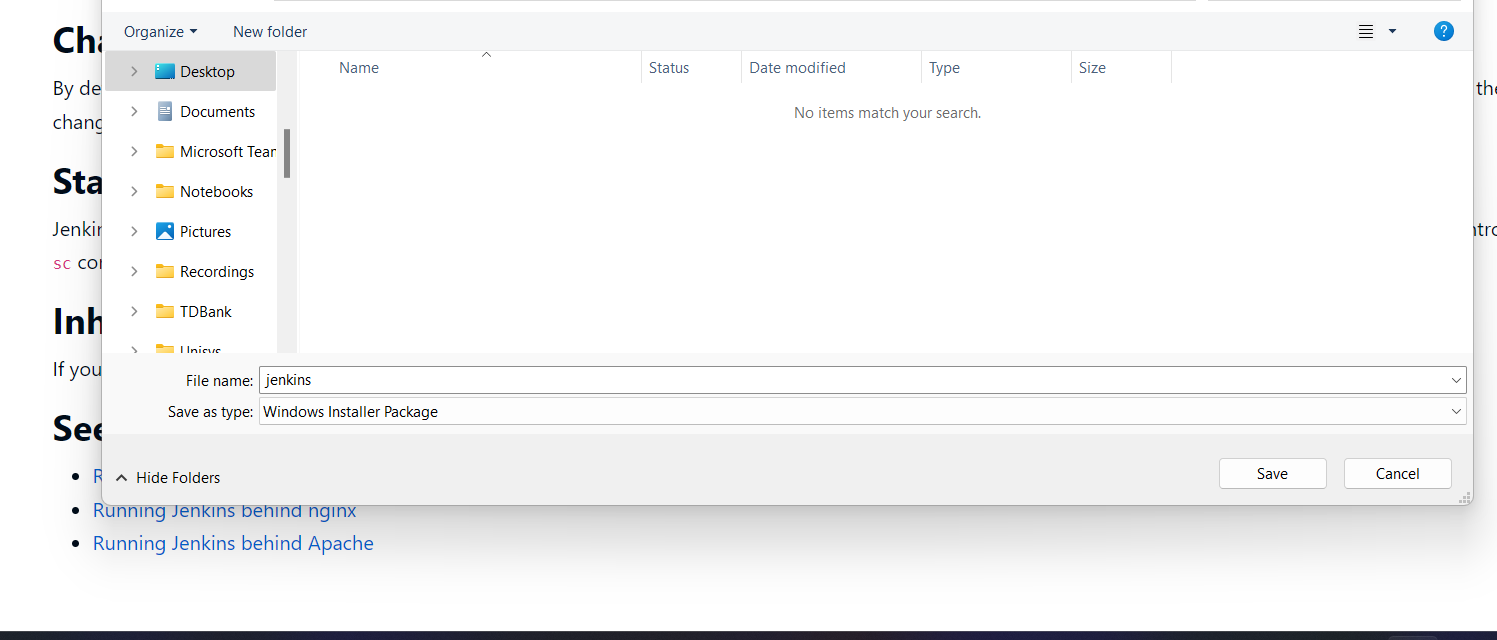
# **Task 2 – A**

# **Jenkins Installation:**

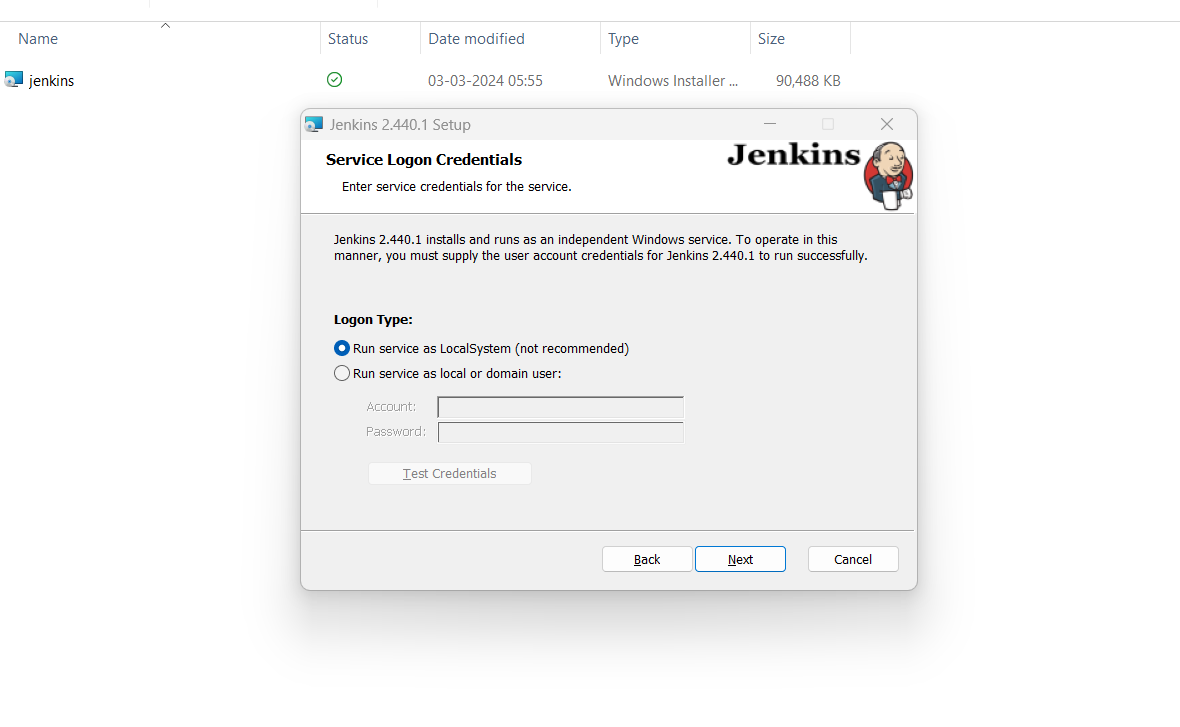
Pre-Requisite

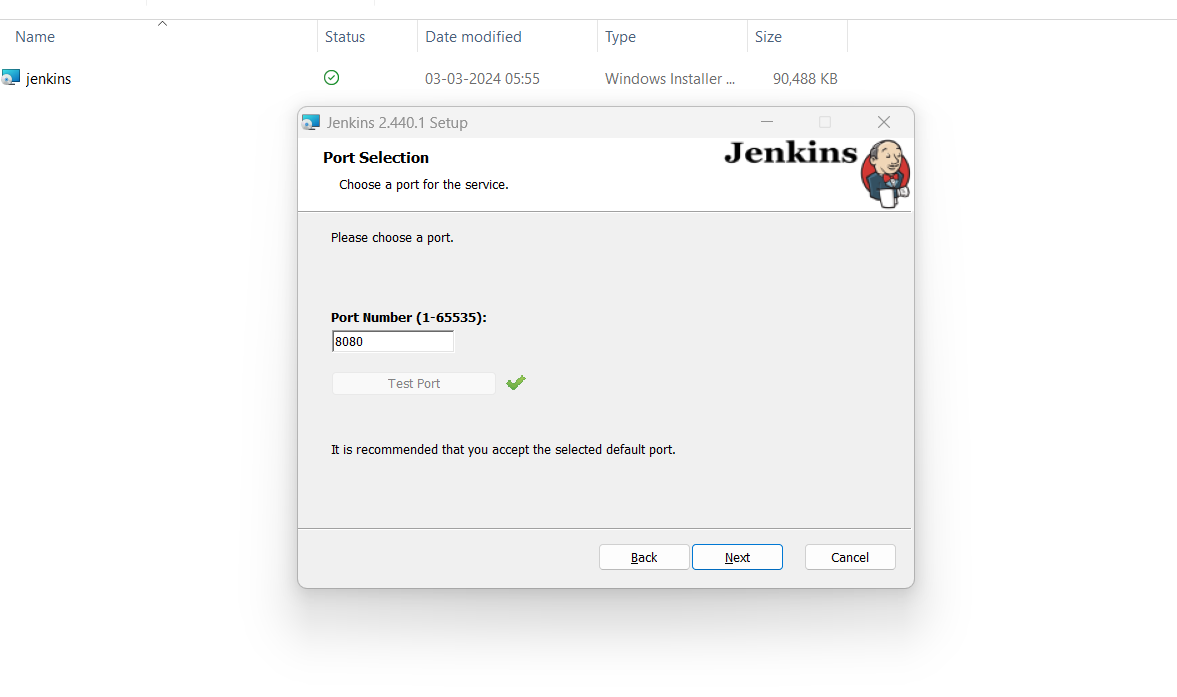


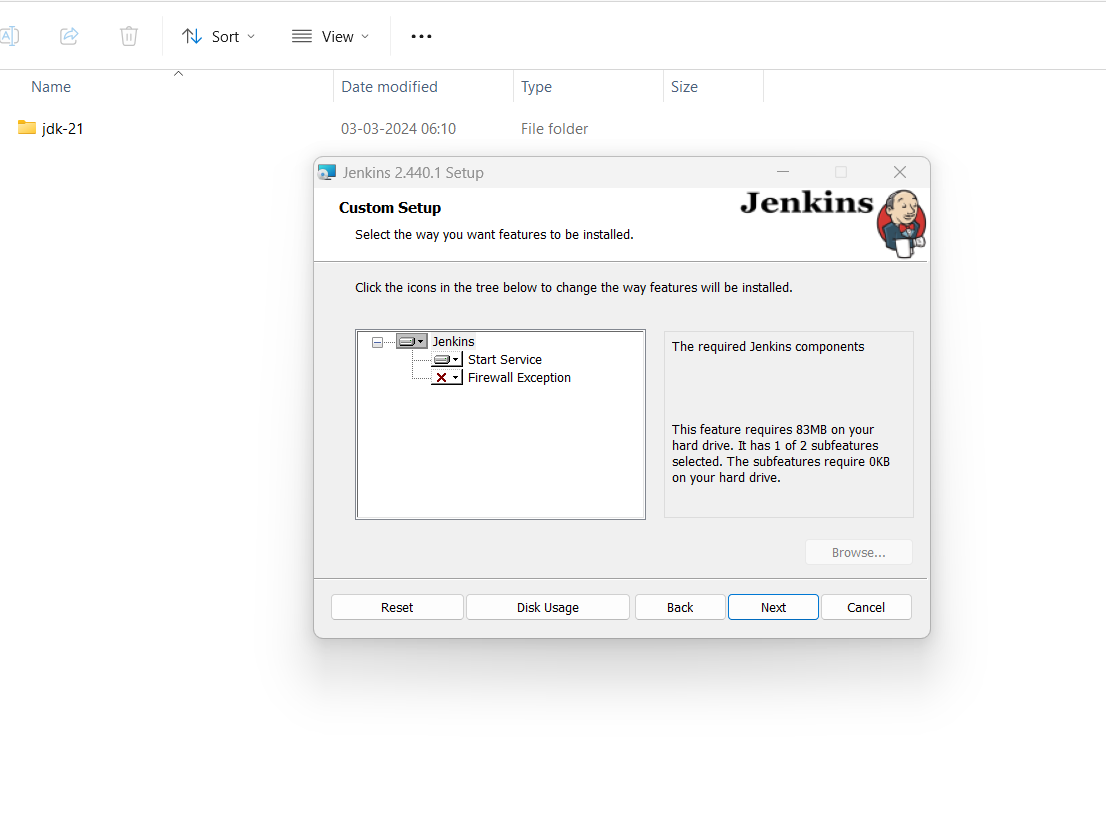


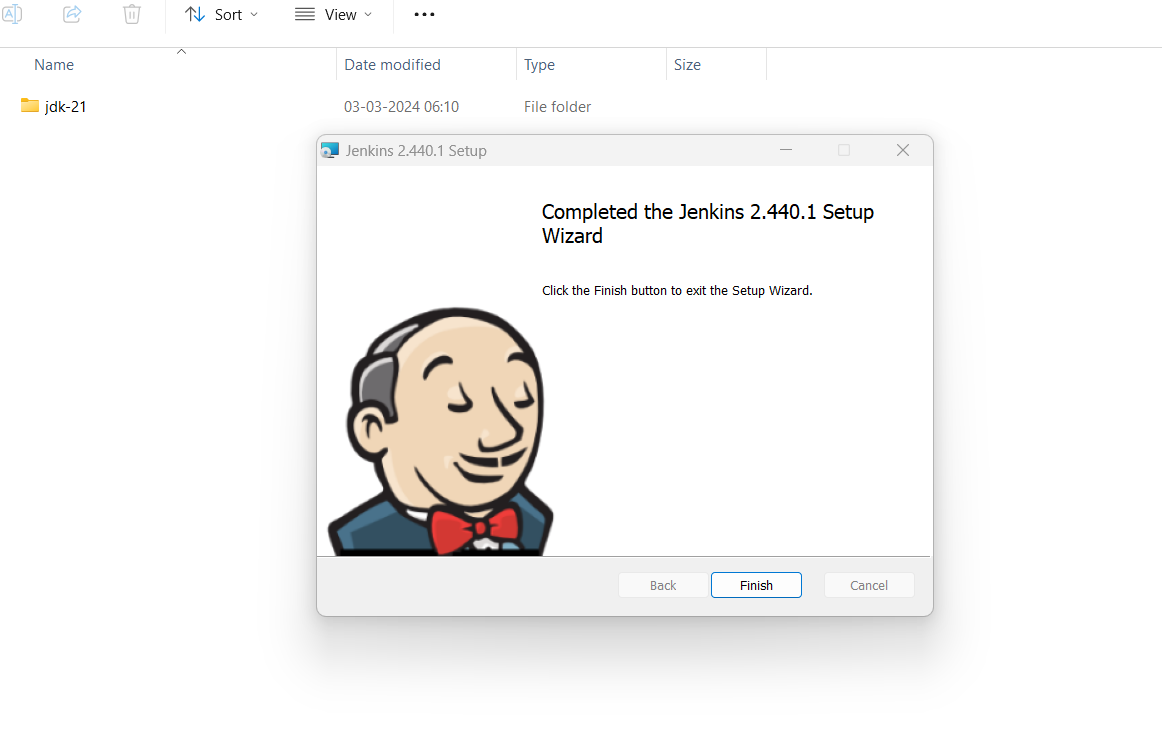


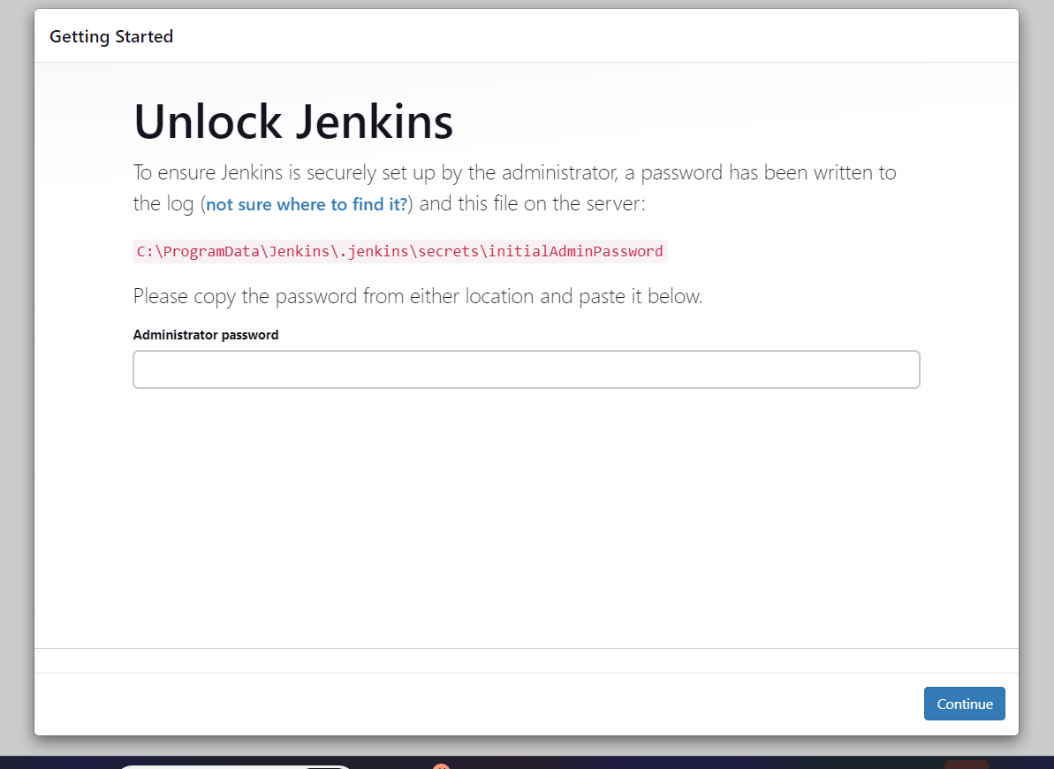


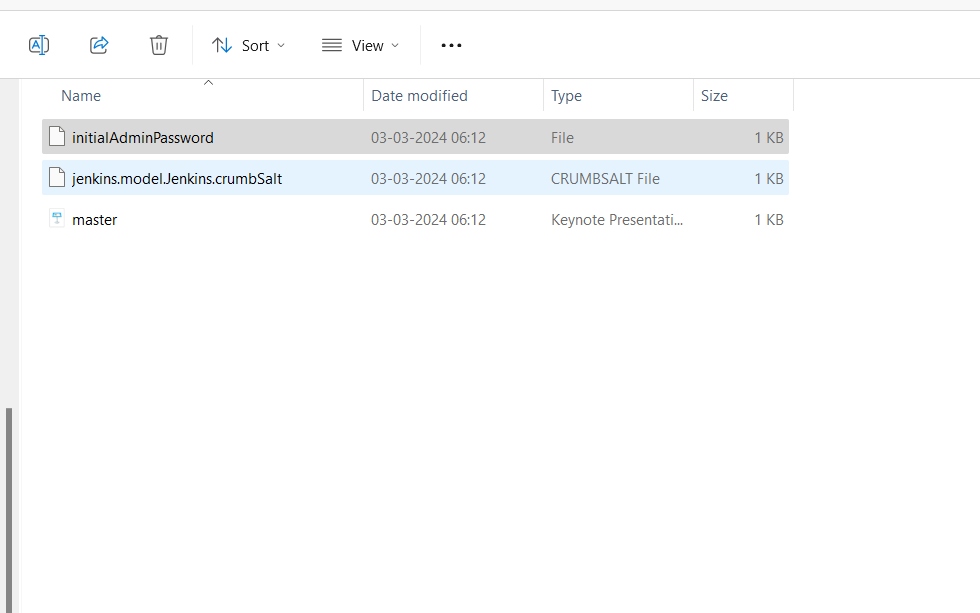


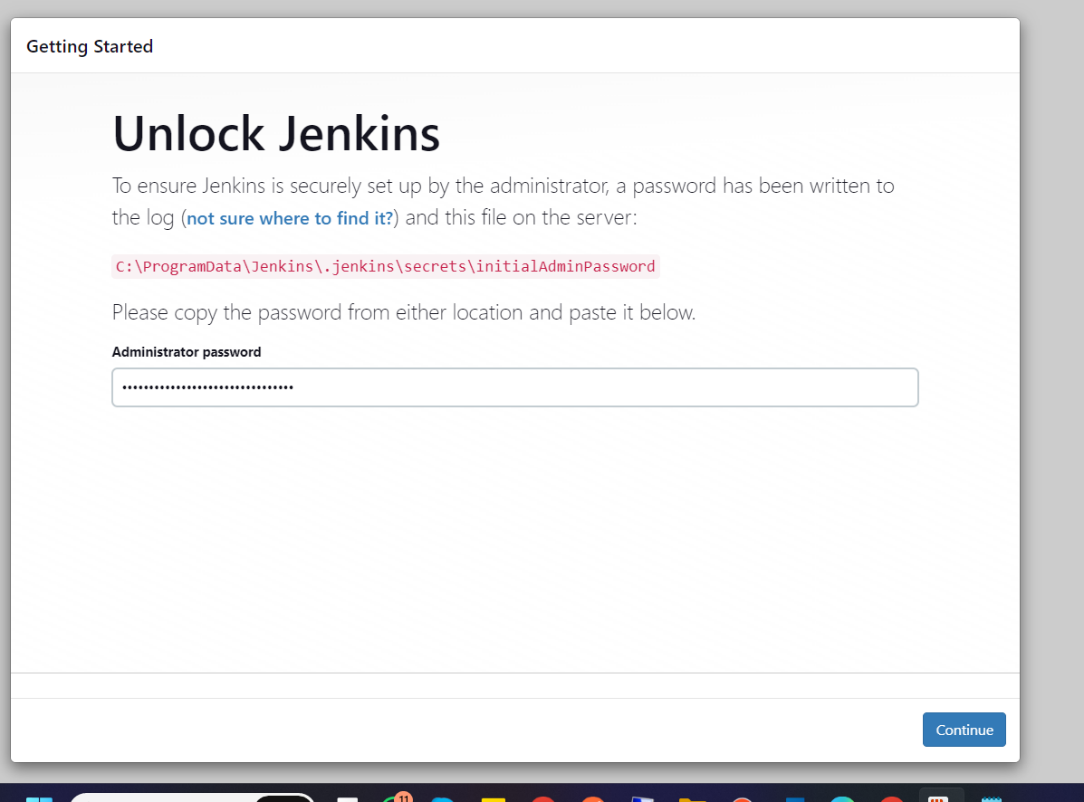


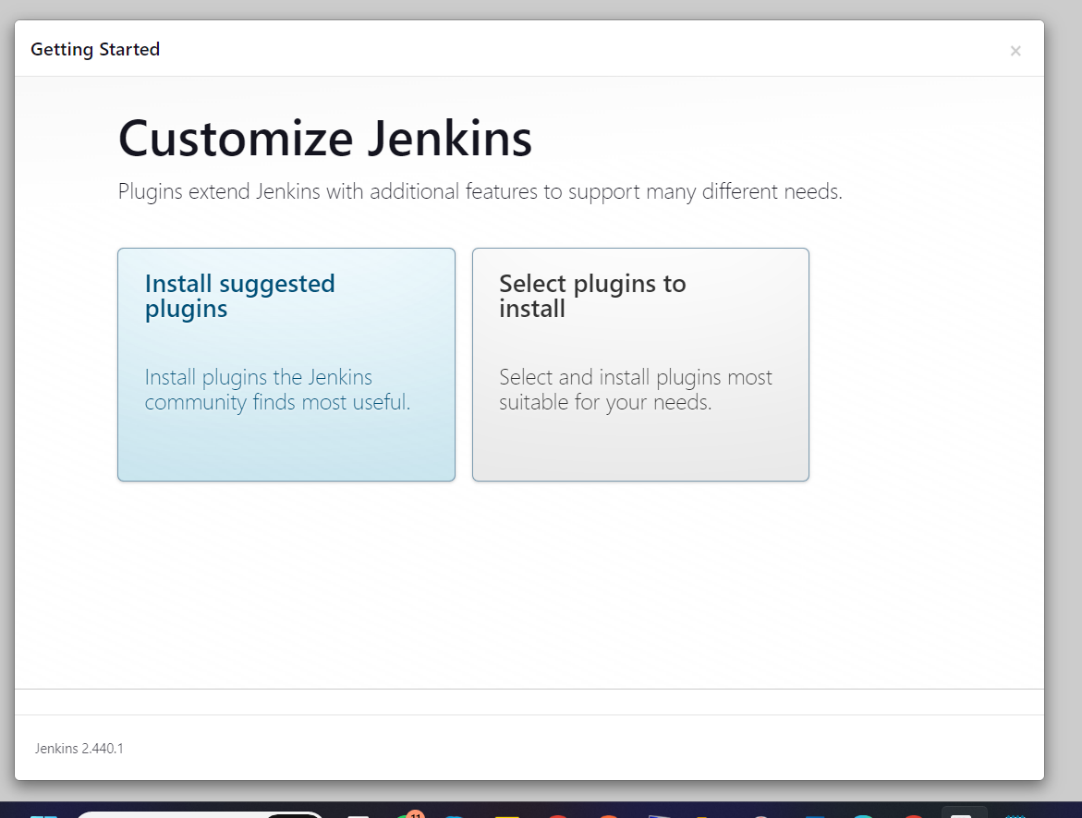


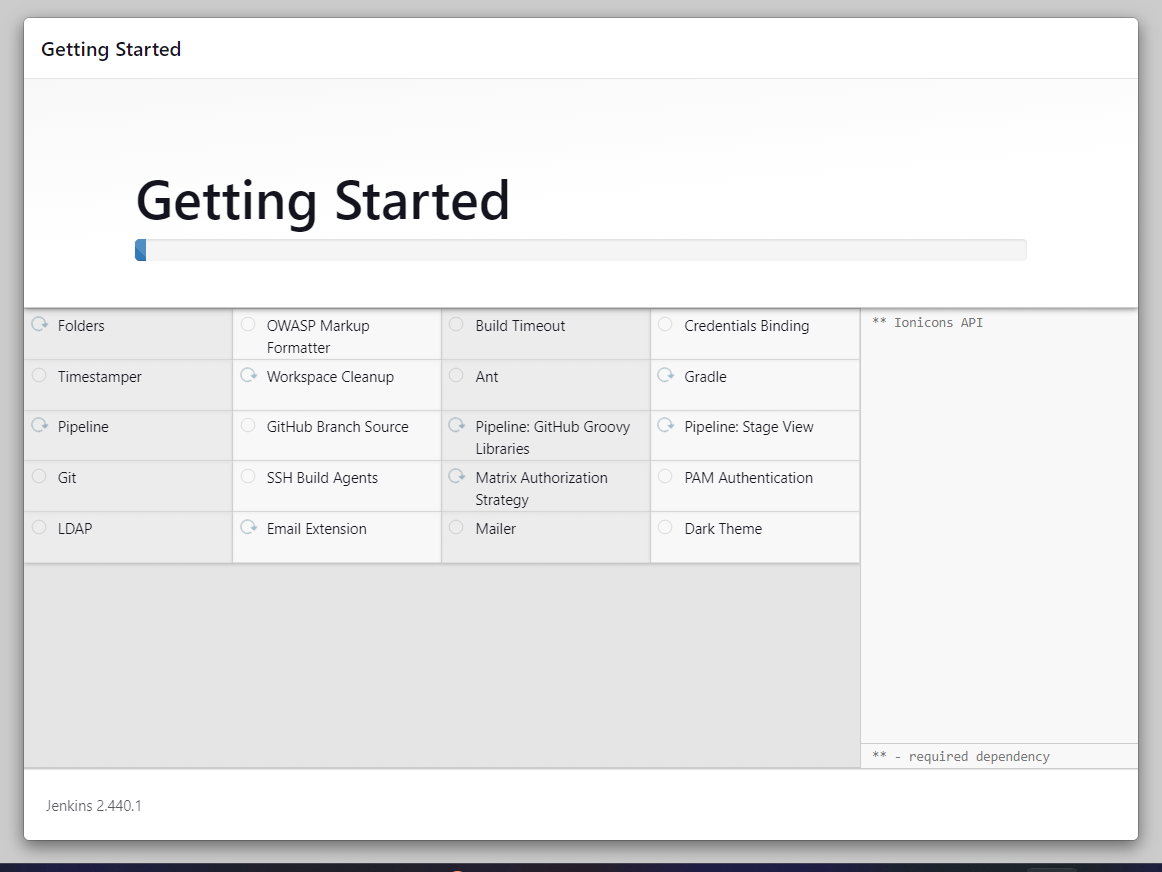


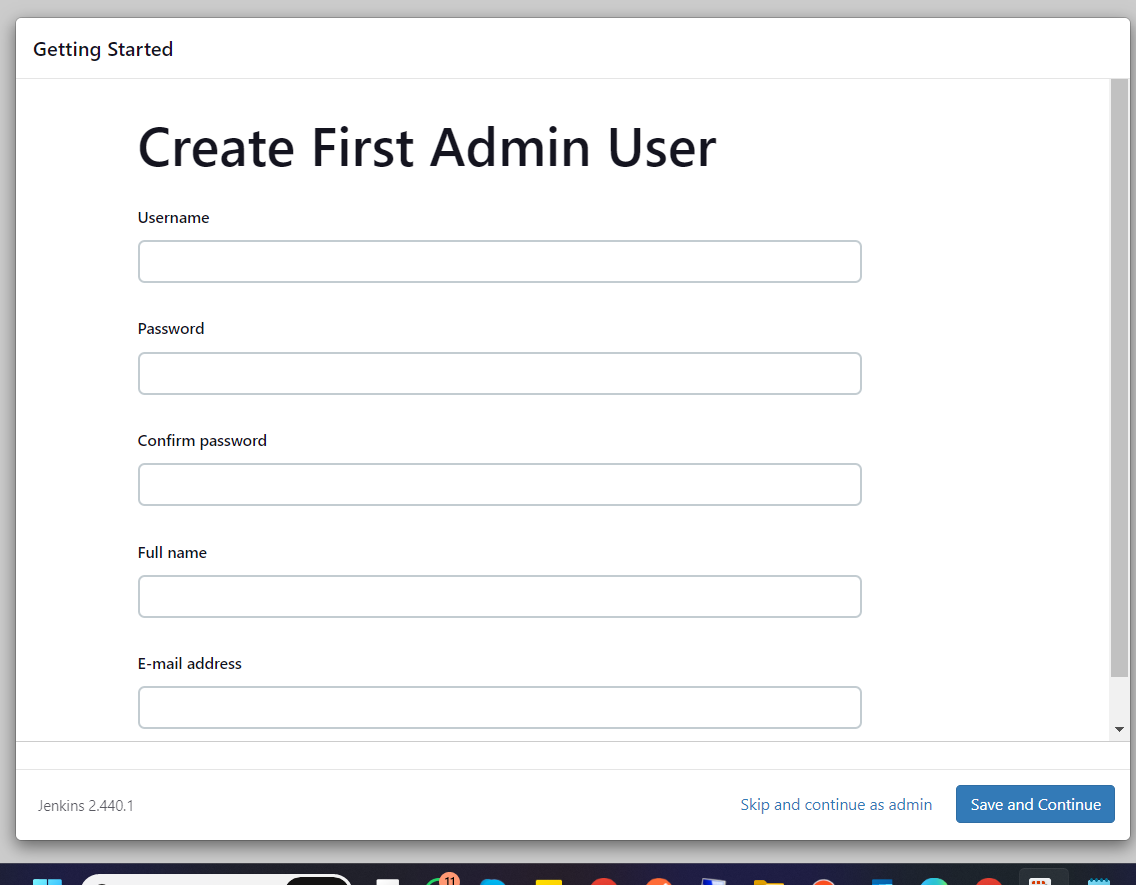


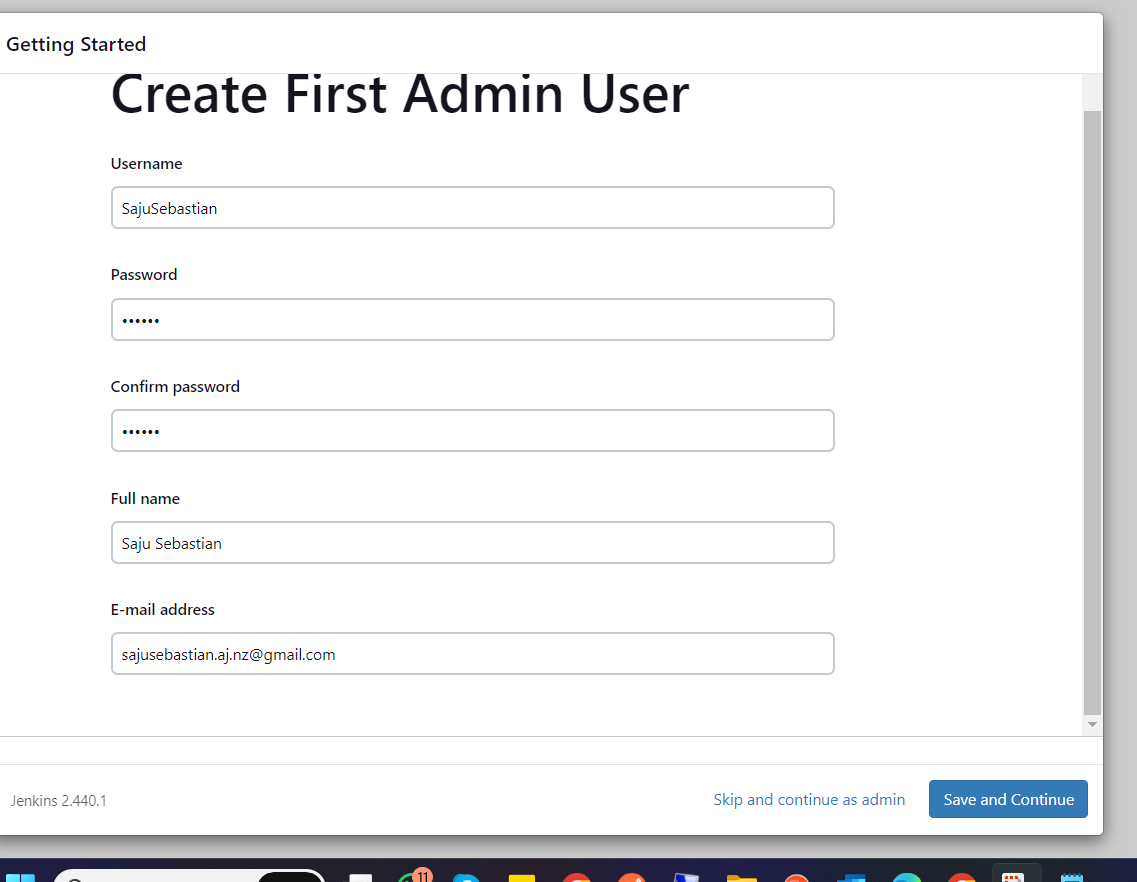


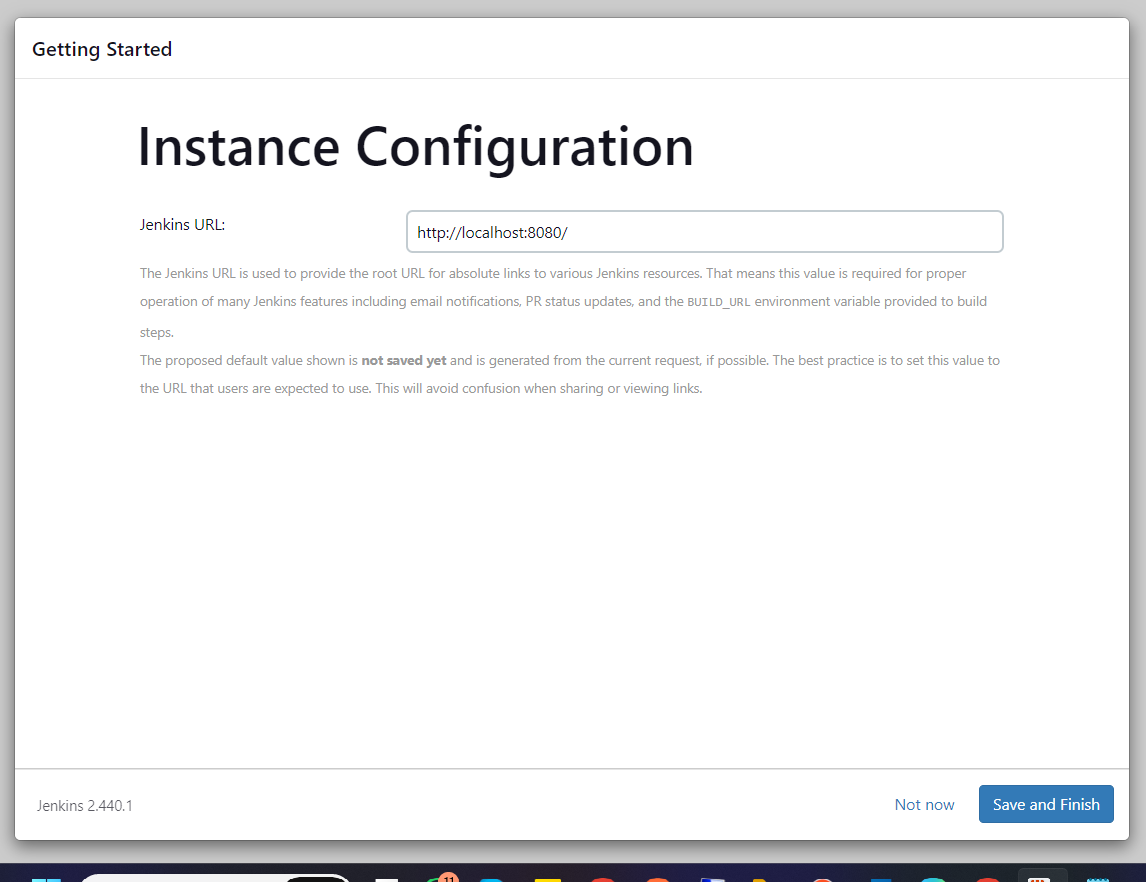


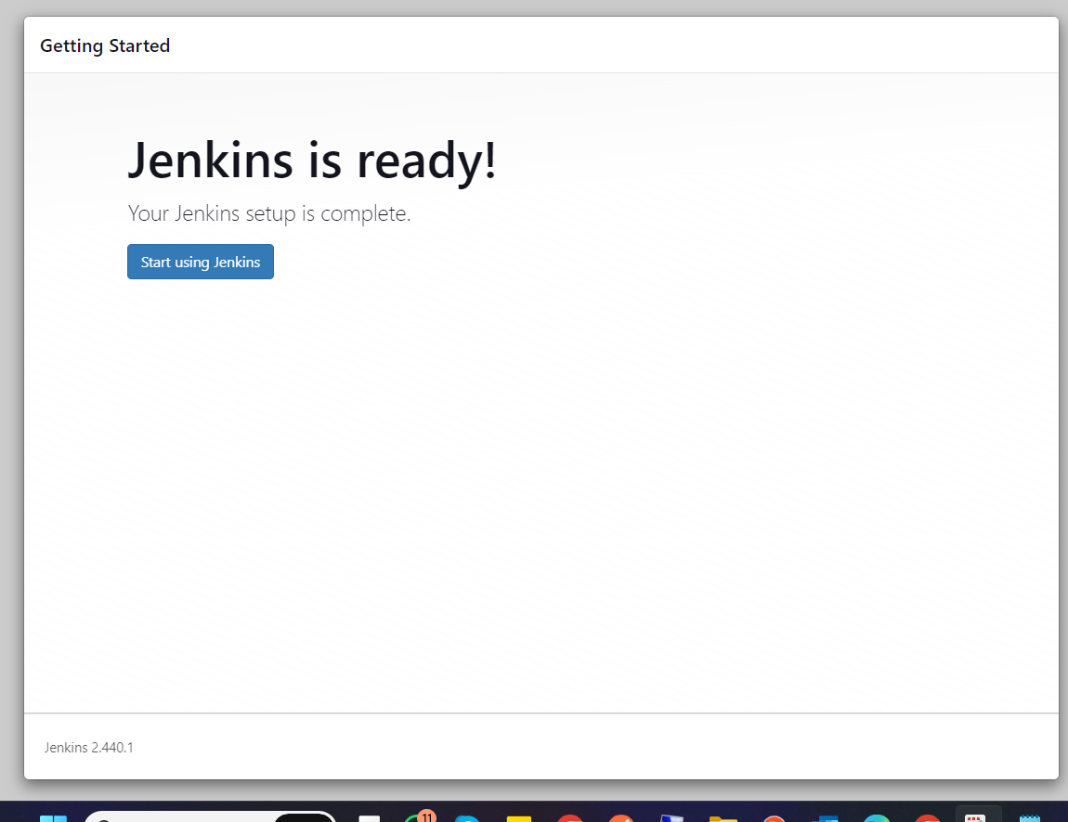


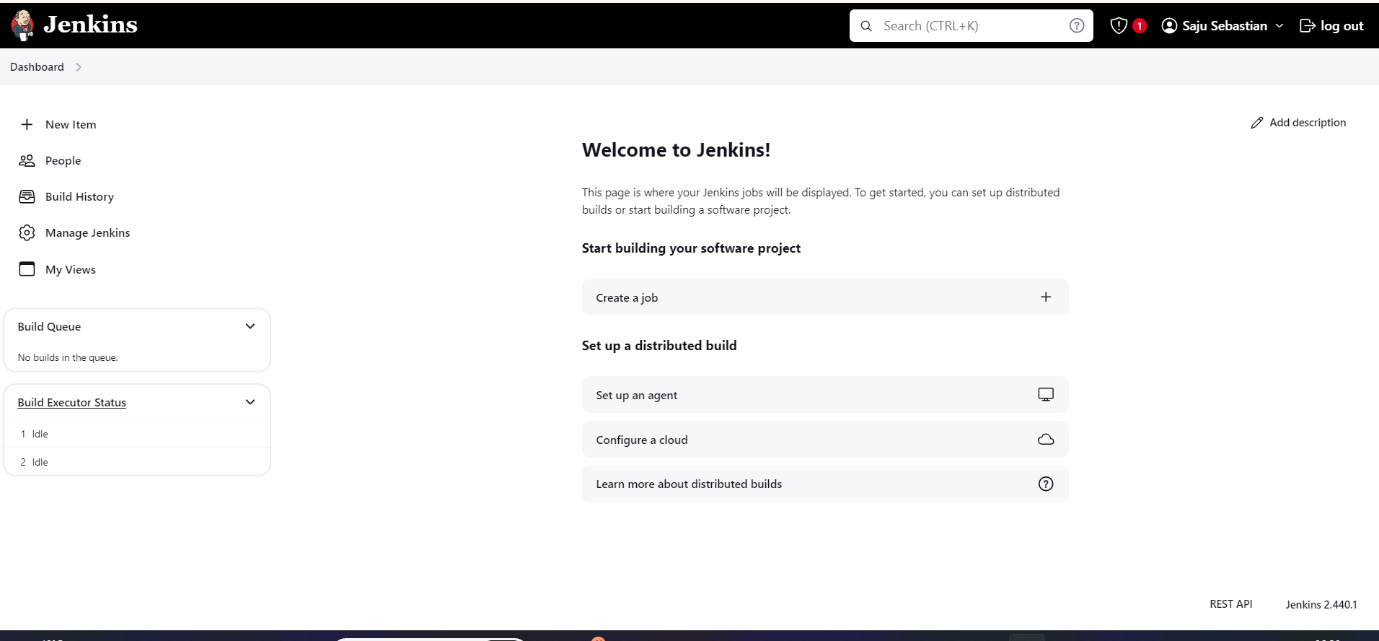








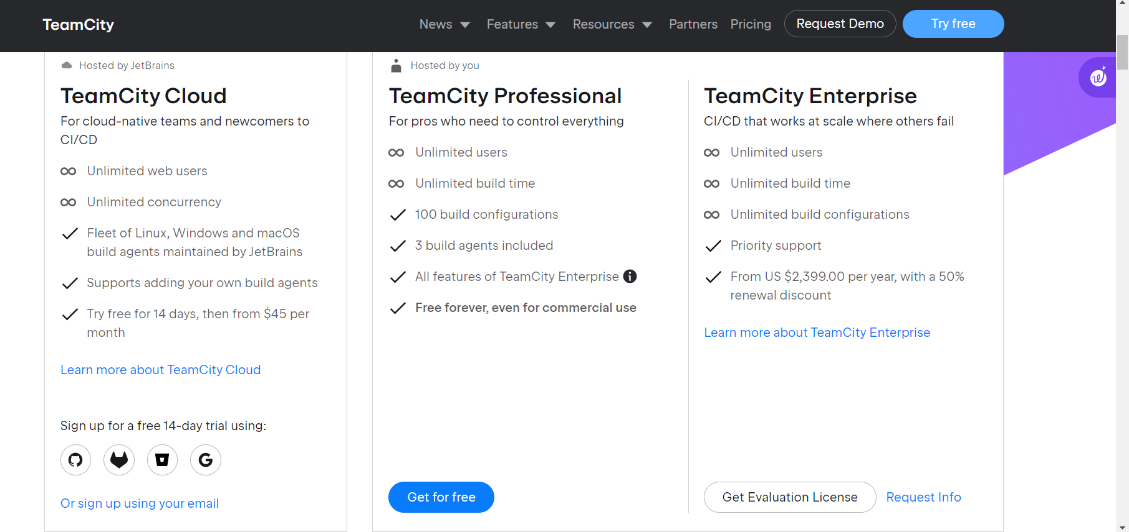


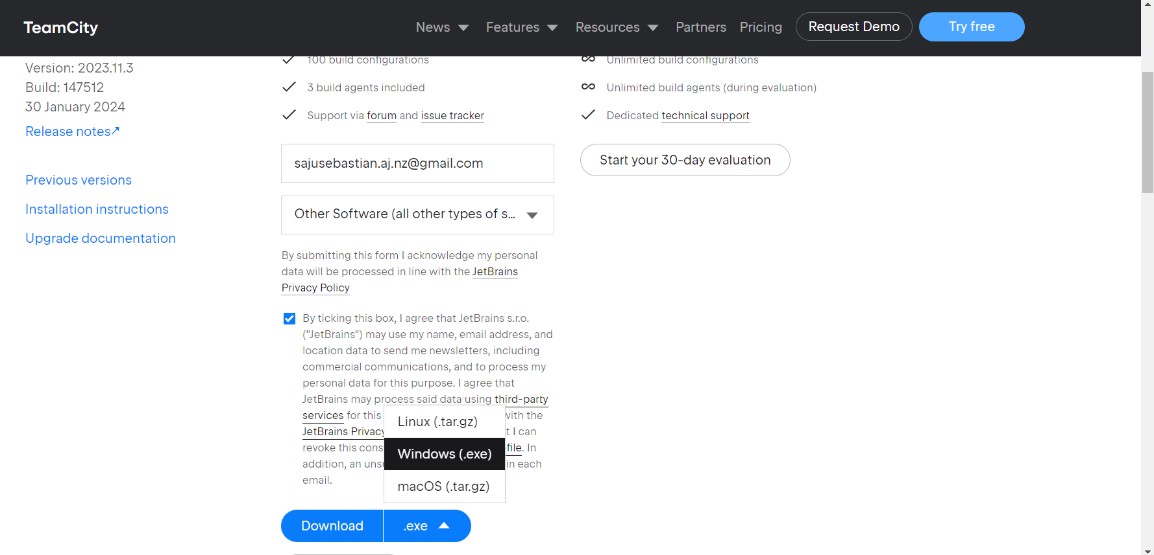


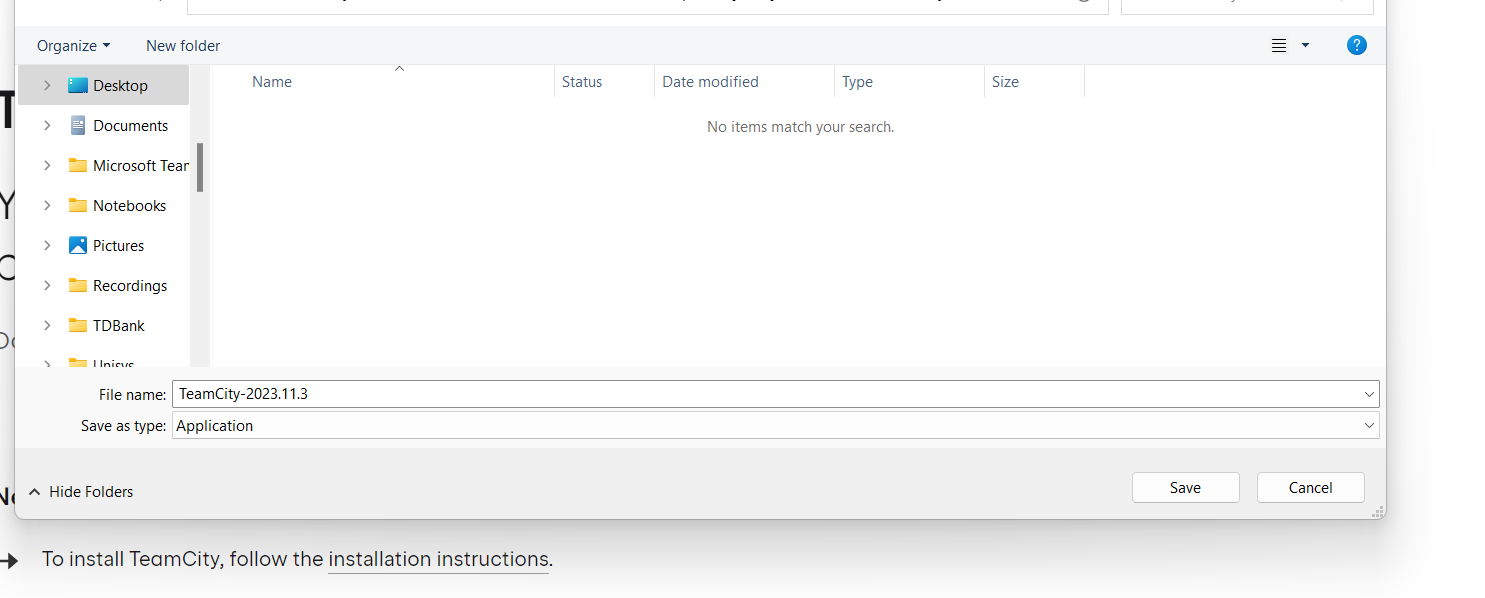
# **TeamCity Installation:**

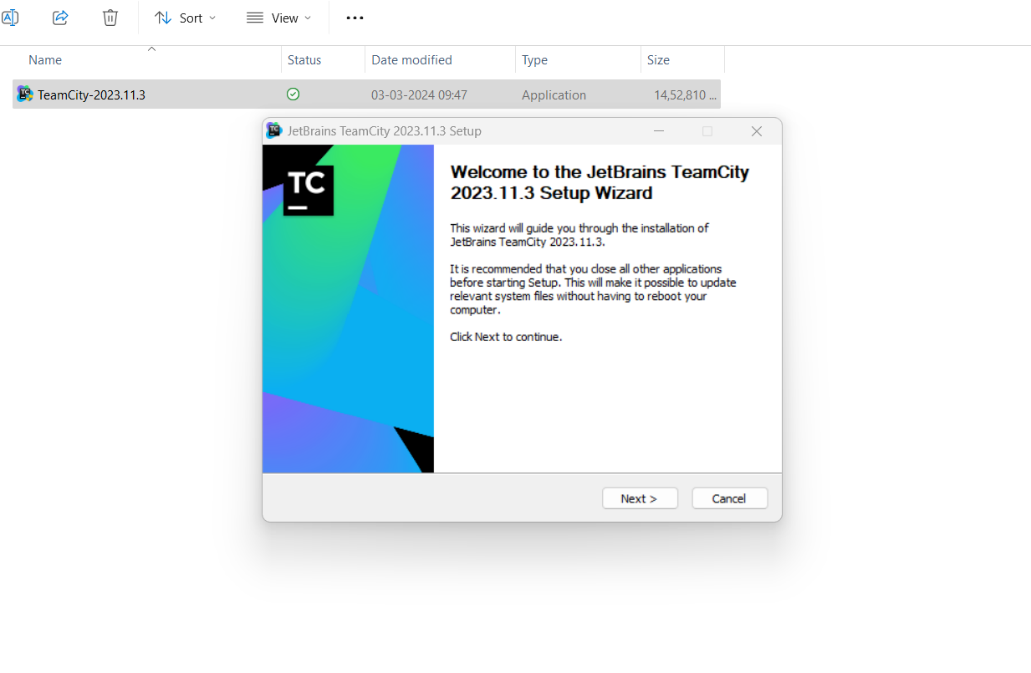
A screenshot of a computer

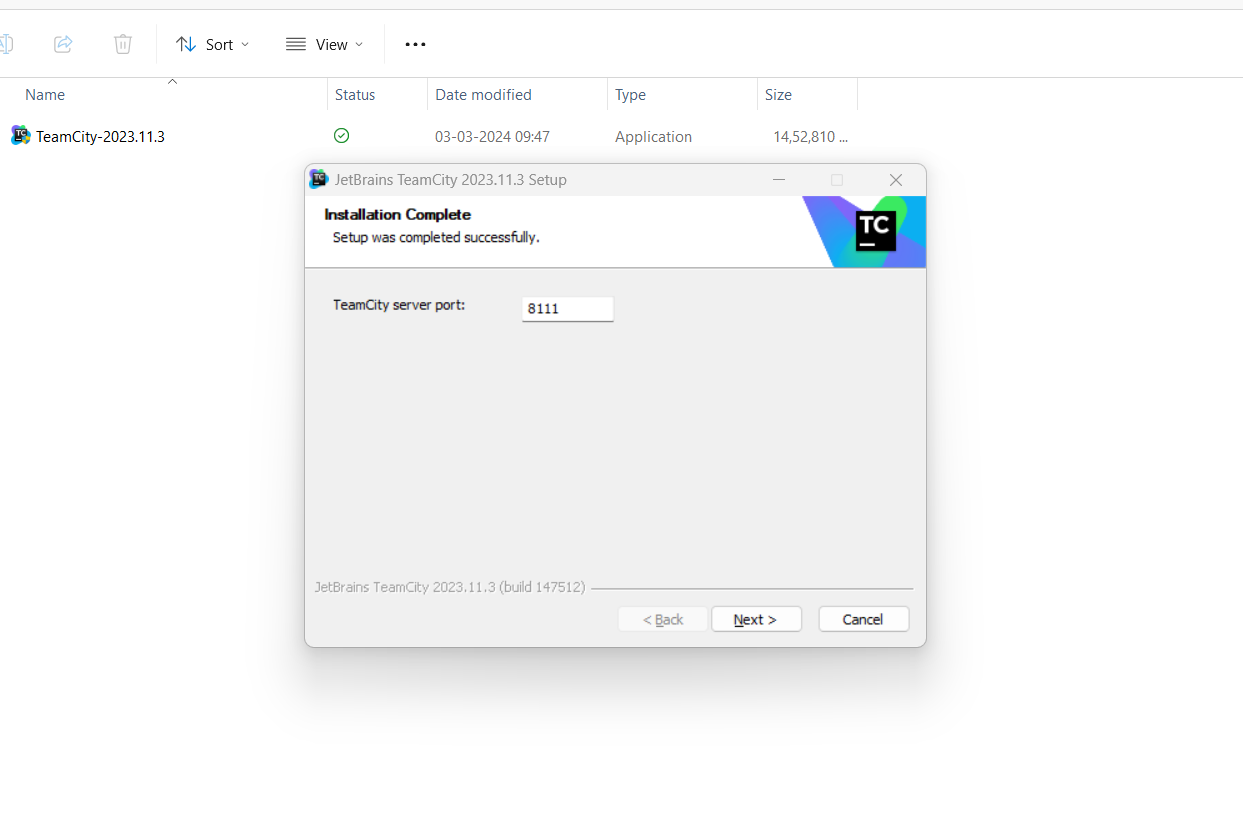
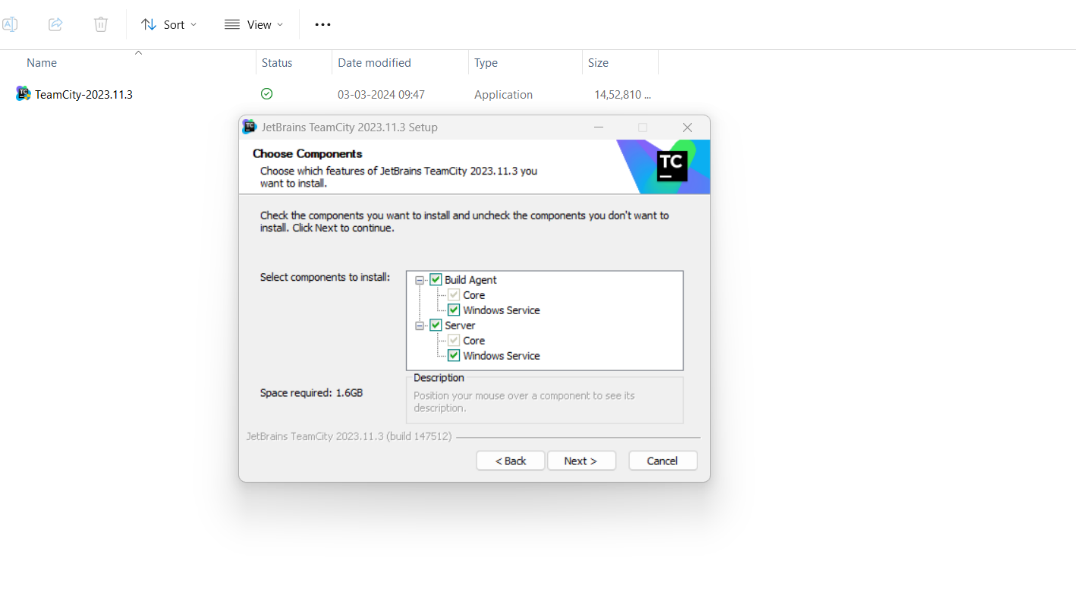
Description automatically generated

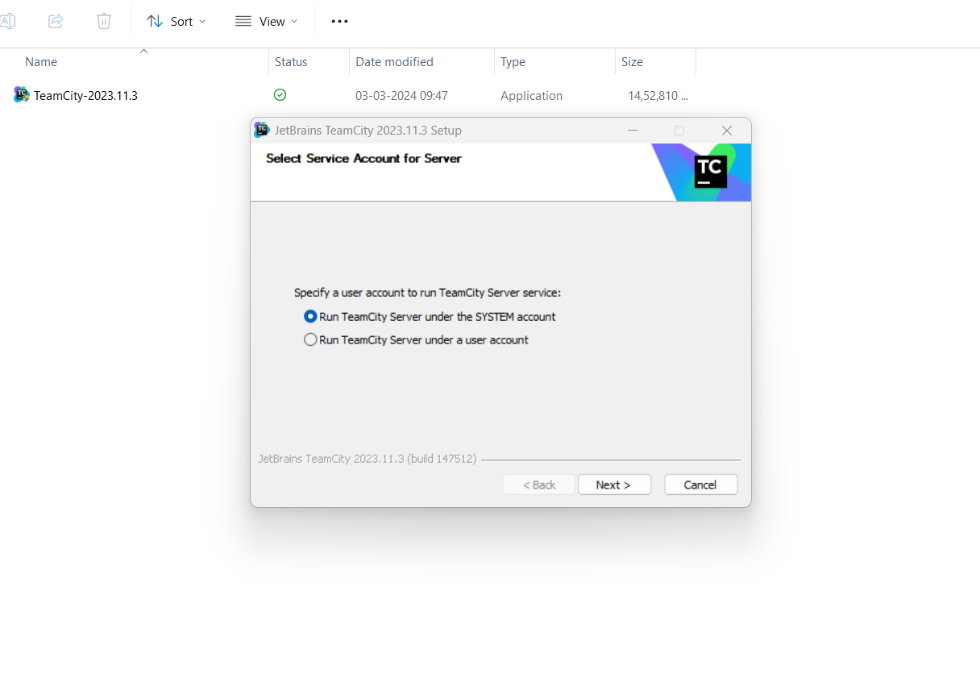
****

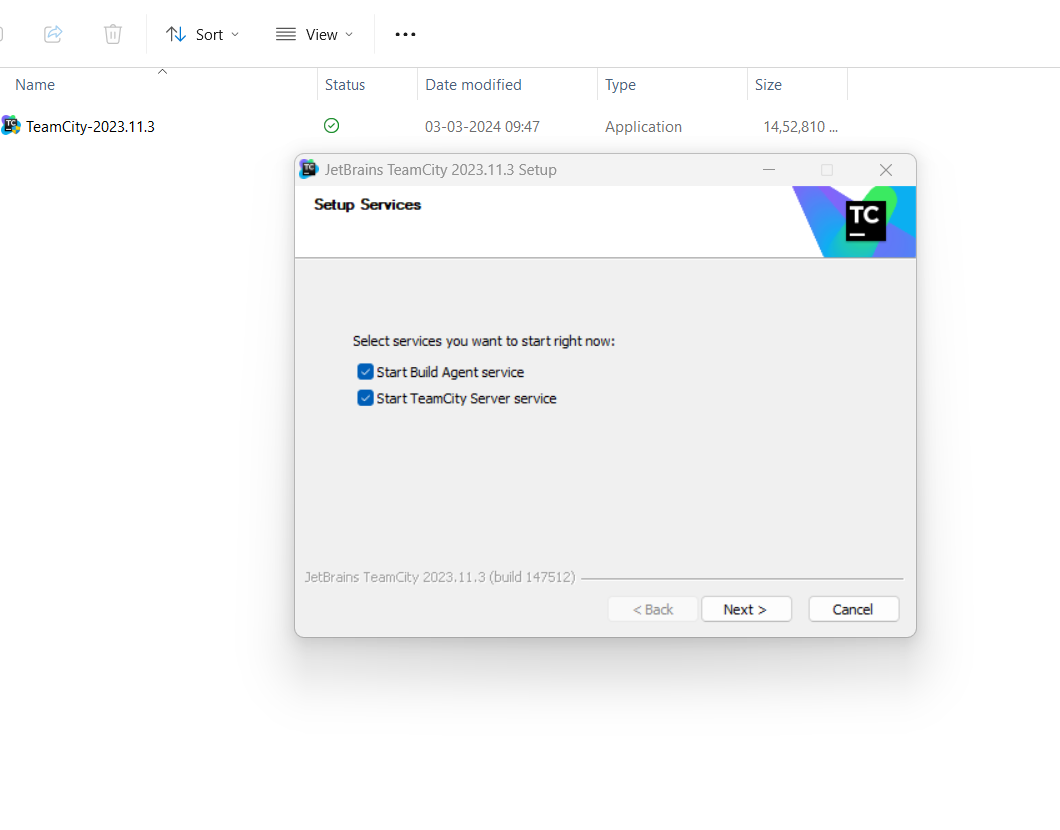
****

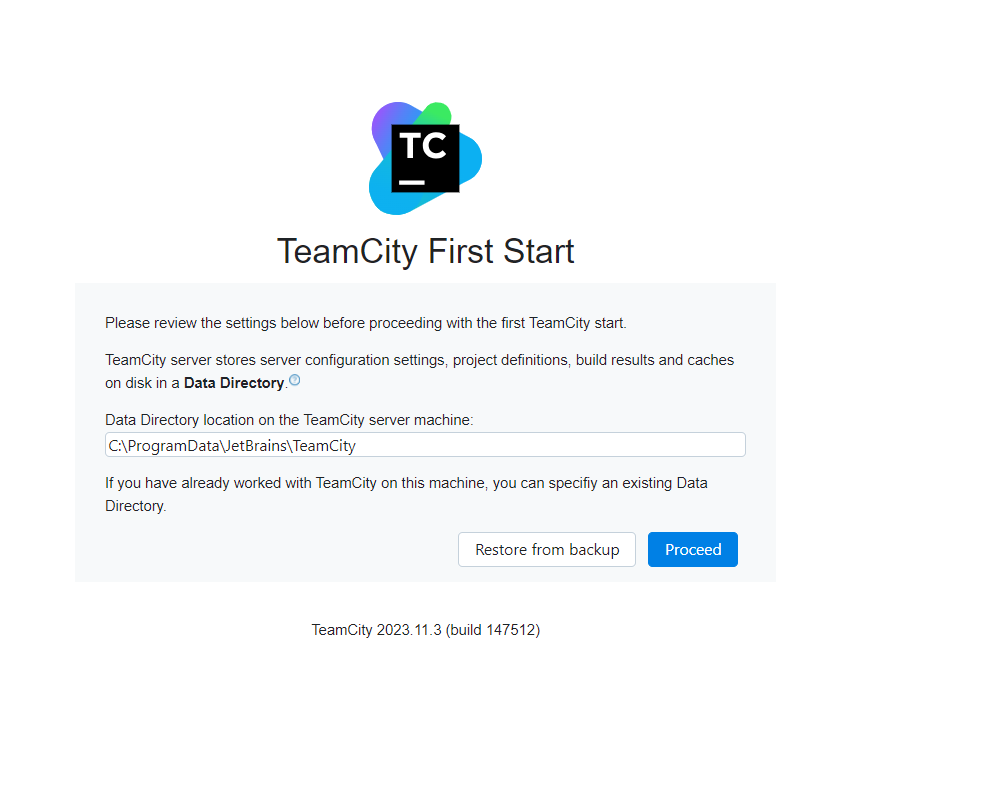
****

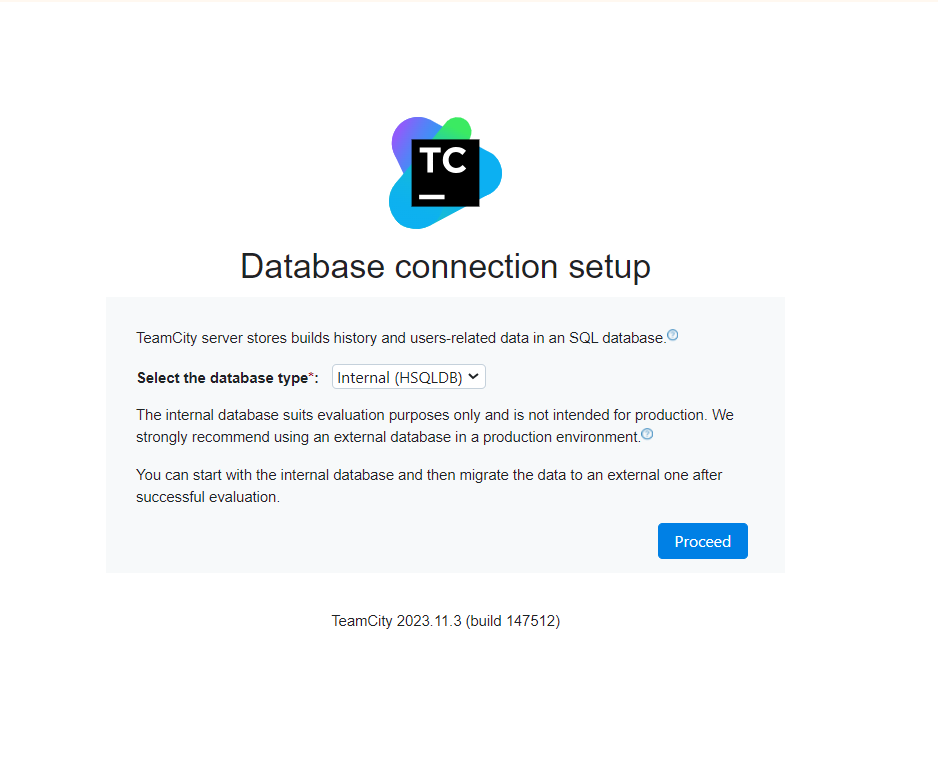
****

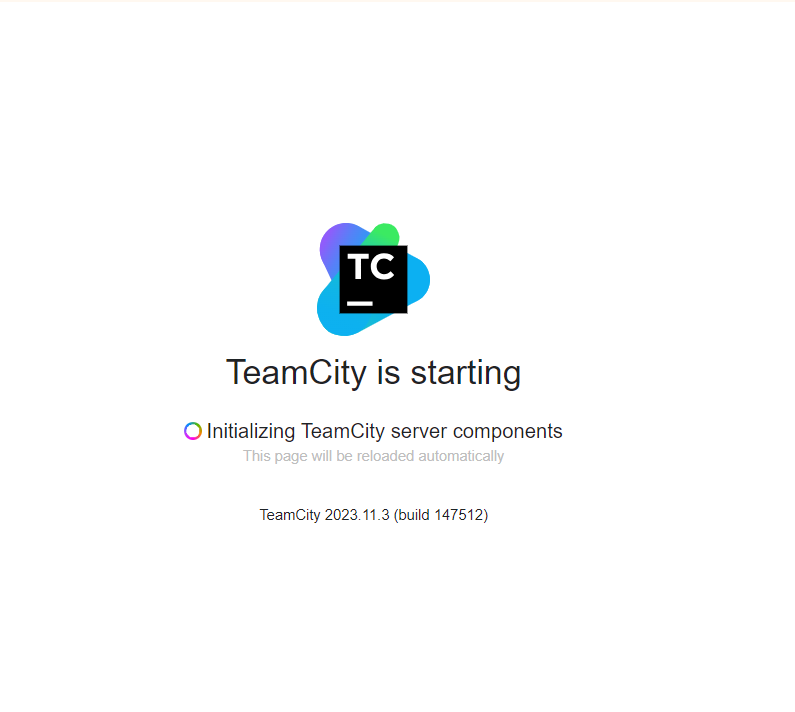
****

****

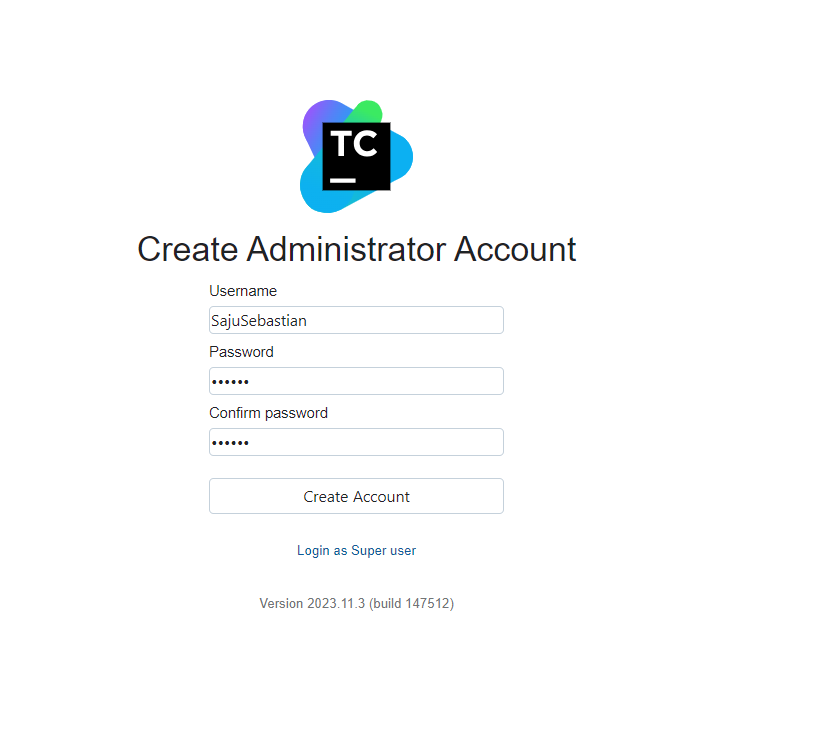
****

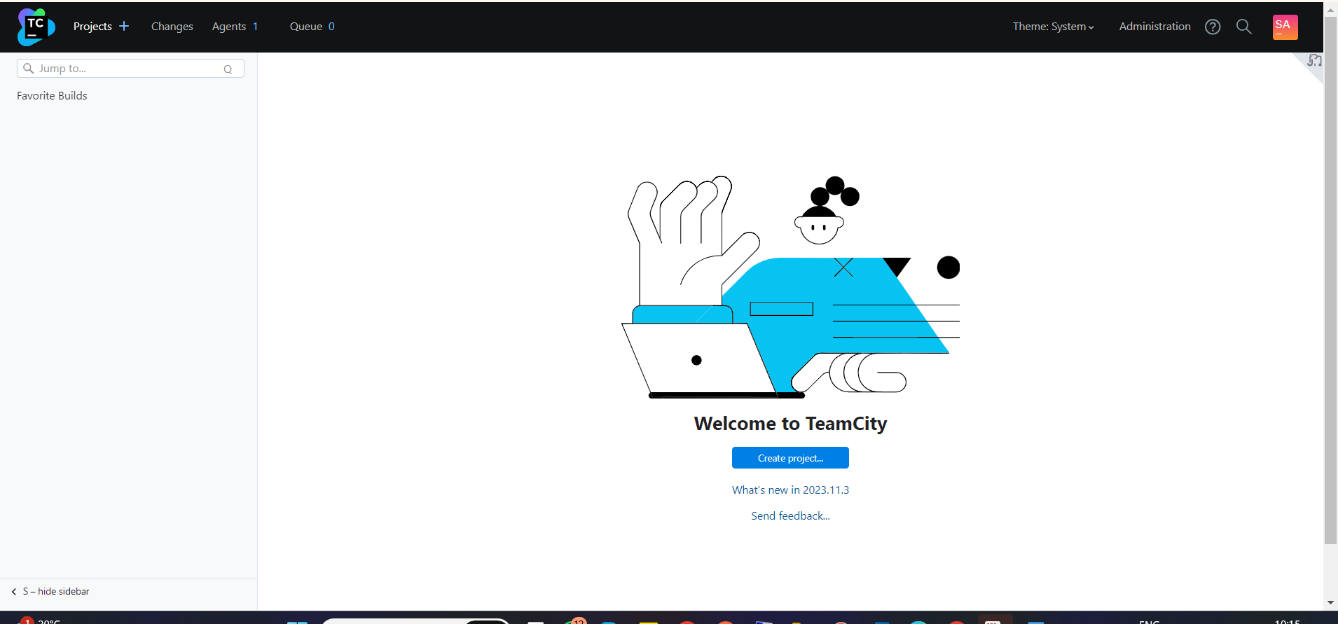
****

****

****

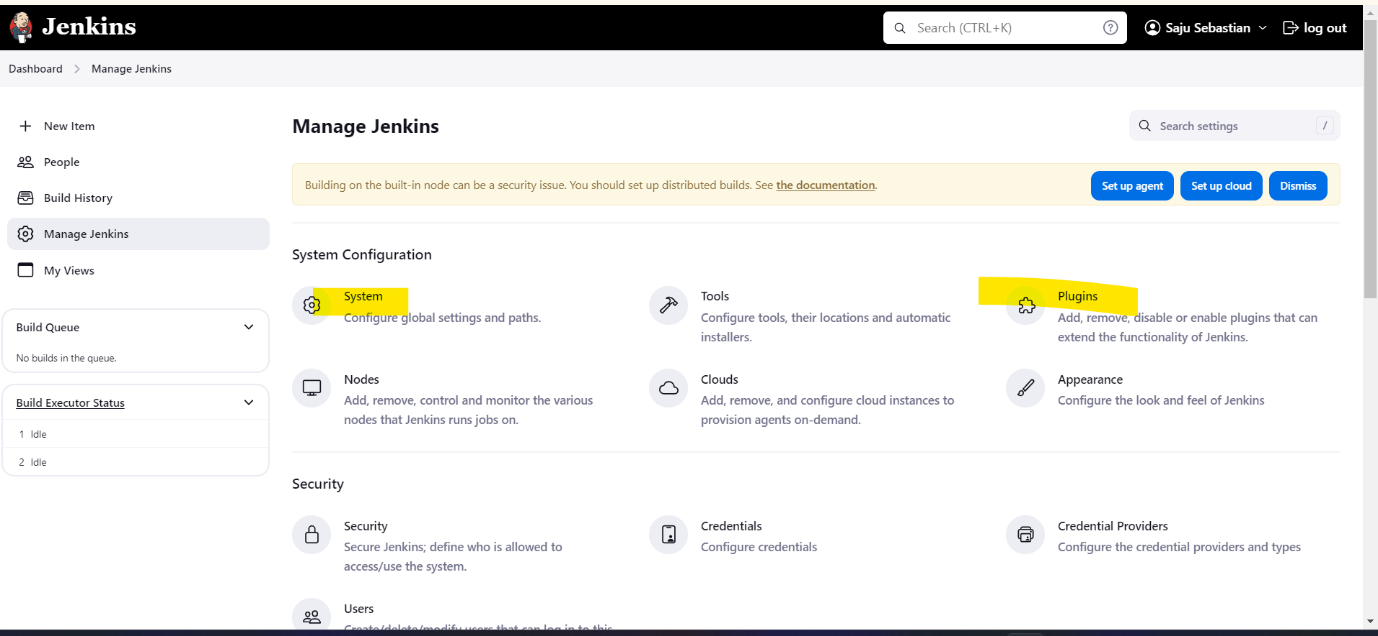
****

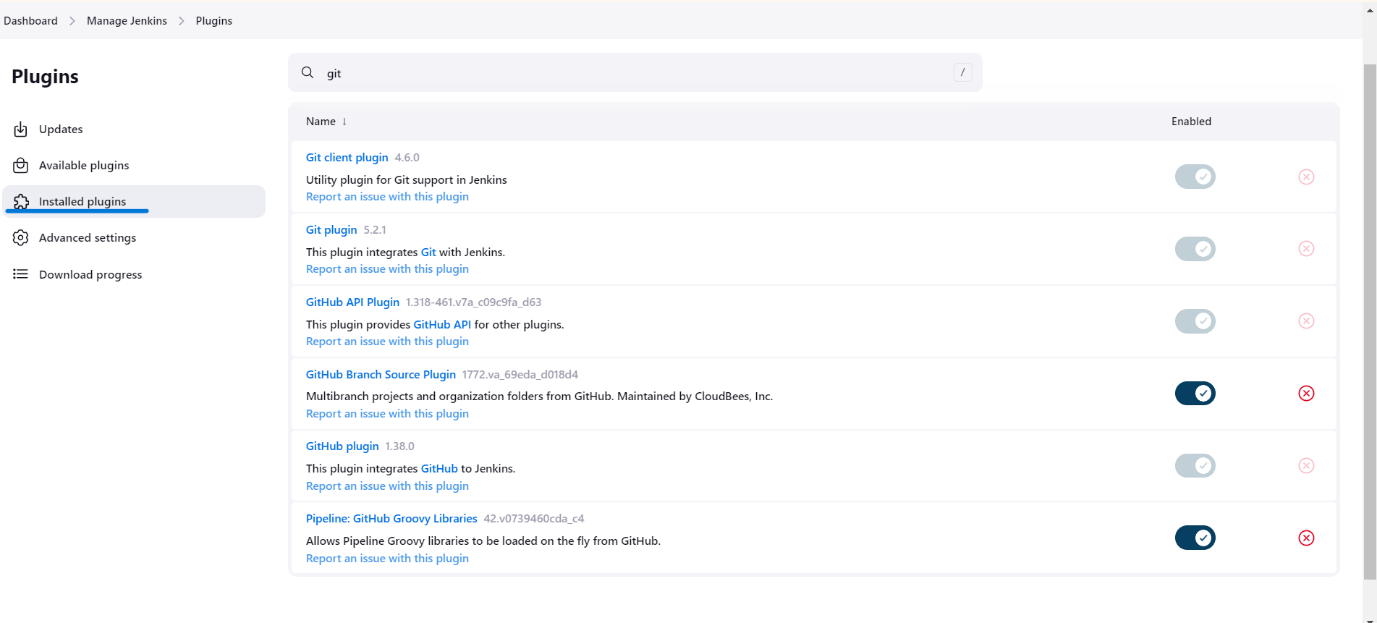
****

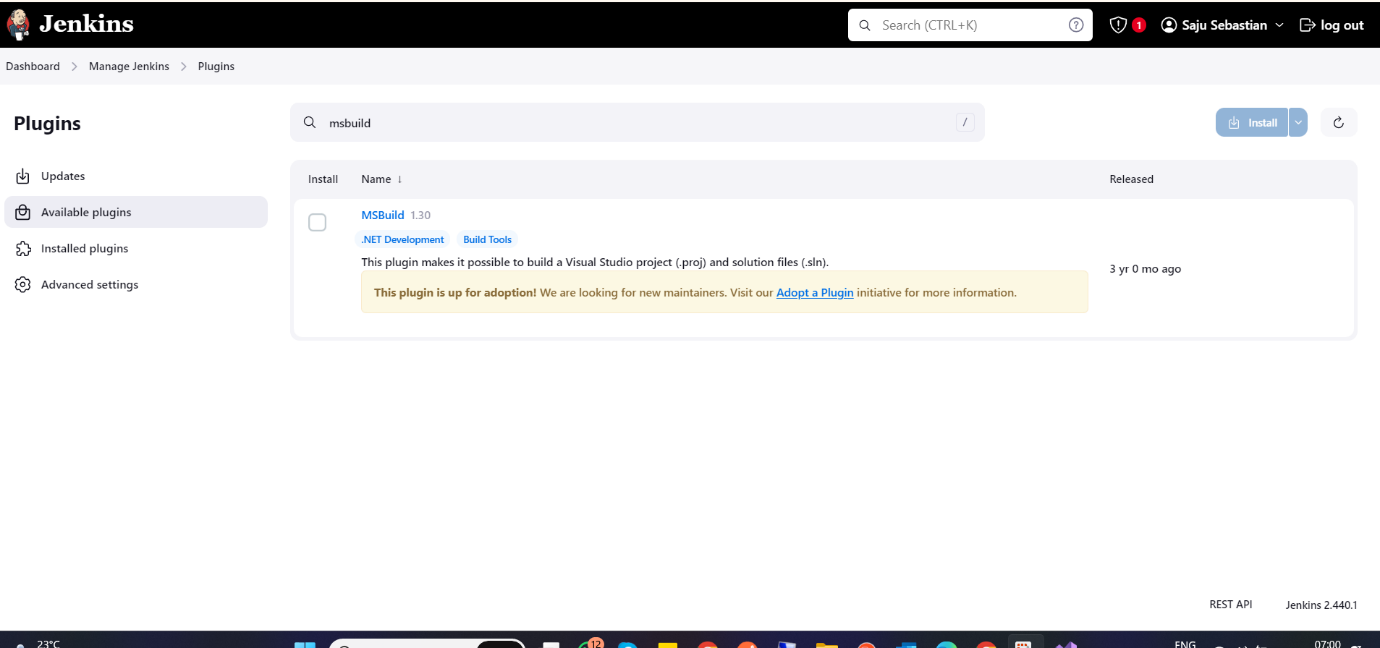
****

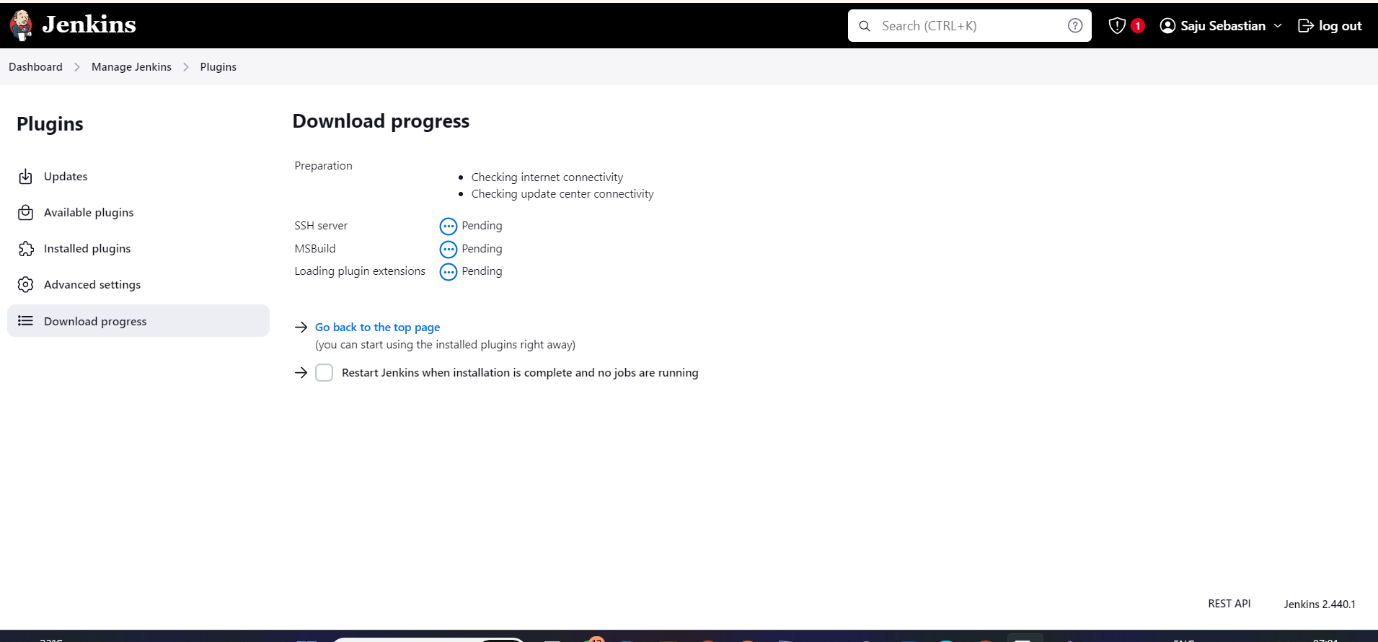
# **Task 2 – B**

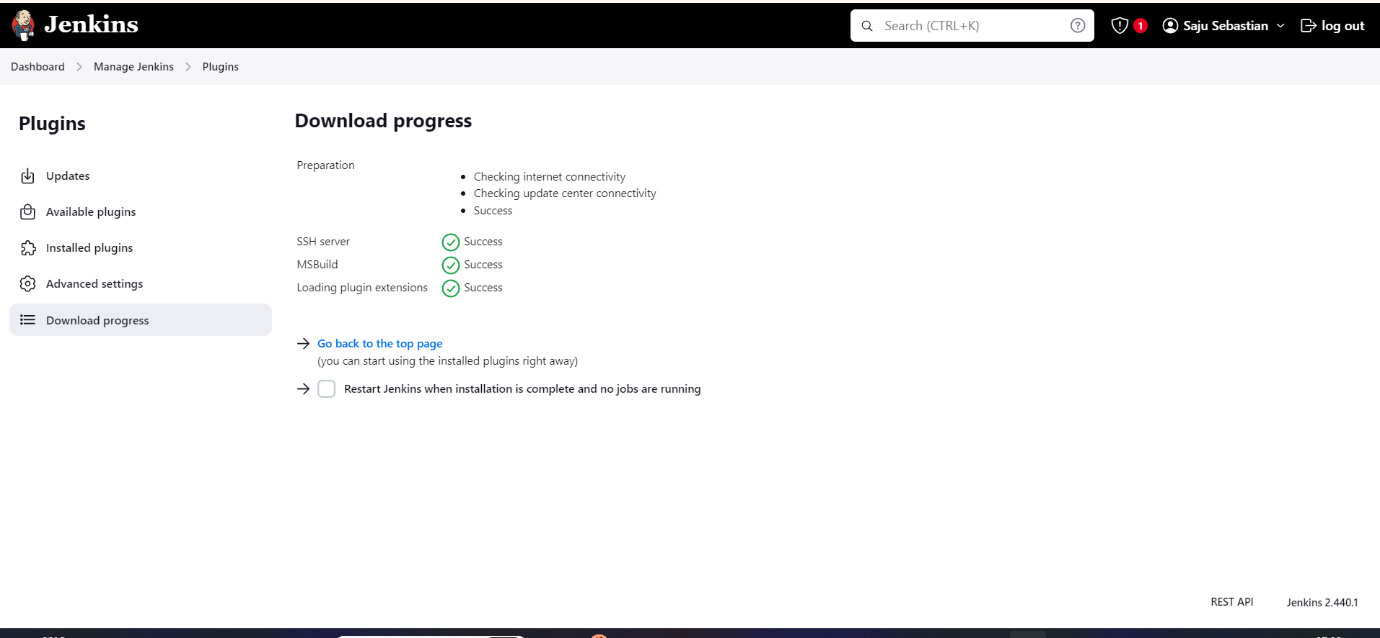
# **Jenkins Plugins & Configurations**











# **TeamCity Plugins & Configurations**

# **Task 2 – C**

# **Jenkins**

# **TeamCity**

# **Task 3 – A**

**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.

**Pros**:

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

**Cons**:

* Outdated or Unsupported Packages: Some packages within Chocolatey’s repository may be outdated or lack support.
* Limited GUI Interface: Chocolatey primarily relies on the CLI, which might be less intuitive for users accustomed to graphical interfaces.
* 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.

# **Task 3 – B**

Based on the evaluation provided, recommending Chocolatey can be justified for several reasons:

Ease of Use: Chocolatey offers an easy-to-use command-line interface (CLI) for managing software packages. This simplicity makes it accessible to users who are comfortable with command-line tools, streamlining the installation, upgrading, and removal of software.

Extensive Package Repository: Chocolatey provides access to a wide range of software packages from its repository. This extensive collection simplifies software installations, ensuring that users can find and install the software they need quickly and conveniently.

Automated Software Management: Chocolatey automates the entire process of downloading, installing, and configuring software packages. This automation saves time and effort for users, allowing them to focus on their tasks without worrying about manual installation procedures.

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. This flexibility caters to the needs of both individual users and enterprise environments, providing options for users with different requirements and budgets.

Overall, recommending Chocolatey is justified based on its ease of use, extensive package repository, automated software management capabilities, and flexibility in pricing and features. It offers a user-friendly solution for managing software packages on Windows systems, making it a valuable tool for individuals and organizations alike.

# **Task 3 – C**

The Package Management Tool that we have chosen for this task is **Chocolatey.**

A screenshot of a computer

Description automatically generated

Click on Try it Now Button (Install)

A screenshot of a computer

Description automatically generated

Select Individual

A screenshot of a computer

Description automatically generated

Copy the Command

A screenshot of a computer

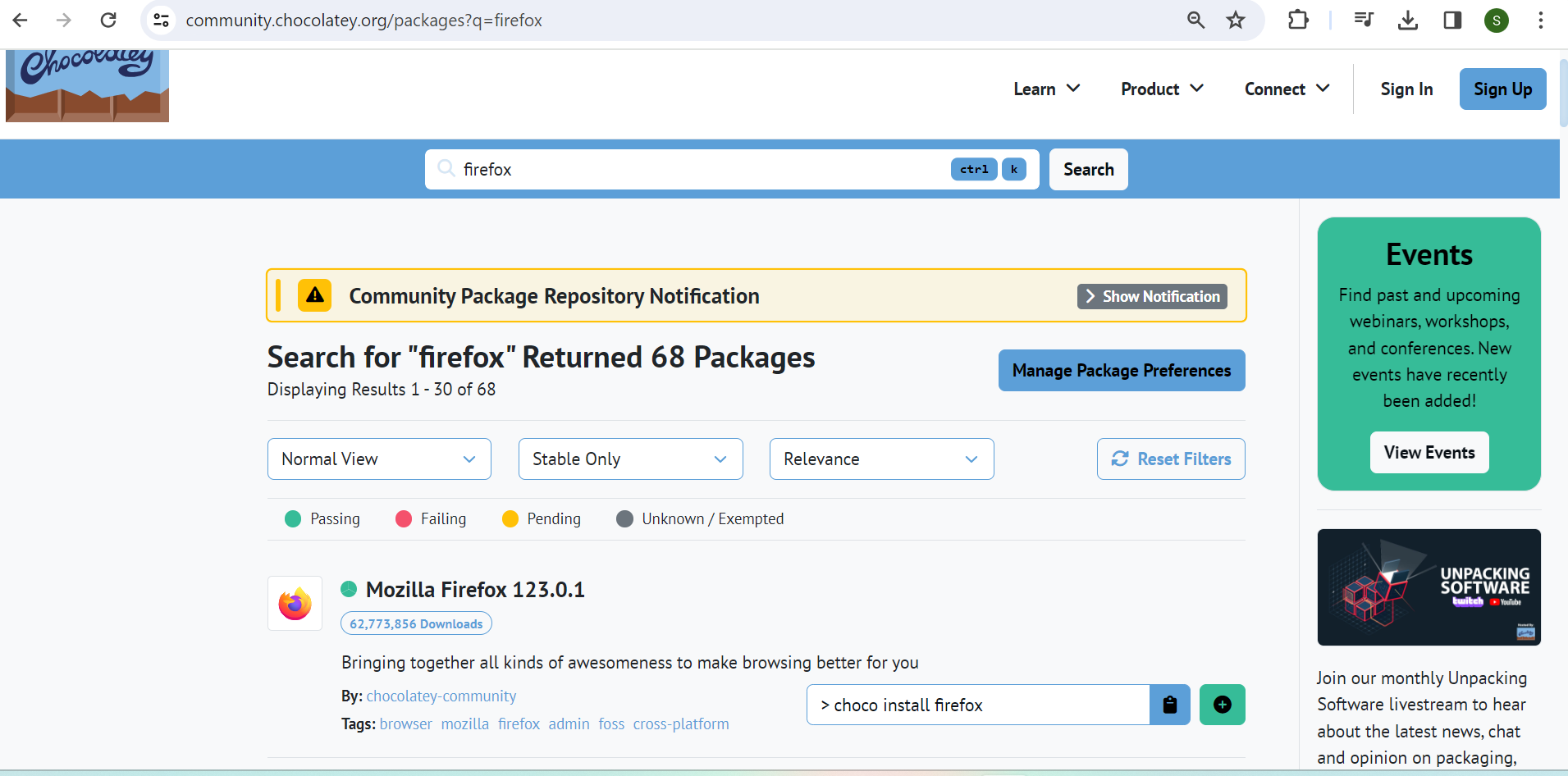
Description automatically generated

A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated



A computer screen with text and icons

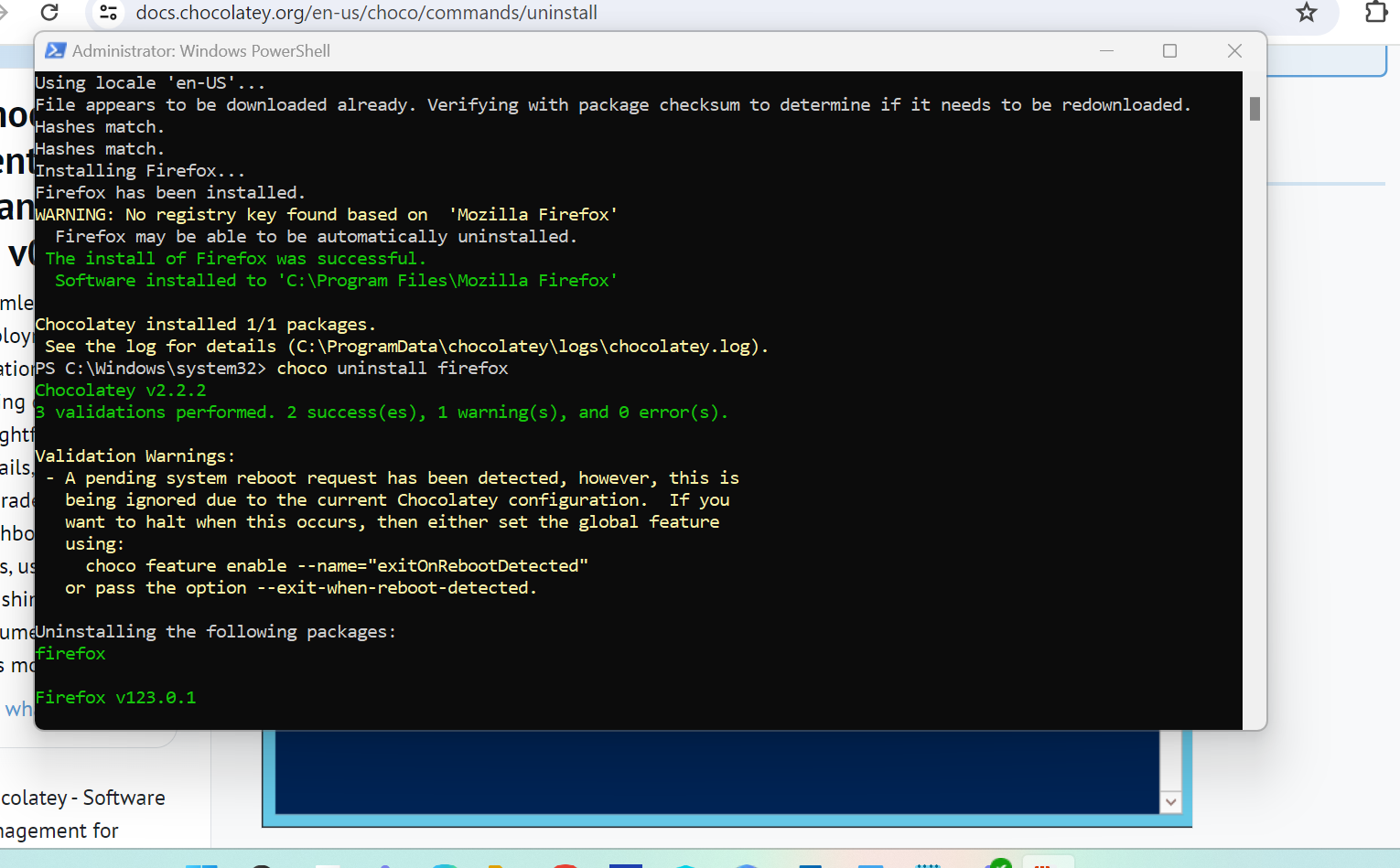
Description automatically generated

A computer screen shot of a black screen

Description automatically generated

A screenshot of a computer program

Description automatically generated



A screenshot of a computer screen

Description automatically generated