International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Migration of Jenkins Pipeline to GitHub Actions

Vidhi Khubchandani

B.Tech. CSAI, IGDTUW

ABSTRACT

Moving your continuous integration and continuous deployment (CI/CD) pipelines from Jenkins to GitHub Actions offers several benefits, including savings and a more flexible and secure hybrid model. Jenkins installations usually use servers hosted in their own data centers, while GitHub operations offer a hybrid model where DevSecOps self-hosted runners can be used for secure workflows and open runners for other common workflows that handle less sensitive data/code. GitHub Actions and Jenkins share some similarities, making the transition to GitHub Actions relatively easy.

INTRODUCTION

What is Jenkins?

Jenkins is an open-source automation server widely used for continuous integration and continuous delivery (CI/CD) in software development. Developed in Java and based on the principles of extensibility, flexibility, and ease of use, Jenkins provides a platform to automate various areas of the software development life cycle.

Jenkins designs and automates code building, testing, and deployment. It integrates seamlessly with version control systems like Git and can trigger automated workflows in response to code changes, allowing developers to identify bugs early in the development process. Jenkins supports a wide ecosystem of plugins that extend its functionality by enabling integration with various tools and technologies commonly used in software development. The Jenkins pipeline is a core feature that allows you to define complex build and deployment workflows in code. This declarative approach improves repeatability, collaboration, and versioning of CI/CD processes.

Jenkins works with a web-based user interface, making it accessible to users of various technical backgrounds. Its extensibility allows organizations to tailor Jenkins to their specific needs, whether they're managing simple builds or complex multi-phase deployment pipelines.

Because Jenkins is open source, it benefits from a large and active community that contributes to its development and supports users through forums, documentation, and plugins. Despite the emergence of alternative CI/CD tools, Jenkins remains a popular choice due to its flexibility, extensibility, and ability to integrate with various development tools and techniques.

What are GitHub Actions?

GitHub Actions is a CI/CD (Continuous Integration/Continuous Delivery) and automation platform provided by GitHub. It allows developers to define and automate workflows directly in their GitHub repositories, seamlessly integrating them with their source code. GitHub Actions is designed to simplify

and improve the software development lifecycle by automating tasks such as building, testing, and deploying applications.

Key features of GitHub Actions include its tight integration with GitHub repositories, allowing developers to trigger workflows based on events such as pushes, pull requests, and comments. Workflows are defined using YAML syntax, which provides a declarative and version-controlled way to define the steps and dependencies of a given automation process. GitHub Actions supports a rich ecosystem of pre-built actions, which are reusable units of code that perform specific tasks. Developers can create custom features or use existing GitHub Marketplace features, enabling easy integration with a variety of tools, services, and platforms.

The platform also offers matrix versions that enable parallel testing and deployment across different operating systems, versions, or configurations. Confidentiality management ensures that sensitive information is securely stored and used in workflows.

GitHub Actions is known for its simplicity, flexibility, and efficiency. It promotes collaboration by bringing CI/CD workflows closer to the codebase, allowing teams to iterate faster and maintain high-quality code. GitHub integration allows developers to benefit from a single platform for code hosting, version control, and CI/CD, making the entire development process more cohesive and collaborative.

Migration of Jenkins to GitHub Actions

The transition from Jenkins to GitHub Actions can be driven by several factors, each contributing to a more modern, efficient, and integrated continuous integration and continuous delivery (CI/CD) process. GitHub Actions offers several benefits that make the switch attractive.

First, GitHub Actions is tightly integrated with GitHub repositories, eliminating the need to manage a separate infrastructure. This advanced integration improves collaboration by bringing CI/CD workflows closer to source code, simplifying version control, and reducing developer context switching.

Second, GitHub Actions provides native support for content workflows that fit well with today's microservices architectures. This enables a more consistent and repeatable build and deployment in a containerized environment, which promotes scalability and resource efficiency.

In addition, GitHub Actions offers a rich ecosystem of instant actions and integration with third-party tools, reducing the need for custom scripting and speeding up workflow setup. Its declarative definition language makes it easy to define and manage complex workflows, which promotes better maintainability and collaboration between team members.

In addition, GitHub Actions supports event-driven workflows, enabling automation triggered by various GitHub events, such as code pushes, pull requests, or issue creation. This event-based model increases the flexibility and responsiveness of CI/CD processes. Finally, moving to GitHub Actions better aligns CI/CD processes with a Git-centric development workflow, promoting a seamless, integrated, and modern

approach to software delivery. However, it is critical to assess specific organizational needs and plan the migration carefully to ensure a smooth transition and maximize the benefits of the new platform.

METHODOLOGY

Components of GitHub Actions

You can set up a GitHub Actions workflow that triggers an event in your repository, such as opening a pull request or creating an issue. Your workflow contains one or more jobs that can be executed sequentially or in parallel. Each job runs in its virtual machine controller or container and has one or more steps that run a script you specify or run an action, which is a reusable plug-in that can simplify your workflow.

Workflows

A workflow is a configurable, automated process that performs one or more tasks. Workflows are defined by a YAML file written to the repository and run when triggered by an event in the repository, or they can be triggered manually or on a specific schedule.

Workflows are defined in the repositories. github/workflows directory, and a repository can have multiple workflows, each of which can perform different tasks. For example, you might have one workflow to create and test requests, another workflow to deploy the application every time a release is created, and another workflow to add a tag every time someone opens a new issue.

Event

An event is a specific action in the repository that triggers the execution of a workflow. For example, activity can come from GitHub when someone creates a pull request, opens an issue, or commits to a repository. You can also run the workflow on a schedule by sending it to the REST API or manually.

Jobs

A job is a collection of workflow steps that execute on the same runner. Each step is either an executable shell script or an executable action. The steps are performed sequentially and are interdependent. Since each stage is managed by the same runner, you can share data from one stage to another. For example, you might have a phase that builds your application, followed by a phase that tests the built application. You can define a job and its dependencies with other jobs. By default, jobs have no dependencies and run in parallel with each other. If a job depends on another job, it waits for the dependent job to finish before it can execute. For example, you can have multiple build jobs for different architectures that have no dependencies, and a batch job that depends on those jobs. The construction works are taking place in parallel, and when they are all completed, the packaging works will continue.

Actions

An action is a specially designed program for the GitHub Actions platform that carries out a difficult but regularly needed activity. Reduce the amount of repetitious code you write in your workflow files by using an action. An action can be used to get your git repository from GitHub, configure your build environment's toolchain, or set up cloud provider credentials.

You have the option of creating your actions or using ones from the GitHub Marketplace to incorporate into your workflows.

Runners

A runner is a server that runs your workflows when they run. Each runner can do one job at a time. GitHub provides tools for Ubuntu Linux, Windows, and macOS to run your workflows; each workflow is executed in a new, freshly made virtual machine. GitHub also offers larger runners available in larger configurations.

Creating GitHub actions workflow

Creating a GitHub Actions workflow involves defining a series of automated steps to be executed whenever specified events occur in a GitHub repository. Utilizing a YAML configuration file, typically stored in the `.github/workflows` directory, developers can specify workflows to trigger events like pushes, pull requests, or schedule-based actions. Each workflow consists of one or more jobs, with each job representing a unit of work to be performed. These jobs run on virtual environments provided by GitHub and can include a set of steps defining tasks such as checking out the repository, setting up the runtime environment, running tests, or deploying applications. GitHub Actions supports parallel execution, matrix builds for testing across multiple environments and integrates seamlessly with a variety of programming languages and tools. With its declarative syntax and integration directly into the GitHub repository, GitHub Actions simplifies the implementation of continuous integration and continuous delivery (CI/CD) pipelines, streamlining development workflows and promoting collaboration among team members.

Build and Test - GitHub Action Workflow

A GitHub actions workflow called "Build and Test" automates the important processes of building and testing a software project. Triggered by both push events and pull requests, this workflow uses a job running on the latest version of the Ubuntu environment provided by GitHub. The sequence of operations starts with checking the repository, followed by configuring Node.js, installing project dependencies using the npm package manager, and running build and test commands. " Configure Node.js " step specifies the runtime environment, which in this example is set to version 14. After this " the Install Dependencies " step ensures that all necessary project dependencies are obtained. " The Build " step starts the build process, which is usually done by an npm script defined in the project. Finally, the " Test " phase implements the project and the test suite and provides a comprehensive assessment of the integrity of the code. This GitHub Actions workflow improves the development process by automatically committing changes, enabling early detection of bugs, and promoting consistency in the build and test processes. Its declarative YAML syntax and seamless integration with GitHub repositories make it a powerful tool for implementing continuous integration and continuous delivery, promoting more efficient and reliable software development workflows.

Continuous Integration - GitHub Action Workflow

A workflow for GitHub operations called "Continuous Integration" serves as a key component in modern software development pipelines, automating the integration of code changes and validating them. Triggered by events such as push or pull requests; this workflow uses a series of defined steps to ensure

the seamless integration of newly deployed code into the existing code base. From the repository control, it continues to define the necessary runtime environment, which developers often specify in a workflow and YAML configuration file. Later steps usually involve installing dependencies, building code, and running tests. By automating these processes, the workflow facilitates the early detection of errors or problems, which contributes to code quality and stability. Parallel executions and matrix structures further improve their efficiency, allowing developers to test simultaneously in different environments. GitHub Actions and #039; integration with popular version control features, its ability to trigger workflows on specific branches or pull requests, and its seamless integration with other GitHub features make it an essential part of a continuous integration (CI) strategy. Overall, "Continuous Integration" The GitHub Actions workflow contributes to the development of robust and reliable software by automating the validation of code changes, promoting collaboration between development teams, and accelerating feedback throughout the software development lifecycle.

Spinnaker pipeline sync - GitHub Action Workflow

A GitHub action workflow called " Spinnaker Pipeline Syn " is designed to synchronise and automate deployment processes using Spinnaker, a powerful continuous delivery platform. This workflow manages the synchronization between GitHub repositories and Spinnaker pipelines. The process usually begins with reviewing the code base, determining the required runtime environment, and determining the dependencies needed to deploy Spinnaker. The workflow then triggers Spinnaker pipelines, which are responsible for managing the deployment workflow, including tasks such as creating artifacts, deploying to different environments, and post-deployment verification. GitHub Action's seamless integration with Spinnaker ensures that code changes are automatically reflected in deployment pipelines, enabling a seamless and consistent approach to continuous delivery. This synchronization workflow helps maintain version consistency between GitHub source code and Spinnaker deployment settings, reducing the risk of incompatibilities and ensuring that the deployment process accurately reflects the state of the codebase. By automating the synchronization of Spinnaker pipelines, this GitHub Actions workflow fosters a more efficient and reliable continuous delivery pipeline, promotes collaboration between development and operations teams, and accelerates the speed of software delivery.

Automation with GitHub Actions Workflow

Achieving automation with GitHub Actions workflows involves defining a series of steps in a YAML configuration file that are automatically executed in response to certain events, such as code pushes, pull requests, or scheduled triggers. These workflows are designed to streamline various aspects of the software development lifecycle, increasing efficiency, consistency, and collaboration between team members.

In the GitHub Actions workflow, each step represents a specific task, such as checking out the code repository, configuring the runtime environment, installing dependencies, building the app, running tests, and deploying code. Actions, which are individual units of code, can be reused in different workflows or shared with the community, enabling a modular and extensible approach to automation.

GitHub Actions supports a wide range of programming languages, tools, and services, giving you the flexibility to customize workflows to meet the specific needs of your project. Parallel execution and matrix compositions enable efficient testing in multiple environments simultaneously.

Secret management ensures that sensitive information such as API keys or credentials are securely stored and used in workflows. In addition, integrations with popular CI/CD tools and cloud services enable full automation from code change to deployment.

With GitHub Actions, teams can automate repetitive and time-consuming tasks, reduce the risk of human error, and create standard processes. This automation not only speeds up development cycles but also improves overall software reliability and quality by enabling continuous integration, continuous delivery, and continuous deployment.

CONCLUSION

With Jenkins and GitHub Actions, you can create workflows that automatically build, test, release, release and deploy code. Jenkins and GitHub Actions share some similarities in workflow definitions:

1. Jenkins creates workflows with declarative pipelines similar to GitHub Actions workflow files.
2. Jenkins uses steps to execute a collection of steps, while GitHub Actions uses jobs to group one or more steps or individual commands.
3. Jenkins and GitHub Actions support containerized builds. For more information, see "Creating a Docker Container Action."
4. Steps or tasks can be reused and shared with the community.

Advantages of Jenkins

1. Jenkins is open source and free, unlike competitor GitHub Actions, which is "Freemium", which is one of its advantages.
2. Plugins support the ability to save files from previous builds and use them in new builds.

Disadvantages of Jenkins

1. Jenkins offers a wide range of plugins to its customers, but unfortunately, the authors do not regularly update some of these plugins. Therefore, plugins must be properly supported before they can be used.
2. Because it relies on plugins, finding core features that aren't plugins can sometimes be difficult.
3. Due to the unpredictable nature of server requirements, usage costs for remote (cloud) Jenkins servers can vary. These downloads include things like the amount of code, number of artifacts, number of edits, etc.

Advantages of GitHub Actions

1. GitHub Actions is incredibly user-friendly for beginners. A YAML file is all you need to get started. For a startup or small business, GitHub Actions is an obvious choice for a CI/CD platform, as your engineers may already have experience with YAML.
2. Jenkins runs on a personal server. This means that you should maintain the Jenkins server regularly. However, GitHub Actions provides you with free runners to run CI/CD processes. Although these runners are owned and maintained by GitHub, you can also add self-hosted runners.

Disadvantages of GitHub Actions

1. When you use GitHub Actions, you are essentially tied into Github's source code management platform. You can store your code in any repository using Jenkins, including Gitlab, BitBucket, and others.
2. Compared to GitHub Actions, Jenkins is more experienced and established. Since Github's efforts are still in their infancy, they don't get much support from the community.

REFERENCES

1. https://k21academy.com/devops-foundation/github-actions-vs-jenkins/#7
2. https://link.springer.com/article/10.1007/s10664-022-10285-5
3. https://ieeexplore.ieee.org/abstract/document/9527496/
4. https://www.theseus.fi/handle/10024/795837
5. https://ieeexplore.ieee.org/abstract/document/10169071/
6. https://ieeexplore.ieee.org/abstract/document/9376148/
7. https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.852
8. https://link.springer.com/chapter/10.1007/978-1-4842-6464-5_1