# GitHub

# Study Guide
## GitHub Actions

Get exam-ready for your GitHub Actions Certification with our comprehensive study guide. We've curated the essential resources and learning activities to better prepare you for the GitHub Actions exam and boost your chances of success.

## Audience Profile

This exam is designed for DevOps engineers, software developers, and IT professionals with intermediate-level experience in GitHub Actions, including workflow creation, automation, and CI/CD pipeline management.

## Objective Domains

An objective domain for a certification exam, often referred to as a "domain" or "exam domain," is a structured outline or framework that defines the specific knowledge, skills, and topics that the certification exam will cover. It provides a clear roadmap for what candidates should expect to encounter on the exam and what they need to study and prepare for.

The domains provided in this study guide are intended to provide insight into the topic categories covered in the GitHub Actions exam, along with the learning objective within each domain.

| Domain Breakdown |
| --- |
| Domain 1: Author and maintain workflows |
| Domain 2: Consume workflows |
| Domain 3: Author and maintain actions |
| Domain 4: Manage GitHub Actions for the enterprise |

## Recommendations and Best Practices for Success

To increase your chances of success in passing the GitHub Actions exam, it's essential to start with a solid foundation of basic experience, exposure, and proficiency of GitHub Actions. The recommended learning paths for this exam provide you with an in-depth study of the learning content, followed by hands-on exercises and preparation assessment questions that were created to enable you to fine-tune your knowledge and readiness for the certification exam.

# Content Resources

The following resources have been created in collaboration with GitHub as recommended content that covers the learning objectives in each domain for the GitHub Actions exam. Both Microsoft Learn and LinkedIn Learning provide a complete learning path for the exam, but offer a different learning experience.

## Microsoft Learn
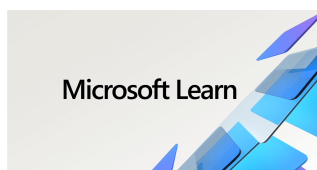
The [GitHub Actions learning path on MS Learn](#) provides a robust collection of learning modules designed to prepare you for the GitHub Actions exam. Dive into the world of code automation with confidence as you embark on a journey of building, testing, and deploying your projects directly from GitHub. Through hands-on exercises and assessment questions, these modules will empower you with the expertise needed to harness the full potential of GitHub Actions in streamlining your software development workflows.

## LinkedIn Learning

Prepare for the GitHub Actions Certification with LinkedIn Learning's comprehensive video series, **Prepare for the GitHub Actions Certification**. Dive into an extensive collection of modules meticulously designed to equip you for the GitHub Actions exam. Learn the intricacies of building, testing, and deploying your code seamlessly from GitHub's intuitive platform. Engage in hands-on exercises and thought-provoking assessments that will empower you with the expertise needed to master GitHub Actions and effortlessly automate your software development workflows. (**This learning path will be available towards the end of February, 2024**)

# Domain 1: Author and maintain workflows

| Work with events that trigger workflows |
| --- |
| Configure workflows to run for one or more events |
| Configure workflows to run for scheduled events |
| Configure workflows to run for manual events |
| Configure workflows to run for webhook events (i.e. check_run, check_suite, deployment, etc.) |
| Demonstrate a GitHub event to trigger a workflow based on a practical use case |

## Use the components of a workflow

Identify the correct syntax for workflow jobs (i.e. indentation and encapsulation of parts of the workflow)

Use job steps for actions and shell commands

Use conditional keywords for steps

Describe how actions, workflows, jobs, steps, runs, and the marketplace work together

Identify scenarios suited for using GitHub-hosted and self-hosted runners

Implement workflow commands as a run step to communicate with the runner

Demonstrate the use of dependent jobs

## Use encrypted secrets and environment variables as part of a workflow

Use encrypted secrets to store sensitive information

Identify the available default environment variables during the construction of the workflow

Identify the location to set custom environment variables in a workflow

Identify when to use the GITHUB_TOKEN secret

Demonstrate how to use workflow commands to set environment variables

## Create a workflow for a particular purpose

Add a script to a workflow

Demonstrate how to publish to GitHub Packages using a workflow

Demonstrate how to publish to GitHub Container Registry using a workflow

Use database and service containers in a GitHub Actions workflow

Use labels to route workflows to specific runners

Use CodeQL as a step in a workflow

Demonstrate how to publish a component as a GitHub release using GitHub Actions

Deploy a release to a cloud provider using a GitHub Actions workflow

| Manage workflow runs |
|---|
| Configure caching of workflow dependencies |
| Identify steps to pass data between jobs in a workflow |
| Remove workflow artifacts from GitHub |
| Add a workflow status badge |
| Add environment protections |
| Define a matrix of different job configurations |
| Implement workflow approval gates |

# Domain 2: Consume workflows

| Interpret the effects of a workflow |
|---|
| Identify the event that triggered a workflow from its effects in a repository, issue, or pull request |
| Describe a workflow's effects from reading its configuration file |
| Diagnose a failed workflow run (i.e. using a workflow run history and its logs, determine why a workflow run may have failed) |
| Identify ways to access the workflow logs from the user interface |
| Identify ways to access the workflow logs from GitHub's REST API |
| Enable step debug logging in a workflow |
| Demonstrate how to use default environment variables in a workflow |
| Demonstrate the correct syntax for passing custom environment variables in a workflow step |

| Locate a workflow, its logs, and artifacts |
|---|
| Describe where to locate a workflow in a repository |
| Explain the difference between disabling and deleting of workflows |
| Demonstrate how to download workflow artifacts from the user interface |
| Describe how to use an organization's templated workflow |

| Use actions in a workflow |
| --- |
| Define the indicators of what makes a trustworthy action |
| Identify an action's type, inputs, and outputs |
| Demonstrate how to use the specific version of an action in a workflow |

# Domain 3: Author and maintain actions

| Use available action types |
| --- |
| Identify the type of action required for a given problem (i.e. JavaScript, Docker container, run step) |
| Demonstrate how to troubleshoot JavaScript actions |
| Demonstrate how to troubleshoot Docker container actions |

| Describe the components of an action |
| --- |
| Identify the files and directory structure needed to create an action |
| Identify the metadata and syntax needed to create an action |
| Implement workflow commands within an action to communicate with the runner (Note: this includes exit codes) |

| Distribute an action |
| --- |
| Identify how to select an appropriate distribution model for an action (i.e., public, private, marketplace) |
| Identify the best practices for distributing custom actions |
| Demonstrate how to create a release strategy for an action (i.e. versioning) |
| Demonstrate how to publish an action to the GitHub marketplace |

# Domain 4: Manage GitHub Actions in the enterprise

| Distribute actions and workflows to the enterprise |
| --- |
| Explain reuse templates for actions and workflows |
| Define an approach for managing and leveraging reusable components (i.e. repos for storage, naming conventions for files/folders, and plans for ongoing maintenance) |
| Define how to distribute actions for an enterprise |
| Define how to control access to actions within the enterprise |
| Configure organizational use policies for GitHub Actions |

| Manage runners for the enterprise |
| --- |
| Describe the effects of configuring IP allow lists on GitHub-hosted and self-hosted runners |
| Describe how to select appropriate runners to support workloads (i.e. using a self-hosted versus GitHub-hosted runner, choosing supported operating systems) |
| Explain the difference between GitHub-hosted and self-hosted runners |
| Configure self-hosted runners for enterprise use (i.e. including proxies, labels, networking) |
| Demonstrate how to manage self-hosted runners using groups (i.e. managing access, moving runners into and between groups) |
| Demonstrate how to monitor, troubleshoot, and update self-hosted runners |

| Manage encrypted secrets in the enterprise |
| --- |
| Identify the scope of encrypted secrets |
| Demonstrate how to access encrypted secrets within actions and workflows |
| Explain how to manage organization-level encrypted secrets |
| Explain how to manage repository-level encrypted secrets |