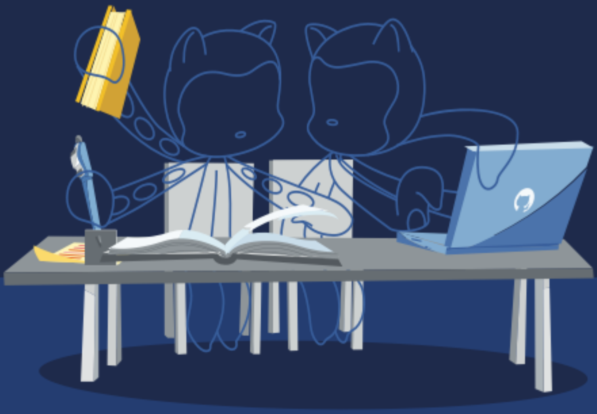
GitHub

# Study Guide
GitHub Administration



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

## Audience Profile

This exam is designed for system administrators, software developers, application administrators, and IT professionals with intermediate-level experience in GitHub Enterprise Administration.

## 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 Administration exam, along with the learning objective within each domain.

| Domain Breakdown | Exam Percentage |
|---|---|
| Domain 1: Support GitHub Enterprise for users and key stakeholders | 15% |
| Domain 2: Manage user identities and GitHub authentication | 20% |
| Domain 3: Describe how GitHub is deployed, distributed, and licensed | 5% |
| Domain 4: Manage access and permissions based on membership | 20% |
| Domain 5: Enable secure software development and ensure compliance | 15% |
| Domain 6: Manage GitHub Actions | 20% |
| Domain 7: Manage GitHub Packages | 5% |

## Recommendations and Best Practices for Success

To increase your chances of success in passing the GitHub Administration exam, it's essential to start with a solid foundation of basic experience, exposure, and proficiency of GitHub Administration. 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

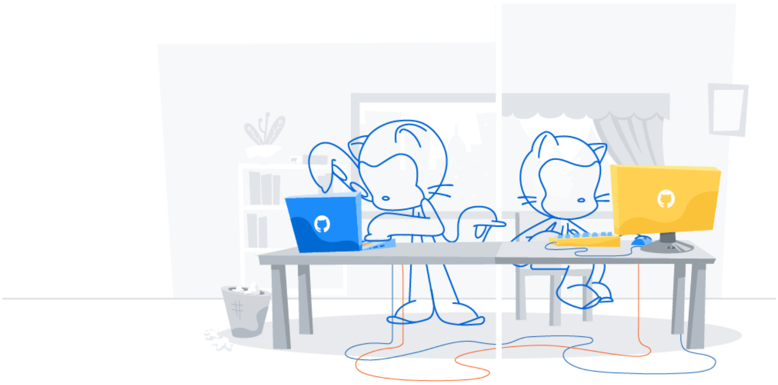
The following resources have been created in collaboration with GitHub as recommended content that covers the learning objectives in each domain for the GitHub Administration exam. Both Microsoft Learn and LinkedIn Learning provide a complete learning path for the exam, but offer a different learning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

### Microsoft Learn

The GitHub Administration learning path on MS Learn provides a robust collection of learning modules designed to prepare you for the GitHub Actions exam. As a GitHub administrator, you need to maintain a healthy, robust, and secure GitHub environment that supports the needs of your organization's requirements and developer workflows. The following modules will provide an overview of the various options and customizations available to you as an administrator on the GitHub platform. Gain more insight into managing user identities and authentication, deployment options, access and permissions, and securing software development and ensuring compliance.

### LinkedIn Learning

Immerse yourself in the Prepare for the GitHub Administration Certification learning path on LinkedIn Learning, featuring an series of video courses designed specifically for the GitHub Administration exam. As a GitHub administrator, your role is pivotal in maintaining a robust, secure, and highly efficient GitHub environment that perfectly aligns with the unique needs of your organization and its developer workflows. Explore our curated modules, guided by industry experts, to delve deep into the plethora of options and customizations available to you as a GitHub administrator. Gain valuable insights into user identity management, authentication, deployment strategies, access control, and security measures critical for software development and compliance.



## Domain 1: Support GitHub Enterprise for users and key stakeholders

| Support GitHub Enterprise for users and key stakeholders |
|---|
| Distinguish problems that can be solved by an administrator from those that need GitHub Support |
| Describe how to generate support bundles and diagnostics |
| Describe how GitHub's products and services are used within the enterprise to identify underutilized features, integrations in use, most active teams, and repositories. |
| Recommend standards for developer workflows, including code collaboration (fork-and-pull versus branching), branching, branch protection rules, code owners, the code review process, automation, and release strategy. |
| Explain the tooling ecosystem at the enterprise |
| Explain the enterprise's CI/CD strategy |
| Discuss how to recommend tooling and workflows to teams within an enterprise |
| Explain how GitHub APIs can be used to extend the capabilities of the administrator from the user interface, such as querying or storing the audit log |
| Locate an asset from the GitHub Marketplace for a specific need (i.e. find the Azure Pipelines GitHub App in the Marketplace, install it, and configure it to deploy your code) |
| Contrast a GitHub App and an action (i.e. their permissions, how they're built, how they're consumed) |
| List the benefits and risks of using apps and actions from the GitHub Marketplace |

## Domain 2: Manage user identities and GitHub authentication

| Manage user identities and GitHub authentication |
|---|
| List the implications of enabling SAML single sign-on (SSO) for an individual organization versus all organizations in an enterprise account |
| List the steps to enable and enforce SAML SSO for a single organization and multiple organizations using enterprise accounts. |
| Explain how to require two-factor authentication (2FA) for an organization. |
| Explain how to choose supported identity providers. |
| Describe how identity management and authorization works on GitHub |
| List the consequences of a user's membership in the instance, an organization, or multiple organizations |
| Describe the authentication and authorization model (specifically, how users get to the system, and how they're granted access to specific things within GitHub) |
| List the supported SCIM providers (Azure, Okta, self-created) |
| Describe how the SCIM protocol works and how GH supports it |
| Describe how Team synchronization works |
| Contrast team synchronization and SCI |

## Domain 3: Describe how GitHub is deployed, distributed, and licensed

| Contrast the capabilities of GHES, GHEC, and GHAE |
|---|
| Describe GitHub Enterprise Cloud (GHEC) |
| Describe GitHub Enterprise Server (GHES) |
| Describe GitHub AE |

| Differentiate how products are billed, including seat licenses, GitHub Actions, and GitHub Packages |
|---|
| Describe pricing for GitHub Actions |
| Describe pricing and support options for organizations |
| Describe how to find statistics of license usage for a specific organization |
| Describe how to find statistics of license usage for machine accounts and peripheral services |
| Explain the consumption of metered products given a report (i.e. GitHub Actions minutes or storage for GitHub Packages) |

## Domain 4: Manage access and permissions based on membership

| Describe enterprise permissions and policies |
|---|
| Explain the benefits and costs of deploying a single organization versus multiple organizations |
| Describe how to set default read permissions versus default write permissions across organizations |
| Describe Team sync through AD |
| Explain maintainability; writing scripts against multiple orgs and multiple access rights |
| Describe how to adjust enterprise policies and organization permissions in alignment with a company's trust and control position |

| Describe enterprise permissions and policies |
|---|
| Define a GitHub organization |
| List the possible roles of an organization member |
| Contrast permissions for organization members, owners, and billing managers |
| Describe the difference between being an organization member and an outside collaborator |
| List the consequences of a user's membership in an instance or organization |
| Explain how to give a user the minimum required permissions for repository, organization, or team access. |
| List the benefits and the drawbacks of creating a new organization |

| Describe team permissions |
|---|
| Define Teams in a GitHub organization |
| List the possible roles of a team member |
| Describe the different permission models |

| Repository permissions |
|---|
| Explain the actions of a user given a list of their permissions, such as repository role, team membership, or organization membership (https://github.com/organizations/<ORG_NAME>/settings/member_privileges) |
| List the repository membership options |
| Explain audit access to a repository |

## Domain 5: Enable secure software development and ensure compliance

| Enable secure software development and ensure compliance |
|---|
| Explain how GitHub supports the enterprise’s security posture |
| Describe scrubbing sensitive data from a Git repository (filter-branch / BFG) |
| Describe scrubbing sensitive data from GitHub (contacting support) |
| Explain how to choose a policy based on how much control is required |
| Explain the impacts of choosing a specific set of policies |
| Define organization policies |
| Define enterprise policies |

| Describe how to use the audit log APIs (Rest and GraphQL) to explain a missing asset |
|---|
| Define the use case for audit logs |
| Describe security and compliance concepts with GitHub |
| Explain how to provide reports for auditing |

| Define and explain the importance of the security features of a GitHub repository |
|---|
| Explain the importance of a security policy |
| Define a vulnerability |
| Describe a vulnerable dependency |
| Explain the importance of secret scanning |
| Explain the importance of code scanning |
| Describe automated code scanning (CodeQL) |
| Explain the dependency graph |
| Explain the importance of a security advisory |
| Describe Dependabot |
| Detect and fix outdated dependencies with security vulnerabilities |
| Describe security vulnerability alerts |
| Create and implement a security response plan that addresses sensitive data on a GitHub repository |
| Describe how to use SSH keys and Deploy keys to access repository data |

| API access and integrations |
|---|
| List supported access tokens (e.g. PAT, Installation Tokens, OAuth and GitHub app OAuth tokens, Device Tokens, Refresh tokens) |
| Explain how to find a token’s rate limits |
| Describe GitHub Apps, their repository permissions, user permissions, and event subscriptions |
| Describe OAuth Apps, their permissions, and event subscriptions |
| Contrast the use of a personal access token (PAT) or a GitHub App for authenticating a machine account |
| Describe the use of machine accounts versus GitHub apps |
| Explain how to approve or deny user-created GitHub Apps and OAuth apps based on a security policy |
| Define an enterprise managed user (EMU) |

## Domain 6: Manage GitHub Actions

| **Distribute actions and workflows to the enterprise** |
|---|
| Identify 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, plans for ongoing maintenance) |
| Define how to distribute actions for an enterprise |
| Explain 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 |
| Configure IP allow lists on internal applications and systems to allow interaction with GitHub-hosted runners |
| List the effects and potential abuse vectors of enabling self-hosted runners on public repositories |
| Select appropriate runners to support workloads (i.e. using a self-hosted versus GitHub-hosted runner, choosing supported operating systems) |
| Contrast GitHub-hosted and self-hosted runners |
| Configure self-hosted runners for enterprise use (i.e. including proxies, labels, networking) |
| Manage self-hosted runners using groups (i.e. managing access, moving runners into and between groups) |
| Monitor, troubleshoot, and update self-hosted runners |

| **Manage encrypted secrets in the enterprise** |
|---|
| Identify the scope of encrypted secrets |
| Explain how to access encrypted secrets within actions and workflows |
| Explain how to manage organization-level encrypted secrets |
| Describe how to manage repository-level encrypted secrets |
| Describe how to use 3rd party vaults |

## Domain 7: Manage GitHub Packages

| **Manage encrypted secrets in the enterprise** |
|---|
| Describe which GH Packages are supported |
| Describe how to access, write, and share GH Packages |
| Describe how to use GH Packages in workflows (i.e. with GH Actions or other CI/CD tools) |
| Explain the differences and use cases between GH Packages and releases |