



*Proprietary & Confidential*

# GitHub

## Enterprise Cloud

### SOC 1

Report on GitHub Management's  
Description of Its System and the Suitability  
of the Design and Operating Effectiveness of Controls



*Integrated SOC 1 Type 2 Report Prepared in Accordance with the  
AICPA Attestation Standards and IAASB ISAE No. 3402 Standards*

OCTOBER 1, 2021 TO SEPTEMBER 30, 2022



MOSSADAMS

# Table of Contents

<b>I. Independent Service Auditor's Report</b>	<b>1</b>
<b>II. GitHub's Assertion</b>	<b>5</b>
<b>III. Description of GitHub's Enterprise Cloud</b>	<b>7</b>
<b>A. Overview of GitHub</b>	<b>7</b>
<b>B. Scope of the Description</b>	<b>9</b>
<b>C. Internal Control Framework</b>	<b>10</b>
1. Control Environment	10
2. Risk Assessment Process	12
3. Monitoring Activities	13
4. Information and Communications	13
5. Control Activities	14
a) <i>Physical and Logical Access</i>	14
b) <i>Information Security</i>	16
Severity	20
SLA 20	
Potential Result	20
c) <i>Change Management</i>	21
d) <i>Backup and Recovery</i>	26
e) <i>Repository History</i>	26
f) <i>Vendor Management</i>	27
<b>D. Control Objectives and Related Controls</b>	<b>27</b>
<b>E. Complementary Subservice Organization Controls</b>	<b>27</b>
<b>F. Complementary User Entity Controls</b>	<b>29</b>
<b>IV. Description of GitHub's Control Objectives and Related Controls, and Independent Service Auditor's Description of Tests of Controls and Results</b>	<b>30</b>
<b>A. Information Provided by the Independent Service Auditor</b>	<b>30</b>
<b>B. Test of Controls and Results</b>	<b>31</b>
1. Physical and Logical Access	31
2. Information Security	37
3. Change Management	42
4. Backup and Recovery	45
5. Repository History	46
6. Vendor Management	48
<b>V. Other Information Provided by the Service Organization</b>	<b>49</b>
<b>A. Management's Response to Identified Testing Exceptions</b>	<b>49</b>

# I. Independent Service Auditor's Report

GitHub, Inc.  
88 Colin P. Kelly Jr. St.  
San Francisco, CA 94107

To the Management of GitHub:

## Scope

We have examined GitHub's description of its system titled "Description of GitHub's Enterprise Cloud" for processing user entities' transactions throughout the period October 1, 2021 to September 30, 2022 (description) and the suitability of the design and operating effectiveness of controls included in the description to achieve the related control objectives stated in the description, based on the criteria identified in "GitHub's Assertion" (assertion). The controls and control objectives included in the description are those that management of GitHub believes are likely to be relevant to user entities' internal control over financial reporting, and the description does not include those aspects of the Enterprise Cloud that are not likely to be relevant to user entities' internal control over financial reporting.

The information included in Section V, "Other Information Provided by the Service Organization" is presented by management of GitHub to provide additional information and is not a part of GitHub's description of its Enterprise Cloud made available to user entities during the period October 1, 2021 to September 30, 2022. Information about GitHub's management responses to identified testing exceptions has not been subjected to the procedures applied in the examination of the description of the Enterprise Cloud and of the suitability of the design and operating effectiveness of controls to achieve the related control objectives stated in the description of the Enterprise Cloud and, accordingly, we express no opinion on it.

GitHub uses the following subservice organizations:

- Sabey for colocation data center services
- QTS for colocation data center services
- CoreSite for colocation data center services
- Equinix for colocation data center services
- Amazon Web Services for infrastructure hosting
- Microsoft Azure for infrastructure hosting



The description includes only the control objectives and related controls of GitHub and excludes the control objectives and related controls of the subservice organizations. The description also indicates that certain control objectives specified by GitHub can be achieved only if complementary subservice organization controls assumed in the design of GitHub's controls are suitably designed and operating effectively, along with the related controls at GitHub. Our examination did not extend to controls at the subservice organizations, and we have not evaluated the suitability of the design or operating effectiveness of such complementary subservice organization controls.

The description indicates that certain control objectives specified in the description can be achieved only if complementary user entity controls assumed in the design of GitHub's controls are suitably designed and operating effectively, along with related controls at the service organization. Our examination did not extend to such complementary user entity controls, and we have not evaluated the suitability of the design or operating effectiveness of such complementary user entity controls.

### **Service Organization's Responsibilities**

In Section II, GitHub has provided an assertion about the fairness of the presentation of the description and suitability of the design and operating effectiveness of the controls to achieve the related control objectives stated in the description. GitHub is responsible for preparing the description and its assertion, including the completeness, accuracy, and method of presentation of the description and assertion, providing the services covered by the description, specifying the control objectives and stating them in the description, identifying the risks that threaten the achievement of the control objectives, selecting the criteria stated in the assertion, and designing, implementing, and documenting controls that are suitably designed and operating effectively to achieve the related control objectives stated in the description.

### **Service Auditor's Responsibilities**

Our responsibility is to express an opinion on the fairness of the presentation of the description and on the suitability of the design and operating effectiveness of the controls to achieve the related control objectives stated in the description, based on our examination.

Our examination was conducted in accordance with attestation standards established by the American Institute of Certified Public Accountants and in accordance with International Standard on Assurance Engagements 3402, *Assurance Reports on Controls at a Service Organization*, issued by the International Auditing and Assurance Standards Board. Those standards require that we plan and perform the examination to obtain reasonable assurance about whether, in all material respects, based on the criteria in management's assertion, the description is fairly presented, and the controls were suitably designed and operating effectively to achieve the related control objectives stated in the description throughout the period October 1, 2021 to September 30, 2022. We believe that the evidence we obtained is sufficient and appropriate to provide a reasonable basis for our opinion.

An examination of a description of a service organization's system and the suitability of the design and operating effectiveness of controls involves:

- performing procedures to obtain evidence about the fairness of the presentation of the description and the suitability of the design and operating effectiveness of the controls to achieve the related control objectives stated in the description, based on the criteria in management's assertion.



- assessing the risks that the description is not fairly presented and that the controls were not suitably designed or operating effectively to achieve the related control objectives stated in the description.
- testing the operating effectiveness of those controls that management considers necessary to provide reasonable assurance that the related control objectives stated in the description were achieved.
- evaluating the overall presentation of the description, suitability of the control objectives stated in the description, and suitability of the criteria specified by the service organization in its assertion.

### **Service Auditor's Independence and Quality Control**

We are required to be independent and to meet our other ethical responsibilities in accordance with the Code of Professional Conduct established by the AICPA and the International Ethics Standards Board for Accountants' Code of Ethics for Professional Accountants.

We applied the Statements on Quality Control Standards established by the AICPA and, accordingly, maintain a comprehensive system of quality control.

### **Inherent Limitations**

The description is prepared to meet the common needs of a broad range of user entities and their auditors who audit and report on user entities' financial statements and may not, therefore, include every aspect of the system that each individual user entity may consider important in its own particular environment. Because of their nature, controls at a service organization may not prevent, or detect and correct, all misstatements in processing or reporting transactions. Also, the projection to the future of any evaluation of the fairness of the presentation of the description, or conclusions about the suitability of the design or operating effectiveness of the controls to achieve the related control objectives, is subject to the risk that controls at a service organization may become ineffective.

### **Description of Tests of Controls**

The specific controls tested and the nature, timing, and results of those tests are listed in Section IV.

### **Opinion**

In our opinion, in all material respects, based on the criteria described in GitHub's assertion:

- the description fairly presents the Enterprise Cloud that was designed and implemented throughout the period October 1, 2021 to September 30, 2022.
- the controls related to the control objectives stated in the description were suitably designed to provide reasonable assurance that the control objectives would be achieved if the controls operated effectively throughout the period October 1, 2021 to September 30, 2022, and the subservice organizations and user entities applied the complementary controls assumed in the design of GitHub's controls throughout the period October 1, 2021 to September 30, 2022.
- the controls operated effectively to provide reasonable assurance that the control objectives stated in the description were achieved throughout the period October 1, 2021 to September 30, 2022, if complementary subservice organization and user entity controls assumed in the design of GitHub's controls operated effectively throughout the period October 1, 2021 to September 30, 2022.



## Restricted Use

This report, including the description of tests of controls and results thereof in Section IV, is intended solely for the information and use of GitHub, user entities of GitHub's Enterprise Cloud during some or all of the period October 1, 2021 to September 30, 2022, and their auditors who audit and report on such user entities' financial statements or internal control over financial reporting and have a sufficient understanding to consider it, along with other information, including information about controls implemented by user entities themselves, when assessing the risks of material misstatements of user entities' financial statements. This report is not intended to be and should not be used by anyone other than these specified parties.

**MOSS ADAMS LLP**

San Francisco, California  
November 11, 2022

## II. GitHub's Assertion

We have prepared the description of GitHub's system titled "Description of GitHub's Enterprise Cloud" for processing user entities' transactions throughout the period October 1, 2021 to September 30, 2022 (description) for user entities of the system during some or all of the period October 1, 2021 to September 30, 2022, and their auditors who audit and report on such user entities' financial statements or internal control over financial reporting and have a sufficient understanding to consider it, along with other information, including information about controls implemented by subservice organizations and user entities of the system themselves, when assessing the risks of material misstatements of user entities' financial statements.

GitHub uses the following subservice organizations:

- Sabey for colocation data center services
- QTS for colocation data center services
- CoreSite for colocation data center services
- Equinix for colocation data center services
- Amazon Web Services for infrastructure hosting
- Microsoft Azure for infrastructure hosting

The description includes only the control objectives and related controls of GitHub and excludes the control objectives and related controls of the subservice organizations. The description also indicates that certain control objectives specified in the description can be achieved only if complementary subservice organization controls assumed in the design of our controls are suitably designed and operating effectively, along with the related controls. The description does not extend to controls of the subservice organizations.

The description indicates that certain control objectives specified in the description can be achieved only if complementary user entity controls assumed in the design of GitHub's controls are suitably designed and operating effectively, along with related controls at the service organization. The description does not extend to controls of the user entities.

We confirm, to the best of our knowledge and belief, that:

- a. The description fairly presents the Enterprise Cloud made available to user entities of the system during some or all of the period October 1, 2021 to September 30, 2022 for processing their transactions as it relates to controls that are likely to be relevant to user entities' internal control over financial reporting. The criteria we used in making this assertion were that the description,
  - i. presents how the system made available to user entities of the system was designed and implemented to process relevant transactions, including, if applicable,
    - (1) the types of services provided, including, as appropriate, the classes of transactions processed.



- (2) the procedures, within both automated and manual systems, by which those services are provided, including, as appropriate, procedures by which transactions are initiated, authorized, recorded, processed, corrected as necessary, and transferred to the reports and other information prepared for user entities of the system.
  - (3) the information used in the performance of the procedures including, if applicable, related accounting records, whether electronic or manual, and supporting information involved in initiating, authorizing, recording, processing, and reporting transactions; this includes the correction of incorrect information and how information is transferred to the reports and other information prepared for user entities.
  - (4) how the system captures and addresses significant events and conditions other than transactions.
  - (5) the process used to prepare reports and other information for user entities.
  - (6) services performed by a subservice organization, if any, including whether the inclusive method or the carve-out method has been used in relation to them.
  - (7) the specified control objectives and controls designed to achieve those objectives, including, as applicable, complementary user entity controls and complementary subservice organization controls assumed in the design of the service organization's controls.
  - (8) other aspects of our control environment, risk assessment process, information and communications (including the related business processes), control activities, and monitoring activities that are relevant to the services provided.
- ii.* includes relevant details of changes to the service organization's system during the period covered by the description.
  - iii.* does not omit or distort information relevant to the service organization's system, while acknowledging that the description is prepared to meet the common needs of a broad range of user entities of the system and the user auditors, and may not, therefore, include every aspect of the Enterprise Cloud that each individual user entity of the system and its auditor may consider important in its own particular environment.
- b. the controls related to the control objectives stated in the description were suitably designed and operating effectively throughout the period October 1, 2021 to September 30, 2022 to achieve those control objectives if subservice organizations and user entities applied the complementary controls assumed in the design of GitHub's controls throughout the period October 1, 2021 to September 30, 2022. The criteria we used in making this assertion were that
    - i.* the risks that threaten the achievement of the control objectives stated in the description have been identified by management of the service organization.
    - ii.* the controls identified in the description would, if operating effectively, provide reasonable assurance that those risks would not prevent the control objectives stated in the description from being achieved.
    - iii.* the controls were consistently applied as designed, including whether manual controls were applied by individuals who have the appropriate competence and authority.





## III. Description of GitHub's Enterprise Cloud

### A. Overview of GitHub

#### COMPANY OVERVIEW

GitHub, an independently operated Microsoft subsidiary, generated its first commit in 2007. It's headquartered in San Francisco, California, with additional offices in Bellevue, WA; Raleigh, NC; Oxford, UK; Tokyo, Japan; Hyderabad, India; and Amsterdam, Netherlands. GitHub currently employs approximately 3,000 employees, with approximately 95 percent of the workforce being remote.

#### SYSTEM DESCRIPTION

GitHub is a web-based software development platform built on the Git version control software. Primarily used to develop and manage software code, GitHub offers the distributed version control and source code management functionality of Git with additional features and enhancements. Specifically, it provides access control, collaboration, bug tracking, feature requests, task management, GitHub advanced security, GitHub actions, pull requests, discussions, issues, pages, projects, docs, and wikis.

GitHub's Enterprise Cloud is GitHub's SaaS solution for collaborative software development. Features of the Enterprise Cloud service include:

#### ORGANIZATIONS

An organization is a collection of user accounts that owns repositories. Organizations have one or more owners, who have administrative privileges for the organization. When a user creates an organization, it does not have any repositories associated with it. At any time, members of the organization with the Owner role can add new repositories or transfer existing repositories.

#### CODE HOSTING

GitHub is one of the largest code hosts in the world, with millions of projects. Private, public, or open-source repositories are equipped with tools to host, version, and release code. Unlimited private repositories allow keeping the code in one place, even when using Subversion (SVN) clients or working with large files using Git Large File Storage (LFS).

Changes can be made to code in precise commits, allowing for quick searches on commit messages in the revision history to find a change. In addition, blame view enables users to trace changes and discover how the file, and code base, has evolved.

GitHub allows sharing, where changes can be packaged from a recently closed milestone or finished project into a new release. Users can draft and publish release notes, publish pre-release versions, attach files to issues, and link directly to the latest download.

#### CODE MANAGEMENT

Code review is a critical path to better code, and it's fundamental to how GitHub works. Built-in review tools make code review an essential part of team development workflows.



A pull request (PR) is a living conversation where ideas can be shared, tasks assigned, details discussed, and reviews conducted. Reviews happen faster because GitHub shows a user exactly what has changed. Diffs compare versions of source code side by side, highlighting the parts that are new, edited, or deleted.

PRs also enable clear feedback, review requests, and comments in context with comment threads within the code. Comments may be bundled into one review or in reply to someone else's comments inline as a conversation.

GitHub allows customers to protect important branches by setting branch protection rules, which define whether collaborators can delete or force push to the branch and set requirements for any pushes to the branch, such as passing status checks or a linear commit history. Protected branches contribute to better quality code management. Repositories can be configured to require status checks, such as continuous integration tests, reducing both human error and administrative overhead.

### GITHUB ACTIONS

GitHub Actions automates Continuous Integration/Continuous Delivery (CI/CD) software workflows by enabling the build, test, and deployment of code directly from GitHub, with code reviews, branch management, and issue triaging customized to work the way developers need.

GitHub Actions initiates workflows for events like push, issue creation, or a new release, and actions can be combined and configured for the services used, built, and maintained by the community.

GitHub Actions supports additional options to do things like build containers, deploy web services, or automate notifications to users of open-source projects using GitHub developers' existing `GITHUB_TOKEN` in collaboration with other GitHub Enterprise Cloud features.

### PROJECT MANAGEMENT

Project boards allow users to reference every issue and PR in a card, providing a drag-and-drop snapshot of the work that teams do in a repository. This feature can also function as an Agile idea board to capture early ideas that come up as part of a standup or team sync, without polluting the issues.

Issues enable team task tracking, with resources identified and assigned tasks within a team. Issues may be used to track a bug, discuss an idea with an @mention, or start distributing work. Issue and PR assignments to one or more teammates make it clear who is doing what work and what feedback and approvals have been requested.

Milestones can be added to issues or PRs to organize and track progress on groups of issues or PRs in a repository.

### TEAM MANAGEMENT

Building software is as much about managing teams and communities as it is about code. Users can set roles and expectations without starting from scratch. Customized common codes of conduct can be created for any project, with pre-written licenses available right from the repository.



GitHub Teams organizes people, provides level-up access with administrative roles, and tunes permissions for nested teams. Discussion threads keep conversations on topic using moderation tools, like issue and PR locking, to help teams stay focused on code. For maintaining open-source projects, user blocking reduces noise and keeps conversations productive.

## DOCUMENTATION

GitHub allows documentation to be created and maintained in any repository, and wikis are available to create documentation with version control. Each wiki is its own repository, so every change is versioned and comparable. With a text editor, users can add docs in the text formatting language of choice, such as Textile or GitHub Flavored Markdown.

## SYSTEM BOUNDARIES

The scope of this report includes GitHub's Enterprise Cloud, and the supporting production systems, infrastructure, software, people, procedures, and data. The following Enterprise Cloud features are included in the scope of this report: Issues, Pull Requests (PRs), Discussions, Wikis, Pages, Projects, Docs, Audit Logging, Advanced Security, Dependabot, and GitHub Actions.

## SUBSERVICE ORGANIZATIONS

GitHub uses multiple subservice organizations in conjunction with providing its Enterprise Cloud product. GitHub uses Sabey, QTS, CoreSite, and Equinix to provide colocation data center services, and Microsoft Azure (Azure) and Amazon Web Services (AWS) to provide infrastructure hosting. These subservice organizations are excluded from the scope of this report. The expected controls for which they are responsible are found in a subsequent section titled *Complementary Subservice Organization Controls*.

## B. Scope of the Description

This description of GitHub's Enterprise Cloud addresses only GitHub's Enterprise Cloud provided to its user entities and excludes other services provided by GitHub. The description is intended to provide information for user entities of the Enterprise Cloud and their independent auditors who audit and report on such user entities' financial statements to be used in obtaining an understanding of the Enterprise Cloud and the controls over that system that are likely to be relevant to user entities' internal control over financial reporting. The description of the system includes certain business process controls and IT general controls that support the delivery of GitHub's Enterprise Cloud.

GitHub uses the following subservice organizations:

- Sabey for colocation data center services
- QTS for colocation data center services
- CoreSite for colocation data center services
- Equinix for colocation data center services
- Amazon Web Services for infrastructure hosting
- Microsoft Azure for infrastructure hosting

The description includes only the control objectives and related controls of GitHub and excludes the control objectives and related controls of the subservice organizations.



## C. Internal Control Framework

This section provides information about the five interrelated components of internal control at GitHub, including GitHub's:

- Control Environment
- Risk Assessment Process
- Monitoring Activities
- Information and Communications
- Control Activities

### 1. Control Environment

The internal control environment reflects the overall attitude, awareness, and actions of executive management and other stakeholders concerning the importance of controls and the emphasis given to controls in the company's policies, procedures, methods, and organizational structure.

Management is responsible for directing and controlling operations and for establishing, communicating, and monitoring policies and procedures. Maintaining sound internal controls and establishing the integrity and ethical values of personnel is a critical management function.

#### CODE OF CONDUCT

During the onboarding process, new employees complete security and privacy awareness training and review and acknowledge the employee guide to company policies and practices, the Hubber Handbook. The Handbook includes the GitHub Standard of Conduct, Security Policy Awareness and Responsibilities, and other information security topics.

#### ORGANIZATIONAL STRUCTURE

GitHub's organizational structure provides the framework within which its activities for achieving company-wide objectives and key results are defined, planned, sponsored, executed, controlled, and monitored. Management believes that establishing a relevant organizational structure includes considering key areas of authority and responsibility and lines of reporting. Senior leadership sets company-wide objectives and key results and also reports and reviews progress towards meeting those objectives semiannually.

GitHub has established appropriate lines of reporting considering the nature, size, and culture of the company. People Operations maintains an organizational chart that outlines security roles across the company. The organizational chart is available for employees and contractors both on GitHub's intranet and internal human resource information system (HRIS). Management continues to evaluate its organizational structure and makes changes as necessary. Hubber responsibilities are communicated through documented job descriptions, stored in the respective teams' GitHub repository.



GitHub's organizational structure is designed to meet the company's security requirements and commitments to customers. GitHub's Senior Leadership team, specifically those reporting to the Chief Executive Officer (CEO), provide direction and oversight and includes members who are independent from control operations. In addition, management is responsible for establishing, communicating, and monitoring policies and procedures as well as aligning operations with leadership's defined objectives and key results.

When leadership changes occur within the organization, the Security team is notified where there may be a compliance impact. New incoming leadership is announced to the organization on the company intranet, and in regular all-hands meetings.

The Security team, headed by the Chief Security Officer and SVP Engineering, is responsible for monitoring and enhancing GitHub's overall security posture. This function includes managing security risks and threats, educating employees on security-related best practices, building security awareness, responding to security incidents, and performing internal security audits and security reviews. Security leadership considers out-of-band changes based on newly identified risk findings or service changes and addresses such changes as deemed appropriate.

Security leadership is also accountable for planning and staffing appropriately to address risk remediation and mitigation as identified in prescribed risk monitoring activities. Security leadership reviews these components as part of an annual headcount and budgeting process.

Management across the company is accountable for ensuring necessary policy, standards, and standard operating procedures aid in assessing and addressing operational risks. These owners review and update these policy-related documents at least annually, to reflect changes and help ensure completeness and accuracy, based upon the annual risk assessment, strategic business initiatives driven by leadership, and other events that dictate changes. Security leadership reviews and approves the materials and any changes on an annual basis.

The Security team administers security and privacy awareness training on an annual basis. The Security team follows up with employees who are delinquent in completing the training until these employees are compliant. Moreover, periodic security awareness notifications are sent to employees as needed, highlighting new controls or warning them of known threats.

## HUMAN RESOURCES

GitHub evaluates candidates' abilities in the interview process against established job descriptions. Fit to the role is scored, tracked, and approved by the hiring manager in GitHub's recruitment management system.

In order to be employed by GitHub, candidates must successfully complete a background check. The background check is initiated after the candidate signs the offer letter. Employees are not allowed to onboard until the background check is cleared. The Talent Coordinator is notified once a candidate clears, which is then relayed to the hiring manager. In instances where negative or incomplete information is obtained, the VP of Global Talent Acquisition, or a delegate, assesses, in consultation with Legal, the potential risk and liabilities related to the job's requirements and makes a final decision on the hire.



Before any adverse action is taken based on a background check, GitHub provides the applicant with a notice that includes an opportunity to respond to or clarify any discrepancies. If GitHub does not hire a candidate based on the results of a background check, GitHub provides the candidate with an adverse-action notice and informs them of their rights to see the information reported and to correct inaccurate information.

New employees undergo a three-day company onboarding session, at which time they are provisioned with a company-issued laptop and their GitHub organization and relevant system credentials as commensurate with their new role. During their first week, new employees are introduced to the roles of Security, GRCC, and Data Protection in GitHub and are required to complete the security and privacy awareness training.

Additionally, product engineers hired to work on the Enterprise Cloud application and services complete Secure Coding Practices training. This training includes existing engineering employees re-taking the training when a refresh is required on coding best practices to help ensure engineers align with changes in the code base or development tools.

Furthermore, GitHub management and peers evaluate and provide feedback to employees annually.

## 2. Risk Assessment Process

GitHub recognizes that risk management is a critical component of its operations and contributes to ensuring customer data is properly protected. GitHub incorporates risk management throughout its business processes and across the organization. The foundation of this process is management's knowledge of its operations, its close working relationship with its customers and vendors, and its understanding of the space in which it operates.

The Security GRCC team is embedded within the larger Security team. This team monitors internal controls and conducts risk monitoring in accordance with relevant regulatory and contractual compliance requirements. This responsibility includes managing the design, implementation, and monitoring of the Enterprise Cloud control environment, as well as assessing, monitoring, and mitigating security, fraud, and compliance risks.

### RISK IDENTIFICATION AND TREATMENT

GitHub has defined risk management processes to identify and manage risks that may affect the system's security. At least annually, Security GRCC performs a risk assessment to identify, track, and treat technical risks related to the product and reports risks to GitHub leadership. Risks identified are documented in a repository where the risks and mitigation actions are tracked to resolution.

### RISK REPORTING

Summary reporting on security risk areas is included in annual leadership reporting as part of the annual security risk assessment. GitHub's Leadership Team as well as extended Leadership Teams across Security, Engineering, IT, Business Systems, Legal, and Privacy, review this annual risk report.



Control activities have been established to help ensure key processes operate as intended. These activities are integrated into the policies, standards, and procedures outlined in the Procedures section above.

### 3. Monitoring Activities

The Security GRCC team at GitHub is responsible for monitoring the internal control environment for each of the compliance frameworks adopted by GitHub.

#### INTERNAL CONTROL REVIEWS

The Security GRCC team conducts internal control assessments annually to assess the effectiveness of the internal control environment. Control assessment results are formally documented and retained. Issues are identified and escalated to the relevant internal teams to resolve control design or effectiveness issues, and the status of the operating effectiveness of controls, identified gaps, and remediation efforts are reported up to the Senior Leadership team for awareness on a quarterly basis.

### 4. Information and Communications

To help align GitHub business strategies and goals with operating performance, management is committed to maintaining effective communication with employees and customers.

#### INTERNAL COMMUNICATIONS

GitHub has published policies and procedures, both included in the Hubber Handbook and published separately, outlining the responsibility of employees to report security and operational failures, incidents, system problems, and complaints. The document owner(s) and Security leadership review and approve security policies and procedures annually. Significant changes to policies result in communication to personnel regarding policy updates.

Every Hubber, depending on their role, is responsible for designing and executing work and services in a secure manner in alignment with company standards and policies, and for reporting issues and findings that impact security up their management chain or directly to the Security team.

#### EXTERNAL COMMUNICATIONS

GitHub communicates the description of Enterprise Cloud systems, features, responsibilities, customer commitments, and instructions to report incidents and complaints using its official blog (<https://blog.github.com>). The blog maintains a changelog, which is a chronological list of information on customer-facing feature changes, bug fixes, or security notifications made available to end users. GitHub's changelog is live on the blog site (<https://blog.github.com/changelog>) and available for RSS feed subscription. It is also accessible via GitHub's Changelog twitter account (@GHChangelog).

Enterprise Cloud users have ready access to support resources at GitHub Help (<https://help.github.com>) and GitHub Guides (<https://guides.github.com>), and they can access customized feature tutorials through the GitHub Learning Labs (<https://lab.github.com>).



Customer commitments and responsibilities are communicated through GitHub's Terms of Service (<https://help.github.com/articles/github-terms-of-service/>) and in respective customer contracts. The Terms of Service includes GitHub's customer Acceptable Use Policy, which provides the basic rules customers are required to follow as members of the community on the Enterprise Cloud product.

The user community can communicate directly with GitHub. GitHub Support receives reports of security issues and many other end-user concerns and questions via email or through the 'Contact Us' web form. A ticketing system is used to document and track these issues to resolution.

## 5. Control Activities

### a) *Physical and Logical Access*

*Controls provide reasonable assurance that physical and logical access to systems and customer data is controlled and limited to properly authorized users.*

#### USER AUTHENTICATION

This section outlines the process for both internal and customer users to access the Enterprise Cloud.

#### GITHUB INTERNAL ACCESS

GitHub has implemented access protection measures to only allow authenticated and authorized users access to the portions of GitHub's instance of the Enterprise Cloud instance that are not explicitly public. Enterprise Cloud requires a valid and unique account ID, password, and two-factor authentication (2FA). GitHub integrates with a third-party, single sign-on (SSO) provider that enforces two-factor authentication using FIDO2 authenticators.

Access to internal systems is restricted through unique account IDs ("handles"), password, and 2FA. To access these environments, GitHub personnel leverage the same GitHub handles as GitHub Enterprise Cloud. Employees are prohibited by policy from using shared accounts or credentials when accessing GitHub internal systems. Exceptions to the policy are documented in the exceptions policy. Employee access to Enterprise Cloud production systems is restricted to authorized personnel with demonstrated need and isolated from the internet by VPN, bastion hosts, and/or SAML enforcing HTTPS reverse proxy. GitHub uses SSH to authenticate to back-end production resources through a bastion host. VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity.

Employee access to production infrastructure must be approved by designated personnel before access is granted, and that approval must be renewed during periodic access reviews. Server and database administrative privileges are restricted to the respective system owners. Individual users, teams, and managers request and manage access via the GitHub Entitlements system. The Security Access Engineering team has the ultimate responsibility for maintaining the Entitlements system, but the access grants and review flow are the responsibility of the user(s) requesting access and their manager.





## GITHUB CUSTOMER ACCESS

Access to customer organizations requires a unique account and password. Customers can optionally enable 2FA for their organization through Enterprise Cloud settings. Audit logging is also available to allow organization administrators to quickly review user actions performed. This log includes details such as who performed the action, what the action was, and when it was performed. Where a repository is configured as private, GitHub Enterprise Cloud is designed to limit access to data and settings to the user who created the repository or who has been explicitly granted access.

GitHub file servers are obfuscated from front-end applications, where each Git operation, such as read, commit, or push in a repository is directed through GitHub's authentication pathway using service handlers. Requests terminate at the Git proxy servers where GitHub, the service handler endpoint, is executed to perform authentication and repository permission checks.

GitHub checks whether the requested repository is active or disabled, is public or private, and checks the user permissions and authentication information (e.g., anonymous user for public only access, username and password, OAuth token, or SSH keys). If the service provides a successful response for the authentication and permission check, GitHub returns the routing information for access to the file server host where the repository can be found. If the response is denied, the request returns a 404 error.

## USER PROVISIONING

New employee permissions are automatically added during the onboarding process based on predefined permissions granted to individuals based on their team and their role within that team. The user's manager and Security Operations review and approve any additional privileges granted outside of those provisioned as part of their role. GitHub's internal systems are configured to automatically provision non role-based entitlements only after the user's manager reviews and approves the access request.

During onboarding, GitHub IT shepherds employees through configuring their account setup and 2FA, for GitHub's SSO provider, and GitHub organization. IT monitors the access configuration, and users are locked out of internal resources until remediated if they are found to not have required security settings enabled in accordance with company requirements.

The Security Operations team is responsible for facilitating a periodic review over access to sensitive systems. Security Operations identifies systems and elevated access that warrant additional review and oversight, given the potential impact on the Enterprise Cloud platform. Specifically, these systems include production, security management tools, user access tools, and vulnerability management tools. Managers review and approve access to critical production and access gateway systems for their direct reports on a semiannual basis. The reviews allow the manager to provide attestation that the level of access currently granted to each individual under their review is appropriate and required for the individual's job function. User access levels remaining after changes from the review are authorized by management, or removed if approval is not granted.



The Infrastructure team reviews physical data center access annually. Accounts belonging to unapproved individuals are removed.

The Threat Hunting, Operations, and Response (THOR) team builds automated monitors of user access events to identify anomalous user access activity. The Security Incident Response Team (SIRT) investigates detections, and if SIRT concludes the activity is potentially malicious, the account is locked to reduce risk of unauthorized access and incident response procedures are initiated.

Deprovisioning occurs within 24 hours of a user's termination. After People Operations enters an employee's termination date in GitHub's HRIS, an offboarding notification is sent to Security Operations daily, for both planned and unplanned exits. The Security Operations team works through a standard offboarding procedure, with most aspects of account termination being automated via the Entitlements system, with manual removal of user access when necessary.

The timeframe for completing access removals for corporate and production network resources, and physical access is within 24 hours of the termination. High-risk access is deprovisioned first, ensuring a terminated employee can no longer access GitHub's internal resources.

## **b) Information Security**

*Controls provide reasonable assurance that confidentiality and information security practices protect the production environment.*

### **POLICIES AND PROCEDURES**

GitHub has published policies and procedures, both included in the Hubber Handbook and published separately, outlining the responsibility of employees to report security and operational failures, incidents, system problems, and complaints. The document owner(s) and Security leadership review and approve security policies and procedures annually. Significant changes to policies result in communication to personnel regarding policy updates.

Every Hubber, depending on their role, is responsible for designing and executing work and services in a secure manner, in alignment with company standards and policies, and for reporting issues and findings that impact security up their management chain or directly to the Security team.

GitHub policies establish procedures and controls to enable security, efficiency, availability, and quality of service. The GitHub Information Security Management System (ISMS) Policy and related policy statements define information security practices, roles, and responsibilities. The ISMS Policy outlines the security roles and responsibilities for the organization and expectations for employees, contractors, and third parties utilizing GitHub systems or data.

GitHub maintains and communicates key standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes.



## VULNERABILITY MANAGEMENT

The GitHub Vulnerability Management Program monitors and interrogates public-facing and internal infrastructure to identify systems that are vulnerable to known exploits, configured in ways that unnecessarily increase risk of compromise, or have software installed which is known to be insecure. The program develops and maintains scanning coverage across hosting platforms, providers, and networks. Vulnerability scanning is executed on at least a monthly basis, with findings prioritized for remediation based on risk, technology dependencies and exposure.

GitHub has a published internal standard based on the Common Vulnerability Scoring System (CVSS) levels for vulnerability management (critical, high, medium, and low) as reported by GitHub's vulnerability scanning and reporting system. GitHub's Vulnerability Management team uses the CVSS score for initial vulnerability assessment and then conducts an impact analysis that takes into consideration the specifics of GitHub's infrastructure. When a credible and actionable vulnerability is identified, the Vulnerability Management team provides the system owner with a description of the perceived impact of the vulnerability and the required timetable for resolution/mitigation. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.

GitHub Product Security Engineering, Vulnerability Management, and Infrastructure teams triage vulnerabilities from vendors and internal and external scanning, and patch those vulnerabilities based on risk and exposure. GitHub prefers to run "known good" software that has been tested and operated in production. Preference is given to running the most stable release of software possible. Patches are not generally applied for the objective of running the latest or "bleeding-edge" version of any package.

Patching occurs to address security fixes, bug-fixes, performance issues, and to accommodate new features. Patches are applied through a combination of automated and manual processes. Linux servers automatically install most security patches via an unattended upgrade process that is orchestrated by GitHub's configuration management system.

In addition to the production network vulnerability assessments, the GitHub Product Security Engineering team provides application security services to the Engineering teams. These services include secure architecture and code review, developing and maintaining internal automated security testing, and secure code training.

## BUG BOUNTY

The Bug Bounty team manages the GitHub Security Bug Bounty program, hosted on HackerOne. Members of the GitHub and security research community are encouraged to submit vulnerabilities through the program. The on-call security resource monitors the submissions and triages accordingly. If a bug is deemed to be legitimate, Security informs the relevant engineers, and the bug is tracked in GitHub issues to resolution. The Bug Bounty team issues bounty rewards for significant finds that lead to security improvements to the platform.



## PENETRATION TESTING

GitHub engages with a third-party security vendor to execute penetration and application security testing annually. The scope of this testing varies from year to year to focus on areas assessed as presenting significant risk, such as new features or services. Results are triaged and the risks reported are assessed against internal environmental considerations. Where remediation is required, solutions are identified and assigned to the relevant engineering teams for resolution with appropriate prioritization. Upon remediation, the fixes performed are shared with the contracted vendor to confirm successful remediation. A customer-facing report of the annual security testing is available to clients under mutual non-disclosure agreements (NDAs).

## SYSTEM HARDENING

System hardening standards have been documented, based on GitHub security practices. The Security Operations team reviews and approves changes to the standard hardening procedures.

Hardened system configurations are maintained in GitHub source code repositories and are deployed to production through a configuration management tool. This configuration management tool monitors system configurations and automatically reverts out-of-compliance systems to the approved hardening baseline.

Changes to the GitHub configurations are peer reviewed and approved prior to deploying the changes to production. Configuration change activities are logged in the GitHub's Puppet repository.

## NETWORK SECURITY

GitHub's network security is enforced through a number of process and configuration controls to protect against unauthorized access and help ensure security of data in transit. Both stateless and stateful network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary.

Access to production systems is brokered through security gateway systems at the production network perimeter. Three technologies provide access: VPN, bastion host, and SAML enforcing HTTPS reverse proxy.

Each of these remote access mechanisms require multiple factors for authentication and employ strong encryption in transit. Employees are provided access to production systems based on their role within the organization. Access is granted through the GitHub Entitlements system, which updates the internal Lightweight Directory Access Protocol (LDAP) store with the correct authorization rights for the user.

Access to machine accounts is limited and activity is logged and audited. Access to production network systems requires the use of an authorized SSH key with 2FA enabled.



GitHub personnel can only elevate privileges for their account and run commands with sudo if they are members of the correct Entitlements group, which is strictly controlled and reviewed. Any of these actions, including elevating to root require using the sudo command and are logged in the security information and event management (SIEM) tool. Alerting is configured to detect unusual or unauthorized activity and, when triggered, response is executed in accordance with the standard procedures defined within the SIRT's monitoring, detection, and response program. As needed, based on the frequency and type of activity detected, the SIRT identifies process improvements to reduce human interactions directly with production servers.

## SYSTEM MONITORING

THOR actively instruments and monitors a wide variety of data logging and telemetry sources across the organization. Production systems and corporate infrastructure transmitting, processing, or storing GitHub data are configured to generate and transmit security event logs. These include:

- *Enterprise* – system monitoring, audit logs, application logs, and network telemetry
- *Corporate* – endpoint system, network telemetry, cloud service, and application logs

These logs are actively monitored for a variety of known security issues, nefarious activity, malicious indicators, privileged actions, unauthorized access, and other anomalous activity. GitHub maintains a comprehensive security detection framework that governs how threat and anomaly detection is performed.

Detection alerts are generated by a variety of systems including log query tools, network or endpoint monitoring systems, or orchestration tooling. Each detection is assigned a severity and initial response SLA based on the risk represented by the event(s) it is designed to detect and surface. SLAs are as follows:

Severity	SLA	Intended Response	Example
Critical	30 minutes – 24/7	Immediate human response	Lost device
High	24 hours	Human response within one day	High confidence IOC match or behavioral detection
Medium	1-7 days	Automated or human response within a week	Low confidence IOC match
Low	N/A	Automated response or contextual only	Uptick in password changes from a single IP



Each alert generated by detection monitoring is assigned to the relevant security team or accountable individual at the time of creation, based on the severity assigned. The assigned team or individual records any actions taken or post-mortem conclusions in the relevant security repository issue for that event and resolves the issue when complete.

## INCIDENT RESPONSE

The THOR and SIRT teams triage, investigate, and respond to a variety of security events reported by internal and external sources, including:

- Suspected internal security compromises
- Critical dotcom vulnerabilities or bugs that may expose user or customer data
- GitHub or customer credential/secret exposure
- External requests for information on dotcom contents or fetches/page views in support of user or customer security incidents

GitHub maintains documented incident response procedures. These procedures are triggered once an event has been detected. GitHub security personnel are informed of security events, whether detected by systems or reported by humans, internal or external, with alerts surfaced to responders based on their severity.

Initial reports are defined as “leads” until appropriately triaged by a member of the relevant THOR or SIRT team member. Once either THOR or SIRT verifies a lead, an escalation procedure is executed to engage appropriate resources to help ensure potential risk of breach or privacy concerns are addressed in accordance with established protocols. Cross-functional accountabilities are documented for reference as escalation and remediation engagement varies depending on the source, scope of impact, and severity of the incident.

Roles for incident response and handling are defined and assigned, and a multi-stage process is followed until the risk has been mitigated and the outcomes documented. Root cause is assessed and addressed as part of the incident response process.

Incident severities are classified as follows; SLAs represent the time from the moment the lead is validated to mitigation of the root cause:

Severity	SLA	Potential Result
<b>Critical</b>	7 days	Catastrophic negative impact to the reputation of GitHub, GitHub personnel (Hubbers), users, or customers  Catastrophic sales losses and/or fiscal penalties associated with breach
<b>High</b>	14 days	Substantial negative impact to the reputation of GitHub, Hubbers, users, or customers  Substantial sales losses and/or fiscal penalties associated with breach



Severity	SLA	Potential Result
<b>Medium</b>	30 days	Moderate negative impact to the reputation of GitHub, Hubbers, users, or customers Moderate sales losses
<b>Low</b>	60 days	Limited to no negative impact to the reputation of GitHub, Hubbers, users, or customers No sales losses

The THOR and SIRT teams also document Legal engagement and external notification processes in the event an incident is validated as a breach or in violation of contractual commitments. GitHub informs affected users and organizations as required. The messaging can be sent via various forms such as email, a changelog post, a blog post, or release notes. Investigations specific to an affected customer are managed directly with that customer, in accordance with contractual terms.

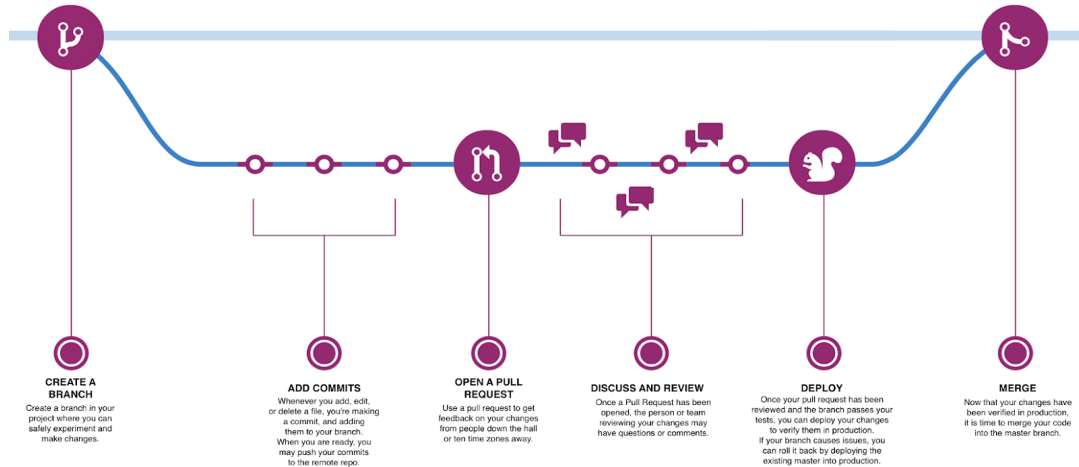
### c) *Change Management*

*Controls provide reasonable assurance that the development of new application functionality and software changes is documented, authorized, tested, and approved before being implemented in the production environment.*

GitHub is a collaborative, organic, and adaptable organization. GitHub's code and system change management processes leverage Enterprise Cloud's native source code and changelog technology available to users and organizations.

#### CHANGE DEVELOPMENT

GitHub application and configuration changes are developed and maintained within GitHub's private enterprise on the Enterprise Cloud product. Individual engineers are grouped by teams based on their areas of code ownership and expertise. Access to branch, deploy, approve, and merge changes to critical Enterprise Cloud GitHub repositories is restricted to members of the Platform and Product Engineering teams. Engineers responsible for the critical EC Repositories of the codebase receive timely notifications of changes pushed and monitor these changes through automated issues posted in the Enterprise Cloud repositories and in Slack.



When developing code for the Enterprise Cloud product, engineers create a branch from the Default Branch, which is the 'gold copy' of GitHub, to begin development of changes. This ensures that test environments are logically separated from production environments. Changes made on these branches are deployed to a pre-production environment for testing. Once pre-production testing has validated the change is functioning as expected, the branch is committed to the Default Branch and is rolled out in staged releases via Canary deployments to GitHub's Kubernetes clusters in production for customer use.

An automated static code analysis tool runs every time new code is committed, to detect insecure coding practices based on third-party provided lexical, syntactic, and semantic rulesets, in addition to GitHub-developed rulesets and test scenarios. An automated alerting issue is posted on the containing issue and in a tracking repository maintained by the Application Security team, to alert them of any potentially insecure coding risks to product development. The developer assesses the potential vulnerabilities, and a peer reviewer follows up on the items during change review and approval.

### PEER REVIEW

GitHub relies heavily on collaboration and peer reviews to help ensure the integrity of the systems. Engineers are trained on the requirements for performing an effective peer review, and each team overlays domain-specific training and checks to the review. The Security team conducts high-level design reviews, including security reviews, and release reviews for large projects and initiatives to assess security risk. For production changes, a code review and, where relevant, a review of database schema occurs to verify the integrity of the changes.

Once the code is ready for discussion and review, developers open a PR for commits added to their branch. PRs are an integral feature of Enterprise Cloud used to coordinate the discussion, approval, testing, and deployment of changes.





Enterprise Cloud repositories come with the ability to designate specific users as “Code Owners.” Code Owners are responsible for reviewing and approving PRs to ensure the appropriateness of commits. Additionally, repository owners have the ability to create a “Code Owners File” that acts as a configuration for the repository contents, enforcing required reviews by specifically nominated users prior to deploying PRs to production and then as part of a merge action to the repository’s Main branch. Specific users may be designated as mandatory reviewers for all or select portions of the repository’s contents/code.

Protected branches (branches where merging to the Main branch is gaited by required review and approval activities) are also enabled on the Enterprise Cloud code repositories, requiring commits to branches via PRs to be reviewed by an independent reviewer prior to merging the code to the Default Branch.

### AUTOMATED TESTING

Continuous Integration (CI) is the process by which code changes are automatically tested and integrated into a single PR. When a change is committed to a branch with an associated PR, the CI system automatically executes the configured CI status checks for that repository. This process includes the following:

- CI is notified of changes to the repository and builds the code for testing.
- CI runs the test suites selected for that repository including end-to-end integration testing, functional/unit testing, and security testing.
- CI successes and failures are referenced in the PR.

Failures in CI will block merging and deploying of those changes. CI testing occurs every time a change is committed and when a change is being deployed to production. GitHub CI testing uses a suite of tools to automate build and test on isolated hosts. Test results are automatically exported and alerted on the respective issues allowing teams to monitor and act on issues prior to deployment.

GitHub engineers collaborate to support the creation of CI test scripts, which include regression test cases, and the implementation of blocking acceptance tests. These acceptance tests typically function at the browser level and focus on the overall functionality of the module or application to help ensure customer commitments and system requirements are met. GitHub prevents any change from deployment to production if required acceptance tests have not passed.

Peer review is the primary method used by GitHub to help ensure the appropriate level of test cases have been implemented prior to deploying changes. Code Owners are domain experts over repositories and/or specific areas of the code. These individuals are ultimately responsible for helping to ensure tests are created and deprecated in accordance with the functional requirements.



## DEPLOYMENT

To support deployment to the production environment, changes are deployed using GitHub's standard and proprietary deployment tools and workflows. GitHub's automated deployment tool, Heaven, sits within GitHub's production environment to manage packages deployed to the production systems. To deploy any application or configuration changes, developers run a Slack Chat Operations (ChatOps) deployment command on the respective team channels associated with the production application. Running the Slack ChatOps creates a traceable record of the action and helps to ensure accountability and transparency within the team and other stakeholders subscribed to the respective Slack channel. Furthermore, Heaven ensures required checks are completed, including passing of mandatory acceptance checks and ensuring the most recent Default Branch build is used, before deploying the change to production. Code commits, whether direct commits or through a pull request, are hashed. The hash is displayed with the commit to facilitate code integrity verification.

## MONITORING, METRICS, AND CHANGE ROLLBACK

To help ensure continued functionality, availability, and security of the Enterprise Cloud product, comprehensive logging and monitoring tools are built into the Enterprise Cloud application, system, and network to monitor real-time telemetrics such as network traffic, successful logins and repository actions, exceptions and error messages from the application and systems, and usage statistics to detect anomalies through fine-tuned fault tolerances and historical trend analyses.

Engineering teams monitor each system component post-deployment to help ensure changes implemented did not detrimentally impact stability. A dedicated Site-Reliability Engineering team monitors and responds to incidents 24/7 through real-time, automated monitoring and incident escalation workflow. In the event of any detrimental impact to production resulting from a change, changes are rolled back to the last known stable version of the Default Branch. As a matter of practice, GitHub uses branch deployments to production, so the roll-back procedure is simple and can be performed by anyone with authorization to deploy.

Once a deployment to production is shown to be faulty or unstable, based on the continuous monitoring described previously, manual testing, or some other measure, the assigned engineer runs a single ChatOps command and the previous version of the Default Branch is deployed back to production. Depending on the severity of the findings, the engineer can make immediate changes and redeploy, or unlock deploys and begin testing again in limited environments.

To keep GitHub personnel in the loop on new feature development or other engineering efforts, the Product teams regularly host internal demos for the organization as a part of regular company and technology all-hands get togethers. These presentations are a way for technology teams to discuss and share projects implemented and goals achieved throughout the quarter.



New customer-impacting product and feature releases, defined as “notable changes”, go through a standardized release process to help ensure the appropriate level of communication given the type of change. A notable change is anything added, changed, deprecated, removed, fixed, or any security fixes (CVEs) that customers should understand. These changes could affect the productivity or workflow of the user, or in the case of security fixes, require notification and awareness for customers. For example, notable changes include API deprecations, User Interface (UI) improvements, or additions to payment options. These changes are communicated at several discrete stages of development and prior to launch. Methods of communication include GitHub blog posts and may also include social posts, and product training-videos.

For new features and major changes, technical reviews are performed to assess security, data, and architecture risk. Cross functional approval is required prior to release of changes to production.

### EMERGENCY DEPLOYS

When an emergency or system failure means a deployment cannot follow the normal procedures, there is an established process to allow engineers to circumvent the usual tooling and force a deployment into production. The emergency deployment process, and knowing when it is justified, is a controlled and necessary component of deployment processes.

Engineers with write access to a repository and privileges to deploy to production have the ability to force deploy, which bypasses required CI and normal approval processes. Forced deployments are requested in the same pre-defined deployment Slack channels used for regular deployments, and the engineer requesting the deployment specifies the emergency justification to give context to the situation to observers and others who are monitoring these deployments.

After an engineer force deploys, the deploy queue is blocked for that repository until the engineer reverts the changes or merges the changes into the Default Branch. Both of these paths require the normally required CI and peer review to complete, ensuring the same level of visibility and testing as any other change to the code base.

When an emergency deployment is executed, an alert is posted to the #incident-command Slack channel for visibility to the on-call team. A post-mortem remediation issue is automatically created in the github/availability repository. The issue is assigned to the engineer who deployed with a post-mortem checklist the assignee completes within 24 hours, giving justification for the deployment.

Post-mortem remediation issues are reviewed weekly in the Production Engineering Availability team meeting. The deploying engineer is invited to the meeting to review the issue and the circumstances, and to verify they have completed the expected post-deployment steps to help ensure no further issues require resolution.



#### **d) Backup and Recovery**

*Controls provide reasonable assurance that system data is regularly backed up and archived data is available for restoration in the event unexpected interruptions occur.*

Data in Git file and database servers is backed up in real time. These backups are encrypted as they are created. The backups from the Git file servers and database servers are maintained in geographically distinct AWS Simple Storage Solution (S3) data center locations.

The Storage Engineering team monitors the health of the Git backups. When delays in the backup jobs are encountered, the Storage Engineering team is notified. Issues requiring intervention are resolved. Common issues that cause backup job delays are documented in the Storage Engineering team's GitHub repository.

A contingency plan has been created that outlines how to recover the system in the event of disaster or other scenarios. The plan is reviewed and tested annually. Any changes identified in the review or test results are incorporated into the plan.

#### **e) Repository History**

*Controls provide reasonable assurance that repository history is complete and accurate.*

Repositories within the application track the history of actions taken within them. Assigned users can add content to the codebase, create issues, and execute pull requests. The history of actions within a specific issue, pull request, or codebase are recorded in the repository's history.

Within an issue, the ability to edit information is restricted to the individual contributor. If an edit is made, evidence of the comment is retained in the issue history, including the name of the contributor and the date and time of the original comment and the date and time it was edited.

Edits to the code are recording in the history. Each edit includes the code changes performed, who performed the change, and the date and time the change occurred.

Pull requests can be used to share commit information with collaborators in the repository. Once submitted, pull request information cannot be edited; however, pull requests can be reverted. If reverted, the revert request links to the original pull request. The original pull request information is not altered after the revert request is sent.

When new code changes are committed to a repository, inherent Git functionality hashes the commit. github.com displays the hash value along with the commit. The hash value can be used by individuals to verify the integrity of the commit. Additionally, GitHub Enterprise Cloud allows users to add their own GPG key that can be used to digitally sign their commits.



## ***f)* Vendor Management**

*Controls provide reasonable assurance that vendors are assessed for their ability to meet security compliance requirements.*

To initiate a vendor relationship, the Legal and Procurement teams negotiate and manage the vendor contract clauses and additional data protection agreements with vendors who process or store GitHub data, customer data, or employee data, as well as systems that connect to GitHub systems.

The Security GRCC team manages the vendor security risk assessment process. GitHub maintains operational processes to assess security risk considerations related to vendors. These vendors are required to undergo an initial security risk assessment prior to contracting with GitHub. Those vendors who do not meet GitHub's baseline security requirements in alignment with their defined business use case do not move forward for procurement.

The Security GRCC team reviews vendor security risk assessments every two years, or upon expansion or changes to the contracted service offering. Vendors deemed high risk, such as data center providers or other vendors storing or processing data in scope for GitHub's regulatory or contractual requirements, undergo reassessment annually.

## **D. Control Objectives and Related Controls**

GitHub has specified the control objectives and identified the controls that are designed to achieve the related control objective. The specified control objectives and related controls are presented in Section IV, "Description of GitHub's Control Objectives and Related Controls, and Independent Service Auditor's Description of Tests of Controls and Results," and are an integral component of GitHub's Enterprise Cloud.

## **E. Complementary Subservice Organization Controls**

GitHub's controls related to the Enterprise Cloud cover only a portion of overall internal control for each user entity of GitHub. It is not feasible for the control objectives related to the Enterprise Cloud to be achieved solely by GitHub. Therefore, each user entity's internal control over financial reporting must be evaluated in conjunction with GitHub's controls and the related tests and results described in Section IV of this report, taking into account the types of controls expected to be implemented by the subservice organization as described below.

Complementary Subservice Organization Controls		Related Control Objective
AWS, Azure		
1	Access to hosted systems requires strong authentication mechanisms.	➤ <b>Control Objective 1:</b> Physical and Logical Access
2	User content is segregated and made viewable only to authorized individuals.	➤ <b>Control Objective 1:</b> Physical and Logical Access



Complementary Subservice Organization Controls		Related Control Objective
3	Network security mechanisms restrict external access to the production environment to authorized ports and protocols	➤ <b>Control Objective 1:</b> Physical and Logical Access
4	New and existing user access and permissions to hosted systems are approved by appropriate personnel prior to be granted.	➤ <b>Control Objective 1:</b> Physical and Logical Access
5	Terminated user access permissions to hosted systems are removed in a timely manner.	➤ <b>Control Objective 1:</b> Physical and Logical Access
6	User access permissions to hosted systems are reviewed by appropriate personnel on a regular basis.	➤ <b>Control Objective 1:</b> Physical and Logical Access
7	Access to the physical facilities housing hosted systems is restricted to authorized users.	➤ <b>Control Objective 1:</b> Physical and Logical Access
8	Connections to the production environment require encrypted communications.	➤ <b>Control Objective 1:</b> Physical and Logical Access
9	Changes to hosted systems are documented, tested, and approved prior to migration to production.	➤ <b>Control Objective 3:</b> Change Management
10	Access to make changes to hosted systems is restricted to appropriate personnel.	➤ <b>Control Objective 3:</b> Change Management
11	Separate production and development environments are maintained.	➤ <b>Control Objective 3:</b> Change Management
<b>QTS, Sabey, CoreSite, Equinix</b>		
1	Access to hosted systems requires strong authentication mechanisms.	➤ <b>Control Objective 1:</b> Physical and Logical Access
2	Network security mechanisms restrict external access to the production environment to authorized ports and protocols	➤ <b>Control Objective 1:</b> Physical and Logical Access
3	New and existing user access and permissions to hosted systems are approved by appropriate personnel prior to be granted.	➤ <b>Control Objective 1:</b> Physical and Logical Access
4	Terminated user access permissions to hosted systems are removed in a timely manner.	➤ <b>Control Objective 1:</b> Physical and Logical Access
5	User access permissions to hosted systems are reviewed by appropriate personnel on a regular basis.	➤ <b>Control Objective 1:</b> Physical and Logical Access
6	Access to the physical facilities housing hosted systems is restricted to authorized users.	➤ <b>Control Objective 1:</b> Physical and Logical Access



## F. Complementary User Entity Controls

GitHub's controls related to the Enterprise Cloud cover only a portion of overall internal control for each user entity of GitHub. It is not feasible for the control objectives related to the Enterprise Cloud to be achieved solely by GitHub. Therefore, each user entity's internal control over financial reporting should be evaluated in conjunction with GitHub's controls and the related tests and results described in Section IV of this report, taking into account the related complementary user entity controls identified below, where applicable.

In order for user entities to rely on the controls reported on herein, each user entity must evaluate its own internal control to determine whether the identified complementary user entity controls have been implemented and are operating effectively. User entities are responsible for:

Complementary User Entity Controls		Related Control Objective	
1	Creating and managing their Organization and Teams, including the proper configuration of access permissions to repositories.	➤	<b>Control Objective 1:</b> Physical and Logical Access
2	Inviting, removing, and managing users in their Organization and Teams, including granting of permission levels and access to repositories, and periodic review of Organization users and outside collaborators.	➤	<b>Control Objective 1:</b> Physical and Logical Access
3	Ensuring authorized users are appointed as Organization owners for administration of the Organization.	➤	<b>Control Objective 1:</b> Physical and Logical Access
4	Enabling SAML for their Enterprise Cloud accounts.	➤	<b>Control Objective 1:</b> Physical and Logical Access
5	Enabling two-factor authentication and ensuring that members and collaborators require two-factor authentication.	➤	<b>Control Objective 1:</b> Physical and Logical Access
6	Maintaining an effective onboarding and offboarding process for their own employees and contractors.	➤	<b>Control Objective 1:</b> Physical and Logical Access
7	Administering and configuring repositories and workflows (GitHub Actions), including permissions, requiring reviews for pull requests, and enabling required status checks before merging.	➤	<b>Control Objective 1:</b> Physical and Logical Access
8	Performing audits of events in the security logs.	➤	<b>Control Objective 5:</b> Repository History
9	Configuring their repositories to utilize GitHub commit signing.	➤	<b>Control Objective 5:</b> Repository History
10	Providing their own GPG keys for commit signing.	➤	<b>Control Objective 5:</b> Repository History



## IV. Description of GitHub's Control Objectives and Related Controls, and Independent Service Auditor's Description of Tests of Controls and Results

### A. Information Provided by the Independent Service Auditor

This integrated SOC 1 Type 2 Report was prepared in accordance with the AICPA attestation standards and in accordance with the International Standard on Assurance Engagements 3402, *Assurance Reports on Controls at a Service Organization*, issued by the International Auditing and Assurance Standards Board and, when combined with an understanding of the controls at user entities, is intended to assist auditors in planning the audit of user entities' financial statements or user entities' internal control over financial reporting and in assessing control risk for assertions in user entities' financial statements that may be affected by controls at GitHub.

Our examination was limited to the control objectives and related controls specified by GitHub in Sections III and IV of the report, and did not extend to controls in effect at user entities.

It is the responsibility of each user entity and its independent auditor to evaluate this information in conjunction with the evaluation of internal control over financial reporting at the user entity in order to assess total internal control. If internal control is not effective at user entities, GitHub's controls may not compensate for such weaknesses.

GitHub's internal control represents the collective effect of various factors on establishing or enhancing the effectiveness of the controls specified by GitHub. In planning the nature, timing, and extent of our testing of the controls to achieve the control objectives specified by GitHub, we considered aspects of GitHub's control environment, risk assessment process, monitoring activities, and information and communications.

The following table clarifies certain terms used in this section to describe the nature of the tests performed:

Test Procedure	Description
<b>Inquiry</b>	➤ Inquiry of appropriate personnel and corroboration with management.
<b>Observation</b>	➤ Observation of the application, performance, or existence of the control.
<b>Inspection</b>	➤ Inspection of documents and reports indicating performance of the control.
<b>Reperformance</b>	➤ Reproduction of the control.

In addition, we evaluated whether the information was sufficiently reliable for our purposes by obtaining evidence about the accuracy and completeness of such information and evaluating whether the information was sufficiently precise and detailed for our purposes.





## B. Test of Controls and Results

### 1. Physical and Logical Access

Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 1:</b> Controls provide reasonable assurance that physical and logical access to systems and customer data is controlled and limited to properly authorized users.		
1. Remote access to internal administration tools is restricted through unique account IDs, passwords, and a one-time token.	<p>Inquired of the Security Operations Manager to confirm remote access to internal administration tools was restricted through unique account IDs, passwords, and a one-time token.</p> <p>Observed the Security Operations Manager authenticate to a sample of the internal administration tools to determine whether these administration tools were restricted through unique account IDs, passwords, and a one-time token.</p> <p>Inspected the production user access listing, the VPN authentication configuration, and the bastion host 2FA configuration to determine whether remote access to internal administration tools required unique account IDs, passwords, and a one-time token.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p>
2. Remote access to Enterprise Cloud production systems is restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which require 2FA.	<p>Inquired of the Security Operations Manager to confirm remote access to Enterprise Cloud production systems was restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which required 2FA.</p> <p>Inspected the authentication configuration to determine whether remote access to Enterprise Cloud production systems was restricted through VPN, bastion hosts, or SAML enforcing HTTPS reverse proxy, which required 2FA.</p> <p>Observed the Security Operations Manager authenticate to Enterprise Cloud production systems to determine whether either a VPN, bastion host, or SAML connection was required.</p> <p>Observed the Security Operations Manager connect over the VPN, the bastion host, and SAML to determine whether the VPN, bastion host, and SAML required 2FA.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p>



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 1:</b> Controls provide reasonable assurance that physical and logical access to systems and customer data is controlled and limited to properly authorized users.		
3. Server and database administrative privileges are restricted to the respective system owners.	<p>Inquired of the Data Infrastructure Engineer to confirm server and database administrative privileges were restricted to the respective system owners.</p> <p>Inspected access listings for servers and databases and employee records to determine whether server and database administrative privileges were restricted to the respective system owners.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p>
4. GitHub policy prohibits the use of shared administrative accounts for Enterprise Cloud production systems. Exceptions to the policy are documented in the Exceptions repository.	<p>Inquired of the Security Operations Manager to confirm GitHub policy prohibited the use of shared administrative accounts for Enterprise Cloud production systems; and exceptions to this policy were documented in the exceptions repository.</p> <p>Inspected the Hubber Handbook to determine whether GitHub prohibited the use of shared administrative accounts for Enterprise Cloud production systems.</p> <p>Inspected the user access listing to Enterprise Cloud production systems to determine whether GitHub identified shared administrative accounts for Enterprise Cloud production systems.</p> <p>Inspected account user lists and the exceptions repository to determine whether shared account exceptions were documented.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p>
5. A unique account and password are required to authenticate customers to their organization in Enterprise Cloud.	<p>Inquired of the Engineering Manager to confirm a unique account and password were required to authenticate customers to their organization in Enterprise Cloud.</p> <p>Inspected the authentication code configuration to determine whether a unique account and password were required to authenticate customers to their organization in Enterprise Cloud.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p>



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 1:</b> Controls provide reasonable assurance that physical and logical access to systems and customer data is controlled and limited to properly authorized users.		
6. Users are only able to access their personal repositories and the repositories where they are granted access.	<p>Inquired of the Engineering Manager to confirm users were only able to access their personal repositories and the repositories where they were granted access.</p> <p>Inspected the GitHub.com repository configuration to determine whether user access was limited to a user's personal repositories and repositories where they were granted access.</p> <p>Observed the repository access restrictions in the system to determine whether a test user was properly restricted from accessing a test repository where the user did not have access.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p>
7. GitHub uses SSH to authenticate to back-end production resources through a bastion host.	<p>Inquired of the Site Reliability Engineer to confirm GitHub used SSH to authenticate to back-end production resources through a bastion host.</p> <p>Inspected the authentication configuration to determine whether GitHub uses SSH to authenticate to back-end production resources through a bastion host.</p> <p>Observed a user authenticate to back-end production resources to determine whether SSH was required for the user to authenticate.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p>
8. VPN and bastion hosts are configured to require user connections to reauthenticate after 24 hours of inactivity.	<p>Inquired of the Security Operations Manager to confirm VPN and bastion hosts were configured to require user connections to reauthenticate after 24 hours of inactivity.</p> <p>Inspected the VPN and bastion hosts inactivity configuration to determine whether VPN and bastion host user connections were required to reauthenticate after 24 hours of inactivity.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p>



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 1:</b> Controls provide reasonable assurance that physical and logical access to systems and customer data is controlled and limited to properly authorized users.		
9. GitHub's internal systems are configured to automatically provision logical and physical access based on the user's job role. GitHub's internal systems are configured to automatically provision non role-based entitlements only after the user's manager reviews and approves the access request.	Inquired of the Security Operations Manager to confirm GitHub's internal systems were configured to automatically provision logical and physical access based on the user's job role.	No exceptions noted.
	Inquired of the Security Operations Manager to confirm GitHub's internal systems were configured to automatically provision non role-based entitlements only after the user's manager reviewed and approved the access request.	No exceptions noted.
	Inspected the system configurations for the role-based entitlements to determine whether GitHub's internal systems were configured to automatically provision logical and physical access based on the user's job role.	No exceptions noted.
	Inspected the system configurations for the non-role-based entitlements to determine whether GitHub's internal systems were configured to automatically provision logical and physical access only after the user's manager reviewed and approved the access request.	No exceptions noted.
	Inspected a pull request for a sample user who received role-based entitlements to determine whether these user entitlements were granted to this user based on the user's role.	No exceptions noted.
	Inspected a pull request for a sample user who received non role-based entitlements to determine whether these user entitlements were granted only after the user's manager reviewed and approved the access request.	No exceptions noted.
	Inspected the entitlements repository revision history and the change population to determine whether these configurations were in place throughout the audit period.	No exceptions noted.



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 1:</b> Controls provide reasonable assurance that physical and logical access to systems and customer data is controlled and limited to properly authorized users.		
10. The Security Operations team revokes logical production access and Production Engineering revokes physical production access for terminated personnel (employee and contractors) within 24 hours of termination.	<p>Inquired of the Security Operations Manager to confirm the Security Operations team revoked logical production access and Production Engineering revoked physical production access for terminated personnel (employee and contractors) within 24 hours of termination.</p> <p>Inspected termination checklists for a sample of terminated personnel to determine whether Security Operations documented the revocation of production access and Production Engineering documented the revocation of physical access for these personnel within 24 hours of their termination.</p> <p>Inspected the production user access levels and physical access levels for a sample of terminated users to determine whether their physical and logical access to production systems was revoked.</p>	<p>No exceptions noted.</p> <p>For 1 of 25 terminated personnel sampled, access was not revoked within 24 hours of termination.</p> <p>See <i>Section V - Other Information Provided by GitHub That Is Not Covered by the Service Auditor's Report</i> for management's response to the noted exceptions.</p> <p>No exceptions noted.</p>
11. Semiannually, the Security Operations team reviews and reauthorizes elevated access permissions to confirm users with this access are restricted based on the principle of least privilege.	<p>Inquired of the Director, Security GRCC to confirm the Security Operations team reviewed elevated access permissions semiannually to confirm users with this access were restricted based on the principle of least privilege; and the access reviewed was reauthorized at the end of the review.</p> <p>Inspected review documentation from both semiannual security reviews performed during the examination period to determine whether the Security Operations team reviewed and reauthorized elevated access permissions.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p>



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 1:</b> Controls provide reasonable assurance that physical and logical access to systems and customer data is controlled and limited to properly authorized users.		
12. The Infrastructure team reviews physical access to production data centers on an annual basis; any unapproved accounts are removed.	Inquired of the Director, Security GRCC to confirm the Infrastructure team reviewed physical access to production data centers on an annual basis; and any unapproved accounts were removed.	No exceptions noted.
	Inspected physical access review documentation to determine whether the Infrastructure team reviewed physical access to production data centers during the examination period.	No exceptions noted.
	Inspected physical access lists and physical access review documentation to determine whether unapproved accounts were removed.	No exceptions noted.



## 2. Information Security

Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 2:</b> Controls provide reasonable assurance that confidentiality and information security practices protect the production environment.		
1. GitHub maintains and communicates key security standards and procedures on the GitHub intranet that address the security of systems, facilities, data, personnel, and processes.	Inquired of the Director, Security GRCC to confirm GitHub maintained and communicated key security standards and procedures on the GitHub intranet; and these standards and procedures addressed the security of systems, facilities, data, personnel, and processes.	No exceptions noted.
	Inspected the GitHub intranet permissions to determine whether key security standards and procedures were made available and communicated to employees.	No exceptions noted.
	Inspected key security standards and procedures on the GitHub intranet to determine whether GitHub maintained these security standards and procedures that addressed the security of systems, facilities, data, personnel, and processes.	No exceptions noted.
2. Security Operations scans internal and external systems monthly. Security Operations reviews identified observations, shares confirmed threats with responsible stakeholders, and tracks these issues to resolution.	Inquired of the Manager, Security Operations to confirm Security Operations scanned internal and external systems monthly, reviewed identified observations, and shared confirmed threats with responsible stakeholders.	No exceptions noted.
	Inspected the vulnerability scanning configuration to determine whether internal and external security vulnerability scanning was configured to execute on a monthly basis.	No exceptions noted.
	Inspected ticketing system documentation for a sample of identified vulnerabilities to determine whether Security Operations reviewed these observations and shared confirmed threats with responsible stakeholders.	No exceptions noted.



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 2:</b> Controls provide reasonable assurance that confidentiality and information security practices protect the production environment.		
3. System owners remediate vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities. Vulnerabilities that cannot be remediated within the required SLA are handled via the documented exception process.	Inquired of the Manager, Security Operations to confirm system owners remediated vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities; and vulnerabilities that could not be remediated within the required SLA were handled via the documented exception process.	No exceptions noted.
	Inspected the Vulnerability Management Process Procedure to determine whether system owners were required to remediate vulnerabilities in accordance with established SLAs based on the severity of the vulnerabilities; and whether vulnerabilities that could not be remediated within the required SLA were handled via the documented exception process.	No exceptions noted.
	Inspected tickets for a sample of vulnerabilities to determine whether these vulnerabilities were remediated within SLA or were handled via the documented exception process; and whether these remediated vulnerabilities were completed in accordance with established SLAs based on the severity of the vulnerabilities.	No exceptions noted.
4. GitHub operates a bug bounty program. The on-call security resource monitors the submissions and triages accordingly. If a bug is deemed to be legitimate, Security informs the relevant engineers and the bug is tracked in GitHub issues to resolution.	Inquired of the Security Engineering Senior Manager to confirm GitHub operated a bug bounty program; on-call security resources monitored the submissions and triaged accordingly; and for legitimate bugs, Security informed the relevant engineers and the bug was tracked in GitHub issues to resolution.	No exceptions noted.
	Observed the functionality of the bug bounty platform to determine whether researchers were able to report bugs.	No exceptions noted.
	Inspected tracking tickets for a sample of bugs in the bug tracking repository to determine whether these bugs were triaged and tracked to resolution.	No exceptions noted.





Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 2:</b> Controls provide reasonable assurance that confidentiality and information security practices protect the production environment.		
5. The Security GRCC team assesses internal security controls through sample-based testing to ensure that control design and operation continues to meet defined criteria and system requirements.	Inquired of the Director, Security GRCC to confirm the Security GRCC team assessed internal security controls annually through sample-based testing to help ensure control design and operation continued to meet defined criteria and system requirements.	No exceptions noted.
	Inspected internal control test results to determine whether the Security GRCC team assessed internal security controls during the examination period through sample-based testing to help ensure control design and operation continued to meet defined criteria and system requirements.	No exceptions noted.
6. GitHub has defined hardening standards based on its security hardening practices. The Security Operations team reviews and approves changes to the standard hardening procedures.	Inquired of the Director, Security GRCC to confirm GitHub had defined hardening standards based on GitHub security hardening practices; and the Security Operations team reviewed and approved changes to the standard hardening procedures.	No exceptions noted.
	Inspected standard hardening procedures to determine whether GitHub had defined hardening standards based on its security hardening practices.	No exceptions noted.
	Inspected revision history approval documentation for a sample of changes to standard hardening procedures to determine whether the Security Operations team reviewed and approved these changes to the standard hardening procedures.	No exceptions noted.
7. Network firewalls restrict external points of connectivity and prevent unauthorized traffic from beyond the system boundary.	Inquired of the Site Reliability Engineer to confirm network firewalls restricted external points of connectivity and prevented unauthorized traffic from beyond the system boundary.	No exceptions noted.
	Inspected firewall configurations to determine whether they restricted external points of connectivity, and whether they were configured to prevent unauthorized traffic from beyond the system boundary.	No exceptions noted.
8. Production traffic is encrypted when transmitted over the public internet.	Inquired of the Security Analyst to confirm production traffic was encrypted when transmitted over the public internet.	No exceptions noted.
	Inspected www.github.com and the corresponding encryption certificate to determine whether production traffic over the internet was encrypted.	No exceptions noted.



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 2:</b> Controls provide reasonable assurance that confidentiality and information security practices protect the production environment.		
9. Logging and monitoring tools are used to collect telemetry and event logs from network devices, applications, and systems. Security alerts are generated, prioritized, and communicated to the designated responders for follow up. GitHub Security teams use these alerts to initiate triage and, if necessary, remediate security issues based on their assigned priority.	Inquired of the SIRT Manager to confirm logging and monitoring tools were used to collect telemetry and event logs from network devices, applications, and systems; and security alerts were generated and communicated to designated responders, and prioritized in a specific GitHub Security team repository for triage, tracking, and remediation.	No exceptions noted.
	Inspected the logging and monitoring tool configuration to determine whether this tool was configured to capture telemetry and event logs from network devices, applications, and systems.	No exceptions noted.
	Inspected the security monitoring tools and security event repositories for a sample of alerts to determine whether these security alerts were prioritized and communicated to designated responders for follow up.	No exceptions noted.
	Inspected the GitHub Security team notification channel and the security event repository for a sample of alerts to determine whether these events were logged and tracked to remediation.	No exceptions noted.
10. GitHub Support receives reports of security issues from customers via emails or through the web form. A ticketing system is used to document and track these issues to resolution.	Inquired of the SIRT Manager to confirm GitHub Support received reports of security issues from customers via emails or through the web form; and a ticketing system was used to document and track these issues to resolution.	No exceptions noted.
	Inspected the customer support communication channels to determine whether security events from customer emails or web requests were available and recorded.	No exceptions noted.
	Inspected tickets for a sample of issues to determine whether these issues were documented and tracked to resolution.	No exceptions noted.



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 2:</b> Controls provide reasonable assurance that confidentiality and information security practices protect the production environment.		
11. A third party performs an annual application penetration assessment. Security Operations triages and tracks identified issues through to resolution.	Inquired of the Director of Application Security Engineering to confirm a third party performed an annual application penetration assessment; and Security Operations triaged and tracked identified issues through to resolution.	No exceptions noted.
	Inspected the penetration test report to determine whether a third party performed an application penetration assessment during the examination period.	No exceptions noted.
	Inspected the penetration test report and tracking documentation to determine whether Security Operations triaged and tracked identified issues through to resolution.	No exceptions noted.



### 3. Change Management

Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 3:</b> Controls provide reasonable assurance that the development of new application functionality and software changes is documented, authorized, tested, and approved before being implemented in the production environment.		
1. Test environments are logically separated from production environments.	<p>Inquired of the Technical Project Manager to confirm test environments were logically separated from production environments.</p> <p>Inspected the configuration of production and test environment clusters to determine whether these environments were logically separated.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p>
2. GitHub engineering code owners create and execute automated regression test cases to address functionality and security requirements.	<p>Inquired of the Technical Project Manager to confirm GitHub engineering code owners created and executed automated regression test cases to address functionality and security requirements.</p> <p>Inspected the automated regression test cases for a sample of code changes to determine whether GitHub engineering code owners created and executed test cases to address functionality and security requirements for these changes.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p>
3. A security code analysis tool scans code during commits and merges. The scanning tool posts potential vulnerabilities on the change pull request. The developer assesses the potential vulnerabilities and a peer reviewer follows up on the items during change review and approval.	<p>Inquired of the Director, Application Security Engineering to confirm a security code analysis tool scanned code during commits and merges, the scanning tool posted potential vulnerabilities on the change pull request, the developer assessed the potential vulnerabilities, and a peer reviewer followed up on the items during change review and approval.</p> <p>Inspected automated code security analysis results for a sample of code changes to determine whether an automated code security analysis was performed for these changes during commits and merges.</p> <p>Inspected pull requests for a sample of code changes to determine whether potential vulnerabilities identified for these changes were posted on the change pull request, whether the developer assessed the potential vulnerabilities, and whether a peer reviewer followed up on these items during change review and approval.</p>	<p>No exceptions noted.</p> <p>No exceptions noted.</p> <p>No exceptions noted.</p>



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 3:</b> Controls provide reasonable assurance that the development of new application functionality and software changes is documented, authorized, tested, and approved before being implemented in the production environment.		
4. Application changes pass automated continuous integration testing prior to deployment to the production environment.	Inquired of the Technical Project Manager to confirm application changes passed automated continuous integration testing prior to deployment to the production environment.	No exceptions noted.
	Inspected pull request documentation for a sample of changes to determine whether these application changes passed automated continuous integration testing prior to deployment to the production environment.	No exceptions noted.
5. An independent reviewer evaluates and approves system change requests via a GitHub pull request.	Inquired of the Director, Software Engineering to confirm an independent reviewer evaluated and approved system change requests via a GitHub pull request.	No exceptions noted.
	Observed and inspected the pull request documentation for a sample of code changes to determine whether an independent reviewer evaluated and approved these changes.	No exceptions noted.
6. Branch protection system configurations enforce peer reviews and integration tests prior to the deployment of changes to the production environment and alert code owners of any forced deployments.	Inquired of the Technical Project Manager to confirm branch protection system configurations enforced peer reviews and integration tests prior to the deployment of changes to the production environment and alerted code owners of any forced deployments.	No exceptions noted.
	Inspected branch protection system configurations for the GitHub and puppet repositories to determine whether system configurations enforced peer reviews and integration tests for these repositories prior to the deployment of changes to the production environment and alerted code owners of any forced deployments.	No exceptions noted.
	Inspected tickets for a sample of code deployments to the production environment to determine whether peer reviews and integration tests were completed prior to the deployment of these changes to the production environment.	No exceptions noted.
	Inspected an alert for a sample forced deployment to determine whether the assigned code owner was alerted of the forced deployment.	No exceptions noted.



Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 3:</b> Controls provide reasonable assurance that the development of new application functionality and software changes is documented, authorized, tested, and approved before being implemented in the production environment.		
7. The Product Engineering team monitors emergency deployments on a real-time basis, and post-mortem reviews are performed for emergency changes impacting production systems.	Inquired of the Technical Project Manager to confirm emergency deployments were monitored on a real-time basis, and post-mortem reviews were performed for emergency changes that impacted production systems.	No exceptions noted.
	Inspected the Software Development Lifecycle Policy to determine whether this policy defined requirements for real-time emergency deployment monitoring, and post-mortem reviews for emergency changes impacting production systems.	No exceptions noted.
	Inspected deploy monitoring tools to determine whether the Product Engineering team monitored emergency deployments on a real-time basis.	No exceptions noted.
	Inspected the post-mortem reviews for a sample of emergency code changes to determine whether a post-mortem review was performed for these changes.	From February 25, 2022, to June 22, 2022 real-time reviews of emergency deployments were not performed in a timely manner.  See Section V - Other Information Provided by GitHub That Is Not Covered by the Service Auditor's Report for management's response to the noted exceptions.
8. Code commits, whether direct commits or through a pull request, are hashed. The hash is displayed with the commit to facilitate code integrity verification.	Inquired of the Director, Security GRCC to confirm code commits, whether direct commits or through a pull request, were hashed; and the hash was displayed with the commit to facilitate code integrity verification.	No exceptions noted.
	Observed the submission of a test direct code commit and a test pull request to determine whether these test commits were hashed.	No exceptions noted.
	Inspected test commits for a sample of changes to determine whether the hash was visible with these commits.	No exceptions noted.



#### 4. Backup and Recovery

Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 4:</b> Controls provide reasonable assurance that system data is regularly backed up and archived data is available for restoration in the event unexpected interruptions occur.		
1. Backups of customer Git repositories are performed on a real-time basis using an automated system. Backups of customer databases are encrypted during creation.	Inquired of the Senior Software Engineer to confirm customer Git repositories were backed up in real time using an automated system; and backups of customer databases were encrypted during creation.	No exceptions noted.
	Inspected the backup configuration to determine whether customer Git repositories were configured to be backed up on a real-time basis using an automated system.	No exceptions noted.
	Inspected the backup encryption configuration to determine whether customer database backups were configured to be automatically encrypted during the backup creation process.	No exceptions noted.
	Inspected backup storage system logs to determine whether backups were created throughout the period.	No exceptions noted.
2. The Storage Engineering team tests GitHub's backup restoration process annually.	Inquired of the Storage Engineering team manager to confirm the Storage Engineering team tested GitHub's backup restoration process annually.	No exceptions noted.
	Inspected GitHub backup restoration plans to determine whether backup restoration procedures were documented.	No exceptions noted.
	Inspected documentation of the most recent backup restoration to determine whether GitHub's defined Git and database data restoration plans were tested during the engagement period.	No exceptions noted.



## 5. Repository History

Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 5:</b> Controls provide reasonable assurance that repository history is complete and accurate.		
1. GitHub.com issue information is editable by individual contributors. Evidence that a comment was made is retained in the issue history, including the name of the contributor, the date and time of the original comment, and the date and time it was edited.	Inquired of the Director, Security GRCC to confirm GitHub.com issue information was editable by individual contributors; and evidence that a comment was made was retained in the issue history, including the name of the contributor, the date and time of the original comment, and the date and time it was edited.	No exceptions noted.
	Inspected a GitHub repository functionality to determine whether the ability to edit issue information was limited to individuals with access to the repository.	No exceptions noted.
	Inspected the history after observing the editing of a test GitHub issue to determine whether edit information was recorded, including the name of the contributor, the date and time of the original comment, and the date and time this issue was edited.	No exceptions noted.
2. Pull requests cannot be edited once they are submitted, but they can be reverted. The revert request links to the original pull request.	Inquired of the Director, Security GRCC to confirm pull requests could not be edited once they were submitted, but they could be reverted; and the revert request linked to the original pull request.	No exceptions noted.
	Observed the submission of a pull request to a test repository and attempted to edit this pull request to determine whether this pull request could not be edited after submission.	No exceptions noted.
	Observed the reverting of a test pull request to determine whether this revert request linked to the original test pull request.	No exceptions noted.





Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 5:</b> Controls provide reasonable assurance that repository history is complete and accurate.		
3. GitHub.com retains a history of actions taken with directly edited code files, including the code changes made, who made the change, and the date and time of the change.	Inquired of Director, Security GRCC to confirm github.com retained a history of actions taken with directly edited code files, including the code changes made, who made the change, and the date and time of the change.	No exceptions noted.
	Observed the edit of test code files in a GitHub codebase to determine whether the edits to these test files were recorded in the codebase history.	No exceptions noted.
	Inspected the edit history to determine whether it included the code change made, the name of the individual that made these test changes, and the date and time of the change.	No exceptions noted.
4. Code commits, whether direct commits or through a pull request, are hashed. The hash is displayed with the commit to facilitate code integrity verification.	Inquired of the Director, Security GRCC to confirm code commits, whether direct commits or through a pull request, were hashed; and the hash was displayed with the commit to facilitate code integrity verification.	No exceptions noted.
	Observed the submission of a test direct code commit and a test pull request to determine whether these test commits were hashed.	No exceptions noted.
	Inspected test commits for a sample of changes to determine whether the hash was visible with these commits.	No exceptions noted.



## 6. Vendor Management

Controls Specified by GitHub	Tests Performed by Moss Adams LLP	Test Results
<b>Control Objective 6:</b> Controls provide reasonable assurance that vendors are assessed for their ability to meet security compliance requirements.		
1. Prior to engaging a new vendor, GitHub reviews vendor security compliance and documents the results in the vendor security repository. On an annual basis, GitHub assesses the controls implemented at subservice organizations who receive or store customer data through the review of relevant subservice organizations' attestation reports.	Inquired of the Director, Security GRCC to confirm GitHub reviewed vendor security compliance prior to engagement a new vendor and documented the results in the vendor security repo; and GitHub assessed the controls implemented at subservice organizations that received or stored customer data on an annual basis through the review of relevant subservice organization attestations' reports.	No exceptions noted.
	Inspected the vendor security compliance review documentation for a sample of new vendors to determine whether GitHub reviewed vendor security compliance for these vendors prior to engagement.	No exceptions noted.
	Inspected the control assessments for a sample of subservice providers that received or stored customer data to determine whether GitHub assessed subservice organization controls during the examination period through the review of these subservice organizations' attestation reports.	No exceptions noted.



## V. Other Information Provided by the Service Organization

The following information is provided for informational purposes only and has not been subjected to the procedures applied in the examination. Accordingly, Moss Adams expresses no opinion on the following information.

### A. Management’s Response to Identified Testing Exceptions

Control #	Controls Specified by GitHub	Exception Noted by Moss Adams LLP	GitHub Management Response
1.10	The Security Operations team revokes logical production access and Production Engineering revokes physical production access for terminated personnel (employee and contractors) within 24 hours of termination.	For 1 of 25 terminated personnel sampled, access was not revoked within 24 hours of termination.	<p>GitHub’s employee termination process is comprised of two main sub-processes:</p> <p>(1) HR’s process to identify an employee/contractor’s date of termination and configure that date manually within GitHub’s HRIS; (2) Security Operations’ automated process to receive the notice of termination (systematically identified upon input of the termination date to the HRIS) and automatically (via system configuration and automation) remove access to GitHub’s internal systems.</p> <p>With regard to the control deviation noted by the Service Auditor, the manual sub-process discussed above was delayed due to miscommunication regarding the effective termination date of the employee in question. As soon as IT was able to confirm the details of the termination, the correct termination date was input/updated within the HRIS and the automated sub-process executed without delay or exception. Additionally, upon further inspection of the user’s internal system access and account activity, GitHub Management determined no inappropriate actions were taken by the employee during the period as a result of the deviation noted by the Service Auditor.</p>



Control #	Controls Specified by GitHub	Exception Noted by Moss Adams LLP	GitHub Management Response
3.7	The Product Engineering team monitors emergency deployments on a real-time basis, and post-mortem reviews are performed for emergency changes impacting production systems.	From February 25, 2022, to June 22, 2022, real-time reviews of emergency deployments were not performed in a timely manner.	With regard to the deviation noted by the Service Auditor, GitHub Management performed a review of all emergency deployments performed during the examination period and determined that post-mortem reviews were performed for all emergency deployments. GitHub Management also determined there was a temporary pause in automated notifications triggering the deployment reviews; the automated alerting functionality was remediated during the examination period and is functioning appropriately. Notwithstanding the deviation noted by the Service Auditor, all other preventive change management controls tested by the Service Auditor for the current examination period were designed and operating effectively throughout the period.

