# Monitoring GitHub

eG Innovations Product Documentation

www.eginnovations.com

eG
Total Performance Visibility

# Table of Contents

# Table of Figures

# Chapter 1: Introduction

GitHub is a web-based version-control and collaboration platform for software developers. Git is used to store the source code for a project and track the complete history of all changes to that code. It allows developers to collaborate on a project more effectively by providing tools for managing possibly conflicting changes from multiple developers. GitHub allows developers to change, adapt and improve software from its public repositories for free, but it charges for private repositories, offering various paid plans. Each public or private repository contains all of a project's files, as well as each file's revision history. Repositories can have multiple collaborators and can be either public or private.

Since the GitHub environment is hosted on the cloud and shared across multiple users, it is important for every GitHub account owner to keep track of the changes happening in his/her account at regular intervals. This is where eG Enterprise giving helping hands to the GitHub account owners. Using the specialized monitoring model offered by eG Enterprise, the GitHub account owner can continuously monitor his/her account.

# Chapter 2: How to Monitor GitHub Using eG Enterprise?

eG Enterprise monitors GitHub using an agentless approach. For this purpose, you need to install an eG agent on a remote Windows host in your . To enable the eG agent to collect the performance metrics of the GitHub user account, specify the credentials of that particular user while configuring the tests using eG admin interface.

The broad steps for monitoring the GitHub using eG Enterprise are as follows:

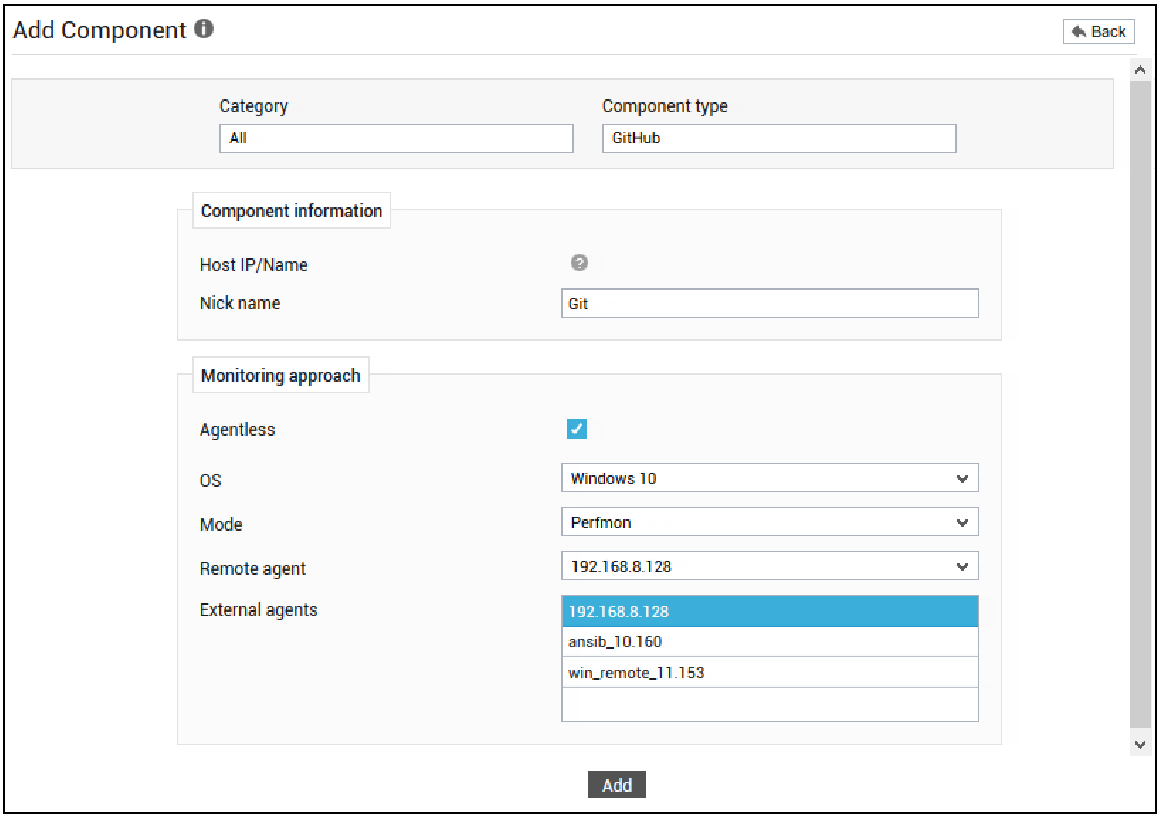
1. Manage the GitHub component using the eG admin interface;
2. Configure the tests for the component.

In this discussion, each of the aforesaid steps will be elaborately dealt in the following sections.

## 2.1 Managing GitHub

To manage the GitHub component, do the following:

1. Log into the eG admin interface.
2. Follow the Components -> Add/Modify menu sequence in the **Infrastructure** tile of the **Admin** menu.
3. In the **Components** page that appears next, select *GitHub* as the **Component type**. Then, click the **Add New Component** button. This will invoke Figure 2.1.



Category: All

Component type: GitHub

Component information

Host IP/Name:

Nick name: Git

Monitoring approach

Agentless: ☑

OS: Windows 10

Mode: Perfmon

Remote agent: 192.168.8.128

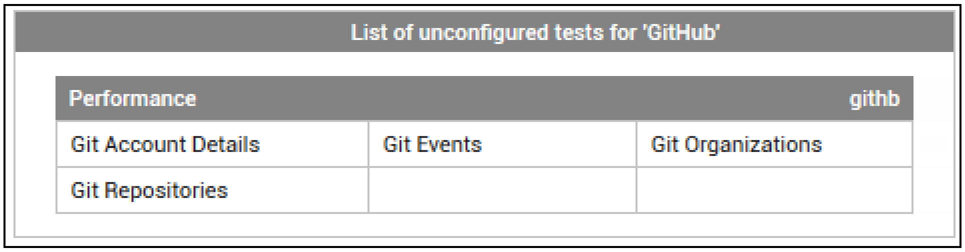
External agents: 192.168.8.128<br>ansib_10.160<br>win_remote_11.153

Figure 2.1: Adding the GitHub

4. Specify the **Nick name** for the GitHub component. Since the GitHub is hosted on the cluster environment, it can be monitored in an agentless manner alone.
5. Therefore, the **Agentless** flag will be enabled by default.
6. Set **Other** as the **OS** and **Other** as the **Mode**.
7. Then, choose the **Remote Agent** that should do agentless monitoring of the GitHub component Chapter 2.
8. Next, assign a **External Agent** to the component.
9. Finally, click the **Add** button to add the GitHub component to the eG Enterprise system. Components manually added will be automatically managed by eG Enterprise.

## 2.2 Configuring Tests

Once the GitHub component is managed, try to sign out of the eG admin interface. This will invoke , Figure 2.2 listing all the unconfigured tests for the GitHub component.

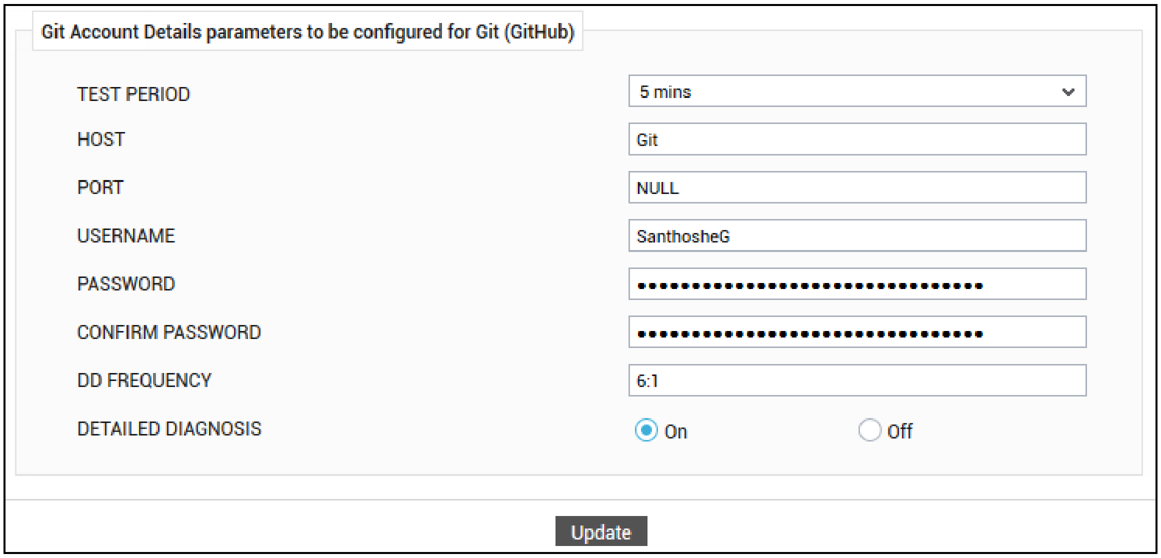


List of unconfigured tests for 'GitHub'

| Performance | | githb |
|---|---|---|
| Git Account Details | Git Events | Git Organizations |
| Git Repositories | | |

Figure 2.2: The list of unconfigured tests for the GitHub component

Click on any test to configure it. For instance, click on the **GitHub Account Details** test. Doing so will invoke Figure 2.



Git Account Details parameters to be configured for Git (GitHub)

| | |
|---|---|
| TEST PERIOD | 5 mins |
| HOST | Git |
| PORT | NULL |
| USERNAME | SanthosheG |
| PASSWORD | •••••••••••••••••••••••••••••••• |
| CONFIRM PASSWORD | •••••••••••••••••••••••••••••••• |
| DD FREQUENCY | 6:1 |
| DETAILED DIAGNOSIS | ● On ○ Off |

Figure 2.3: Configuring the Git Account Details test

In Figure 2, you have to specify the valid credentials of a GitHub user against the **USERNAME** and **PASSWORD** parameters to enable the eG agent to monitor the GitHub environment of the specified user. To know more details on configuring the tests, refer to **Monitoring GitHub**.

Once the test is configured, sign out of the eG admin interface.

# Chapter 3: Monitoring GitHub

eG Enterprise offers a dedicated monitoring model for GitHub which periodically monitors the repositories, the organizations and the events reported during various activities in a GitHub user account.

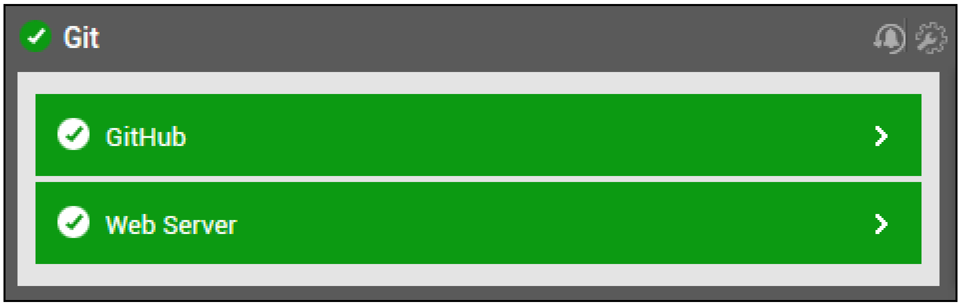


Figure 3.1: Layer model of the GitHub

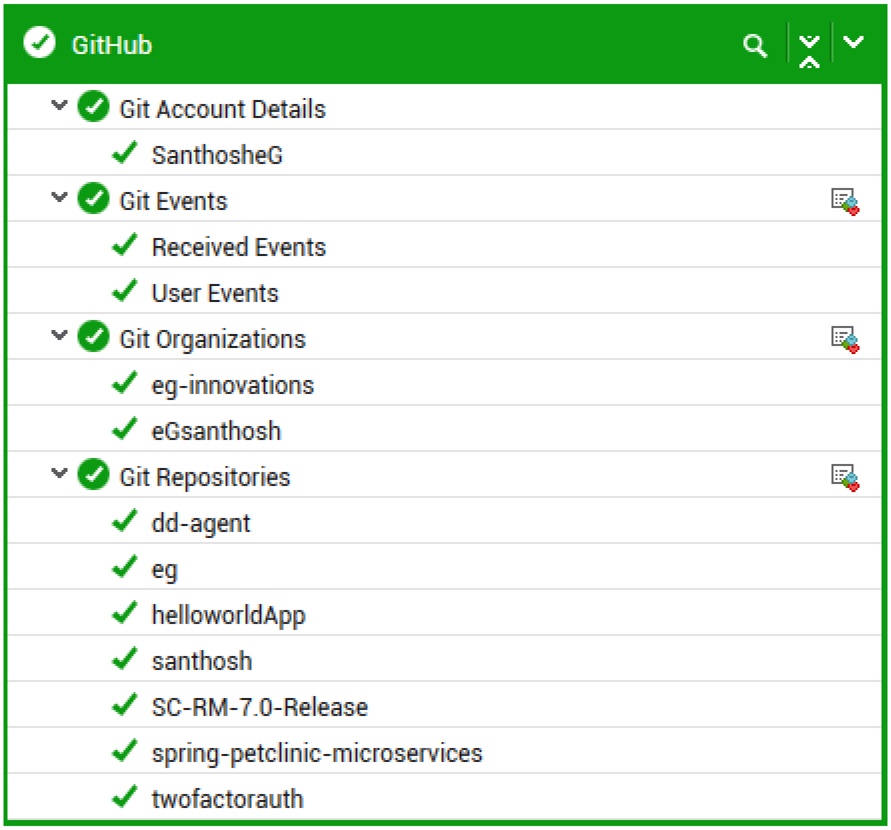
Using the metrics reported by the tests mapped to the layers, administrators can find quick and accurate answers to certain persistent performance queries, such as the following:

- What is the size of each repository?
- Is any repository populated with more number of open issues?
- How well the storage space is being utilized in the target user account?
- How many private and public repositories are owned by the target user account?
- Which organization has more number of outside collaborators?
- Did create/delete events were performed in the user account?
- How many commit operations were performed in each repository?

Since the **Web Server** layer in the Figure 3.1 has already been discussed in the *Monitoring Microsoft IIS Web Server* document, the section to come will focus on the **GitHub** layer alone.

## 3.1 GitHub Layer

Using the tests mapped to this layer, administrators can continuously monitor the GituHub user and reveal details on organizations and repositories. In the process, you can also determine the kind of events that occurred frequently in the account.



GitHub

- ✓ Git Account Details
  - ✓ SanthosheG
- ✓ Git Events
  - ✓ Received Events
  - ✓ User Events
- ✓ Git Organizations
  - ✓ eg-innovations
  - ✓ eGsanthosh
- ✓ Git Repositories
  - ✓ dd-agent
  - ✓ eg
  - ✓ helloworldApp
  - ✓ santhosh
  - ✓ SC-RM-7.0-Release
  - ✓ spring-petclinic-microservices
  - ✓ twofactorauth

Figure 3.2: The tests mapped to the GitHub layer

### 3.1.1 Git Account Details Test

With a valid GitHub user account, you can do the following:

- access all features and components of Git;
- create the components such as repositories, branches and gists;
- share repositories with other users;
- get access to use the storage space allocated on the cloud;
- follow other GitHub accounts;
- collaborate with other GitHub users;
- track the history of changes and events over time;
- revert back your changes anytime to any version, etc.

Using this test, you can get an overview of the GitHub user account that is being configured for monitoring. In the process, this test reveals the count of private and public repositories and gists owned by the GitHub user account. In the process, this test sheds light on the disk space utilization in the user account. This helps you determine whether adequate storage space is available to ensure uninterrupted functioning of the repositories owned by the target account. Additionally, this test provides the detailed diagnosis using which you can identify the users who are currently following the target account and the users who are followed by the target account.

**Target of the test :** GitHub

**Agent deploying the test :** A remote agent

**Outputs of the test :** One set of the results for the GitHub user account being monitored.

**Configurable parameters for the test**

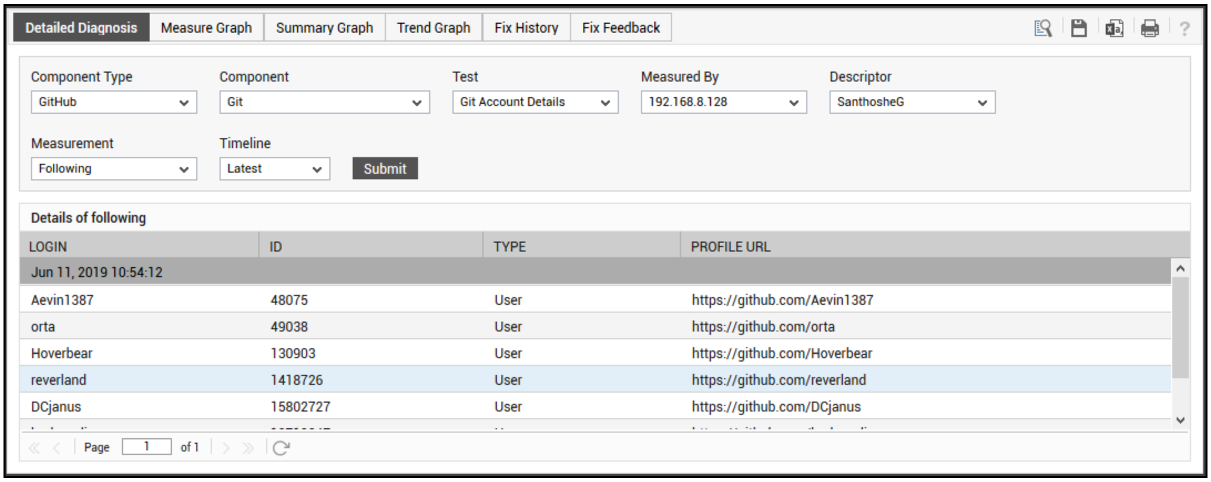
| Parameter | Description |
|---|---|
| Test Period | How often should the test be executed. By default, this is set to 1800 seconds. |
| Host | The IP address of the host for which this test is to be configured. |
| Port | The port at which the specified host listens. By default, this is *NULL* |
| Username and Password | Specify the valid credentials of a user of the GitHub against the Username and Password parameters. |
| Confirm Password | Confirm the password by retyping it here. |
| DD Frequency | Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *6:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency. |
| Detailed Diagnosis | To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.<br>The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:<br>• The eG manager license should allow the detailed diagnosis capability<br>• Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0. |

### Measurements made by the test

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| Public repos | Indicates the number of public repositories in the target user account. | Number | |
| Private repos | Indicates the number of private repositories in the target user account. | Number | |
| Public gists | Indicates the number of public gists in the target user account. | Number | With gists, you can share single files, parts of files, and full applications with other users.<br>Public gists show up in Discover, where people can browse new gists as they're created. They are also searchable, so you can use them if you'd like other people to find and see your work. After creating a gist, you cannot convert it from public to secret. |
| Private gists | Indicates the number of private gists in the target user account. | Number | |
| Collaborators limit | Indicates the maximum number of collaborators who can access the repositories in the target user account. | Number | The value of this measure varies depending upon the type of GitHub account. For instance, if you're using GitHub Free account, you can add unlimited collaborators on public repositories, and up to three collaborators on private repositories owned by your personal account. |
| Private repos limit | Indicates the maximum number of repositories that can be maintained personal to the target user account. | Number | |
| Max storage limit | Indicates the maximum amount of storage space allocated for the target | MB | GitHub doesn't have any set user disk quota and provides abundant storage for all Git repositories, although there |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | user account. | | are hard limits for file and repository sizes. Keeping repositories small ensures that our servers are fast and downloads are quick for our users. |
| Disk usage | Indicates the amount of storage space utilized in the target user account. | MB | |
| Free disk | Indicates the amount of storage space available for in the target user account. | MB | The value of this measure is desired to be high. |
| Disk Utilization | Indicates the amount of storage space available for use in the target user account. | Percent | |
| Collaborators | Indicates the number of collaborators in the target user account. | Number | A collaborator is an outside user who has been granted write access to the main repository owned by the user account. |
| Followers | Indicates the number of users who follows the target user account. | Number | The detailed diagnosis of this measure reveals the login name, ID and type of every user who is following the user account being monitored and the URL of the follower profile. |
| Following | Indicates the number of users who are all followed by the target user account. | Number | The detailed diagnosis of this measure reveals the login name, ID, type and URL of every user who is being followed by the user account being monitored. |

Use the detailed diagnosis of the *Follwers* measure to view the login name, ID and type of every user who is following the user account being monitored and the URL of the follower profile.



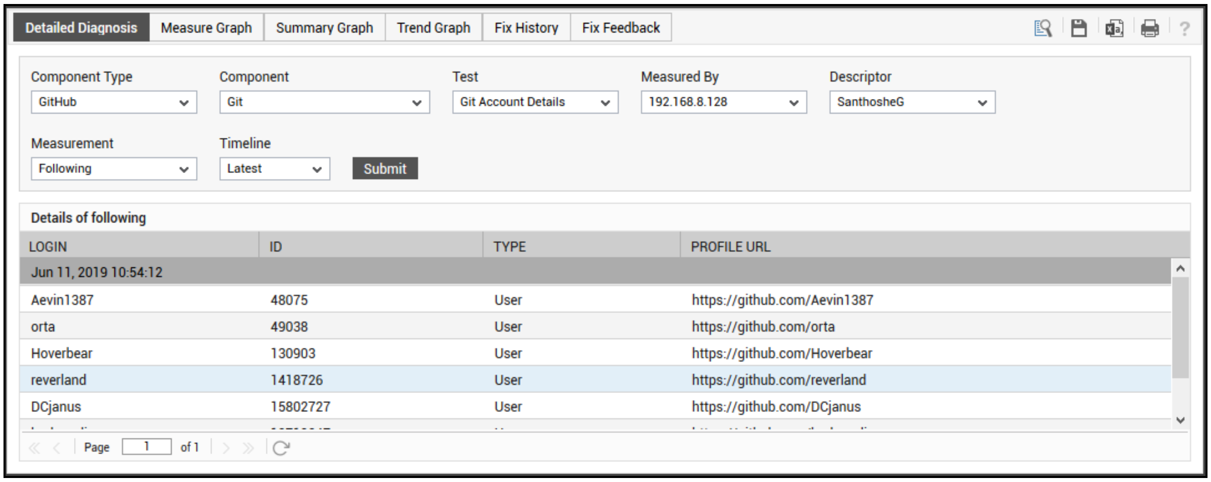
Component Type: GitHub
Component: Git
Test: Git Account Details
Measured By: 192.168.8.128
Descriptor: SanthosheG
Measurement: Following
Timeline: Latest
Submit

Details of following

| LOGIN | ID | TYPE | PROFILE URL |
|---|---|---|---|
| Jun 11, 2019 10:54:12 | | | |
| Aevin1387 | 48075 | User | https://github.com/Aevin1387 |
| orta | 49038 | User | https://github.com/orta |
| Hoverbear | 130903 | User | https://github.com/Hoverbear |
| reverland | 1418726 | User | https://github.com/reverland |
| DCjanus | 15802727 | User | https://github.com/DCjanus |

Figure 3.3: The detailed diagnosis of the Followers measure

Use the detailed diagnosis of the *Following* measure to view the login name, ID, type and URL of every user who is being followed by the user account being monitored.



Component Type: GitHub
Component: Git
Test: Git Account Details
Measured By: 192.168.8.128
Descriptor: SanthosheG
Measurement: Following
Timeline: Latest
Submit

Details of following

| LOGIN | ID | TYPE | PROFILE URL |
|---|---|---|---|
| Jun 11, 2019 10:54:12 | | | |
| Aevin1387 | 48075 | User | https://github.com/Aevin1387 |
| orta | 49038 | User | https://github.com/orta |
| Hoverbear | 130903 | User | https://github.com/Hoverbear |
| reverland | 1418726 | User | https://github.com/reverland |
| DCjanus | 15802727 | User | https://github.com/DCjanus |

Figure 3.4: The detailed diagnosis of the Following measure

### 3.1.2 Git Events Test

A user with a valid account on the GitHub site can perform various actions within his/her account. Every action performed by the user is recorded as an event in a read-only API for GitHub events. These recorded events are helpful for tracking the history of past activities performed in the account. By analyzing the history of events, the user can instantly know the type of events that was performed frequently, the changes on repositories, whether the events were created by himself/herself or from outside, the details on the status of issues, etc. To help the users in this regard, this test monitors the events recorded in the read-only API and reports the number of different events reported over the time.

**Target of the test :** GitHub

**Agent deploying the test :** A remote agent

**Outputs of the test :** One set of the results for each event type.

**Configurable parameters for the test**

| Parameter | Description |
|---|---|
| Test Period | How often should the test be executed. By default, this is set to 1800 seconds. |
| Host | The IP address of the host for which this test is to be configured. |
| Port | The port at which the specified host listens. By default, this is *NULL* |
| Username and Password | Specify the valid credentials of a user of the GitHub against the Username and Password parameters. |
| Confirm Password | Confirm the password by retyping it here. |
| DD Frequency | Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *6:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency. |
| Detailed Diagnosis | To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.<br>The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:<br>• The eG manager license should allow the detailed diagnosis capability<br>• Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0. |

### Measurements made by the test

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| Check run events | Indicates the number of events of this type triggered when a check run was created, requested, rerequested and completed. | Number | The check runs API enables you to build GitHub Apps that run powerful checks against code changes in a repository. |
| Check suite events | Indicates the number of events of this type triggered when a check suite was created, requested, rerequested and completed | Number | A check suite is a collection of the check runs created by a single GitHub App for a specific commit. Check suites summarize the status and conclusion of the check runs that a suite includes. |
| Commit comment events | Indicates the number of events of this type triggered when a commit comment was created. | Number | |
| Content reference events | Indicates the number of events of this type triggered when the body or comment of an issue or pull request includes a URL that matches a configured content reference domain | Number | These events are triggered based on the specificity of the domain you register. For example, if you register a subdomain (https://subdomain.example.com) then only URLs for the subdomain trigger this event. If you register a domain (https://example.com) then URLs for domain and all subdomains trigger this event. |
| Create events | Indicates the number of events of this type triggered when branches or tags were created. | Number | |
| Delete events | Indicates the number of events of this type triggered when branches or tags were deleted. | Number | |
| Deployment events | Indicates the number of events of this type | Number | Deployments are requests to deploy a specific ref (branch, SHA, tag). GitHub |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | triggered when new deployments were created. | | dispatches a deployment event that external services can listen for and act on when new deployments are created. Deployments enable developers and organizations to build loosely coupled tooling around deployments, without having to worry about the implementation details of delivering different types of applications (e.g., web, native). |
| Deployment status events | Indicates the number of events of this type triggered when new deployments and deployment statuses were created. | Number | These events allows third-party integrations to receive respond to deployment requests and update the status of a deployment as progress is made.<br>Deployment statuses allow external services to mark deployments with an error, failure, pending, in progress, queued, or success state that systems listening to deployment status events can consume.<br>Deployment statuses can also include an optional description and log url, which are highly recommended because they make deployment statuses more useful. The log URL is the full URL to the deployment output, and the description is a high-level summary of what happened with the deployment. |
| Download events | Indicates the number of events of this type triggered when a new download was created. | Number | |
| Follow events | Indicates the number of events of this type triggered when a user follows another user. | Number | |
| Fork events | Indicates the number of | Number | A fork is a copy of a repository. Forking a |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | events of this type triggered when a user forks a repository. | | repository allows you to freely experiment with changes without affecting the original project. |
| Fork apply events | Indicates the number of events of this type triggered when a patch was applied in the fork queue. | Number | |
| Github app authorization events | Indicates the number of events of this type triggered when a user revoked his authorization of a GitHub App. | Number | Anyone can revoke their authorization of a GitHub App from their GitHub account settings page. Revoking the authorization of a GitHub App does not uninstall the GitHub App. You should program your GitHub App so that when it receives this event, it stops calling the API on behalf of the person who revoked the token. If your GitHub App continues to use a revoked access token, it will receive the *Bad Credentials* error. |
| Gist events | Indicates the number of events of this type triggered when a Gist was created or updated. | Number | |
| Gollum events | Indicates the number of events of this type triggered when a Wiki page was created or updated. | Number | Gollum is a simple wiki system built on top of Git. A Gollum Wiki is simply a git repository of a specific nature:<br>• A Gollum repository's contents are human-editable. Pages are unique text files which may be organized into directories any way you choose, as long as they have a recognized file extension. Other content can also be included, for example images, PDFs and headers/footers.<br>• Gollum pages: |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | | | ○ May be written in a variety of markup languages.<br>○ Can be edited with your favourite system editor or IDE or with the built-in web interface.<br>○ Can be displayed in all versions, and can easily be rolled back.<br>• Gollum supports advanced functionality like UML diagrams, macros, metadata, and more. |
| Installation events | Indicates the number of events of this type triggered when a user installed/uninstalled the GitHub App, or accepted new permissions for the GitHub App | Number | |
| Installation repositories events | Indicates the number of events of this type triggered when a repository was added or removed from an installation. | Number | |
| Issue comment events | Indicates the number of events of this type triggered when an issue comment was created, edited or deleted. | Number | |
| Issues events | Indicates the number of events of this type triggered when an issue was opened, edited, deleted, transferred, pinned, unpinned, closed, | Number | |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | reopened, assigned, unassigned, labeled, unlabeled, locked, unlocked, milestoned or demilestoned. | | |
| Label events | Indicates the number of events of this type triggered when a repository's label was created, edited, or deleted. | Number | |
| Marketplace purchase events | Indicates the number of events of this type triggered when changes were happened to the a user's plan from the Marketplace purchase. | Number | These events are recorded when a user purchased a GitHub Marketplace plan, canceled the plan, upgraded the plan (effective immediately), downgraded the plan that remains pending until the end of the billing cycle, or cancels a pending plan. |
| Member events | Indicates the number of events of this type triggered when a user accepted an invitation or was removed as a collaborator to a repository, or had his/her permissions changed. | Number | |
| Membership events | Indicates the number of events of this type triggered when a user was added or removed from a team. | Number | |
| Milestone events | Indicates the number of events of this type triggered when a milestone was created, closed, opened, edited or deleted. | Number | The milestones are helpful to track progress on groups of issues or pull requests in a repository. When you create a milestone, you can associate it with issues and pull requests.<br>From the milestone page, you can see: |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | | | • A user-provided description of the milestone, which can include information like a project overview, relevant teams, and projected due dates<br>• The milestone's due date<br>• The milestone's completion percentage<br>• The number of open and closed issues and pull requests associated with the milestone<br>  • A list of the open and closed issues and pull requests associated with the milestone<br>Additionally, you can edit the milestone from the milestone page and create new issues that are, by default, associated with the milestone. |
| Organization events | Indicates the number of events of this type triggered when an organization was deleted and renamed, and when a user was added, removed or invited to the organization. | Number | Organizations are shared accounts where businesses and open-source projects can collaborate across many projects at once. Owners and administrators can manage member access to the organization's data and projects with sophisticated security and administrative features. |
| Org block events | Indicates the number of events of this type triggered when an organization owner blocked or unblocked a | Number | |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | user. | | |
| Pagebuild events | Indicates the number of events of this type triggered on push to a GitHub Pages enabled branch (gh-pages for project pages, master for user and organization pages). | Number | A PageBuild Events represents whether an attempted build of a GitHub Pages site is successful or not. |
| Project card events | Indicates the number of events of this type triggered when a project card was created, edited, moved, converted to an issue, or deleted. | Number | |
| Project column events | Indicates the number of events of this type triggered when a project column was created, updated, moved or deleted. | Number | |
| Project events | Indicates the number of events of this type triggered when a project was created, updated, closed, reopened or deleted. | Number | |
| Public events | Indicates the number of events of this type triggered when a private repository was made public. | Number | |
| Pull request events | Indicates the number of events of this type triggered when a pull request was assigned, unassigned, labeled, | Number | |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | unlabeled, opened, edited, closed, reopened, synchronized, ready for review, locked, unlocked or when a pull request review was requested or removed. | | |
| Pull request review events | Indicates the number of events of this type triggered when a pull request review was submitted into a non-pending state, the body of the review was edited, or the review was dismissed. | Number | Reviews allow collaborators to comment on the changes proposed in pull requests, approve the changes, or request further changes before the pull request is merged. Repository administrators can require that all pull requests are approved before being merged. |
| Pull request review comment events | Indicates the number of events of this type triggered when a comment on a pull request's unified diff was created, edited or deleted. | Number | |
| Push events | Indicates the number of events of this type triggered when repository branch pushes and repository tag pushes were performed. | Number | |
| Release events | Indicates the number of events of this type triggered when a release was published, unpublished, created, edited, deleted or prereleased. | Number | |
| Repository events | Indicates the number of events of this type triggered when a | Number | |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | repository was created, archived, unarchived, renamed, edited, transferred, made public, or made private. | | |
| Repository import events | Indicates the number of events of this type triggered when a successful, canceled, or failed repository import was finished for a GitHub organization or a personal repository. | Number | |
| Repository vulnerability alert events | Indicates the number of events of this type triggered when a security alert was created, dismissed or resolved. | Number | |
| Security advisory events | Indicates the number of events of this type triggered when a new security advisory was published, updated or withdrawn. | Number | A security advisory provides information about security-related vulnerabilities in software on GitHub. Security Advisory webhooks are available to GitHub Apps only. The security advisory dataset also powers the GitHub security alerts. |
| Status events | Indicates the number of events of this type triggered when the status of a Git commit changed. | Number | |
| Team events | Indicates the number of events of this type triggered when an organization's team was created, deleted, edited, added to repository, or removed from repository. | Number | |
| Team add events | Indicates the number of events of this type | Number | |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | triggered when a repository was added to a team. | | |
| Watch events | Indicates the number of events of this type triggered when a user starred a repository. | Number | |

### 3.1.3 Git Organizations Test

Organizations are shared accounts where businesses and open-source projects can collaborate across many projects at once. Owners and administrators can manage member access to the organization's data and projects with sophisticated security and administrative features. Organizations include:

- A free option, GitHub Team for Open Source, with unlimited collaborators on unlimited public repositories.
- The option to upgrade to GitHub Team or GitHub Enterprise Cloud for additional features, including private repositories, sophisticated user authentication and management, and escalated support options.
- Unlimited membership with a variety of roles that grant different levels of access to the organization and its data.
- The ability to give members a range of access permissions to your organization's repositories.
- Nested teams that reflect your company or group's structure with cascading access permissions and mentions.
- The ability for organization owners to view members' two-factor authentication (2FA) status.
- The option to require all organization members to use two-factor authentication.

Using this test, administrators can instantly identify how many public members, teams and outside collaborators are in each organization of the target GitHub account. This test also sheds light on the issues recorded in the repositories in each organization. This helps administrators to identify the organization that has more number of open issues and take essential measures to resolve them in time.

**Target of the test :** GitHub

**Agent deploying the test :** A remote agent

**Outputs of the test :** One set of the results for every organization created.

### Configurable parameters for the test

| Parameter | Description |
|---|---|
| Test Period | How often should the test be executed. By default, this is set to 1800 seconds. |
| Host | The IP address of the host for which this test is to be configured. |
| Port | The port at which the specified host listens. By default, this is *NULL* |
| Username and Password | Specify the valid credentials of a user of the GitHub against the Username and Password parameters. |
| Confirm Password | Confirm the password by retyping it here. |
| DD Frequency | Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *6:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency. |
| Detailed Diagnosis | To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.<br>The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:<br>• The eG manager license should allow the detailed diagnosis capability<br>• Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0. |

### Measurements made by the test

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| Public members | Indicates the number of public members in this | Number | |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | organization. | | |
| Teams | Indicates the number of teams in this organization. | Number | |
| Outside collaborators | Indicates the total number of outside collaborators added to repositories in this organization. | Number | An outside collaborator is a person who is not explicitly a member of the organization, but who has Read, Write, or Admin permissions to one or more repositories in the organization. |
| Projects | Indicates the number of Projects created in this Organization. | Number | |
| Issues | Indicates the number of issues reported in the repositories in this Organization. | Number | If more number of issues are left unsolved in an organization, it may impact the reliability and data integrity of the repositories in the organization.<br>Compare the value of this measure across the organizations to find out which organization has more number of issues. |
| Webhooks | Indicates the number of webhooks created in this Organization. | Number | Webhooks allow you to build or set up integrations, such as GitHub Apps or OAuth Apps, which subscribe to certain events on GitHub.com. When one of those events is triggered, a HTTP POST payload will be sent to the webhook's configured URL. Webhooks can be used to update an external issue tracker, trigger CI builds, update a backup mirror, or even deploy to a production server. |

### 3.1.4 Git Repositories Test

A repository, also called as a project folder, is the most basic element of GitHub. You can create a new repository on your personal account or any organization where you have sufficient permissions. The repository contains all of the project's files and stores each file's revision history. You can also discuss and manage your project's work within the repository. Repositories can be either public or private and can have multiple collaborators. Public repositories are visible to everyone, on the other hand, a private repository can be viewed and contributed only by the owner and collaborators. As the repositories are the basic units and shared among multiple users, it is important for the account owners to be up-to-date about the size and workload of the repositories in their account. The **Git Repositories** test helps the account owners in this regard.

This test auto-discovers the repositories in the target GitHub account and reports the count of commit operations performed on each repository. In addition, the detailed diagnosis provided by the test reveals the record of what changes were made when and by who. This helps the account owners to keep track of the commit operations and to know if any of the changes were done by unauthorized users. This test also reveals the workload on each repository in terms of issues and pull requests. This way, the unsolved issues are brought to the immediate attention of the account owners, so that they can investigate the reason for the same and fix them. In the process, this test also sheds light on the size of each repository. Using this revelation, the account owners can instantly know whether the repository size is within the acceptable limit or has increased abnormally. If the repositories are detected to be in large size, the account owners can initiate remedial measures to prevent slowness during fetching files from the repositories.

**Target of the test :** GitHub

**Agent deploying the test :** A remote agent

**Outputs of the test :** One set of the results for every repository created.

**Configurable parameters for the test**

| Parameter | Description |
|---|---|
| Test Period | How often should the test be executed. By default, this is set to 1800 seconds. |
| Host | The IP address of the host for which this test is to be configured. |
| Port | The port at which the specified host listens. By default, this is *NULL* |
| Username and Password | Specify the valid credentials of a user of the GitHub against the Username and Password parameters. |
| Confirm Password | Confirm the password by retyping it here. |
| DD Frequency | Refers to the frequency with which detailed diagnosis measures are to be generated for this test. The default is *6:1*. This indicates that, by default, detailed measures will be generated every time this test runs, and also every time the test detects a problem. You can modify this frequency, if you so desire. Also, if you intend to disable the |

| Parameter | Description |
|---|---|
| | detailed diagnosis capability for this test, you can do so by specifying *none* against DD frequency. |
| Detailed Diagnosis | To make diagnosis more efficient and accurate, the eG Enterprise suite embeds an optional detailed diagnostic capability. With this capability, the eG agents can be configured to run detailed, more elaborate tests as and when specific problems are detected. To enable the detailed diagnosis capability of this test for a particular server, choose the **On** option. To disable the capability, click on the **Off** option.<br>The option to selectively enable/disable the detailed diagnosis capability will be available only if the following conditions are fulfilled:<br>• The eG manager license should allow the detailed diagnosis capability<br>• Both the normal and abnormal frequencies configured for the detailed diagnosis measures should not be 0. |

### Measurements made by the test

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| Commits | Indicates the number of commit operations performed in this repository. | Number | A commit, or "revision", is an individual change to a file or set of files in the repository. When you save a file, except with Git, every time it creates a unique ID (a.k.a. the "SHA" or "hash") that allows you to keep record of what changes were made when and by who. Commits usually contain a commit message which is a brief description of what changes were made.<br>The detailed diagnosis of this measure reveals the name and email address of an author who created the file for which the commit operation was performed. |
| Pull requests | Indicates the number of pull requests made to this repository during the last measurement period. | Number | Pull requests are proposed changes to a repository submitted by a user and accepted or rejected by a repository's collaborators. |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | | | Once a pull request is opened, the particular user can discuss and review the potential changes with the collaborators and add follow-up commits before the changes are merged into the base branch. |
| Branches | Indicates the number of branches created in this repository. | Number | A branch is a parallel version of a repository. It is contained within the repository, but does not affect the primary or master branch allowing you to work freely without disrupting the live version. When you have made the changes you want to make, you can merge your branch back into the master branch to publish your changes. |
| Open issues | Indicates the number of issues that are still open/ unresolved in this repository. | Number | Issues are suggested improvements, tasks or questions related to the repository. Issues can be created by anyone (for public repositories), and are moderated by repository collaborators. Each issue contains its own discussion forum, can be labeled and assigned to a user.<br>A high value may indicate that more number of tasks and improvements are not completed yet which in turn may decrease the data integrity and reliability of the repositories. |
| Repository size | Indicates the size of this repository. | Number | The value reported by this measure helps a repository administrator to know whether the size of the repository is within an acceptable limit or has increased abnormally. If the size of the repository has increased abnormally, administrator should consider removing the large files from the repository to keep the repository |

| Measurement | Description | Measurement Unit | Interpretation |
|---|---|---|---|
| | | | smaller. The smaller repositories ensure that the GitHub server is fast and downloads are quick for users. |
| Watchers | Indicates the number of users who can watch the activities performed in this repository. | Number | Watchers are Github users who have asked to be notified of activity in a repository. |
| Repository ranking | Indicates the ranking of this repository. | Number | In GitHub, You can star repositories and topics to keep track of projects you find interesting and discover related content in your news feed. Starring makes it easy to find a repository or topic for future reference.<br>Starring a repository also shows appreciation to the repository administrator for their work. Many of GitHub's repository rankings depend on the number of stars a repository has. Therefore, a high value is preferred for this measure. |
| Forks | Indicates the number of forks in this repository. | Number | A fork is a personal copy of another user's repository that lives on your GitHub account. Forks allow you to freely make changes to a project without affecting the original. Forks remain attached to the original, allowing you to submit a pull request to the original's author to update with your changes. You can also keep your fork up to date by pulling in updates from the original. |

# About eG Innovations

eG Innovations provides intelligent performance management solutions that automate and dramatically accelerate the discovery, diagnosis, and resolution of IT performance issues in on-premises, cloud and hybrid environments. Where traditional monitoring tools often fail to provide insight into the performance drivers of business services and user experience, eG Innovations provides total performance visibility across every layer and every tier of the IT infrastructure that supports the business service chain. From desktops to applications, from servers to network and storage, from virtualization to cloud, eG Innovations helps companies proactively discover, instantly diagnose, and rapidly resolve even the most challenging performance and user experience issues.

eG Innovations is dedicated to helping businesses across the globe transform IT service delivery into a competitive advantage and a center for productivity, growth and profit. Many of the world's largest businesses use eG Enterprise to enhance IT service performance, increase operational efficiency, ensure IT effectiveness and deliver on the ROI promise of transformational IT investments across physical, virtual and cloud environments.

To learn more visit www.eginnovations.com.

### Contact Us

For support queries, email support@eginnovations.com.

To contact eG Innovations sales team, email sales@eginnovations.com.