Site admin dashboard

In this article

Explore
Audit log
Reports
Indexing
Reserved logins
Dormant users
Suspended users

You can use the site admin dashboard to manage users, organizations, and repositories on your GitHub Enterprise Server instance.

To access the dashboard, in the upper-right corner of any page, click 🚀.

Explore

Data for GitHub's trending page is calculated into daily, weekly, and monthly time spans for both repositories and developers. You can see when this data was last cached and queue up new trending calculation jobs from the **Explore** section.

Audit log

GitHub Enterprise Server keeps a running log of audited actions that you can query.

By default, the audit log shows you a list of all audited actions in reverse chronological order. You can filter this list by entering key-value pairs in the **Query** text box and then clicking **Search**, as explained in "Searching the audit log for your enterprise."

For more information on audit logging in general, see "About the audit log for your enterprise." For a full list of audited actions, see "Audit log events for your enterprise."

Reports

If you need to get information on the users, organizations, and repositories in your GitHub Enterprise Server instance, you would ordinarily fetch JSON data through the GitHub API. Unfortunately, the API may not provide all of the data that you want and it requires a bit of technical expertise to use. The site admin dashboard offers a **Reports** section as an alternative, making it easy for you to download CSV reports with most of the information that you are likely to need for users, organizations, and repositories.

Specifically, you can download CSV reports that list

- all users
- all active users
- all dormant users

- all users who have been suspended
- all organizations
- all repositories

You can also access these reports programmatically via standard HTTP authentication with a site admin account. You must use a personal access token (classic) with the `site_admin` scope. For more information, see "Managing your personal access tokens."

For example, here is how you would download the "all users" report in a `curl` command:

```
curl --remote-name \
     --location \
     --user 'USERNAME:TOKEN' \
     http(s)://HOSTNAME/stafftools/reports/all_users.csv
```

To access the other reports programmatically, replace `all_users` with `active_users`, `dormant_users`, `suspended_users`, `all_organizations`, or `all_repositories`.

Note: The initial `curl` request will return a 202 HTTP response if there are no cached reports available; a report will be generated in the background. You can send a second request to download the report. You can use a password or an OAuth token with the `site_admin` scope in place of a password.

User reports

Key	Description
`created_at`	When the user account was created (as an ISO 8601 timestamp)
`id`	Account ID for the user or organization
`login`	Account's login name
`email`	Account's primary email address
`role`	Whether the account is an admin or an ordinary user
`suspended?`	Whether the account has been suspended
`last_logged_ip`	Most recent IP address to log into the account
`repos`	Number of repositories owned by the account
`ssh_keys`	Number of SSH keys registered to the account
`org_memberships`	Number of organizations to which the account belongs
`dormant?`	Whether the account is dormant
`last_active`	When the account was last active (as an ISO 8601 timestamp)
`raw_login`	Raw login information (in JSON format)
`2fa_enabled?`	Whether the user has enabled two-factor authentication

Organization reports

Key	Description
`id`	Organization ID
`created_at`	When the organization was created
`login`	Organization's login name
`email`	Organization's primary email address
`owners`	Number of organization owners
`members`	Number of organization members
`teams`	Number of organization teams
`repos`	Number of organization repositories
`2fa_required?`	Whether the organization requires two-factor authentication

Repository reports

Key	Description
`created_at`	When the repository was created
`owner_id`	ID of the repository's owner
`owner_type`	Whether the repository is owned by a user or an organization
`owner_name`	Name of the repository's owner
`id`	Repository ID
`name`	Repository name
`visibility`	Whether the repository is public or private
`readable_size`	Repository's size in a human-readable format
`raw_size`	Repository's size as a number
`collaborators`	Number of repository collaborators
`fork?`	Whether the repository is a fork
`deleted?`	Whether the repository has been deleted

Indexing

GitHub's search features are powered by Elasticsearch. This section of the site admin dashboard shows you the current status of your Elasticsearch cluster and provides you with several tools to control search and index behavior.

For more information about code search, see "Search on GitHub documentation." For more information about Elasticsearch, see the Elasticsearch website.

> **Note**: In normal use, site administrators do not need to create new indices or schedule repair jobs. For troubleshooting or other support purposes, GitHub Support may instruct you to run a repair job.

Index management

GitHub Enterprise Server reconciles the state of the search index with data on the instance automatically and regularly.

- Issues, pull requests, repositories, and users in the database
- Git repositories (source code) on disk

Your instance uses repair jobs to reconcile the data, and schedules a repair job in the background when the following events occur.

- A new search index is created.
- Missing data needs to be backfilled.
- Old search data needs to be updated.

You can create a new index, or you can click on an existing index in the list to manage the index. You can perform the following operations on an index.

- Make the index searchable.
- Make the index writable.
- Update the index.
- Delete the index
- Reset the index repair state.
- Start a new index repair job.
- Enable or disable index repair jobs.

A progress bar shows the current status of a repair job across background workers. The bar is the percentage difference of the repair offset with the highest record ID in the database. You can ignore the value shown in the progress bar after a repair job has completed. The progress bar shows the difference between the repair offset and the highest record ID in the database, and will decrease as more repositories are added to your GitHub Enterprise Server instance even though those repositories are actually indexed.

To minimize the effects on I/O performance and reduce the chances of operations timing out, run the repair job during off-peak hours. As the job reconciles the search index with database and Git repository data, one CPU will be used. Monitor your system's load averages and CPU usage with a utility like `top`. If you don't notice any significant increase in resource consumption, it should also be safe to run an index repair job during peak hours.

Repair jobs use a "repair offset" for parallelization. This is an offset into the database table for the record being reconciled. Multiple background jobs can synchronize work based on this offset.

Code search

This allows you to enable or disable both search and index operations on source code.

Reserved logins

Certain words are reserved for internal use in your GitHub Enterprise Server instance, which means that these words cannot be used as usernames.

For example, the following words are reserved, among others:

- `admin`
- `enterprise`
- `login`
- `staff`
- `support`

For the full list or reserved words, navigate to "Reserved logins" in the site admin dashboard.

Dormant users

Here you can see and suspend inactive users on your GitHub Enterprise Server instance. For more information, see "Suspending and unsuspending users".

A user account is considered to be inactive ("dormant") when it:

- Has existed for longer than the dormancy threshold that's set for your GitHub Enterprise Server instance.
- Has not generated any activity within that time period.
- Is not a site administrator.

The dormancy threshold is the length of time a user must be inactive to be considered dormant. The default dormancy threshold is 90 days, however you can customize the dormancy threshold for your GitHub Enterprise Server instance. For more information, see "Managing dormant users."

Suspended users

Here you can see all of the users who have been suspended on your GitHub Enterprise Server instance, and initiate an SSH key audit.

Legal

© 2023 GitHub, Inc. Terms Privacy Status Pricing Expert services Blog