Github Actions

Saturday, February 15, 2025 12:20 PM

By specifying the event types as [created, edited], the workflow will only run when branch protection rules are created or edited, excluding the deletion of branch protection rules. This configuration ensures that the workflow execution is limited to the desired events.

GitHub Packages has support for the following package registries:

- JavaScript - Node package manager (npm)
- Ruby - RubyGems package manager (gem)
- Java - Apache Maven project management and comprehension tool (mvn)
- Java - Gradle build automation tool for Java (gradle)
- .NET - NuGet package management for .NET (dotnet)
- N/A Docker container management

The Checks API is the correct choice because GitHub Actions use this API to output statuses, results, and logs for a workflow. It allows workflows to create detailed status checks, annotations, and summaries for each job and step in the workflow, providing visibility into the execution and results of the workflow.

The Actions API in GitHub is used to interact with GitHub Actions workflows, such as triggering workflows, listing workflow runs, and getting workflow run details. It is not specifically used to output statuses, results, and logs for a workflow in GitHub Actions.

The Health API in GitHub is used to check the health status of GitHub services and systems, providing information on the operational status of GitHub's infrastructure. It is not directly related to outputting statuses, results, and logs for a workflow in GitHub Actions.

Filtering on "failure" will show only the workflow runs that have failed specifically. This is the correct status to filter on if you want to see only the failed workflow runs on the GitHub Actions tab.

If a job is not approved within 30 days while awaiting review in a workflow, the job will automatically fail as the approval process has not been completed within the specified timeframe. This ensures that the workflow does not get stuck indefinitely and allows for proper handling of job statuses.

GitHub provides runners with a scoped installation token that has read access to the repository where the actions are stored. This token is automatically generated and expires after one hour, ensuring access control and security by limiting the duration of access to the actions.

You can view a GitHub Actions workflow run status for a pull request on the pull request before a merge, on the checks tab of the pull request and within the GitHub Actions tab of the repository.

GitHub adds the "Set up job" step at the beginning of each job to prepare the environment and set up any necessary configurations. The "Complete job" step is added at the end of the job to finalize any actions or clean up resources used during the job execution.

Updating the IP address allowlist for GitHub-hosted runners on a weekly basis can be cumbersome for the operations team because it requires manual effort and is prone to errors. This repetitive task can consume valuable time and resources, leading to potential security vulnerabilities if not done accurately and promptly.

The default environment variables starting with "GITHUB_" and "RUNNER_" cannot be overwritten using the GITHUB_ENV file in a workflow. These variables are reserved and cannot be modified to maintain the integrity and security of GitHub Actions workflows.

Using a combination of GitHub-hosted and self-hosted runners can be beneficial when dealing with resource-intensive tasks. GitHub-hosted runners may have limitations in terms of resources, so adding self-hosted runners with higher capabilities can help handle tasks that require more processing power or memory.

Using the args keyword in the action metadata file allows you to pass the input value as an argument to the Docker container. This argument can then be accessed within the container as an environment variable, enabling you to retrieve the corresponding input value efficiently.

Modifying the entrypoint.sh script to explicitly set executable permissions before running will resolve the permission denied error. By setting the executable permissions, the script will be allowed to run as intended within the Docker container action

When drafting a new release and publishing an action to GitHub Marketplace, it is essential that the action's metadata file's category matches an existing GitHub Marketplace category. This ensures that the action is listed in the correct category for users to discover and use effectively.

To store secrets larger than 48 KB, you have to use encryption with GPG. The encrypted file is stored in the repository, and the decryption passphrase is stored as a secret on GitHub.

treat environment variables as case-sensitive

In GitHub Actions, the exit code of an action determines its success or failure status. A zero exit code indicates success, allowing the workflow to execute subsequent tasks. Conversely, a nonzero exit code signifies failure, leading to canceling concurrent actions and skipping future actions in the workflow. This distinction enables GitHub to accurately assess the outcome of each action and manage workflow execution accordingly.

Docker container actions can only be executed on runners with a Linux operating system. Self-hosted runners must use a Linux operating system and have Docker installed to run Docker container actions.

GitHub ensures the security of secrets printed to workflow logs within GitHub Actions by automatically redacting them. When secrets are printed to workflow logs, GitHub replaces them with placeholders to prevent their exposure. This security measure helps protect sensitive information from being unintentionally leaked or accessed by unauthorized users, ensuring the confidentiality and integrity of secrets used within GitHub Actions workflows.

Sam would like to trigger a workflow when a push is made to any branch in the repository, or somebody creates a tag. How can Sam specify these events within the GitHub workflow configuration?
on: [push, create]

When troubleshooting connectivity issues with a self-hosted runner and validating its access to GitHub services, the development team must use the --check feature alongside the runner's URL and authentication token. This combination allows them to diagnose and resolve any network connectivity issues efficiently, ensuring smooth workflow execution. By providing the necessary credentials, the team can validate that the self-hosted runner can access all required GitHub services effectively.

The recommended location to store action files when a GitHub repository contains multiple custom actions is within the **.github/actions** directory. Each action should have its own subdirectory within this directory to maintain organization and clarity.

restore-keys allows you to specify a list of alternative restore keys. These keys are used sequentially in the order provided to find and restore a cache in case of a cache miss.

Service containers are configured for each job in a workflow, and GitHub creates a fresh Docker container for each service configured in the workflow. These service containers are then destroyed when the job completes

GitHub limits the maximum number of jobs generated by a matrix to 256 jobs per workflow run

In GitHub Actions, debug messages printed using workflow commands are not displayed in the logs by default. They are only shown when debug logging is explicitly enabled for the workflow run. If debug logging is not enabled, the debug messages will not appear in the logs, even though they were printed successfully.
You must create a secret named ACTIONS_STEP_DEBUG with the value true to see the debug messages set by this command in the log.

Workflow commands prefixed with the :: can be used to customize the runner environment, such as setting environment variables or modifying the working directory

In GitHub Actions, you can use the group workflow command to create expandable groups in the workflow logs. By specifying a title with the group command, you can organize related log messages within a collapsible section, improving the readability and navigation of the workflow logs.

- The echo command is used to print messages to the workflow log but does not create expandable groups in the logs.
- The add-mask command will mask a value to prevent a string or variable from being printed in the log.
- The stop-commands command will stops processing any workflow commands. This special command allows you to log anything without accidentally running a workflow command.

To trigger a workflow when a comment is created on an issue within a GitHub repository, Dani should use the issue_comment event within the GitHub workflow configuration. Here's how Dani can specify this event:

1. on:
2. issue_comment:
3. types: [created]

If a variable with the same name exists at multiple levels, the variable at the lowest level takes precedence. For example, if an organization-level variable has the same name as a repository-level variable, then the repository-level variable takes precedence. Similarly, if an organization, repository, and environment have a variable with the same name, the environment-level variable takes precedence.

When an input is specified in a workflow file or using a default input value, GitHub creates an environment variable for the input using the naming convention of converting input names to UPPERCASE letters and replacing spaces with _ characters.

You can delete a workflow run that has been completed or is more than two weeks old.

the **check_suite** event is designed to exclusively trigger workflow runs on the default branch

The self-hosted runner connects to GitHub to receive job assignments and to download new versions of the runner application. The self-hosted runner uses an HTTPS *long poll* that opens a connection to GitHub for 50 seconds, and if no response is received, it then times out and creates a new long poll. The application must be running on the machine to accept and run GitHub Actions jobs.
Since the self-hosted runner opens a connection to GitHub.com, you do not need to allow GitHub to make inbound connections to your self-

hosted runner.
You must ensure that the machine has the appropriate network access to communicate with the GitHub hosts listed below. Some hosts are required for essential runner operations, while other hosts are only required for certain functionality.

The key benefit of using service containers in a GitHub Actions workflow is to provide easy access to actual or simulated external dependencies, such as databases or services needed for testing

The GITHUB_ACTIONS variable is always set to true when GitHub Actions is running the workflow. This can be used to differentiate whether tests are being run locally or GitHub Actions.

The “Publish” checkbox may be disabled if the account that owns the repository has not accepted the GitHub Marketplace Developer Agreement

To draft a new release and publish the action to GitHub Marketplace, follow these instructions:

1. On GitHub.com, navigate to the main page of the repository.
2. Navigate to the action metadata file in your repository (action.yml or action.yaml), and you'll see a banner to publish the action to GitHub Marketplace. Click **Draft a release**.
3. Under "Release Action", select **Publish this Action to the GitHub Marketplace**.
   Note: The "Publish" checkbox is disabled if the account that owns the repository has not yet accepted the GitHub Marketplace Developer Agreement. If you own the repository or are an organization owner, click the link to "accept the GitHub Marketplace Developer Agreement", then accept the agreement. If there is no link, send the organization owner a link to this "Release Action" page and ask them to accept the agreement.
4. If the labels in your metadata file contain any problems, you will see an error message. Address them by updating your metadata file. Once complete, you will see an "Everything looks good!" message.
5. Select the **Primary Category** dropdown menu and click a category that will help people find your action in GitHub Marketplace.
6. Optionally, select the **Another Category** dropdown menu and click a secondary category.
7. In the tag field, type a version for your action. This helps people know what changes or features the release includes. People will see the version in the action's dedicated GitHub Marketplace page.
8. In the title field, type a release title.
9. Complete all other fields and click **Publish release**. Publishing requires you to use two-factor authentication.

To enable step debug logging, set the following secret or variable in the repository that contains the workflow: ACTIONS_STEP_DEBUG to true. If both the secret and variable are set, the value of the secret takes precedence over the variable.

The branches and branches-ignore keywords accept glob patterns that use characters like *, **, +, ?, ! and others to match more than one branch name. If a name contains any of these characters and you want a literal match, you need to escape each of these special characters with \.

Fill in the blank: When using `push` event trigger filters you can use <____> patterns to target multiple branches…. Glob

Your Pull Request analysis workflow uses multiple code analysis tools and takes about 20minutes to fully complete. It is triggered on `pull_request` event with `branches` filter set to `master`. Therefore if a developer pushes multiple commits within few minutes multiple workflows are running in parallel. How can you stop all previous workflow runs and only run the one with latest changes? questions Question 029

1. Use concurrency with cancel-in-progress

```
concurrency:
  group: ${{ github.workflow }}-${{ github.ref }}
  cancel-in-progress: true
```

Which is true about `Starter Workflows` ? (Select three.)

- They allow users to leverage ready-to-use (or requiring minimal changes) workflow templates
- GitHub provides and maintains starter workflows for different categories, languages and tooling
- Your organization can create custom starter workflows for users in your organization

How can you require manual approvals by a maintainer if the workflow run is targeting the `production` environment?

1. Using deployment protection rules

Which is true about environments? questions Question 057

Each job in a workflow can reference a single environment.

How can organizations which are using GitHub Enterprise Server enable automatic syncing of third party GitHub Actions hosted on GitHub.com to their GitHub Enterprise Server instance?

1. Using GitHub Connect

Where can you find network connectivity logs for a GitHub self-hosted-runner?

1. In the _diag folder directly on the runner machine

https://docs.github.com/en/actions

How can you use the GitHub API to download workflow run logs?

1. GET /repos/{owner}/{repo}/actions/runs/{run_id}/logs

How can you use the GitHub API to create or update a repository secret?

1. PUT /repos/{owner}/{repo}/actions/secrets/{secret_name}

Artifacts are used to preserve data after a job has completed and/or share that data with another job within the same workflow.
Cache can be reused across workflows within one repository
Environment variables can be scoped to a step, job or a workflow. They cannot be shared across workflows/repositories or organizations

What's the maximum amount of reusable workflows that can be called from a single workflow file? 20

With keyword is used for giving input

What is the default timeout for a GitHub Actions job? - 360 min

In a GitHub Actions workflow, how do you specify a specific version of Node.js to use in a job?

```
uses: actions/setup-node@v4
with:
  node-version: 20
```

What is the default shell used by GitHub Actions on Windows runners? Powershell
Linux - bash
Mac = zsh\\

What is the purpose of the `restore-keys` parameter in `actions/cache` in GitHub Actions?

1. provide alternative keys to use in case of a cache miss

How can you skip the following workflow run when you commit or create a PR? questions Question 118

By including any one of the following keywords in the commit message or in the title of the pull-request
[skip ci]
[ci skip]
[no ci]
[skip actions]
[actions skip]

What is the correct syntax for specifying a cleanup script in a container action?

```
runs:
  using: 'docker'
  image: 'Dockerfile'
  entrypoint: 'entrypoint.sh'
  post-entrypoint: 'cleanup.sh'
```

In GitHub Actions, how can you ensure that a specific job in a workflow only runs if changes were made to files in either of two different directories?
Create a preliminary job to check for changes in the specified directories and use its output in the if condition of the dependent jobs.

In GitHub Actions, what is the correct approach to ensure that a workflow is triggered by a push event only when specific files or directories change?
Use the on: push: paths: ['specific-path/*'] syntax to trigger the workflow only when changes occur in files or directories under 'specific-path'.

When creating a custom GitHub Action in a public repository, what is the best practice for ensuring the action's code adheres to consistent coding standards and best practices?
Implement a linter in the action's development workflow to automatically check code submissions for adherence to defined coding standards.

In the context of consuming workflows in GitHub Actions, how can you trigger a workflow in one repository as a result of an event in a separate repository?

Use the on: repository_dispatch event in the consuming repository, and send a repository dispatch event from the source repository.

Scoping secrets to specific branches or environments within a repository helps maintain tight access control and ensures that secrets are only available where absolutely necessary.

In GitHub Actions, how would you configure a workflow to automatically cancel previous runs of the same workflow on the same branch when a new run is triggered?

Use the concurrency keyword with a unique group name that includes the branch name to automatically cancel overlapping runs.

The concurrency keyword allows you to manage the concurrency of workflow runs. By setting it with a group name that includes the branch name (e.g., concurrency: { group: workflow-${{ github.ref }} }), GitHub Actions will automatically cancel any in-progress job or workflow in the same concurrency group (i.e., the same branch in this case) when a new run is triggered.

When authoring a JavaScript-based custom GitHub Action, what is the recommended approach to manage third-party dependencies that the action requires?
Bundle the dependencies with a tool like Webpack, and commit the bundled file along with your action code to the repository.

When developing a custom GitHub Action that interacts with external APIs, what is the best strategy to manage and rotate API keys or tokens to enhance security?

Store the API keys or tokens as encrypted secrets in the GitHub repository and reference them in the action's code.

In the context of GitHub Actions, what is the correct use of environment keyword in a workflow file?
environment is utilized to specify the deployment environment, such as production, staging, or development, and can enforce additional rules like manual approvals.

For a custom GitHub Action you are developing, which method is most appropriate for debugging issues that occur during the action's execution in a workflow?
Utilize console.log statements in the action's code and review the output in the GitHub Actions workflow logs.

In GitHub Actions, you want to consume a workflow from another repository and trigger it whenever a new issue is opened in your repository. How can you achieve this?
Use the on: repository_dispatch event in the target repository's workflow and dispatch an event from your repository when a new issue is opened.

Proper network configuration, including setting up proxies and IP allow lists, is crucial to ensuring that self-hosted runners operate securely and efficiently within an enterprise environment.

In GitHub Actions, how can you dynamically generate a matrix for a job to run against multiple configurations, using data from an external JSON file hosted in the same repository?
Implement a custom action that reads the JSON file and outputs the matrix configuration, then use this output in the job's matrix setting.

Configure a repository_dispatch event in the source repository and trigger it manually when a new release is published.

When configuring IP allow lists for GitHub-hosted and self-hosted runners, what is the primary effect?
It restricts which IP addresses can interact with your runners, enhancing security.

In GitHub Actions, how can you utilize artifacts generated in one workflow in a separate, subsequent workflow within the same repository?
Use the on: workflow_run trigger in the subsequent workflow and utilize the actions/download-artifact@v2 action to fetch the artifacts

How to create a manual approval step in a workflow ?

▼ show

There are different steps. First you need to create an **Environment** (let's call it "dev").

Then, in this environment you need to define **protection rules** and especially the rule "Required reviewers"

Finally, you need for the job you want an approval for, to reference the newly created/configured environment :

```
jobs:
  build-and-publish:
    runs-on: ubuntu-latest
    steps:

  deploy-dev:
    runs-on: 'ubuntu-latest'
    environment: 'dev'
```

Which variable contains the name of the repository and the name of the owner ?

▼ show

GITHUB_REPOSITORY

Which variable contains the name of the user who triggered the workflow ?

▼ show

GITHUB_ACTOR

Which variable contains the name of the branch or tag which triggered the current workflow ?

▼ show

GITHUB_REF_NAME

/!\ GITHUB_REF will contains the full name *(refs/heads/<branch_name>)*

Which fields are mandatory in the metadata file ?

- [ ] name
- [ ] description
- [ ] version
- [ ] inputs
- [ ] author

▼ show

Name and **description**. **Inputs** and **author** are optional. **Version** does not exist

How can you customize the icon and the background color of your custom GitHub Action ?

▼ show

By defining a branding section in the metadata file:

```
branding:
  icon: 'award'
  color: 'green'
```

What is the docker command to publish a container image on GitHub Packages ?

▼ show

```
docker push ghcr.io/OWNER/IMAGE_NAME:latest
```

The core.warning() function from the @actions/core package is used to create a warning message in the workflow logs. This is an annotation type that informs users about issues that don't require failing the build but still need attention.

- azure/webapps-deploy@v1
- azure/login@v1 that we saw previously
- azure/docker-login@v1

A workflow that uses another workflow is referred to as a "caller" workflow. The reusable workflow is a "called" workflow.

When a reusable workflow is triggered by a caller workflow, the github context is always associated with the caller workflow. The called workflow is automatically granted access to github.token and secrets.GITHUB_TOKEN.

The dependency graph is a summary of the manifest and lock files stored in a repository and any dependencies that are submitted for the repository using the dependency submission API.

The dependency graph is automatically generated for all public repositories. You can choose to enable it for forks and for private repositories.

Workflow templates created by users can only be used to create workflows in public repositories. Organizations using GitHub Enterprise Cloud can also use workflow templates to create workflows in private repositories

Create a metadata file inside the workflow-templates directory. The metadata file must have the same name as the workflow file, but instead of the .yml extension, it must be appended with .properties.json.

If a secret was used in the job, GitHub automatically redacts secrets printed to the log. You should avoid printing secrets to the log intentionally

You can store up to 1,000 organization secrets, 100 repository secrets, and 100 environment secrets.

A workflow created in a repository can access the following number of secrets:

- All 100 repository secrets.
- If the repository is assigned access to more than 100 organization secrets, the workflow can only use the first 100 organization secrets (sorted alphabetically by secret name).
- All 100 environment secrets.

Secrets are limited to 48 KB in size. To store larger secrets, see the Storing large secrets workaround below.

gpg --symmetric --cipher-algo AES256 my_secret.json

As a habit of best practice, you should mask all sensitive information that is not a GitHub secret by using ::add-mask::VALUE.

Copy the code example and save it to a file called delete-logs.sh.

Grant it the execute permission with chmod +x delete-logs.sh.

Run the following command, where REPOSITORY_NAME is the name of your repository and WORKFLOW_NAME is the file name of your workflow.

./delete-logs.sh REPOSITORY_NAME WORKFLOW_NAME

gh run view RUN_ID --log

gh run view --job JOB_ID --log

You can connect up to four levels of workflows.

You can call a maximum of 20 unique reusable workflows from a single workflow file. This limit includes any trees of nested reusable workflows that may be

called starting from your top-level caller workflow file.

Any environment variables set in an env context defined at the workflow level in the caller workflow are not propagated to the called workflow.

Similarly, environment variables set in the env context, defined in the called workflow, are not accessible in the env context of the caller workflow. Instead, you must use outputs of the reusable workflow

To reuse variables in multiple workflows, set them at the organization, repository, or environment levels and reference them using the vars context.

You reference reusable workflow files using one of the following syntaxes:

- {owner}/{repo}/.github/workflows/{filename}@{ref} for reusable workflows in public and private repositories.
- ./.github/workflows/{filename} for reusable workflows in the same repository.

Re-running a workflow or jobs in a workflow uses the same GITHUB_SHA (commit SHA) and GITHUB_REF (Git ref) of the original event that triggered the workflow run.

You can re-run a workflow or jobs in a workflow for up to 30 days after the initial run. You cannot re-run jobs in a workflow once its logs have passed their retention limits.

```
gh run rerun RUN_ID
```

```
gh run rerun --job JOB_ID --debug
```

Workflows will not run on pull_request activity if the pull request has a merge conflict. The merge conflict must be resolved first.

Workflows don't run in forked repositories by default. You must enable GitHub Actions in the **Actions** tab of the forked repository.

In a public repository, scheduled workflows are automatically disabled when no repository activity has occurred in 60 days.

You can't use workflow_run to chain together more than three levels of workflows. For example, if you attempt to trigger five workflows (named B to F) to run sequentially after an initial workflow A has run (that is: A → B → C → D → E → F), workflows E and F will not be run.

```
steps:
    - run: gh issue comment $ISSUE --body "Thank you for opening this issue!"
```

::group::{title}
::endgroup::

This workflow uses an imaginary secret store, secret-store, which has imaginary commands store-secret and retrieve-secret

```
console.log("The running PID from the main action is: " + process.env.STATE_processID);
```

```
permissions: write-all
```

You can store up to 1,000 organization variables, 500 variables per repository, and 100 variables per environment. The total combined size limit for organization and repository variables is 256 KB per workflow run.

```
jobs:
  deployment:
    runs-on: ubuntu-latest
    environment: production
    steps:
      - name: deploy
        # ...deployment-specific steps
```

You can use jobs.<job_id>.concurrency to ensure that only a single job or workflow using the same concurrency group will run at a time.

To also cancel any currently running job or workflow in the same concurrency group, specify cancel-in-progress: true.

A matrix will generate a maximum of 256 jobs per workflow run. This limit applies to both GitHub-hosted and self-hosted runners.

Package managers	setup-* action for caching
npm, Yarn, pnpm	setup-node
pip, pipenv, Poetry	setup-python
Gradle, Maven	setup-java
RubyGems	setup-ruby
Go `go.sum`	setup-go
.NET NuGet	setup-dotnet

The cache action will attempt to restore a cache based on the key you provide. When the action finds a cache that *exactly* matches the key, the action restores the cached files to the path you configure. You can optionally provide a list of restore-keys to use in case the key doesn't match an existing cache. A list of restore-keys is useful when you are restoring a cache from another branch because restore-keys can *partially* match cache keys.

gh cache delete $cacheKey

https://docs.github.com/en/actions/writing-workflows/workflow-syntax-for-github-actions#onworkflow_dispatchinputsinput_idtype

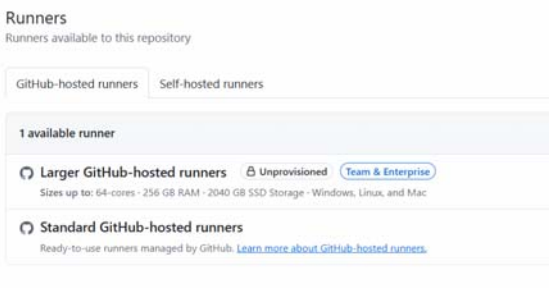
Can be nested to have up to 10 composite actions in one workflow

Reausable - Can connect a maximum of four levels of workflows

Checking in your node_modules directory can cause problems. As an alternative, you can use a tool called @vercel/ncc to compile your code and modules into one file used for distribution.

You can share actions and reusable workflows from your private repository, without making them public, by allowing GitHub Actions workflows to access a private repository that contains the action or reusable workflow.

Using GitHub-hosted runners requires network access with at least 70 kilobits per second upload and download speeds.



Plan	Storage	Minutes (per month)
GitHub Free	500 MB	2,000
GitHub Pro	1 GB	3,000
GitHub Free for organizations	500 MB	2,000
GitHub Team	2 GB	3,000
GitHub Enterprise Cloud	50 GB	50,000

Jobs that run on Windows and macOS runners that GitHub hosts consume minutes at 2 and 10 times the rate that jobs on Linux runners consume. For example, using 1,000 Windows minutes would consume 2,000 of the minutes included in your account. Using 1,000 macOS minutes, would consume 10,000 minutes included in your account.

Per-minute rates for standard runners

Operating system	Per-minute rate (USD)
Linux 2-core	$0.008
Windows 2-core	$0.016
macOS 3-core or 4-core (M1 or Intel)	$0.08

Per-minute rates for x64-powered larger runners

Operating system	Per-minute rate (USD)
Linux Advanced 2-core	$0.008
Linux 4-core	$0.016
Linux 8-core	$0.032

If GitHub Actions services are temporarily unavailable, then a workflow run is discarded if it has not been queued within 30 minutes of being triggered.

Linux- sudo, macos- brew, windows- choco

Larger runners are not eligible for the use of included minutes on private repositories. For both private and public repositories, when larger runners are in use, they will always be billed at the per-minute rate.

Autoscaling is only available for larger runners with Linux or Windows operating systems.

Private networking for GitHub-hosted runners does not support static IP addresses for larger runners.

You can use up to 10 larger runners with static IP address ranges in total across all your larger runners.

If runners are unused for more than 30 days, their IP address ranges are automatically removed and cannot be recovered.

Larger runners are only available for organizations and enterprises using the GitHub Team or GitHub Enterprise Cloud plans.

Enterprise or organization owners can manage larger runners.

For example, the following diagram has a runner group named grp-ubuntu-20.04-16core at the enterprise level. Before the repository named octo-repo can use the runners in the group, you must first configure the group at the enterprise level to allow access to the octo-org organization. You must then configure the group at the organization level to allow access to octo-repo.

```
runs-on:
    group: ubuntu-runners
    labels: ubuntu-20.04-16core
```

A self-hosted runner is automatically removed from GitHub if it has not connected to GitHub Actions for more than 14 days. An ephemeral self-hosted runner is automatically removed from GitHub if it has not connected to GitHub Actions for more than 1 day.

Each job in a workflow can run for up to 5 days of execution time. If a job reaches this limit, the job is terminated and fails to complete.

You can execute up to 1,000 requests to the GitHub API in an hour across all actions within a repository. If requests are exceeded, additional API calls will fail which might cause jobs to fail.

You can have a maximum of 10,000 self-hosted runners in one runner group. If this limit is reached, adding a new runner will not be possible.

You must ensure that the machine has the appropriate network access with at least 70 kilobits per second upload and download speed to communicate with the GitHub hosts listed below.

The status can be one of the following:

- **Idle:** The runner is connected to GitHub and is ready to execute jobs.
- **Active:** The runner is currently executing a job.
- **Offline:** The runner is not connected to GitHub. This could be because the machine is offline, the self-hosted runner application is not

running on the machine, or the self-hosted runner application cannot communicate with GitHub.

In addition to --check, you must provide two arguments to the script:

- --url with the URL to your GitHub repository, organization, or enterprise. For example, --url https://github.com/octo-org/octo-repo.
- --pat with the value of a personal access token (classic), which must have the workflow scope, or a fine-grained personal access token with workflows read and write access.

To disable TLS certification verification in the self-hosted runner application, set the GITHUB_ACTIONS_RUNNER_TLS_NO_VERIFY environment variable to 1

You can monitor the status of the self-hosted runner application and its activities. Log files are kept in the _diag directory where you installed the runner application, and a new log is generated each time the application is started. The filename begins with Runner_, and is followed by a UTC timestamp of when the application was started.

The self-hosted runner application creates a detailed log file for each job that it processes. These files are stored in the _diag directory where you installed the runner application, and the filename begins with Worker_.

You can use journalctl to monitor the real-time activity of the self-hosted runner:

You can use systemctl to check the service status:

Actions Runner Controller (ARC) is a Kubernetes operator that orchestrates and scales self-hosted runners for GitHub Actions.

run: gh issue comment "$NUMBER" --body "$BODY"

GitHub Actions is available on all GitHub products, but GitHub Actions is not available for private repositories owned by accounts using legacy per-repository plans.

GitHub plan	Total concurrent jobs	Maximum concurrent macOS jobs
Free	20	5
Pro	40	5
Team	60	5
Enterprise	500	50

GitHub-hosted larger runners

GitHub plan	Total concurrent jobs	Maximum concurrent macOS jobs	Maximum concurrent GPU jobs
Team	1000	5	100
Enterprise	1000	50	100

- For public repositories: you can change this retention period to anywhere between 1 day or 90 days.
- For private repositories: you can change this retention period to anywhere between 1 day or 400 days.

When you customize the retention period, it only applies to new artifacts and log files, and does not retroactively apply to existing objects.

The workflow runs in a repository's workflow run history are retained for 400 days. After 400 days, workflow runs are archived. 10 days after archival, they are permanently deleted. The retention period for workflow runs cannot be modified.

- *: Matches zero or more characters, but does not match the / character. For example, Octo* matches Octocat.
- **: Matches zero or more of any character.
- ?: Matches zero or one of the preceding character.
- +: Matches one or more of the preceding character.
- [] Matches one alphanumeric character listed in the brackets or included in ranges. Ranges can only include a-z, A-Z, and 0-9. For example, the range[0-9a-z] matches any digit or lowercase letter. For example, [CB]at matches Cat or Bat and [1-2]00 matches 100 and 200.
- !: At the start of a pattern makes it negate previous positive patterns. It has no special meaning if not the first character.