Autoscaling with self-hosted runners

You can automatically scale your self-hosted runners in response to webhook events.

About autoscaling ⊘

You can automatically increase or decrease the number of self-hosted runners in your environment in response to the webhook events you receive with a particular label. For example, you can create automation that adds a new self-hosted runner each time you receive a `workflow_job` webhook event with the `queued` activity, which notifies you that a new job is ready for processing. The webhook payload includes label data, so you can identify the type of runner the job is requesting. Once the job has finished, you can then create automation that removes the runner in response to the `workflow_job` `completed` activity.

Recommended autoscaling solutions ⊘

GitHub recommends and partners closely with two open source projects that you can use for autoscaling your runners. One or both solutions may be suitable, based on your needs.

The following repositories have detailed instructions for setting up these autoscalers:

- actions/actions-runner-controller - A Kubernetes controller for GitHub Actions self-hosted runners.
- philips-labs/terraform-aws-github-runner - A Terraform module for scalable GitHub Actions runners on Amazon Web Services.

Each solution has certain specifics that may be important to consider.

	actions-runner-controller	terraform-aws-github-runner
Runtime	Kubernetes	Linux and Windows VMs
Supported Clouds	Azure, Amazon Web Services, Google Cloud Platform, on-premises	Amazon Web Services
Where runners can be scaled	Enterprise, organization, and repository levels. By runner	Organization and repository levels. By runner label and

How runners can be scaled	Webhook events, Scheduled, Pull-based	Webhook events, Scheduled (org-level runners only)

Using ephemeral runners for autoscaling 🔗

GitHub recommends implementing autoscaling with ephemeral self-hosted runners; autoscaling with persistent self-hosted runners is not recommended. In certain cases, GitHub cannot guarantee that jobs are not assigned to persistent runners while they are shut down. With ephemeral runners, this can be guaranteed because GitHub only assigns one job to a runner.

This approach allows you to manage your runners as ephemeral systems, since you can use automation to provide a clean environment for each job. This helps limit the exposure of any sensitive resources from previous jobs, and also helps mitigate the risk of a compromised runner receiving new jobs.

To add an ephemeral runner to your environment, include the `--ephemeral` parameter when registering your runner using `config.sh`. For example:

```
./config.sh --url https://github.com/octo-org --token example-token --ephemeral
```

The GitHub Actions service will then automatically de-register the runner after it has processed one job. You can then create your own automation that wipes the runner after it has been de-registered.

> **Note:** If a job is labeled for a certain type of runner, but none matching that type are available, the job does not immediately fail at the time of queueing. Instead, the job will remain queued until the 24 hour timeout period expires.

Alternatively, you can create ephemeral, just-in-time runners using the REST API. For more information, see "[Actions](#)."

Controlling runner software updates on self-hosted runners 🔗

By default, self-hosted runners will automatically perform a software update whenever a new version of the runner software is available. If you use ephemeral runners in containers then this can lead to repeated software updates when a new runner version is released. Turning off automatic updates allows you to update the runner version on the container image directly on your own schedule.

To turn off automatic software updates and install software updates yourself, specify the `--disableupdate` flag when registering your runner using `config.sh`. For example:

```
./config.sh --url https://github.com/YOUR-ORGANIZATION --token EXAMPLE-TOKEN --disableupdate
```

If you disable automatic updates, you must still update your runner version regularly. New functionality in GitHub Actions requires changes in both the GitHub Actions service *and* the runner software. The runner may not be able to correctly process jobs that take advantage of new features in GitHub Actions without a software update.

If you disable automatic updates, you will be required to update your runner version within 30 days of a new version being made available. You may want to subscribe to

notifications for releases in the `actions/runner` repository. For more information, see "Configuring notifications."

For instructions on how to install the latest runner version, see the installation instructions for the latest release.

> **Note:** If you do not perform a software update within 30 days, the GitHub Actions service will not queue jobs to your runner. In addition, if a critical security update is required, the GitHub Actions service will not queue jobs to your runner until it has been updated.

Using webhooks for autoscaling

You can create your own autoscaling environment by using payloads received from the `workflow_job` webhook. This webhook is available at the repository, organization, and enterprise levels, and the payload for this event contains an `action` key that corresponds to the stages of a workflow job's life-cycle; for example when jobs are `queued`, `in_progress`, and `completed`. You must then create your own scaling automation in response to these webhook payloads.

- For more information about the `workflow_job` webhook, see "Webhook events and payloads."
- To learn how to work with webhooks, see "Webhooks documentation."

Authentication requirements

You can register and delete repository and organization self-hosted runners using the API. To authenticate to the API, your autoscaling implementation can use an access token or a GitHub app.

Your access token will require the following scope:

- For private repositories, use an access token with the `repo` scope.
- For public repositories, use an access token with the `public_repo` scope.
- For organizations, use an access token with the `admin:org` scope.

To authenticate using a GitHub App, it must be assigned the following permissions:

- For repositories, assign the `administration` permission.
- For organizations, assign the `organization_self_hosted_runners` permission.

You can register and delete enterprise self-hosted runners using the API. To authenticate to the API, your autoscaling implementation can use an access token.

Your access token will require the `manage_runners:enterprise` scope.