Running scripts before or after a job

In this article

Scripts can automatically execute on a self-hosted runner, directly before or after a job.

About pre- and post-job scripts

You can automatically execute scripts on a self-hosted runner, either before a job runs, or after a job finishes running. You could use these scripts to support the job's requirements, such as building or tearing down a runner environment, or cleaning out directories. You could also use these scripts to track telemetry of how your runners are used.

The custom scripts are automatically triggered when a specific environment variable is set on the runner; the environment variable must contain the absolute path to the script. For more information, see "Triggering the scripts" below.

The following scripting languages are supported:

- **Bash**: Uses `bash` and can fallback to `sh`. Executes by running `-e {pathtofile}`.
- **PowerShell**: Uses `pwsh` and can fallback to `powershell`. Executes by running `-command \". '{pathtofile}'\"`.

Writing the scripts

Your custom scripts can use the following features:

- **Variables**: Scripts have access to the default variables. The full webhook event payload can be found in `GITHUB_EVENT_PATH`. For more information, see "Variables."
- **Workflow commands**: Scripts can use workflow commands. For more information, see "Workflow commands for GitHub Actions". Scripts can also use environment files. For more information, see Environment files.

Your script files must use a file extension for the relevant language, such as `.sh` or `.ps1`, in order to run successfully.

> **Note**: Avoid using your scripts to output sensitive information to the console, as anyone with read access to the repository might be able to see the output in the UI logs.

Handling exit codes

For pre-job scripts, exit code `0` indicates that the script completed successfully, and the

job will then proceed to run. If there is any other exit code, the job will not run and will be marked as failed. To see the results of your pre-job scripts, check the logs for `Set up runner` entries. For more information on checking the logs, see "Using workflow run logs."

The `continue-on-error` setting is not supported for use by these scripts.

Triggering the scripts

The custom scripts must be located on the runner, but should not be stored in the `actions-runner` application directory. The scripts are executed in the security context of the service account that's running the runner service.

> **Note**: The triggered scripts are processed synchronously, so they will block job execution while they are running.

The scripts are automatically executed when the runner has the following environment variables containing an absolute path to the script:

- `ACTIONS_RUNNER_HOOK_JOB_STARTED`: The script defined in this environment variable is triggered when a job has been assigned to a runner, but before the job starts running.
- `ACTIONS_RUNNER_HOOK_JOB_COMPLETED`: The script defined in this environment variable is triggered at the end of the job, after all the steps defined in the workflow have run.

To set these environment variables, you can either add them to the operating system, or add them to a file named `.env` within the self-hosted runner application directory (that is, the directory into which you downloaded and unpacked the runner software). For example, the following `.env` entry will have the runner automatically run a script, saved as `/opt/runner/cleanup_script.sh` on the runner machine, before each job runs:

```
ACTIONS_RUNNER_HOOK_JOB_STARTED=/opt/runner/cleanup_script.sh
```

> **Note**: The script defined in `ACTIONS_RUNNER_HOOK_JOB_COMPLETED` is executed at the end of the job, before the job completes. This makes it unsuitable for use cases that may interrupt a runner, such as deleting the runner machine as part of an autoscaling implementation.

Troubleshooting

Permission denied

If you get a "permission denied" error when you attempt to run a script, make sure that the script is executable. For example, in a terminal on Linux or MacOS you can use the following command to make a file executable.

```
chmod +x PATH/TO/FILE
```

For information about using workflows to run scripts, see "Essential features of GitHub Actions."

No timeout setting

There is currently no timeout setting available for scripts executed by `ACTIONS_RUNNER_HOOK_JOB_STARTED` or `ACTIONS_RUNNER_HOOK_JOB_COMPLETED`. As a result,

you could consider adding timeout handling to your script.

Reviewing the workflow run log

To confirm whether your scripts are executing, you can review the logs for that job. The scripts will be listed within separate steps for either `Set up runner` or `Complete runner`, depending on which environment variable is triggering the script. For more information on checking the logs, see "Using workflow run logs."