Understanding GitHub Actions

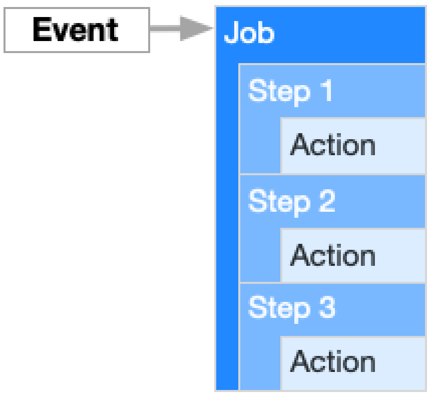
docs.github.com/en/actions/learn-github-actions/understanding-github-actions

Learn the basics of GitHub Actions, including core concepts and essential terminology.

Overview

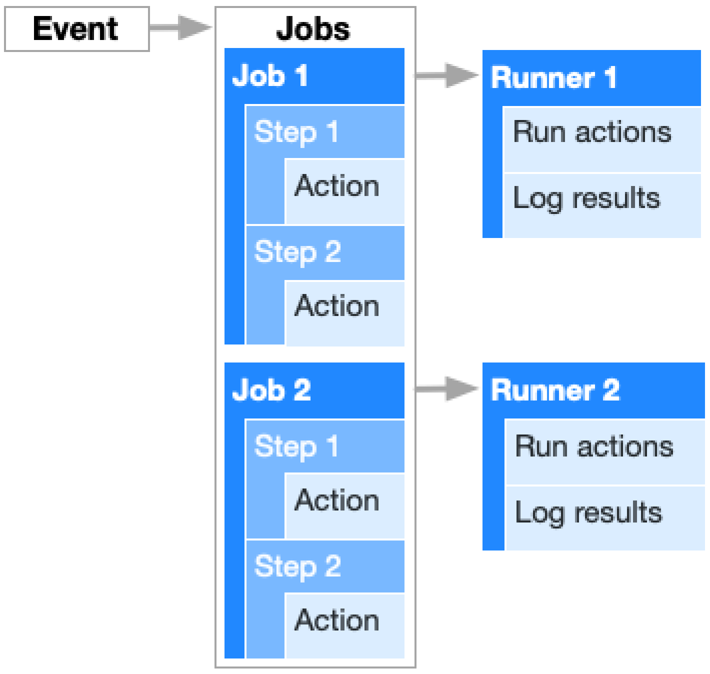
GitHub Actions help you automate tasks within your software development life cycle. GitHub Actions are event-driven, meaning that you can run a series of commands after a specified event has occurred. For example, every time someone creates a pull request for a repository, you can automatically run a command that executes a software testing script.

This diagram demonstrates how you can use GitHub Actions to automatically run your software testing scripts. An event automatically triggers the *workflow*, which contains a *job*. The job then uses *steps* to control the order in which *actions* are run. These actions are the commands that automate your software testing.



The components of GitHub Actions

Below is a list of the multiple GitHub Actions components that work together to run jobs. You can see how these components interact with each other.



Workflows

The workflow is an automated procedure that you add to your repository. Workflows are made up of one or more jobs and can be scheduled or triggered by an event. The workflow can be used to build, test, package, release, or deploy a project on GitHub. You can reference a workflow within another workflow, see "Reusing workflows."

Events

An event is a specific activity that triggers a workflow. For example, activity can originate from GitHub when someone pushes a commit to a repository or when an issue or pull request is created. You can also use the repository dispatch webhook to trigger a workflow when an external event occurs. For a complete list of events that can be used to trigger workflows, see Events that trigger workflows.

Jobs

A job is a set of steps that execute on the same runner. By default, a workflow with multiple jobs will run those jobs in parallel. You can also configure a workflow to run jobs sequentially. For example, a workflow can have two sequential jobs that build and test code, where the test job is dependent on the status of the build job. If the build job fails, the test job will not run.

Steps

A step is an individual task that can run commands in a job. A step can be either an *action* or a shell command. Each step in a job executes on the same runner, allowing the actions in that job to share data with each other.

Actions

Actions are standalone commands that are combined into *steps* to create a *job*. Actions are the smallest portable building block of a workflow. You can create your own actions, or use actions created by the GitHub community. To use an action in a workflow, you must include it as a step.

Runners

A runner is a server that has the GitHub Actions runner application installed. You can use a runner hosted by GitHub, or you can host your own. A runner listens for available jobs, runs one job at a time, and reports the progress, logs, and results back to GitHub. GitHub-hosted runners are based on Ubuntu Linux, Microsoft Windows, and macOS, and each job in a workflow runs in a fresh virtual environment. For information on GitHub-hosted runners, see "About GitHub-hosted runners." If you need a different operating system or require a specific hardware configuration, you can host your own runners. For information on self-hosted runners, see "Hosting your own runners."

Create an example workflow

GitHub Actions uses YAML syntax to define the events, jobs, and steps. These YAML files are stored in your code repository, in a directory called `.github/workflows`.

You can create an example workflow in your repository that automatically triggers a series of commands whenever code is pushed. In this workflow, GitHub Actions checks out the pushed code, installs the software dependencies, and runs `bats -v`.

1. In your repository, create the `.github/workflows/` directory to store your workflow files.
2. In the `.github/workflows/` directory, create a new file called `learn-github-actions.yml` and add the following code.

   ```
   name: learn-github-actions
   on: [push]
   jobs:
     check-bats-version:
       runs-on: ubuntu-latest
       steps:
         - uses: actions/checkout@v2
         - uses: actions/setup-node@v2
           with:
             node-version: '14'
         - run: npm install -g bats
         - run: bats -v
   ```

3. Commit these changes and push them to your GitHub repository.

Your new GitHub Actions workflow file is now installed in your repository and will run automatically each time someone pushes a change to the repository. For details about a job's execution history, see "Viewing the workflow's activity."

Understanding the workflow file

To help you understand how YAML syntax is used to create a workflow file, this section explains each line of the introduction's example:

`name: learn-github-actions`	*Optional* - The name of the workflow as it will appear in the Actions tab of the GitHub repository.
`on: [push]`	Specify the event that automatically triggers the workflow file. This example uses the `push` event, so that the jobs run every time someone pushes a change to the repository. You can set up the workflow to only run on certain branches, paths, or tags. For syntax examples including or excluding branches, paths, or tags, see "Workflow syntax for GitHub Actions."
`jobs:`	Groups together all the jobs that run in the `learn-github-actions` workflow file.
`check-bats-version:`	Defines the name of the `check-bats-version` job stored within the `jobs` section.
`runs-on: ubuntu-latest`	Configures the job to run on an Ubuntu Linux runner. This means that the job will execute on a fresh virtual machine hosted by GitHub. For syntax examples using other runners, see "Workflow syntax for GitHub Actions."
`steps:`	Groups together all the steps that run in the `check-bats-version` job. Each item nested under this section is a separate action or shell command.
`- uses: actions/checkout@v2`	The `uses` keyword tells the job to retrieve `v2` of the community action named `actions/checkout@v2` . This is an action that checks out your repository and downloads it to the runner, allowing you to run actions against your code (such as testing tools). You must use the checkout action any time your workflow will run against the repository's code or you are using an action defined in the repository.

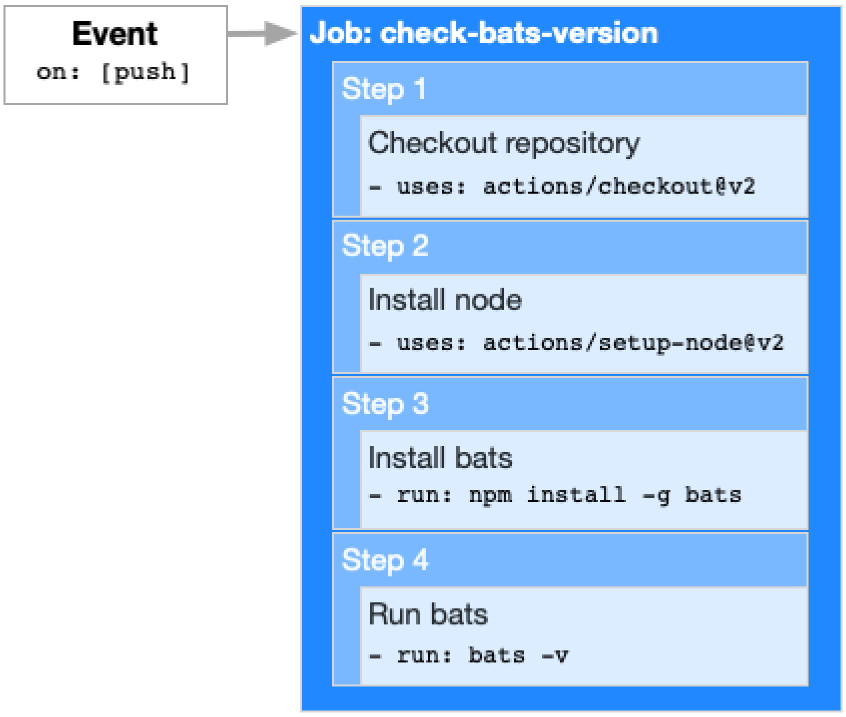
`- uses: actions/setup-node@v2`
`  with:`
`    node-version: '14'`
	This step uses the `actions/setup-node@v2` action to install the specified version of the `node` software package on the runner, which gives you access to the `npm` command.

`- run: npm install -g bats`	The `run` keyword tells the job to execute a command on the runner. In this case, you are using `npm` to install the `bats` software testing package.

`- run: bats -v`	Finally, you'll run the `bats` command with a parameter that outputs the software version.

Visualizing the workflow file

In this diagram, you can see the workflow file you just created and how the GitHub Actions components are organized in a hierarchy. Each step executes a single action or shell command. Steps 1 and 2 use prebuilt community actions. Steps 3 and 4 run shell commands directly on the runner. To find more prebuilt actions for your workflows, see "Finding and customizing actions."

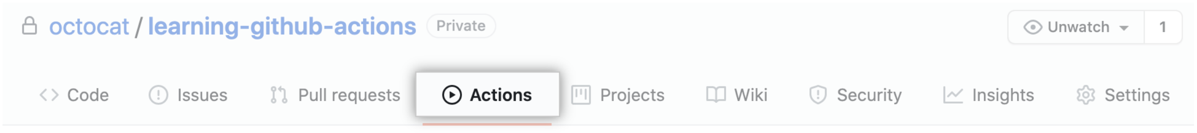


Viewing the job's activity

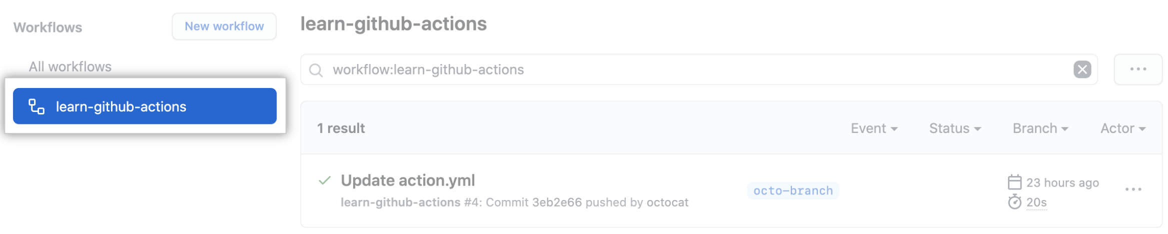
Once your job has started running, you can see a visualization graph of the run's progress and view each step's activity on GitHub.

1. On GitHub, navigate to the main page of the repository.

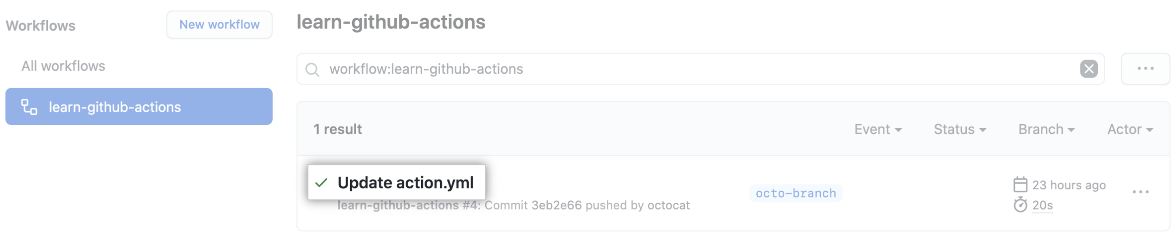
2. Under your repository name, click **Actions**.



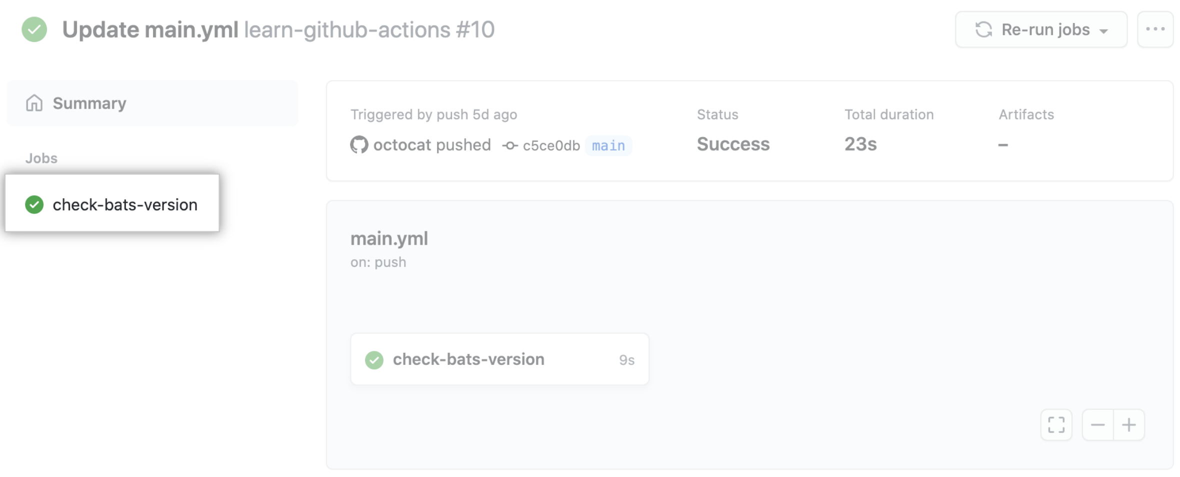
3. In the left sidebar, click the workflow you want to see.



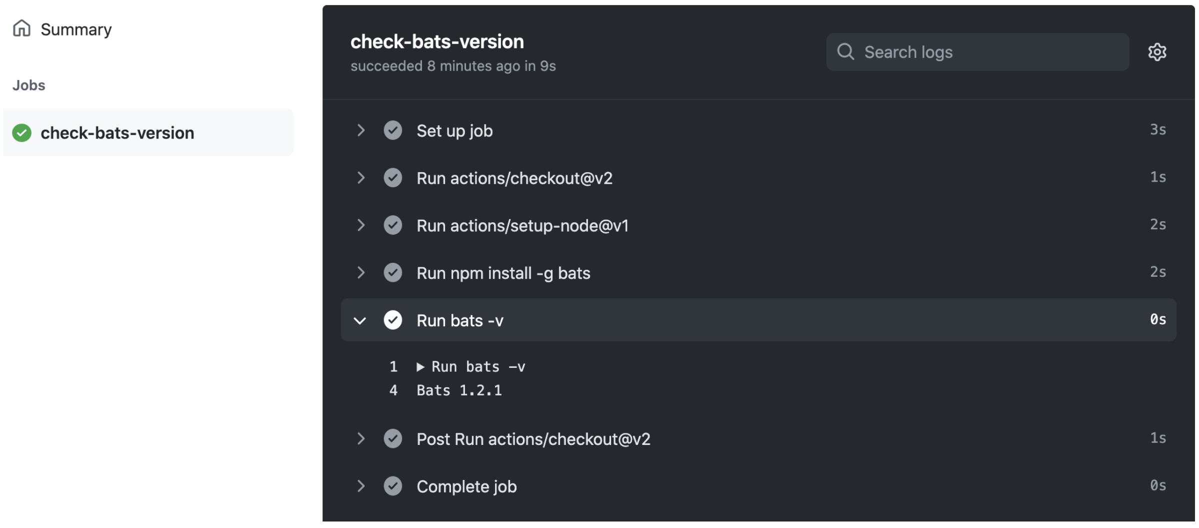
4. Under "Workflow runs", click the name of the run you want to see.



5. Under **Jobs** or in the visualization graph, click the job you want to see.



6. View the results of each step.


Summary
Jobs
check-bats-version
check-bats-version
succeeded 8 minutes ago in 9s
Search logs
Set up job 3s
Run actions/checkout@v2 1s
Run actions/setup-node@v1 2s
Run npm install -g bats 2s
Run bats -v 0s

```
1  ▶ Run bats -v
4  Bats 1.2.1
```

Post Run actions/checkout@v2 1s
Complete job 0s

Next steps

To continue learning about GitHub Actions, see "Finding and customizing actions."

To understand how billing works for GitHub Actions, see "About billing for GitHub Actions".

Contacting support

If you need help with anything related to workflow configuration, such as syntax, GitHub-hosted runners, or building actions, look for an existing topic or start a new one in the GitHub Community Support's GitHub Actions category.

If you have feedback or feature requests for GitHub Actions, share those in the Feedback form for GitHub Actions.

Contact GitHub Support for any of the following, whether your use or intended use falls into the usage limit categories:

- If you believe your account has been incorrectly restricted
- If you encounter an unexpected error when executing one of your Actions, for example: a unique ID
- If you encounter a situation where existing behavior contradicts expected, but not always documented, behavior