

Building and testing Go

In this article

Introduction

Prerequisites

Using a Go starter workflow

Specifying a Go version

Installing dependencies

Building and testing your code

Packaging workflow data as artifacts

You can create a continuous integration (CI) workflow to build and test your Go project.

Introduction

This guide shows you how to build, test, and publish a Go package.

GitHub-hosted runners have a tools cache with preinstalled software, which includes the dependencies for Go. For a full list of up-to-date software and the preinstalled versions of Go, see "[Using GitHub-hosted runners](#)."

Prerequisites


You should already be familiar with YAML syntax and how it's used with GitHub Actions. For more information, see "[Workflow syntax for GitHub Actions](#)."

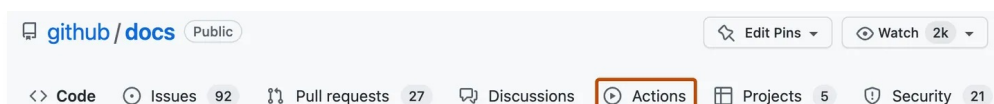
We recommend that you have a basic understanding of the Go language. For more information, see [Getting started with Go](#).

Using a Go starter workflow

To get started quickly, add a starter workflow to the `.github/workflows` directory of your repository.

GitHub provides a Go starter workflow that should work for most Go projects. The subsequent sections of this guide give examples of how you can customize this starter workflow.

- 1 On GitHub.com, navigate to the main page of the repository.
- 2 Under your repository name, click  **Actions**.

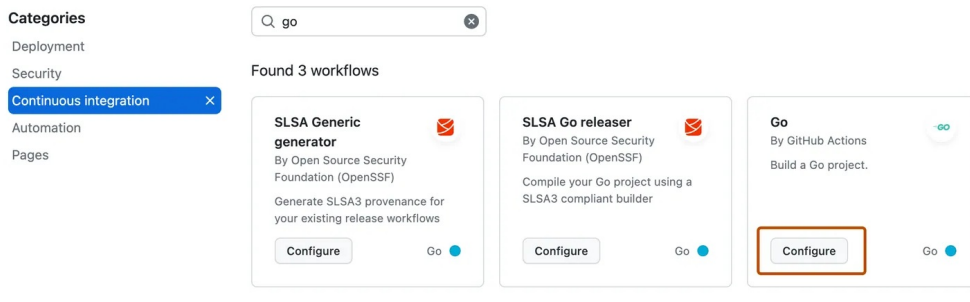


- 3 If you already have a workflow in your repository, click **New workflow**.
- 4 The "Choose a workflow" page shows a selection of recommended starter workflows. Search for "go".
- 5 Filter the selection of workflows by clicking **Continuous integration**.
- 6 On the "Go - by GitHub Actions" workflow, click **Configure**.

Choose a workflow

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →



- 7 Edit the workflow as required. For example, change the version of Go.
- 8 Click **Commit changes**.

The `go.yml` workflow file is added to the `.github/workflows` directory of your repository.

Specifying a Go version [↗](#)

The easiest way to specify a Go version is by using the `setup-go` action provided by GitHub. For more information see, the [setup-go action](#).

To use a preinstalled version of Go on a GitHub-hosted runner, pass the relevant version to the `go-version` property of the `setup-go` action. This action finds a specific version of Go from the tools cache on each runner, and adds the necessary binaries to `PATH`. These changes will persist for the remainder of the job.

The `setup-go` action is the recommended way of using Go with GitHub Actions, because it helps ensure consistent behavior across different runners and different versions of Go. If you are using a self-hosted runner, you must install Go and add it to `PATH`.

Using multiple versions of Go [↗](#)

YAML



```
name: Go

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      matrix:
        go-version: [ '1.19', '1.20', '1.21.x' ]
```

```

steps:
  - uses: actions/checkout@v4
  - name: Setup Go ${ matrix.go-version }
    uses: actions/setup-go@v4
    with:
      go-version: ${ matrix.go-version }
  # You can test your matrix by printing the current Go version
  - name: Display Go version
    run: go version

```

Using a specific Go version [↗](#)

You can configure your job to use a specific version of Go, such as `1.20.8`. Alternatively, you can use semantic version syntax to get the latest minor release. This example uses the latest patch release of Go 1.21:

YAML



```

- name: Setup Go 1.21.x
  uses: actions/setup-go@v4
  with:
    # Semantic version range syntax or exact version of Go
    go-version: '1.21.x'

```

Installing dependencies [↗](#)

You can use `go get` to install dependencies:

YAML



```

steps:
  - uses: actions/checkout@v4
  - name: Setup Go
    uses: actions/setup-go@v4
    with:
      go-version: '1.21.x'
  - name: Install dependencies
    run: |
      go get .
      go get example.com/octo-examplemodule
      go get example.com/octo-examplemodule@v1.3.4

```

Caching dependencies [↗](#)

You can cache and restore dependencies using the [setup-go action](#). By default, caching is enabled when using the `setup-go` action.

The `setup-go` action searches for the dependency file, `go.sum`, in the repository root and uses the hash of the dependency file as a part of the cache key.

You can use the `cache-dependency-path` parameter for cases when multiple dependency files are used, or when they are located in different subdirectories.

YAML



```

- name: Setup Go
  uses: actions/setup-go@v4
  with:

```

```
go-version: '1.17'
cache-dependency-path: subdir/go.sum
```

If you have a custom requirement or need finer controls for caching, you can use the [cache action](#). For more information, see "[Caching dependencies to speed up workflows](#)."

Building and testing your code [↗](#)

You can use the same commands that you use locally to build and test your code. This example workflow demonstrates how to use `go build` and `go test` in a job:

YAML



```
name: Go
on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4
      - name: Setup Go
        uses: actions/setup-go@v4
        with:
          go-version: '1.21.x'
      - name: Install dependencies
        run: go get .
      - name: Build
        run: go build -v ./...
      - name: Test with the Go CLI
        run: go test
```

Packaging workflow data as artifacts [↗](#)

After a workflow completes, you can upload the resulting artifacts for analysis. For example, you may need to save log files, core dumps, test results, or screenshots. The following example demonstrates how you can use the `upload-artifact` action to upload test results.

For more information, see "[Storing workflow data as artifacts](#)."

YAML



```
name: Upload Go test results

on: [push]

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      matrix:
        go-version: [ '1.19', '1.20', '1.21.x' ]

    steps:
      - uses: actions/checkout@v4
      - name: Setup Go
        uses: actions/setup-go@v4
        with:
          go-version: ${ matrix.go-version }
```

```
- name: Install dependencies
  run: go get .
- name: Test with Go
  run: go test -json > TestResults-${{ matrix.go-version }}.json
- name: Upload Go test results
  uses: actions/upload-artifact@v3
  with:
    name: Go-results-${{ matrix.go-version }}
    path: TestResults-${{ matrix.go-version }}.json
```

Legal

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Status](#) [Pricing](#) [Expert services](#) [Blog](#)