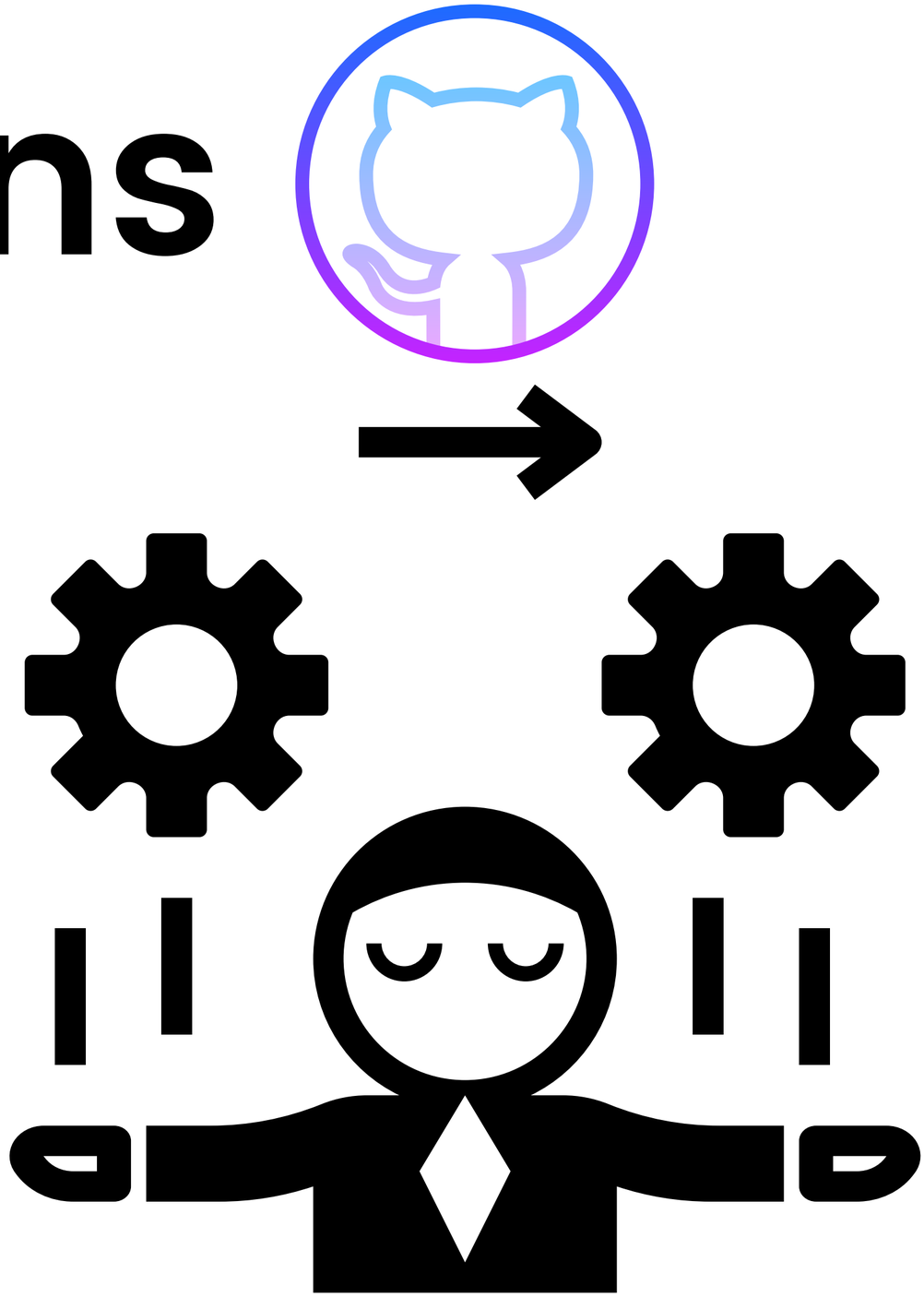
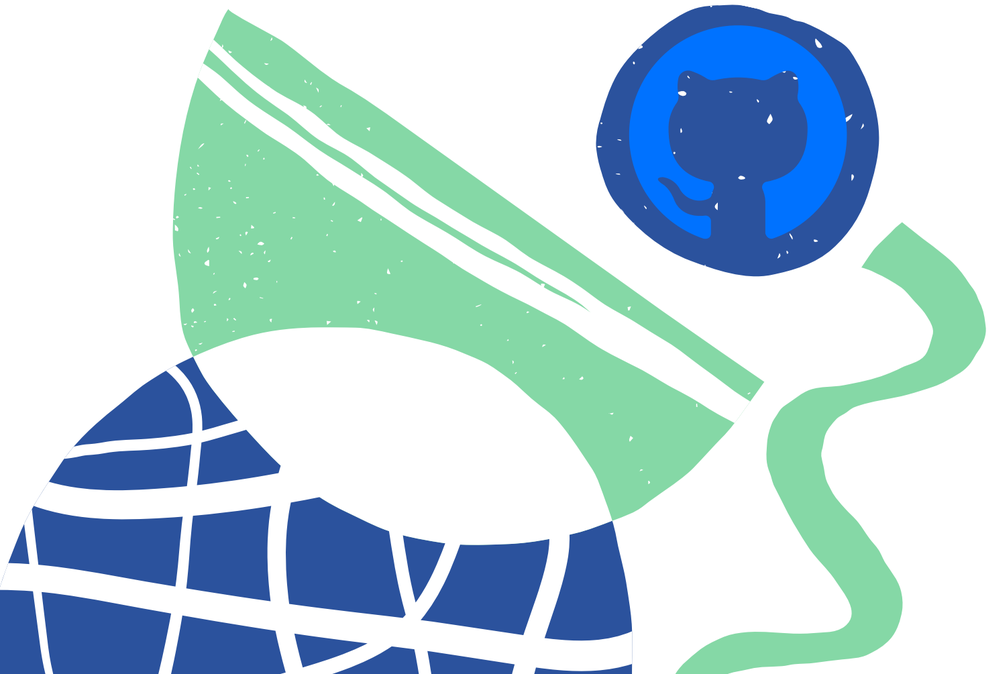
GitHub Actions

EP 1



Topics Covered



- What is GitHub Actions?
- GitHub Actions Components
- Pricing
- Community Actions
- Custom Runners Setup + Demo
- GitHub Actions Demo: NodeJS

WHAT IS GITHUB ACTIONS?

We can automate, customize, and execute our software development workflows right in our repository with GitHub Actions. We can discover, create, and share actions to perform any job we need, including CI/CD, and combine actions in a completely customized workflow.

GitHub Actions are an automated process that allows us to build, test, release and deploy any code project on GitHub, but we can also use it to automate any step of our workflow such as merging pull requests, assigning levels, triaging issues etc

In short:

GitHub Actions are a custom software developmet workflow automation tool

GITHUB ACTIONS COMPONENTS

1. WORKFLOW
2. ON EVENTS
3. JOBS
4. RUNNERS
5. STEPS
6. MATRIX
7. ENV
8. SECRETS
9. CACHE

GITHUB ACTIONS COMPONENTS

1. WORKFLOW

```yaml
name: My Workflow
on:
  push:
    branches:
      - 'releases/*'
      - '!releases/**-alpha'
env:
  message: 'conversation'
  my_token: ${{ secrets.GITHUB_TOKEN }}
jobs:
  my_build:
    runs-on: ubuntu-latest
    steps:
      - name: Checking out our code
        uses: actions/checkout@master
      - name: Say something
        run: |
          echo "A little less ${message}"
          echo "A little more action"
  my_job:
    needs: my_build
    container:
      image: node:10.16-jessie
      env:
        NODE_ENV: development
      ports:
        - 80
      volumes:
        - my_docker_volume:/volume_mount
      options: --cpus 1
    services:
      redis:
        image: redis
        ports:
          - 6379/tcp
```

Workflow is a configurable, automated process that we can use in our repository to build, test, package, release, or deploy your project. Workflows are made up of one or more “jobs” and can be triggered by GitHub events.

Workflow Files

Workflows can be created inside the .github/workflows directory by adding a .yml/.yaml workflow file. For example, add .github/workflows/build_and_deploy_action.yaml to your project.

Workflow Sytax

Github Actions files are written using YAML syntax and have either a .yml or .yaml file extension

Workflow Name

The name of your workflow is displayed on the Github actions page. If you omit this field, it is set to the file name.

GITHUB ACTIONS COMPONENTS

2. ON EVENTS

The on keyword defines the Github events that trigger the workflow. We can provide a single event, array, or events or a configuration map that schedules a workflow.

```yaml
on:
  push:
    branches:
      - main
```

```yaml
# Triggers the workflow on push or pull request events
on: [push, pull_request]
```

```yaml
on:
  # Trigger the wor
  # but only for th
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
  # Also trigger or
  page_build:
  release:
    types: # This c
      - created
```

Scheduled Events

schedule event allows us to trigger a workflow at a scheduled time

```yaml
on:
  schedule:
  - cron: '*/15 * * * *'
```

GITHUB ACTIONS COMPONENTS

3. JOBS

A workflow run is made up of one or more jobs. Jobs define the functionality that will be run in the workflow and run in parallel by default.

```yaml
jobs:
  my-job:
    name: My Job
    runs-on: ubuntu-latest
    steps:
    - name: Print a greeting
      run: |
        echo Hello there!
```

Job Collection

A workflow usually consists of one or more jobs identified by unique job id (e.g. my-job), jobs ruins in parallel unless queued with the "needs" attribute. Each job runs in a fresh instance of the virtual environment specified by "runs-on"

Needs

Identifies any job that needs to be completed successfully before this job runs.

Runs-on

Type of Virtual Machine to run the job on.
example: ubuntu-latest,windows-latest, macOS-latest, self-hosted

GITHUB ACTIONS COMPONENTS

4. RUNNERS

A runner is a machine with the Github Actions runner application installed. Then runner waits for available jobs it can then execute. After picking up a job they run the job's actions and report the progress and results back to Github. Runners can be hosted on Github or self-hosted on your own machines/servers.

GitHub Hosted Runers

A GitHub-hosted runner is a virtual machine hosted by GitHub with the GitHub Actions runner application installed. GitHub offers runners with Linux, Windows, and macOS operating systems.The virtual machine contains an environment of tools, packages, and settings available for GitHub Actions to use.

- Windows and Linux virtual machines: 2-core CPU,7 GB RAM, 14GB SSD Disk space
- MacOs Runner: 3-core CPU, 14GB RAM, 14GB SSD Disk Space

Self Hosted Runers

We can host our own runners and customize the environment used to run jobs in your GitHub Actions workflows. Self-hosted runners offer more control of hardware, operating system, and software tools than GitHub-hosted runners provide. Supported os as below:

Linux: RHEL Linux 7 +, CentOS 7 +, Oracle Linux 7+, Fedora 29+, Debain 9+, Ubutu 16.04 +, Linux Mint 18+ etc

Windows (64bit): Windows 7, 8.1, 10, Server 2012 R2, Server 2016 , Server 2019

Mac: macOS 10.13 (High Sierra) or later

GITHUB ACTIONS COMPONENTS

5. STEPS

A Job contains a sequence of tasks called **steps**.
Steps can run commands and actions.

```yaml
name: GitHub Actions Demo
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "🎉 The job was automatically triggered by a
      - run: echo "🐧 This job is now running on a ${{ runner.o
      - run: echo "🔎 The name of your branch is ${{ github.ref
      - name: Check out repository code
        uses: actions/checkout@v2
      - run: echo "💡 The ${{ github.repository }} repository h
      - run: echo "🖥️ The workflow is now ready to test your co
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
      - run: echo "🍏 This job's status is ${{ job.status }}."
```

```yaml
- name: Use Node.js
  uses: actions/setup-node@v1
  with:
    node-version: '12.x'
```

Step Name

Specify the label to display for this step on GitHub

Uses

Specify an action to run as part of step in our workflow job

With

A map of input parameters is defined by the action in it's actions.yml file

Run

Instead of running an existing action, a command-line program can be run using an operating system shell.
Multiple commands can be run in a single shell instance by using "|" (pipe) operator

GITHUB ACTIONS COMPONENTS

6. MATRIX STRATEGY

A build matrix allows you to test across multiple operating systems, platforms, and language versions at the same time. You can specify a build matrix using the strategy keyword and pass it to runs-on.

```
runs-on: ${{ matrix.os }}
strategy:
  matrix:
    os: [ubuntu-16.04, ubuntu-18.04]
    node: [6, 8, 10]
steps:
  - uses: actions/setup-node@v1
    with:
      node-version: ${{ matrix.node }}
```

GITHUB ACTIONS COMPONENTS

7. ENV

GitHub sets default environment variables for each GitHub Actions workflow run. We can also set custom environment variables in our workflow file using "**env**".

```
env:
  message: 'conversation'
  my_token: ${{ secrets.GITHUB_TOKEN }}
jobs:
  my_build:
    runs-on: ubuntu-latest
    steps:
      - name: Checking out our code
```

Env defines a map of environment variables that are available to all jobs and steps in the workflow. You can also set environment variables that are only available to a job or step.

Default Environment Variables

CI, GITHUB_WORKFLOW, GITHUB_RUN_ID, GITHUB_RUN_NUMBER, GITHUB_ACTION, GITHUB_ACTIONS, GITHUB_ACTOR, GITHUB_REPOSITORY, GITHUB_EVENT_NAME, GITHUB_EVENT_PATH, GITHUB_WORKSPACE, GITHUB_SHA, GITHUB_REF, GITHUB_HEAD_REF, GITHUB_BASE_REF, GITHUB_SERVER_URL, GITHUB_API_URL GITHUB_GRAPHQL_URL

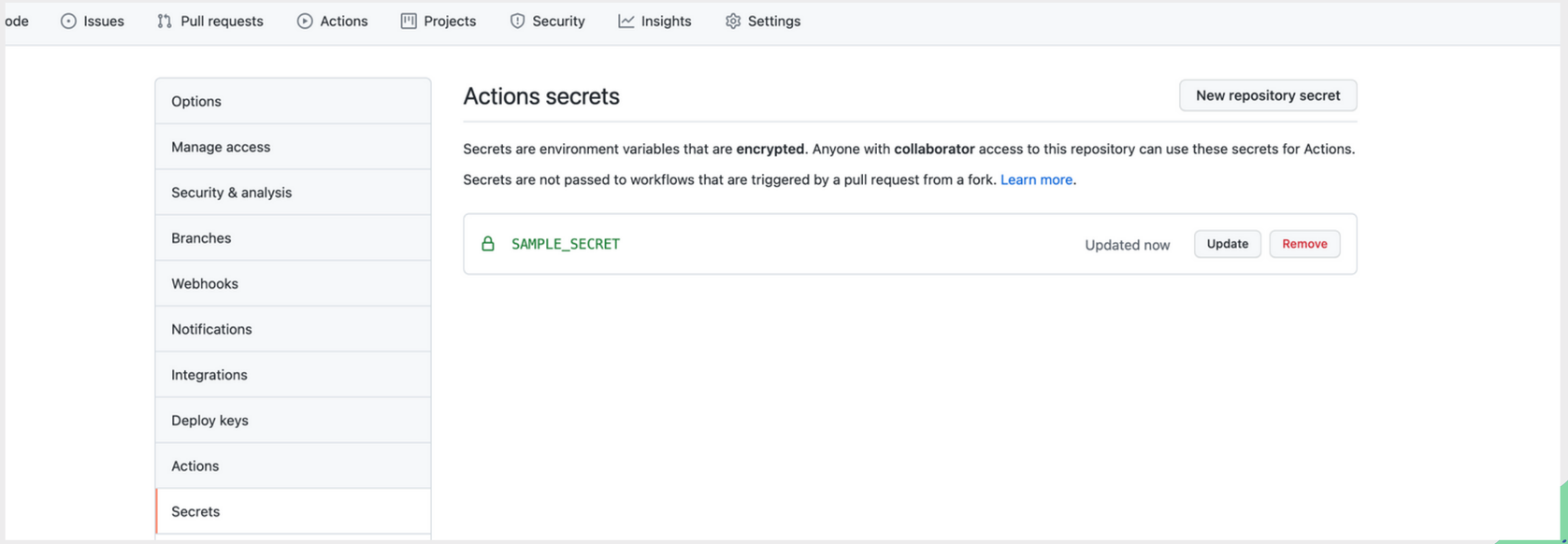
GITHUB ACTIONS COMPONENTS

8. SECRETS

```yaml
steps:
  - shell: bash
    env:
      SUPER_SECRET: ${{ secrets.SuperSecret }}
    run: |
      example-command "$SUPER_SECRET"
```

Secrets are encrypted environment variables that we create in an organization, repository, or repository environment. The secrets that we create are available to use in GitHub Actions workflows. Sensitive values should never be stored as plaintext in workflow files, but rather as secrets

Secrets get accessed in workflow like this: ${{ secrets.SECRET_KEY_HERE }}



Limit **We can store up to 1,000 organization secrets, 100 repository secrets, and 100 environment secrets.**

GITHUB ACTIONS COMPONENTS

9. CACHE

```
- uses: actions/checkout@v1
- name: Cache node modules
  uses: actions/cache@v1
  with:
    path: node_modules
    key: x-y-${{hashFiles('**/package-lock.json')}}
    restore-keys: |
      x-y-
      x-
```

The workflow runs often reuse the same output as the previous ones and can therefore be cached for an increase in performance. Every job run on Github-hosted runners starts in a clean virtual environment and doesn't use cache by default.

Caching is made possible by the Github cache action which will attempt to restore a cache based on the key you provide. If no match for the cache key is found it will create a new one after the successful completion of the job.

Input parameters:

- key (Required): The key identifies the cache and is created when saving the cache.
- path (Required): The file path of the directory you want to cache or restore.
- restore-key (Optional): An ordered list of alternative keys to use for finding the cache if no cache hit occurred for the key.

Output parameters:

- cache-hit: Boolean variable with success state of the cache action

PRICING

Included storage and minutes

Product	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

Minute multipliers

Operating system	Minute multiplier
Linux	1
macOS	10
Windows	2

Per-minute rates

Operating system	Per-minute rate (USD)
Linux	$0.008
macOS	$0.08
Windows	$0.016

GitHub Actions usage is free for both public repositories and self-hosted runners.

For private repositories, each GitHub account receives a certain amount of free minutes and storage, depending on the product used with the account.

Any usage beyond the included amounts is controlled by spending limits. 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.

If your account's usage surpasses these limits and you have set a spending limit above $0 USD, you will pay $0.25 USD per GB of storage per month and per-minute usage depending on the operating system used by the GitHub-hosted runner

COMMUNITY ACTIONS

Actions are standalone commands that are combined into steps to create a job. Actions are the smallest portable building block of a workflow. We can create our own actions or use actions created by the GitHub community, the actions created by GitHub community and ready to use, are called **Community Actions**.

We can explore all community actions here: https://github.com/marketplace?type=actions

Example Usage

```
- uses: actions/checkout@v2
```

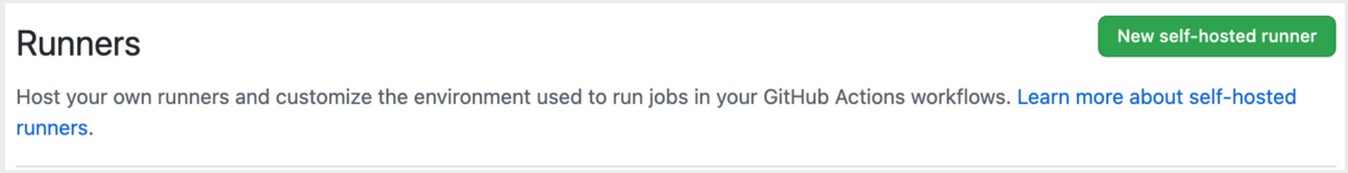
The `uses` keyword tells the job to retrieve `v2` of the community action named `actions/checkout@v2`. This is an action that checks out your repository and downloads it to the runner, allowing you to run actions against your code (such as testing tools). You must use the checkout action any time your 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'
```

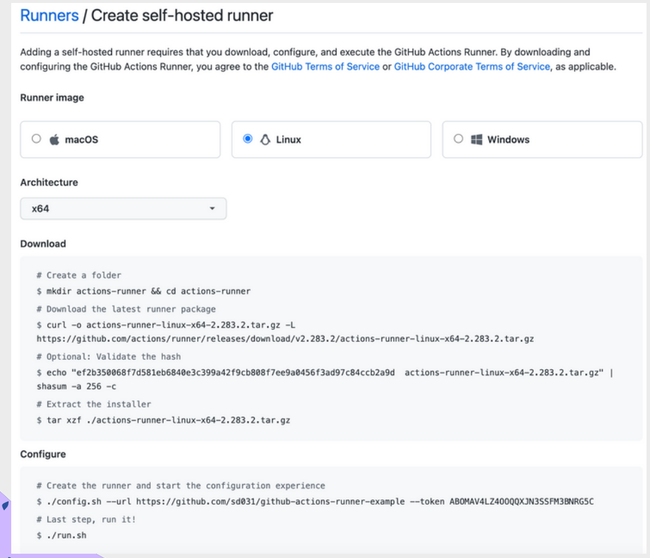
This step uses the `actions/setup-node@v2` action to install the specified version of the `node` software package on the runner, which gives you access to the `npm` command.

CUSTOM RUNNERS SETUP & DEMO

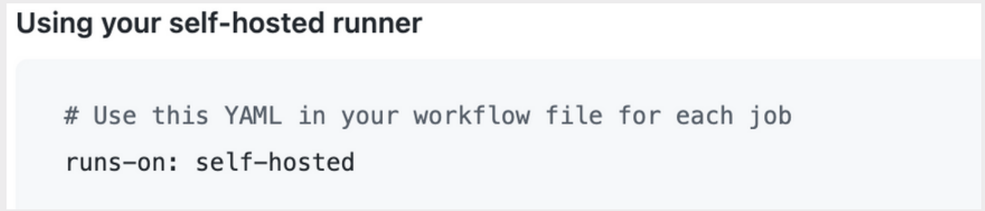
Repo → **Settings** → **Actions** → **Runners**



Available Choices Are MacOs, Linux, Windows. For Our Demo We will use LInux



After Selecting desired os type installation, simply follow the commands




```yaml
name: GitHub Actions Demo
on: [push]
jobs:
  hellow-world-demo-o-self-hosted:
    runs-on: self-hosted
    steps:
      - run: echo "Hellow world"
```

GITHUB ACTIONS DEMO: NODEJS

```yaml
name: Node.js GitHub Actions Demo

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2
      - name: Use Node.js
        uses: actions/setup-node@v2
        with:
          node-version: '14.x'
      - name: Install dependencies
        run: npm install
      - run: npm run build --if-present
      # - run: npm test

  publish:
    runs-on: ubuntu-latest
    needs: build
    steps:
      - name: Create release
        id: create_release
        uses: actions/create-release@v1
        env:
          GITHUB_TOKEN: ${{ secrets.github_token }} # This token is provided by Ac
        with:
          tag_name: ${{ github.run_number }}
          release_name: Release ${{ github.run_number }}
          body: New release for ${{ github.sha }}. Release notes can be found in t
          draft: false
          prerelease: false
      - name: Download artifact
        uses: actions/download-artifact@v2
        with:
          name: zipped-bundle
      - name: Upload release asset
        uses: actions/upload-release-asset@v1
        env:
          GITHUB_TOKEN: ${{ secrets.github_token }}
        with:
          upload_url: ${{ steps.create_release.outputs.upload_url }}
          asset_path: ./${{ github.sha }}.zip
          asset_name: source_code_with_libraries.zip
          asset_content_type: application/zip
```

In this demo, we will use GitHub actions to make Node.js build and make a GitHub release from it.

In this demo, we will be utilizing multiple GitHub Community Actions