Using GitHub Actions:

Actions, Inputs, Env, Secrets & Artifacts

Quick-Reference + Paste-and-Run Example

What you get. A concise reference for daily GitHub Actions work (using marketplace/local/container actions; passing inputs; working with `env`, secrets, and artifacts) *plus* a complete, runnable sample you can paste into any repo to verify your pipeline end-to-end.

1 Overview

- **Actions**: Reusable steps packaged as Docker containers, JavaScript, or Composite actions.
- **Triggers**: Defined under `on:` (e.g., `push`, `pull_request`, `workflow_dispatch`).
- **Runners**: `runs-on:` labels such as `ubuntu-latest`.
- **Context**: Read runtime data via expressions like `${{ github.actor }}` or `${{ env.MY_VAR }}`.

2 Using Actions

2.1 Marketplace action

```yaml
- uses: actions/checkout@v4
```

2.2 Action from another repo (pin versions)

```yaml
- uses: octocat/my-cool-action@v1        # tag
- uses: octocat/my-cool-action@9fceb02   # commit SHA (strongest pin)
- uses: octocat/my-cool-action@main      # branch (avoid in prod)
```

2.3 Action from a subdirectory in a repo

```yaml
- uses: octocat/my-cool-action/path/to/action@v2
```

2.4 Local action (same repository)

```yaml
- uses: ./.github/actions/lint
```

2.5 Container action (Docker image)

```yaml
- uses: docker://python:3.12
# or a registry image:
# - uses: docker://ghcr.io/owner/image:1.2.3
```

3 Inputs (with)

Pass inputs to an action with `with:`.

```yaml
- uses: actions/checkout@v4
  with:
    repository: apache/tomcat
    ref: main
    path: tomcat
```

4 Environment Variables (env)

Define variables at workflow, job, or step scope. Read in shells or expressions.

4.1 Define at different levels

```yaml
name: Example
on: [push]
env:
  PROJECT_NAME: my-app           # workflow-level default

jobs:
  build:
    runs-on: ubuntu-latest
    env:
      NODE_ENV: production       # job-level
    steps:
      - uses: actions/checkout@v4
      - name: Show env
        run: |
          echo "PROJECT_NAME=${PROJECT_NAME}"
          echo "NODE_ENV=${NODE_ENV}"
      - name: Step override
        env:
          NODE_ENV: test         # step-level
        run: echo "NODE_ENV=${NODE_ENV}"
```

4.2 Read syntax

- Bash: `$PROJECT_NAME`
- PowerShell: `$Env:PROJECT_NAME`
- YAML expression: `${{ env.PROJECT_NAME }}`

4.3 Useful built-in env/context

`GITHUB_EVENT_NAME`, `GITHUB_ACTOR`, `GITHUB_WORKSPACE`, `${{ github.repository }}`, `${{ github.sha }}`.

5 Secrets

- Store under **Settings → Secrets and variables → Actions**.
- Use in YAML via `${{ secrets.MY_SECRET }}`.
- Pass as `env` or as action inputs. Values are masked in logs.

5.1 Example

```yaml
- name: Login to AWS
  env:
    AWS_ACCESS_KEY_ID:     ${{ secrets.AWS_ACCESS_KEY_ID }}
    AWS_SECRET_ACCESS_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}
    AWS_DEFAULT_REGION:    us-east-1
  run: aws sts get-caller-identity
```

6 Artifacts

Upload outputs to share across jobs or to download from the run.

6.1 Upload

```yaml
- uses: actions/upload-artifact@v4
  with:
    name: build-outputs
    path: ./dist/**
```

6.2 Download

```yaml
- uses: actions/download-artifact@v4
  with:
    name: build-outputs
```

6.3 Notes

Retention defaults (often 90 days) and storage quotas apply. Names must match between upload and download.

7 Complete, Working Example (Paste & Run)

This example uses **Bun** to build a tiny executable and demonstrates: *actions*, *env*, *artifacts*, and *job dependencies*. Paste these files into your repo and push.

Project files

- `random-number-generator.js`
- `package.json`
- `.gitignore`
- `.github/workflows/create-artifacts.yml`

7.1 random-number-generator.js

```
#!/usr/bin/env bun
/**
 * Deterministic-ish RNG runner:
 *   - No arg       -> random numbers with time seed
 *   - String arg   -> hashed to seed
 *   - Number arg   -> used as seed
 * Appends a short report to ./report.txt
 */

function xfnv1a(str) {
  let h = 2166136261 >>> 0;
  for (let i = 0; i < str.length; i++) {
    h ^= str.charCodeAt(i);
    h = Math.imul(h, 16777619);
  }
  return h >>> 0;
}
function mulberry32(seed) {
  let a = seed >>> 0;
  return function () {
    a |= 0;
    a = (a + 0x6D2B79F5) | 0;
    let t = Math.imul(a ^ (a >>> 15), 1 | a);
    t ^= t + Math.imul(t ^ (t >>> 7), 61 | t);
    return ((t ^ (t >>> 14)) >>> 0) / 4294967296;
  };
}

const arg = process.argv[2];
let seed;
if (arg === undefined) {
  seed = (Date.now() ^ (Math.random() * 1e9)) >>> 0;
} else if (!Number.isNaN(Number(arg))) {
  seed = Number(arg) >>> 0;
} else {
  seed = xfnv1a(String(arg));
}
const rand = mulberry32(seed);

// Produce a tiny report (3 numbers, 0-100)
const nums = Array.from({ length: 3 }, () => Math.floor(rand() * 101));
const mode = arg === undefined ? "no-seed" : (Number.isNaN(Number(arg)) ?
↪  `seed:${arg}` : `seed:${Number(arg)}`);
const line = `[${new Date().toISOString()}] (${mode}) -> ${nums.join(", ")}\n`;

await Bun.write("report.txt", line, { append: true });
console.log(line.trim());
```

7.2 package.json

```
{
  "name": "create-artifacts",
  "version": "1.0.0",
  "private": true,
  "scripts": {
    "build": "bun build --compile ./random-number-generator.js --outfile
    ↪  random-number-generator-linux",
    "test": "bun run random-number-generator.js && bun run random-number-generator.js
    ↪  \"test-seed\" && bun run random-number-generator.js 1906"
  }
}
```

7.3 .github/workflows/create-artifacts.yml

```
name: Create Artifacts
on: [push, workflow_dispatch]

jobs:
  test:
    name: Test Code
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Setup Bun
        uses: oven-sh/setup-bun@v2
        with:
          bun-version: latest

      - name: Install dependencies
        run: bun install

      - name: Test with different seeds
        run: |
          echo "Testing with no seed..."
          bun run random-number-generator.js
          echo

          echo "Testing with string seed..."
          bun run random-number-generator.js "test-seed"
          echo

          echo "Testing with numeric seed..."
          bun run random-number-generator.js 1906
          echo "Generated report:"
          cat report.txt

      - name: Upload test report
        uses: actions/upload-artifact@v4
        with:
          name: test-report
```

```
        path: report.txt

build:
  name: Build Executables
  runs-on: ubuntu-latest
  needs: test
  steps:
    - name: Checkout
      uses: actions/checkout@v4

    - name: Setup Bun
      uses: oven-sh/setup-bun@v2
      with:
        bun-version: latest

    - name: Install dependencies
      run: bun install

    - name: Build Linux executable
      run: bun run build

    - name: List executables
      run: ls -la random-number-generator-*

    - name: Upload executables
      uses: actions/upload-artifact@v4
      with:
        name: random-number-generator-executables
        path: ./random-number-generator-*

test-linux:
  name: Test Linux Executable
  runs-on: ubuntu-latest
  needs: build
  steps:
    - name: Download executables
      uses: actions/download-artifact@v4
      with:
        name: random-number-generator-executables

    - name: Make binary executable
      run: chmod +x random-number-generator-linux

    - name: Run smoke tests
      run: |
        echo "Testing with no seed..."
        ./random-number-generator-linux
        echo

        echo "Testing with string seed..."
        ./random-number-generator-linux "test-seed"
        echo

        echo "Testing with numeric seed..."
```

```
        ./random-number-generator-linux 1906
        echo

    - name: Upload linux test report
      uses: actions/upload-artifact@v4
      with:
        name: linux-test-report
        path: report.txt
```

7.4 .gitignore

```
# Prevent project files from being tracked by git
*.bun-build
bun.lock
report.txt
package-lock.json
random-number-generator-*
```

Run it

1. Commit the four files and push to any branch.
2. Open **Actions** in GitHub; select the latest run.
3. Verify three artifacts exist:
   - `test-report`
   - `random-number-generator-executables`
   - `linux-test-report`

8 Common Pitfalls & Quick Fixes

- **Indentation / misplaced steps:** Ensure steps are under `jobs.<job>.steps`.
- **Unpinned actions:** Use a tag or SHA; avoid `@main` for reliability.
- **Secrets not visible:** Expected—GitHub masks values. Pass via `env` or `with`.
- **Artifacts missing:** Confirm matching `name` on upload/download and `needs:` between jobs.
- **Windows env syntax:** Use `$Env:NAME` in PowerShell vs `$NAME` in Bash.
- **Multiple OS binaries:** Start with Linux; add matrix builds later if you need macOS/Windows.

9 Build Notes (minted)

- Compile with `-shell-escape` so `minted` can call `pygmentize`.
- If your LaTeX toolchain caches styles, avoid `frozencache` unless you manage the cache files.
- The examples use only ASCII to avoid Unicode issues in some LaTeX setups.

Tip: Swap `actions/checkout@v4` for a pinned SHA in regulated environments, and pin `oven-sh/setup-bun@v2` to a tag/commit as well.