GitHub Actions & Workflows

Key Patterns & Drop-in Recipes

How to use this document

Copy the snippets into your repository under `.github/workflows/`. Each recipe is crafted to be production-ready and highlights a specific pattern (matrix builds, path filters, concurrency, reusable workflows, etc.). All code blocks use `minted`; compile with `-shell-escape`.

1 Workflow Anatomy (90-second refresher)

- **`name`**: Human-readable workflow label (optional).
- **`on`**: Triggers (e.g., `push`, `pull_request`, `workflow_dispatch`, `schedule`, `workflow_run`).
- **`jobs`**: One or more jobs; each runs on a runner (e.g., `ubuntu-latest`).
- **`steps`**: Inside jobs; `uses` an action or `run`s shell commands.
- **`needs`**: Job dependencies (enforces order).
- **`strategy.matrix`**: Expand a job across OS/language versions etc.
- **`env/secrets`**: Environment variables and secrets.
- **`permissions`**: Token scopes (*least privilege*).
- **`concurrency`**: Cancel in-flight runs for a branch/PR.

2 Recipe 1 – First Workflow: two jobs, multi-OS sanity check

Listing 1: Basic two-job workflow to verify runners, shells, and checkout

```yaml
name: First Workflow
on: [push]

jobs:
  linux:
    name: Linux env
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Print env (bash)
        run: env | sort

  windows:
    name: Windows env
    runs-on: windows-latest
    steps:
      - uses: actions/checkout@v4
      - name: Print env (PowerShell)
        run: Get-ChildItem Env: | Sort-Object Name
```

3 Recipe 2 – Matrix Build & Test (Go example)

Listing 2: Matrix build across Ubuntu/macOS/Windows with Go toolchain cache

```yaml
name: CI (Go matrix)
on:
  push:
    branches: [main]
  pull_request:

jobs:
  build:
    name: Build & Test (${{ matrix.os }})
    runs-on: ${{ matrix.os }}
    strategy:
      fail-fast: false
      matrix:
        os: [ubuntu-latest, macos-latest, windows-latest]
        include:
          - os: ubuntu-latest
            bin: main
            runbin: ./main
          - os: macos-latest
            bin: main
            runbin: ./main
          - os: windows-latest
            bin: main.exe
            runbin: .\main.exe
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-go@v5
        with:
          go-version: '1.21'
          cache: true
      - name: Build
        run: go build -o ${{ matrix.bin }} ./...
      - name: Unit tests
        run: go test ./... -count=1 -race -v
      - name: Run binary smoke test
        run: ${{ matrix.runbin }} --help
```

Optional `main.go` for smoke test

```go
package main

import (
  "flag"
  "fmt"
)

func main() {
  help := flag.Bool("help", false, "show help")
  flag.Parse()
  if *help {
    fmt.Println("demo app: flags: --help")
    return
  }
  fmt.Println("hello from CI")
}
```

4 Recipe 3 – Cross-compile & Publish Artifacts

Listing 3: Cross-compile after matrix (`needs: build`) and upload artifacts

```yaml
name: Build + Cross-Compile
on:
  push:
    branches: [main]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-go@v5
        with:
          go-version: '1.21'
          cache: true
      - run: go build -o main ./...

  cross:
    runs-on: ubuntu-latest
    needs: build
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-go@v5
        with:
          go-version: '1.21'
          cache: true
      - name: Cross-compile
        shell: bash
        run: |
          set -euo pipefail
          mkdir -p dist
          GOOS=linux   GOARCH=amd64 go build -o dist/app-linux-amd64 ./...
          GOOS=darwin  GOARCH=arm64 go build -o dist/app-macos-arm64 ./...
          GOOS=windows GOARCH=amd64 go build -o dist/app-windows-amd64.exe ./...
      - name: Upload artifacts
        uses: actions/upload-artifact@v4
        with:
          name: app-${{ github.sha }}
          path: dist/*
```

5 Recipe 4 – Branch, Tag, and Path Filters

Listing 4: Precise event filters for branches, tags, and paths

```yaml
name: Filtered CI
on:
  push:
    branches: ["main", "release/*"]
    tags: ["v*"]
    paths:
      - "cmd/**"
      - "internal/**"
      - "!docs/**"
  pull_request:
    branches: ["main"]
    paths-ignore:
      - "docs/**"
```

6 Recipe 5 – Concurrency (Cancel In-Progress)

Listing 5: Concurrency to avoid duplicate long-running jobs

```yaml
name: Lint & Test
on: [push, pull_request]

concurrency:
  group: ${{ github.workflow }}-${{ github.ref }}
  cancel-in-progress: true

jobs:
  ci:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: echo "do work"
```

7 Recipe 6 – Least-Privilege `permissions`

Listing 6: Lock down `GITHUB_TOKEN` scopes

```yaml
name: Secure Permissions
on: [pull_request]

permissions:
  contents: read
  pull-requests: write   # e.g., workflow needs to comment on PRs

jobs:
  annotate:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Add PR comment (example)
        uses: marocchino/sticky-pull-request-comment@v2
        with:
          message: "CI results are in!"
```

8 Recipe 7 – Cache Dependencies (generic)

Listing 7: Generic caching with a robust key and restore keys

```yaml
name: Node CI
on: [push, pull_request]

jobs:
  ci:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Use Node
        uses: actions/setup-node@v4
        with:
          node-version: '20.x'

      - name: Cache npm
        uses: actions/cache@v4
        with:
          path: ~/.npm
          key: npm-${{ runner.os }}-${{ hashFiles('**/package-lock.json') }}
          restore-keys: |
            npm-${{ runner.os }}-

      - run: npm ci
      - run: npm test -- --ci
```

9 Recipe 8 – Artifacts and Build Outputs Between Jobs

Listing 8: Upload in one job, download in a dependent job

```yaml
name: Build → E2E
on: [pull_request]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: npm ci && npm run build
      - uses: actions/upload-artifact@v4
        with:
          name: web-dist
          path: dist/

  e2e:
    runs-on: ubuntu-latest
    needs: build
    steps:
      - uses: actions/download-artifact@v4
        with:
          name: web-dist
          path: dist/
      - run: npx playwright install --with-deps
      - run: npx playwright test
```

10 Recipe 9 – Reusable Workflows (`workflow_call`)

1) Reusable workflow (in same repo) Save as `.github/workflows/reusable-ci.yml`.

```yaml
name: Reusable CI
on:
  workflow_call:
    inputs:
      node:
        required: true
        type: string

jobs:
  ci:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: ${{ inputs.node }}
      - run: npm ci && npm test
```

2) Caller workflow

```yaml
name: App CI
on: [push, pull_request]

jobs:
  call-shared:
    uses: ./.github/workflows/reusable-ci.yml
    with:
      node: "20"
```

11 Recipe 10 – Scheduled and Manual Triggers

Listing 9: Nightly maintenance and manual `workflow_dispatch`

```yaml
name: Maintenance
on:
  schedule:
    - cron: "17 3 * * *"   # 03:17 UTC daily
  workflow_dispatch:
    inputs:
      task:
        description: "Choose a task"
        required: true
        type: choice
        options: [vacuum-db, refresh-caches]

jobs:
  run-task:
    runs-on: ubuntu-latest
    steps:
      - run: echo "Running ${{ github.event.inputs.task }} ..."
```

12 Recipe 11 – Safer PRs from Forks (`pull_request_target`)

Listing 10: Guarded use of `pull_request_target`

```yaml
name: PR Labeler (safe)
on:
  pull_request_target:
    types: [opened, synchronize, reopened]

permissions:
  contents: read
  pull-requests: write

jobs:
  label:
    runs-on: ubuntu-latest
    steps:
      - name: Label by title
        uses: actions-ecosystem/action-add-labels@v1
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          labels: |
            needs-triage
```

13 Recipe 12 – Monorepo Path Filters (per package)

Listing 11: Monorepo CI per package directory

```yaml
name: Package A CI
on:
  push:
    branches: [main]
    paths:
      - "packages/pkg-a/**"
  pull_request:
    paths:
      - "packages/pkg-a/**"

jobs:
  test-a:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: cd packages/pkg-a && npm ci && npm test
```

14 Recipe 13 – Conditional Steps/Jobs (`if:`)

Listing 12: Conditionally skip when only docs changed

```yaml
name: Conditional Work
on: [push, pull_request]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Detect docs-only
        id: changed
        run: |
          git fetch --depth=2 origin ${{ github.base_ref || 'HEAD~1' }}
          CHANGED=$(git diff --name-only HEAD^ HEAD | tr -d '\r')
          echo "changed=$CHANGED" >> $GITHUB_OUTPUT

      - name: Skip if docs-only
        if: ${{ startsWith(steps.changed.outputs.changed, 'docs/') }}
        run: echo "Docs-only change; skipping build."

      - name: Build
        if: ${{ !startsWith(steps.changed.outputs.changed, 'docs/') }}
        run: echo "Do the real build..."
```

15 Recipe 14 – Self-Hosted Runners (labels and timeouts)

Listing 13: Self-hosted with custom labels and timeout

```yaml
name: Self-Hosted CI
on: [push]

jobs:
  heavy:
    runs-on: [self-hosted, linux, x64, gpu]
    timeout-minutes: 60
    steps:
      - uses: actions/checkout@v4
      - run: nvidia-smi || true
```

16 Common Gotchas (quick checklist)

- **`needs`** references *job IDs* (the YAML keys), not `name`.
- Default shells differ: bash on Linux/macOS; PowerShell on Windows.
- Put per-event branch/path filters under each event key inside `on:`.
- Use `concurrency` to cancel old runs on the same branch/PR.
- Lock down `permissions`; grant elevated scopes only where needed.
- For forked PRs, prefer `pull_request`; use `pull_request_target` only for trusted, no-checkout workflows.
- Artifact size and log size are limited—upload just what you need.

Appendix – Local commands you may find handy

```
# Validate YAML locally (requires yq or yamllint)
yamllint .github/workflows

# Render your matrix to see what would run (pseudo; GitHub handles expansion)
# Tip: keep matrix small and explicit when in doubt.
```

PowerShell snippet (Windows runner tips)

```
# Show environment variables
Get-ChildItem Env: | Sort-Object Name

# Fail a step explicitly
Write-Error "Stopping this step on purpose."
```