Advanced GitHub Actions — Very Detailed Cheat Sheet

Hands-on, copy-pasteable examples

Contents

1 Events & Triggers (on:)

1.1 Common Triggers

```yaml
name: Triggers and Filters
on:
  push:
    branches: [ "main", "release/**" ]
    tags: [ "v*", "release-*" ]
    paths:
      - "src/**"
      - "!docs/**"           # exclude
  pull_request:
    types: [opened, synchronize, reopened, ready_for_review]
    branches: [ "main" ]
    paths:
      - "src/**"
      - "package.json"
  workflow_dispatch:
    inputs:
      env:
        description: "Environment to deploy"
        type: choice
        options: [dev, staging, prod]
        required: true
      ref:
        description: "Ref (branch/sha) to run"
        default: "main"
        required: false
  schedule:
    - cron: "15 6 * * 1-5"   # 06:15 UTC Mon-Fri
  workflow_run:
    workflows: ["Build"]
    types: [completed]
```

Notes

- **Paths filters** are evaluated per trigger: use to limit runs in monorepos.
- `pull_request_target` runs in the base repo context (caution with secrets).
- `workflow_dispatch.inputs` support `type` (`string|choice|boolean|environment`).

2 Jobs, Runners, Strategy, Defaults

```yaml
jobs:
  build:
    name: Build (${{ matrix.os }} / Node ${{ matrix.node }})
    runs-on: ${{ matrix.os }}
    timeout-minutes: 30
    continue-on-error: false  # or {matrix: allow-failure == 'true'}
    strategy:
      fail-fast: false
      matrix:
        os: [ubuntu-latest, windows-latest, macos-latest]
        node: [18, 20]
        include:
          - os: ubuntu-latest
            node: 20
            allow-failure: 'true'
        exclude:
          - os: windows-latest
            node: 18
    defaults:
      run:
        shell: bash
        working-directory: .
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: ${{ matrix.node }}
      - run: npm ci
      - run: npm test -- --ci
```

Key Points

- `runs-on`: hosted images or self-hosted labels.
- `strategy.fail-fast=false`: do not cancel siblings on first fail.
- `timeout-minutes`: cancel long-running jobs automatically.

3 Contexts Overview

- `github`, `env`, `vars`, `secrets`, `runner`, `job`, `steps`, `matrix`, `needs`, `inputs`, `strategy`.

```yaml
- name: Dump contexts (redact secrets yourself!)
  run: |
    echo 'actor: ${{ github.actor }}'
    echo 'ref_name: ${{ github.ref_name }}'
    echo 'event: ${{ github.event_name }}'
    echo 'runner: ${{ toJSON(runner) }}'
    echo 'job: ${{ toJSON(job) }}'
    echo 'matrix: ${{ toJSON(matrix) }}'
```

4 Expressions & Built-in Functions

4.1 Operators & Types

- Types: `boolean`, `null`, `number`, `string`.
- Operators: `==`, `!=`, `<`, `<=`, `>`, `>=`, `&&`, `||`, `!`.

4.2 Functions (with examples)

```yaml
if: contains(github.ref, 'refs/heads/release/') && !cancelled()

# String helpers
if: startsWith(github.ref, 'refs/tags/') || endsWith(github.ref_name, '-rc')

# JSON coercion
env:
  FLAG: 'false'
if: ${{ fromJSON(env.FLAG) }}   # false -> step skipped

# Formatting & joining
env:
  MSG: ${{ format('Deploy {0} to {1}', github.sha, inputs.env) }}
  LIST: ${{ join(fromJSON('["a","b","c"]'), ',') }}

# Hashing files for cache keys
key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements.txt') }}
```

4.3 Status Checks in `if`

```yaml
if: success()     # default for steps/jobs
if: failure()     # previous step failed
if: always()      # run regardless
if: cancelled()   # run if run cancelled
```

5 Variables, Secrets, and `vars`

5.1 Scopes & Precedence

- Declare at **workflow**, **job**, or **step** via `env:`.
- `vars.X` are configuration variables (repo/org/environment).
- `secrets.X` are masked, not echoed by default.

```yaml
env:
  WF_VAR: "workflow-level"

jobs:
  demo:
    env:
      JOB_VAR: "job-level"
    steps:
      - name: Step vars
        env:
          STEP_VAR: "step-level"
        run: |
          echo "WF_VAR=${WF_VAR}"
          echo "JOB_VAR=${JOB_VAR}"
          echo "STEP_VAR=${STEP_VAR}"
          echo "CONF VAR: ${{ vars.MY_CONFIG }}"
          echo "SECRET: ${{ secrets.MY_SECRET }}"
```

Tip

- Booleans in `env` are strings; coerce with `fromJSON`.

6 Permissions & `GITHUB_TOKEN`

```yaml
permissions:              # default -> fine-grained least-privilege recommended
  contents: read
  pull-requests: write
  id-token: write         # required for OIDC federation
  actions: read
  checks: write

jobs:
  secure:
    permissions:
      contents: read
      packages: write
```

Notes

- Prefer explicit `permissions` at workflow/job scope.
- Exercise care with `pull_request_target`.

7 Job/Step Outputs, Artifacts, and Cache

7.1 Step & Job Outputs

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    outputs:
      image_sha: ${{ steps.meta.outputs.sha }}
    steps:
      - id: meta
        run: echo "sha=${GITHUB_SHA}" >> "$GITHUB_OUTPUT"

  deploy:
    needs: [build]
    runs-on: ubuntu-latest
    steps:
      - run: echo "Got image sha = ${{ needs.build.outputs.image_sha }}"
```

7.2 Artifacts

```yaml
- name: Upload build
  uses: actions/upload-artifact@v4
  with:
    name: web-dist
    path: dist/

- name: Download build
  uses: actions/download-artifact@v4
  with:
    name: web-dist
    path: ./dist
```

7.3 Cache

```yaml
- uses: actions/cache@v4
  with:
    path: ~/.cache/pip
    key: ${{ runner.os }}-pip-${{ hashFiles('**/requirements.txt') }}
    restore-keys: |
      ${{ runner.os }}-pip-
```

8 Concurrency & Cancel In-Progress

```yaml
concurrency:
  group: ${{ github.workflow }}-${{ github.ref_name }}
  cancel-in-progress: true
```

Pattern For PRs, use `github.head_ref`; for branches use `ref_name`.

9 Environments & Required Reviews

```yaml
jobs:
  deploy:
    runs-on: ubuntu-latest
    environment:
      name: staging
      url: https://staging.example.com
    steps:
      - run: ./deploy.sh
```

Note Environments can enforce approvals and restrict secrets.

10 Reusable Workflows (`workflow_call`)

10.1 Called Workflow

```yaml
# .github/workflows/reusable.yml
name: Reusable
on:
  workflow_call:
    inputs:
      service:
        type: string
        required: true
    secrets:
      DEPLOY_KEY:
        required: true
    outputs:
      version:
        description: "Output version"
        value: ${{ jobs.publish.outputs.version }}

jobs:
  publish:
    runs-on: ubuntu-latest
    outputs:
      version: ${{ steps.tag.outputs.version }}
    steps:
      - id: tag
        run: echo "version=1.2.3" >> "$GITHUB_OUTPUT"
```

10.2 Caller Workflow

```yaml
# .github/workflows/caller.yml
name: Caller
on: workflow_dispatch
jobs:
  use-reusable:
    uses: ./.github/workflows/reusable.yml
    with:
      service: api
    secrets:
      DEPLOY_KEY: ${{ secrets.DEPLOY_KEY }}
```

11 Composite Actions (local `action.yml`)

11.1 Metadata

```yaml
# .github/actions/greet/action.yml
name: "Greet"
description: "Say hello"
inputs:
  who:
    description: "Name to greet"
    required: true
runs:
  using: "composite"
  steps:
    - shell: bash
      run: echo "Hello, ${{ inputs.who }}!"
```

11.2 Use the Composite Action

```yaml
- uses: ./.github/actions/greet
  with:
    who: "Jordan"
```

12 Containers & Service Containers

12.1 Job Container

```yaml
jobs:
  dockerized:
    runs-on: ubuntu-latest
    container:
      image: node:20-bullseye
      options: --cpus 2
    steps:
      - uses: actions/checkout@v4
      - run: node -v
```

12.2 Service Container (Postgres)

```yaml
jobs:
  integ:
    runs-on: ubuntu-latest
    services:
      pg:
        image: postgres:16
        env:
          POSTGRES_PASSWORD: postgres
        ports: ["5432:5432"]
        options: >-
          --health-cmd="pg_isready -U postgres"
          --health-interval=10s --health-timeout=5s --health-retries=5
    steps:
      - run: |
          psql "postgresql://postgres:postgres@localhost:5432/postgres" -c "SELECT 1"
```

13 OpenID Connect (OIDC) to Cloud (Example: AWS)

```yaml
permissions:
  id-token: write
  contents: read

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: aws-actions/configure-aws-credentials@v4
        with:
          aws-region: us-east-1
          role-to-assume: arn:aws:iam::123456789012:role/GitHubOIDCDeployRole
      - run: aws sts get-caller-identity
```

Note Configure an AWS IAM role with a trust policy allowing GitHub OIDC provider and repository conditions.

14 Logging & Debugging Tricks

```yaml
- name: Group logs
  run: |
    echo "::group::Build"
    npm ci && npm run build
    echo "::endgroup::"

- name: Enable step debug (set once in repo secrets)
  run: echo "ACTIONS_STEP_DEBUG=true" >> "$GITHUB_ENV"

- name: Mask a value (extra safety)
  run: echo "::add-mask::${SECRET_VALUE}"

- name: Export env for later steps
  run: echo "FOO=bar" >> "$GITHUB_ENV"

- name: Prepend to PATH
  run: echo "$HOME/bin" >> "$GITHUB_PATH"
```

15 Monorepo Patterns (Paths, Concurrency)

15.1 Paths-based Workflows

```yaml
on:
  push:
    paths:
      - "services/web/**"
      - ".github/workflows/web-*.yml"
```

15.2 Per-branch Concurrency

```yaml
concurrency:
  group: ${{ github.workflow }}-${{ github.head_ref || github.ref_name }}-web
  cancel-in-progress: true
```

16 Production-Ready Recipes

16.1 Node Build & Test (Matrix, Cache)

```yaml
name: CI
on:
  pull_request:
  push:
    branches: [ "main" ]
jobs:
  node:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node: [18, 20]
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-node@v4
        with:
          node-version: ${{ matrix.node }}
          cache: npm
      - run: npm ci
      - run: npm run lint
      - run: npm test -- --ci
```

16.2 Python Build & Test with Pip Cache

```yaml
jobs:
  python:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - uses: actions/setup-python@v5
        with:
          python-version: "3.11"
          cache: "pip"
      - run: python -m pip install -U pip
      - run: pip install -r requirements.txt
      - run: pytest -q
```

16.3 Docker Build & Push to GHCR

```yaml
jobs:
  docker:
    runs-on: ubuntu-latest
    permissions:
      contents: read
      packages: write
    steps:
      - uses: actions/checkout@v4
      - name: Log in to GHCR
        run: echo "${{ secrets.GITHUB_TOKEN }}" | docker login ghcr.io -u ${{ github.actor }}
        ↪ --password-stdin
      - name: Build
        run: |
          IMAGE=ghcr.io/${{ github.repository_owner }}/myapp:${{ github.sha }}
          docker build -t "$IMAGE" .
          echo "image=$IMAGE" >> "$GITHUB_OUTPUT"
        id: build
      - name: Push
        run: docker push "${{ steps.build.outputs.image }}"
```

16.4 Deploy with Environment Protection

```yaml
jobs:
  deploy:
    runs-on: ubuntu-latest
    environment:
      name: prod
      url: https://app.example.com
    steps:
      - run: ./deploy-prod.sh
```

16.5 PR Label Gate (Example)

```yaml
jobs:
  gated:
    if: contains(join(github.event.pull_request.labels.*.name, ','), 'ready')
    runs-on: ubuntu-latest
    steps:
      - run: echo "Label 'ready' is present; proceeding."
```

16.6 Required Check/Manual Retry Pattern

```yaml
jobs:
  flaky-test:
    runs-on: ubuntu-latest
    steps:
      - name: Run flaky test (retry)
        uses: nick-invision/retry@v3
        with:
          timeout_minutes: 10
          max_attempts: 3
          command: npm run test:flaky
```

17 Security Best Practices (Quick Hits)

- Use `permissions` least-privilege explicitly.
- Prefer OIDC over long-lived cloud keys.
- Avoid `pull_request_target` unless you know why; sanitize inputs.
- Never print secrets; rely on masking and `::add-mask::` for derived values.
- Pin third-party actions by version or commit SHA.

18 Appendix: Handy Snippets

18.1 Only on Tags

```
if: startsWith(github.ref, 'refs/tags/')
```

18.2 Mark Step as Skipped by Condition

```
- name: Skip if docs-only change
  if: "!contains(join(github.event.commits.*.modified, ','), 'src/')"
  run: echo "Docs-only change."
```

18.3 Read JSON and Access Fields

```
- name: Parse JSON
  id: parse
  run: |
    echo 'json={"a":1,"b":2}' >> "$GITHUB_OUTPUT"
- name: Use JSON
  run: |
    echo '${{ fromJSON(steps.parse.outputs.json).a }}'
```

This cheat sheet focuses on portability and clean compilation under `minted`. Replace action versions with pinned SHAs in production and validate runner images as your platform evolves.