name

• name at the **workflow level** specifies the overall workflow name.

• name at the **job level** defines the name of a job in the workflow.

• run-name provides a custom name for each workflow run and can include dynamic context variables.

run-name

• **Customizes the name** of each run in the GitHub Actions interface.

• Helps identify runs more easily, especially in workflows that trigger frequently.

• Supports dynamic context variables to make the name more informative.

What happens without run-name:

• The workflow run will still execute as usual.

• The name of the run will be more generic, based on the filename and run number.

• It may be harder to distinguish between different runs if you have multiple workflows triggered frequently, as there won't be a descriptive label to help identify them.

When to use run-name:

You would typically use run-name when you want to make your workflow runs more **descriptive** and easier to identify, especially when using dynamic context variables like the branch name or commit SHA. This can be helpful in large projects with multiple workflows.

Triggers:

1. **push** - Triggered on pushes to branches/tags.

2. **pull_request** - Triggered on pull request events.

3. **workflow_dispatch** - Manual trigger via UI.

4. **schedule** - Cron-like scheduling for periodic runs.

5. **release** - Triggered on release events (creation, publishing).

6. **issues** - Triggered on issue activity (opened, closed, etc.).

7. **fork** - Triggered when a repository is forked.

8. **delete** - Triggered when a branch or tag is deleted.

9. **workflow_run** - Triggered when another workflow completes.

10. **push (with path filters)** - Triggered on push, but filters based on file paths.

JOBS

• **jobs** are the building blocks of a GitHub Actions workflow.

• Each job runs in its own environment, isolated from others.

• Jobs can run in **parallel** or **sequentially** based on dependencies (using needs).

• **runs-on** specifies the type of runner (e.g., ubuntu-latest, windows-latest).

• Jobs consist of a series of **steps** which define specific actions or commands to be executed.

• You can use **matrices** to run the same job across different configurations (e.g., different Node.js versions).

Jobs are crucial in breaking down workflows into smaller, manageable tasks that can be executed independently or in a specific order

```
name: CI Workflow
```

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

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Install dependencies
        run: npm install
      - name: Run build
        run: npm run build

  test:
    runs-on: ubuntu-latest
    needs: build  # This job depends on the "build" job
    steps:
      - name: Checkout code
        uses: actions/checkout@v2
      - name: Run tests
        run: npm test
```

actions

• uses is for **reusing existing actions** from GitHub Marketplace or other repositories.

• It saves time by avoiding the need to write repetitive scripts.

• You can specify versions using tags (@v4), commits (@commit_sha), or branches (@main).

• It can be combined with run for more flexibility.

Conditional Step Execution

```
jobs:
  check-env:
    runs-on: ubuntu-latest
    steps:
      - name: Run for production
        if: github.ref == 'refs/heads/main'
        run: echo "Deploying to Production"

      - name: Run for staging
        if: github.ref == 'refs/heads/staging'
        run: echo "Deploying to Staging"

      - name: Run for any other branch (Fallback)
        if: github.ref != 'refs/heads/main' && github.ref != 'refs/heads/staging'
        run: echo "Deploying to Development"
```

sharing

✅ **Reusable Workflows** → Like shared pipelines (use workflow_call)

✅ **Composite Actions** → Like shared functions (define in .github/actions/)

✅ **Shared GitHub Actions** → Public or private actions in separate repos