Complete Guide: Activating `workflow_run` in GitHub Actions

What is `workflow_run`?

`workflow_run` is a GitHub Actions event trigger that starts one workflow automatically after another workflow finishes.

It is mainly used for:

- CI/CD pipelines
- Running deployment only after tests pass
- Security scanning after build
- Multi-stage automation
- Triggering advanced workflows

Example:

1. Workflow A → Runs tests
2. Workflow B → Starts automatically after Workflow A completes

---

Basic Architecture

```
Push Code
   ↓
Workflow A (Build/Test)
   ↓
workflow_run Trigger
   ↓
Workflow B (Deploy/Notify/etc.)
```

Step-by-Step Setup

Step 1: Create the First Workflow

This workflow will run first.

Path:

```
.github/workflows/build.yml
```

Example:

```yaml
name: Build Workflow

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Run build
        run: echo "Building project..."
```

Step 2: Create the `workflow_run` Workflow

Path:

```
.github/workflows/deploy.yml
```

Example:

```yaml
name: Deploy Workflow

on:
  workflow_run:
    workflows: ["Build Workflow"]
    types:
      - completed
```

```yaml
jobs:
  deploy:
    runs-on: ubuntu-latest

    if: ${{ github.event.workflow_run.conclusion == 'success' }}

    steps:
      - name: Deploy application
        run: echo "Deploying project..."
```

Important Explanation

workflows

```yaml
workflows: ["Build Workflow"]
```

This must match the exact:

```yaml
name:
```

of the first workflow.

Example:

```yaml
name: Build Workflow
```

Even one spelling mistake breaks the trigger.

---

types

```yaml
types:
  - completed
```

Available types:

Type	Meaning
completed	Runs after workflow finishes

Type	Meaning
requested	Runs when workflow is requested
in_progress	Runs while workflow is running

Usually `completed` is used.

Checking Success or Failure

```
if: ${{ github.event.workflow_run.conclusion == 'success' }}
```

Possible values:

Value	Meaning
success	Workflow passed
failure	Workflow failed
cancelled	Workflow cancelled
skipped	Workflow skipped

Full Real Example

CI Workflow

```
name: CI Pipeline

on:
  push:
    branches:
      - main

jobs:
  test:
    runs-on: ubuntu-latest

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

```
      - name: Install Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 20

      - name: Install dependencies
        run: npm install

      - name: Run tests
        run: npm test
```

Deployment Workflow

```
name: Production Deploy

on:
  workflow_run:
    workflows: ["CI Pipeline"]
    branches: [main]
    types:
      - completed

jobs:
  deploy:
    runs-on: ubuntu-latest

    if: ${{ github.event.workflow_run.conclusion == 'success' }}

    steps:
      - name: Deploy to server
        run: echo "Deploying to production..."
```

How to Activate `workflow_run`

Many people create the YAML correctly but it still does not trigger.

These are the required conditions.

Requirement 1: Workflow Must Exist on Default Branch

The workflow file must exist in:

```
main
```

or your repository's default branch.

If it only exists in another branch, GitHub may not activate it.

Requirement 2: First Workflow Must Run Successfully

The first workflow must actually execute.

Example:

```
Push → Build Workflow runs → workflow_run activates
```

If Workflow A never runs:

- Workflow B never triggers

Requirement 3: Correct Workflow Name

This is the most common mistake.

Wrong:

```
workflows: ["build workflow"]
```

Correct:

```
workflows: ["Build Workflow"]
```

GitHub is case-sensitive.

Requirement 4: Actions Must Be Enabled

Go to:

```
Repository → Settings → Actions → General
```

Make sure:

- Actions are enabled
- Workflows have permission to run

Requirement 5: Repository Permissions

Go to:

```
Settings → Actions → General
```

Enable:

```
Read and write permissions
```

for workflow permissions if needed.

Viewing Workflow Logs

Go to:

```
Repository → Actions
```

You can inspect:

- Trigger history
- Errors
- Logs
- Execution steps

Common Problems and Fixes

Problem 1: Workflow Not Triggering

Causes

- Wrong workflow name
- Workflow file not in main branch
- YAML syntax error
- First workflow never executed

Fix

Check:

```
workflows: ["Exact Workflow Name"]
```

Problem 2: Workflow Runs Twice

This happens when:

- push trigger exists
- workflow_run also exists

Example:

```
on:
  push:
  workflow_run:
```

This can create duplicate runs.

Problem 3: Infinite Workflow Loop

Dangerous configuration:

```
Workflow A triggers Workflow B
Workflow B triggers Workflow A
```

This creates an infinite loop.

Avoid circular triggers.

Problem 4: workflow_run Does Not Work from Forks

GitHub blocks some workflow behavior for security reasons in forked repositories.

Especially:

- secrets
- write permissions
- deployment access

Using Secrets with workflow_run

Example:

```
env:
  API_KEY: ${{ secrets.API_KEY }}
```

Store secrets here:

```
Settings → Secrets and variables → Actions
```

Advanced Example: Trigger Only on Success

```
if: |
  github.event.workflow_run.conclusion == 'success'
```

Advanced Example: Trigger on Failure

```
if: |
  github.event.workflow_run.conclusion == 'failure'
```

Useful for:

- alerts
- Discord notifications
- Slack notifications
- automatic rollback

Accessing Workflow Run Data

You can access details from the previous workflow.

Examples:

```
${{ github.event.workflow_run.id }}
${{ github.event.workflow_run.name }}
${{ github.event.workflow_run.conclusion }}
${{ github.event.workflow_run.head_branch }}
```

Example: Print Previous Workflow Data

```
steps:
  - name: Print info
    run: |
```

```
      echo "Workflow Name: ${{ github.event.workflow_run.name }}"
      echo "Status: ${{ github.event.workflow_run.conclusion }}"
```

Security Notes

Be careful with:

- deployment workflows
- secrets
- public repositories
- fork pull requests

Never expose:

- API keys
- tokens
- passwords

inside logs.

Best Practices

Recommended Structure

```
build.yml
↓
test.yml
↓
security-scan.yml
↓
deploy.yml
```

Keep Workflows Small

Instead of one huge workflow:

- split workflows logically
- use workflow_run chaining

Benefits:

- easier debugging
- cleaner logs
- faster maintenance

Use Conditions

Always use:

```
if: ${{ github.event.workflow_run.conclusion == 'success' }}
```

Otherwise deployment may happen even after failure.

Difference Between `workflow_run` and `workflow_call`

Feature	workflow_run	workflow_call
Trigger after workflow ends	Yes	No
Reusable workflows	No	Yes
Workflow chaining	Yes	Limited
Direct function-like call	No	Yes
Common use	CI/CD pipelines	Shared templates

When to Use `workflow_run`

Use it when:

- One workflow depends on another
- You want staged CI/CD
- Deployment should happen after testing
- Security scans run after build
- Notifications after completion

Recommended Project Structure

```
.github/
└── workflows/
    ├── build.yml
    ├── test.yml
    ├── deploy.yml
    └── notify.yml
```

Debugging Checklist

If `workflow_run` is not working:

Check these:

- Workflow name correct?
- YAML syntax valid?
- Actions enabled?
- Workflow on default branch?
- First workflow executed?
- Permissions enabled?
- Infinite loop blocked?
- Repository private/public restrictions?

Final Working Minimal Example

Workflow A

```
name: Test Workflow

on:
  push:

jobs:
  test:
    runs-on: ubuntu-latest
```

```
    steps:
      - run: echo "Testing"
```

Workflow B

```
name: Deploy Workflow

on:
  workflow_run:
    workflows: ["Test Workflow"]
    types:
      - completed

jobs:
  deploy:
    if: ${{ github.event.workflow_run.conclusion == 'success' }}

    runs-on: ubuntu-latest

    steps:
      - run: echo "Deploying"
```

Conclusion

`workflow_run` is one of the most powerful GitHub Actions automation features.

It allows you to:

- chain workflows
- build advanced CI/CD pipelines
- separate testing and deployment
- improve security
- organize automation professionally

Most problems happen because:

- workflow names do not match
- files are not on the default branch
- permissions are disabled
- the first workflow never runs

If configured correctly, `workflow_run` becomes extremely reliable and scalable for professional DevOps workflows.