5 Things You Didn’t Know GitHub Actions Could Do

Audience

- Assume you know what GitHub Actions is
- Want quick tips you can use immediately after this talk is over

Who am I?

- Director of Engineering at Lean TECHniques
- Microsoft MVP
- Co-organizer of Iowa .NET User Group
- Dometrain Author

#1. Sparse Checkouts

- What is it?
  - Allows you to checkout just a subset of the repository

- Why should I care?
  - Target certain parts of your repo
  - Checkout just your IAC folder
  - Useful for monorepos
  - One client went from 2.5 mins checkout of 1GB repo to 10 seconds

#1. Sparse Checkouts – How?

```yaml
- uses: actions/checkout@v4
  with:
    sparse-checkout:
      terraform
```

#2. Run Scheduled Jobs

- What is it?
  - Allows you to run certain jobs on a schedule like daily, hourly, etc.

- Why should I care?
  - Automate simple tasks (ie email me this daily report)
  - Automate daily security scans for repos that don’t get updated often
  - This is NOT an enterprise Cron Job solution, no guarantee of running at the exact time
  - I’ve seen ~15 min variability

#2. Run Scheduled Jobs – How?

```yaml
on:
  schedule:
    - cron: "0 * * * *"
```

#3. Reusing Workflows

- What is it?
  - Allows you to reuse code in your workflows

- Why should I care?
  - Stop copying and pasting YAML
  - Reduce duplication which promotes consistency and maintainability
  - Deploying to different environments usually shares 95% of the same code
  - Consistency across different repos in your enterprise

#3. Reusing Workflows – How?

```yaml
on:
  workflow_call:
    inputs:
      env:
        required: true
        type: string
    secrets:
      azure_client_id:
        required: true
      azure_tenant_id:
        required: true
      azure_subscription_id:
        required: true

jobs:
  deploy:
    name: Deploy ${{ inputs.env }}
    runs-on: ubuntu-latest
    environment: ${{ inputs.env }}

    # more stuff
```

```yaml
  deploy_dev:
    name: Deploy Dev
    needs: build
    uses: ./.github/workflows/step-deploy.yml
    with:
      env: dev
    secrets: inherit

  deploy_prod:
    name: Deploy Prod
    needs: deploy_dev
    uses: ./.github/workflows/step-deploy.yml
    with:
      env: prod
    secrets: inherit
```

#3. Reusing Workflows – How?

```yaml
deploy_dev:
  name: Deploy Dev
  needs: build
  uses: my-org/shared-workflows/.github/workflows/step-deploy.yml@main
  with:
    env: dev
  secrets: inherit

deploy_prod:
  name: Deploy Prod
  needs: deploy_dev
  uses: my-org/shared-workflows/.github/workflows/step-deploy.yml@main
  with:
    env: prod
  secrets: inherit
```

#4. Path Filters

- What is it?
  - Allows you to only run workflow when files that match a pattern change

- Why should I care?
  - Only trigger a workflow when certain files change
  - Optimizes what runs based on what changed – saves time and $
  - Could be a change in a /terraform folder on a PR kicks off a a terraform plan
  - Could be a change in a /sql folder kicks off the SQL migrations

#4. Path Filters – How?

```yaml
on:
  push:
    paths:
      - 'terraform/**'
```

```yaml
on:
  push:
    paths-ignore:
      - 'terraform/**'
```

#5. Trigger actions based on external events

- What is it?
  - Run a workflow when calling GitHub REST API

- Why should I care?
  - Able to integrate GitHub Actions with other systems
  - Could be Slack or Microsoft Teams
  - Could be Datadog or PagerDuty or equivalents
  - Could be Jira or other work tracking tool
  - Could be integrating across multiple repos
  - Could be a CI system or another source control system

#5. Trigger actions based on external events

```yaml
on:
  repository_dispatch:
    types: [slack_message]
```

```json
{
  "event_type": "slack_message",
  "client_payload": {
    "message": "Something custom"
  }
}
```

Takeaways

- Ways to optimize your workflow for speed
- See that GitHub is more than just a CI/CD tool, it’s an automation tool

Thanks!

Slides up at scottsauber.com