LUCES CAMARA Y GITHUB ACTIONS

IXCHEL RUIZ

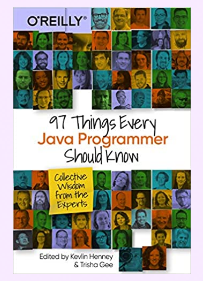
Maven

ABOUT ME

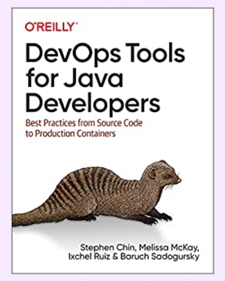
@ixchelruiz@mastodon.social

linkedin/in/ixchelruiz

Senior Software Developer


O'REILLY

97 Things Every Java Programmer Should Know


O'REILLY

DevOps Tools for Java Developers

Why GitHub Actions?

Octoverse 2023

GitHub by the numbers

+100M developers are on GitHub

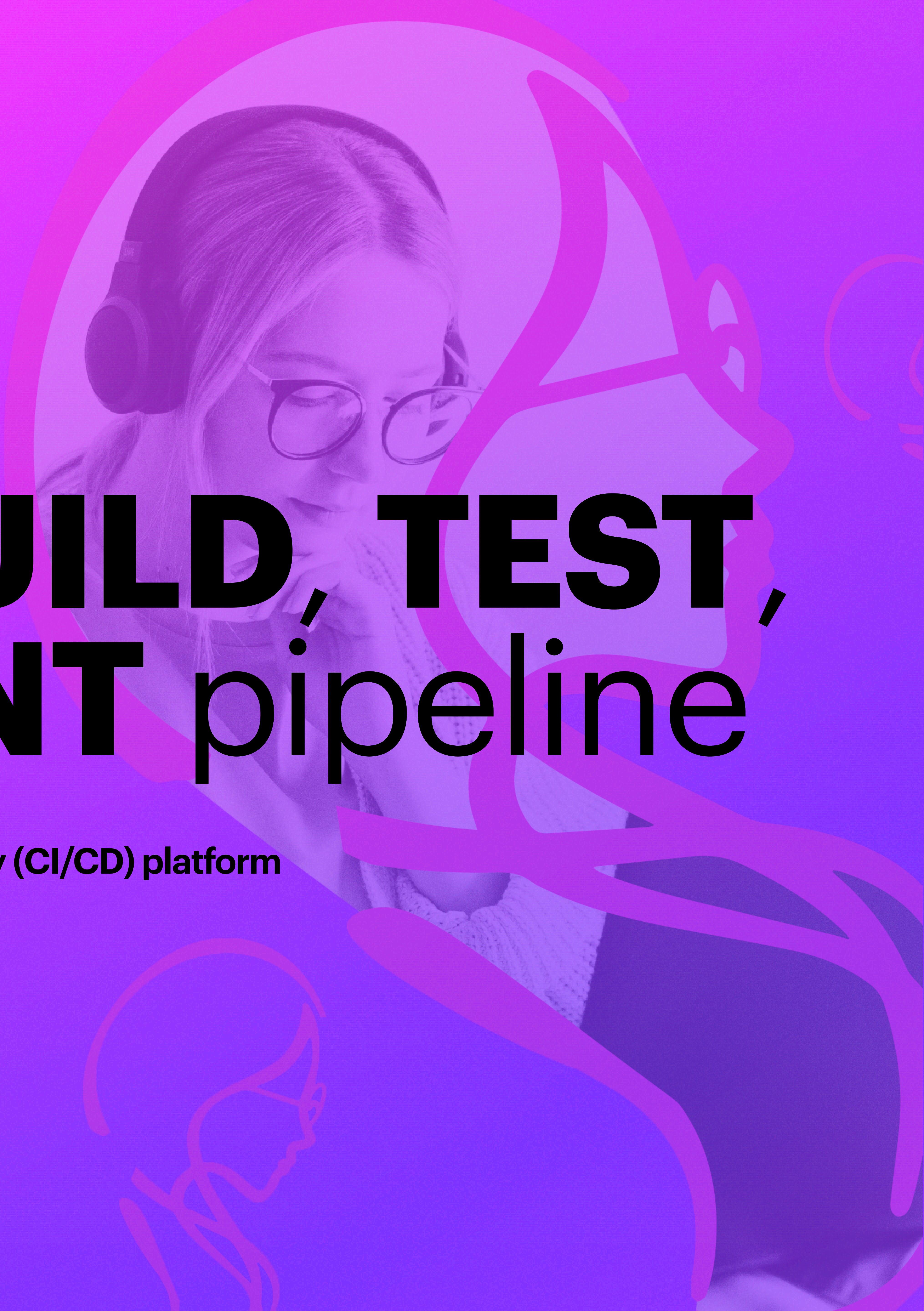
+20M GitHub Actions minutes a day in public projects.

+20K GitHub Actions in the GitHub Marketplace.

Top programming languages : JAVASCRIPT(TYPESCRIPT), PYTHON, **JAVA**

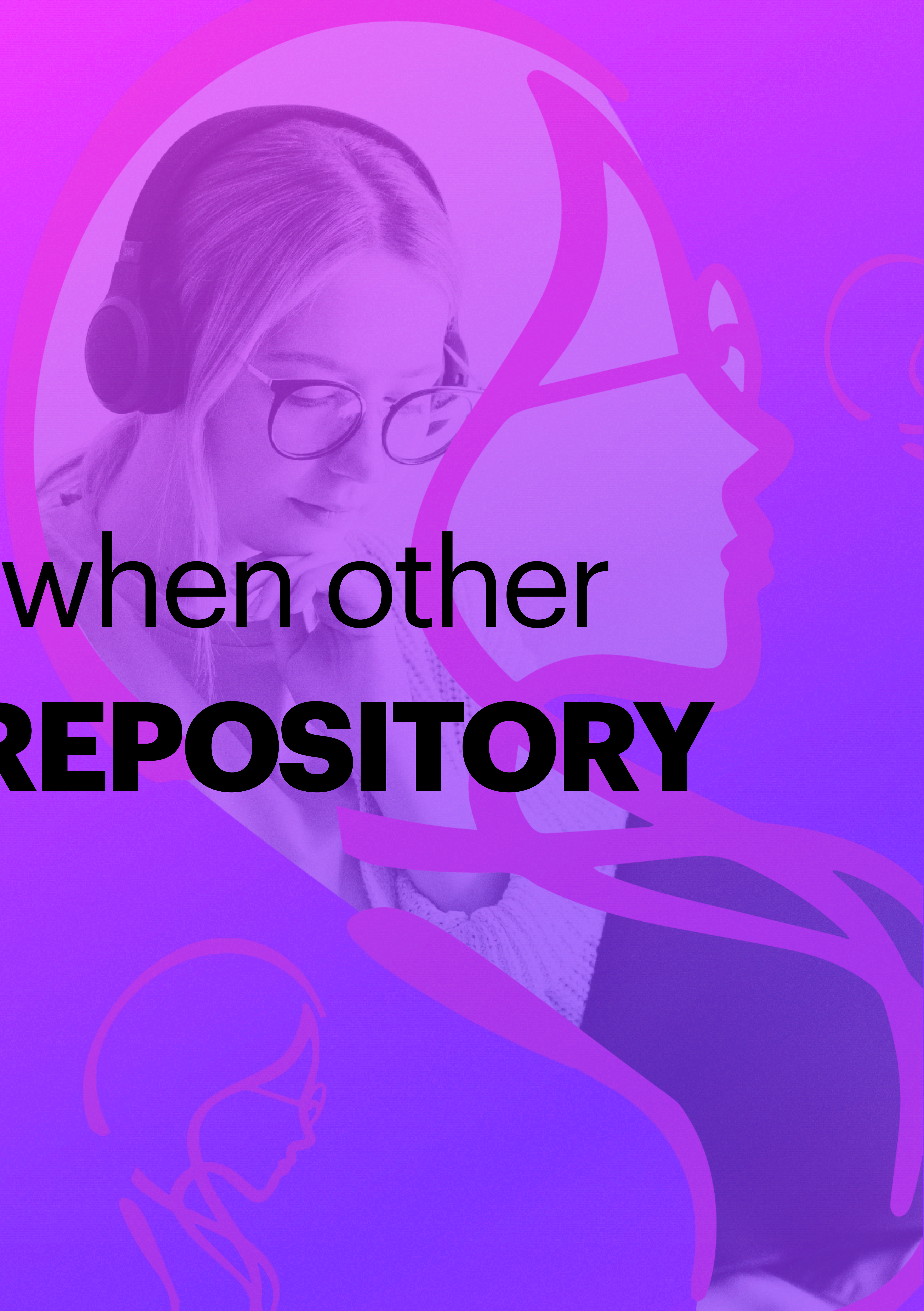
https://octoverse.github.com/

What are GitHub Actions?

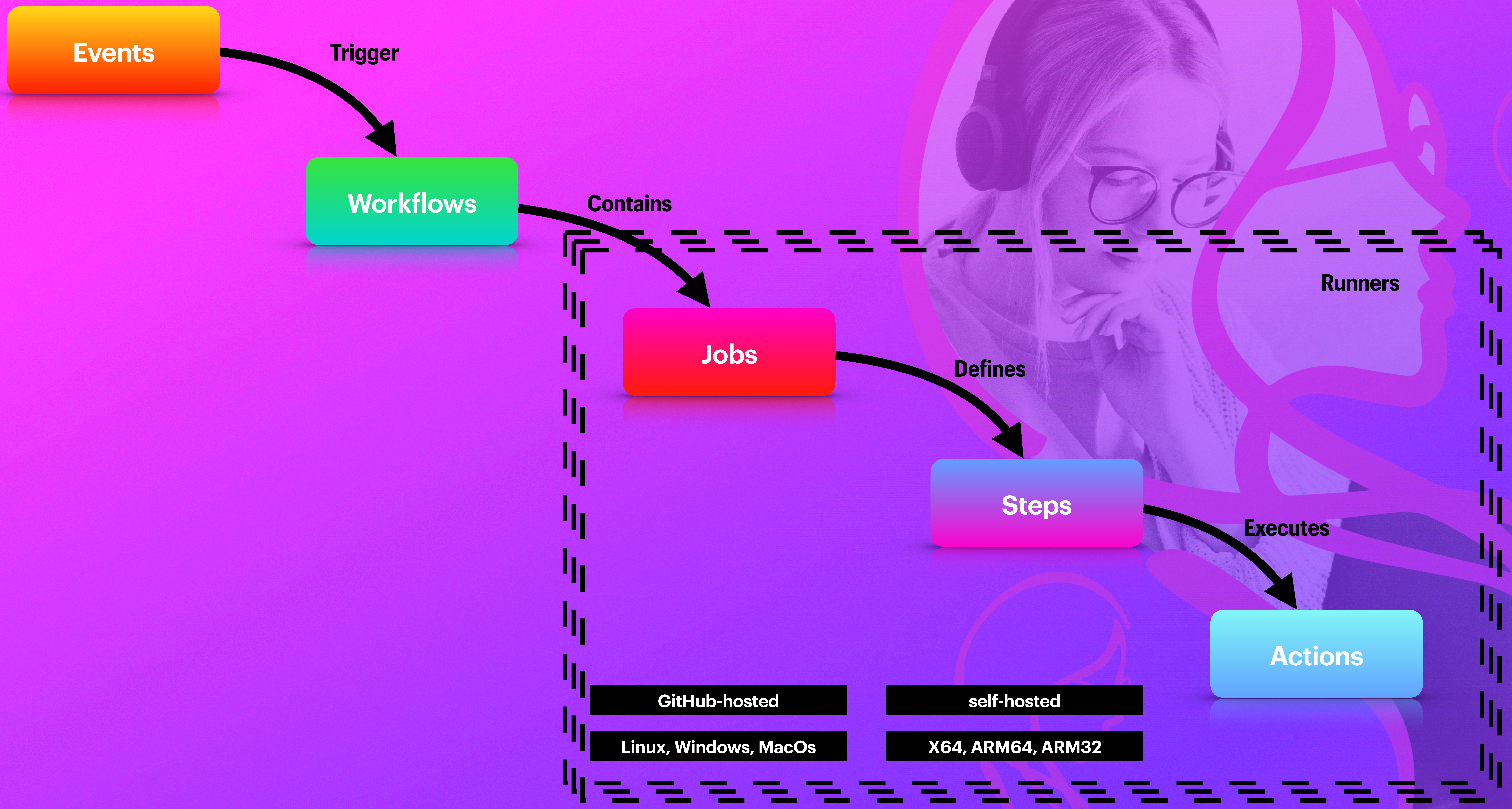


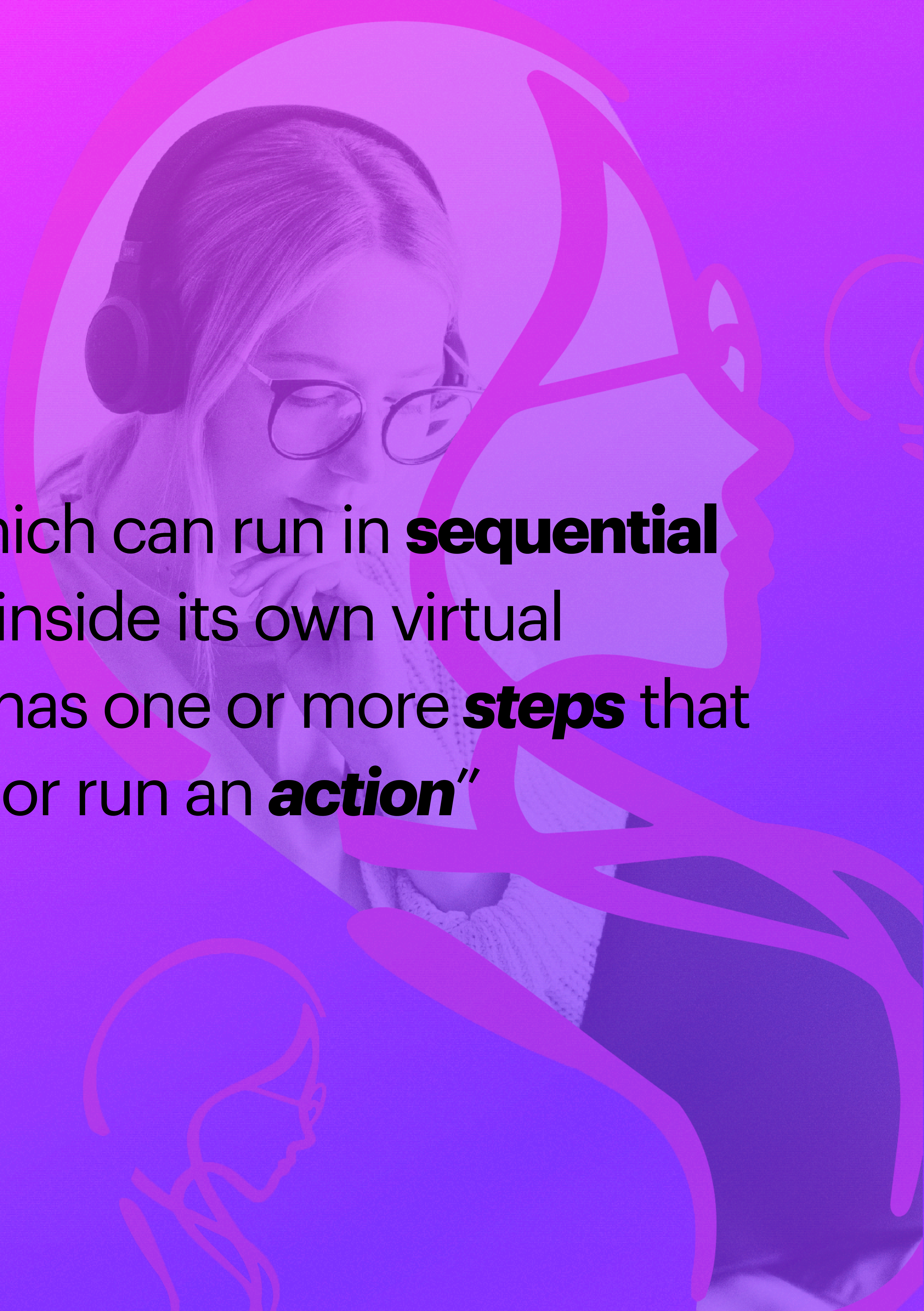
Automate your **BUILD**, **TEST**, and **DEPLOYMENT** pipeline

Continuous integration and continuous delivery (CI/CD) platform



Run **WORKFLOWS** when other **EVENTS** happen in a **REPOSITORY**


Events
Trigger
Workflows
Contains
Jobs
Defines
Steps
Executes
Actions
Runners
GitHub-hosted
Linux, Windows, MacOs
self-hosted
X64, ARM64, ARM32



"A workflow **contains** one or more ***jobs*** which can run in **sequential** order or in **parallel**. Each job will run inside its own virtual machine ***runner***, or inside a container, and has one or more ***steps*** that either run a **script** that you define or run an ***action***"

EVENTS : Triggering a workflow

Workflow triggers

Types

- Events **in** your workflow's repository
- Events **outside** of GitHub and trigger a `repository_dispatch` event on GitHub
- **Scheduled** times
- **Manual**

Some events also require the workflow file to be PRESENT on the default branch of the repository in order to run.

Workflow triggers

Events

branch_protection_rule
check_run
check_suite
create
delete
deployment
deployment_status
discussion
discussion_comment
fork
gollum
issue_comment
issues
label
merge_group
milestone
page_build
project
project_card
project_column
public
pull_request
pull_request_comment (use issue_comment)
pull_request_review
pull_request_review_comment
pull_request_target
push
registry_package
release
repository_dispatch
schedule
status
watch
workflow_call
workflow_dispatch
workflow_run

Workflow triggers

Events

branch_protection_rule
check_run
check_suite
create
delete
deployment
deployment_status
discussion
discussion_comment
fork
gollum
issue_comment
issues
label
merge_group
milestone
page_build
project
project_card
project_column
public
pull_request
pull_request_comment (use issue_comment)
pull_request_review
pull_request_review_comment
pull_request_target
push
registry_package
release
repository_dispatch
schedule
status
watch
workflow_call
workflow_dispatch
workflow_run

Workflow triggers

Events

```
create:
When a Git branch or tag is created.

push:
When there is a push to a repository branch.
  includes
     a commit is pushed,
     a commit tag is pushed,
     a branch is deleted,
     a tag is deleted,
     a repository is created from a template
```

will not occur when more than three tags are created at once.

Workflow triggers

Scheduled

```
on:
  schedule:
    - cron:  '30 5,17 * * *'
```

Every day at 5:30 and 17:30 UTC

1. Minute [0,59]
2. Hour [0,23]
3. Day of the month [1,31]
4. Month of the year [1,12]
5. Day of the week ([0,6] with 0=Sunday)

```
on:
  schedule:
    - cron: '30 5 * * 1,3'
    - cron: '30 5 * * 2,4'

jobs:
  test_schedule:
    runs-on: ubuntu-latest
    steps:
      - name: Not on Monday or Wednesday
        if: github.event.schedule != '30 5 * * 1,3'
        run: echo "This step will be skipped on Monday
and Wednesday"
      - name: Every time
        run: echo "This step will always run"
```

if: github.event.schedule != '30 5 * * 1,3'

Run at 5:30 UTC every Monday-Thursday, but skips “Not on Monday or Wednesday” step on Monday and Wednesday.

Triggering workflows

Workflow triggers

Using events & event types

```
on: push
```

Single

```
on: [ push, fork ]
```

Multiple

```
on:
  label:
    types:
      - created
  push:
    branches:
      - main
  page_build:
```

Event activity types

```
on:
  issues:
    types:
      - opened
      - labeled
```

! **Issue with 2 labels is open ? workflow triggers 3 times.**

Workflow triggers

Using filters

```
on:
  pull_request:
#Sequence of patterns matched against refs/heads
    branches:
      - main
      - 'mona/octocat'
      - ‘releases/**'
```

branches

```
on:
  pull_request:
   branches-ignore:
      - 'mona/octocat'
      - ‘releases/**-alpha’
```

Ignore branches

```
on:
  pull_request:
   branches:
       - ‘releases/**'
      - ‘!releases/**-alpha'
```

Combined ?

! **The order that you define patterns matters.**

Workflow triggers

Using filters

```
on:
  push:
# Sequence of patterns matched against refs/heads
    branches:
      - main
      - 'mona/octocat'
      - 'releases/**'
# Sequence of patterns matched against refs/tags
    tags:
      - v2
      - v1.*
```

tags

```
on:
  push:
# Sequence of patterns matched against refs/heads
    branches-ignore:
      - 'mona/octocat'
      - 'releases/**-alpha'
# Sequence of patterns matched against refs/tags
    tags-ignore:
      - v2
      - v1.*
```

Ignore tags

Works with paths too!

Communication

Communication

Passing values

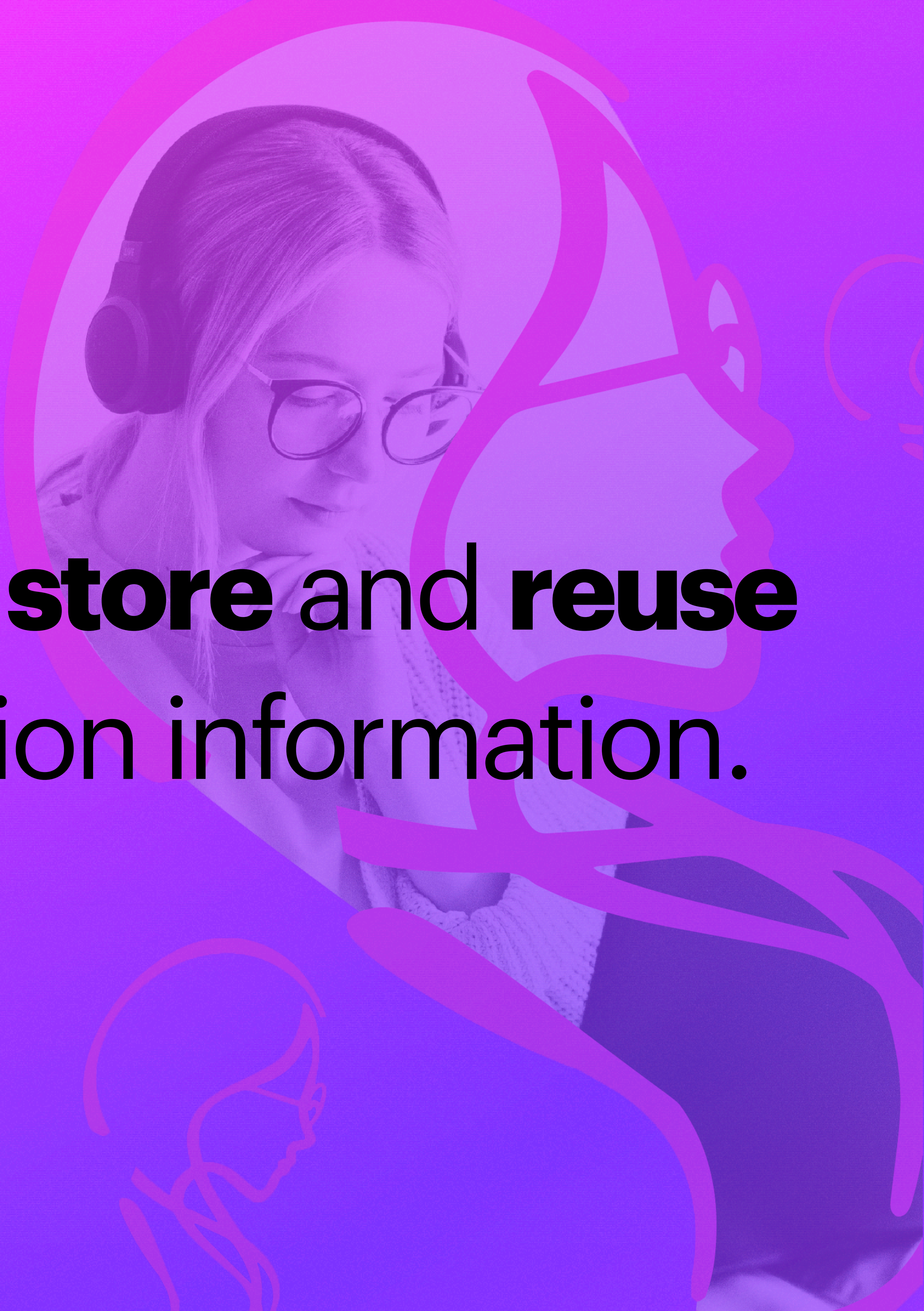
```
Context
Variables
  Environment Variables
Outputs
Inputs
```

A way to *access* **information** about workflow runs, **variables**, **runner** environments, **jobs**, and **steps**.

Context

```
${{ <context> }}
```

You can access contexts using the expression syntax



Variables provide a way to **store** and **reuse** **non-sensitive** configuration information.

Environment Variables

Single workflow

Scope of a custom variable

⋆ **Entire** workflow >> `env` (at top level)

⋆A **job** within a workflow >> `jobs.<job_id>.env`

⋆A specific **step** within a job >> `jobs.<job_id>.steps[*].env`

Environment variables ***exist*** *only* on the **runner** that is executing a *job*

Using env context

to access environment variable values

Workflow level

Job level

Step level

```yaml
env:
  DAY_OF_WEEK: Monday
jobs:
  greeting_job:
    runs-on: ubuntu-latest
    env:
      Greeting: Hello
    steps:
      - name: "Say Hello Mona it's Monday"
        if: ${{ env.DAY_OF_WEEK == 'Monday' }}
        run: echo "$Greeting $First_Name. Today is $DAY_OF_WEEK!"
        env:
          First_Name: Mona
```

! **cannot use runner environment variables in parts of a workflow that are processed by GitHub Actions and are not sent to the runner.**

Use a context in an "if" conditional statement to access the value of an variable.

Environment Variables

Single workflow or Multiple workflows.

Scope of a custom variable

⋆ **Entire** workflow >> `env` (at top level)

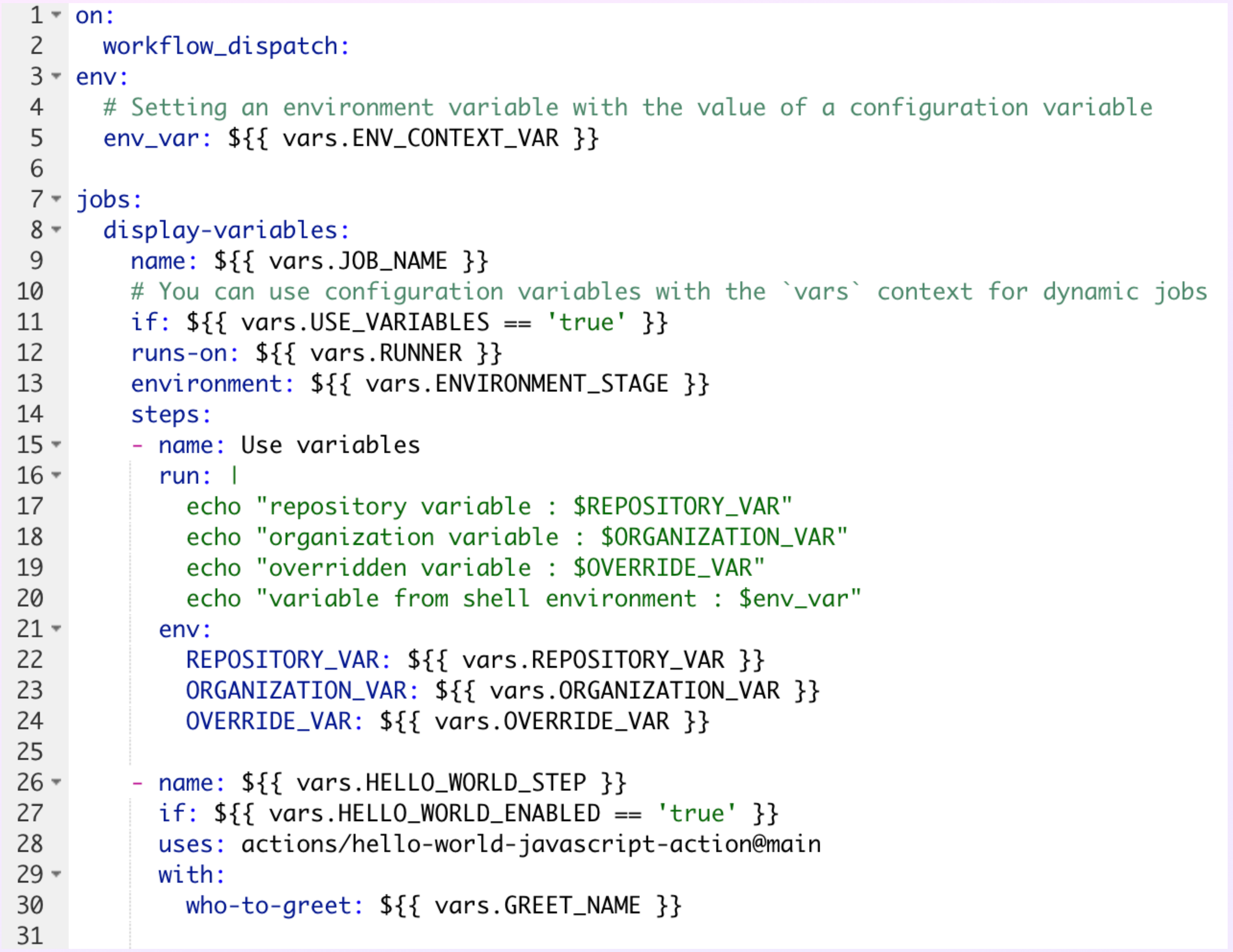
⋆A **job** within a workflow >> `jobs.<job_id>.env`

⋆A specific **step** within a job >> `jobs.<job_id>.steps[*].env`

⋆ Configuration variables can be accessed **across the workflows** >> `vars` context.

Using the vars context

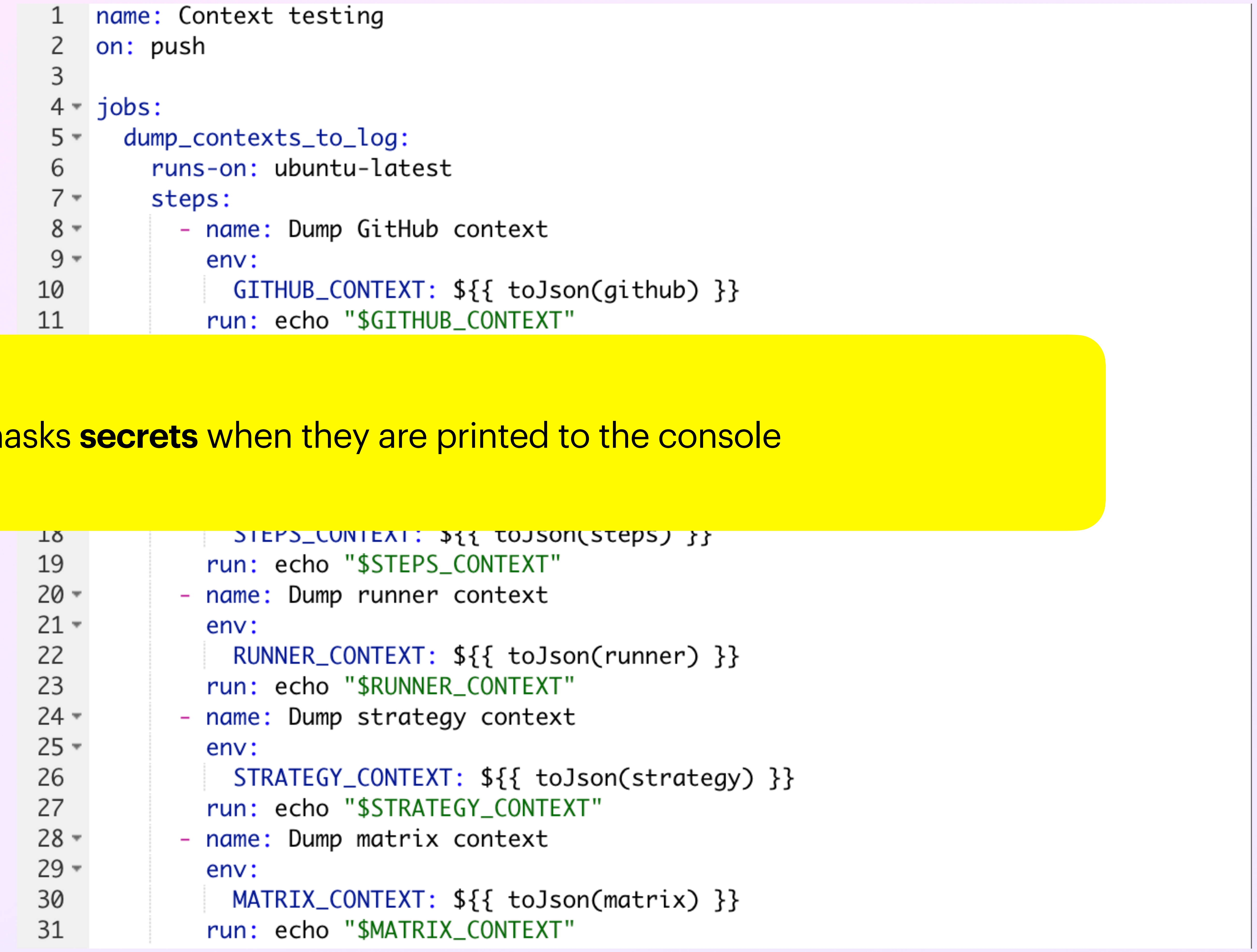
to access configuration variable values



```yaml
on:
  workflow_dispatch:
env:
  # Setting an environment variable with the value of a configuration variable
  env_var: ${{ vars.ENV_CONTEXT_VAR }}

jobs:
  display-variables:
    name: ${{ vars.JOB_NAME }}
    # You can use configuration variables with the `vars` context for dynamic jobs
    if: ${{ vars.USE_VARIABLES == 'true' }}
    runs-on: ${{ vars.RUNNER }}
    environment: ${{ vars.ENVIRONMENT_STAGE }}
    steps:
    - name: Use variables
      run: |
        echo "repository variable : $REPOSITORY_VAR"
        echo "organization variable : $ORGANIZATION_VAR"
        echo "overridden variable : $OVERRIDE_VAR"
        echo "variable from shell environment : $env_var"
      env:
        REPOSITORY_VAR: ${{ vars.REPOSITORY_VAR }}
        ORGANIZATION_VAR: ${{ vars.ORGANIZATION_VAR }}
        OVERRIDE_VAR: ${{ vars.OVERRIDE_VAR }}

    - name: ${{ vars.HELLO_WORLD_STEP }}
      if: ${{ vars.HELLO_WORLD_ENABLED == 'true' }}
      uses: actions/hello-world-javascript-action@main
      with:
        who-to-greet: ${{ vars.GREET_NAME }}

```

Example: printing context information to the log

pretty-print JSON objects to the log



```yaml
name: Context testing
on: push

jobs:
  dump_contexts_to_log:
    runs-on: ubuntu-latest
    steps:
      - name: Dump GitHub context
        env:
          GITHUB_CONTEXT: ${{ toJson(github) }}
        run: echo "$GITHUB_CONTEXT"
```

```yaml
          STEPS_CONTEXT: ${{ toJson(steps) }}
        run: echo "$STEPS_CONTEXT"
      - name: Dump runner context
        env:
          RUNNER_CONTEXT: ${{ toJson(runner) }}
        run: echo "$RUNNER_CONTEXT"
      - name: Dump strategy context
        env:
          STRATEGY_CONTEXT: ${{ toJson(strategy) }}
        run: echo "$STRATEGY_CONTEXT"
      - name: Dump matrix context
        env:
          MATRIX_CONTEXT: ${{ toJson(matrix) }}
        run: echo "$MATRIX_CONTEXT"
```

! `github.token` GitHub masks **secrets** when they are printed to the console

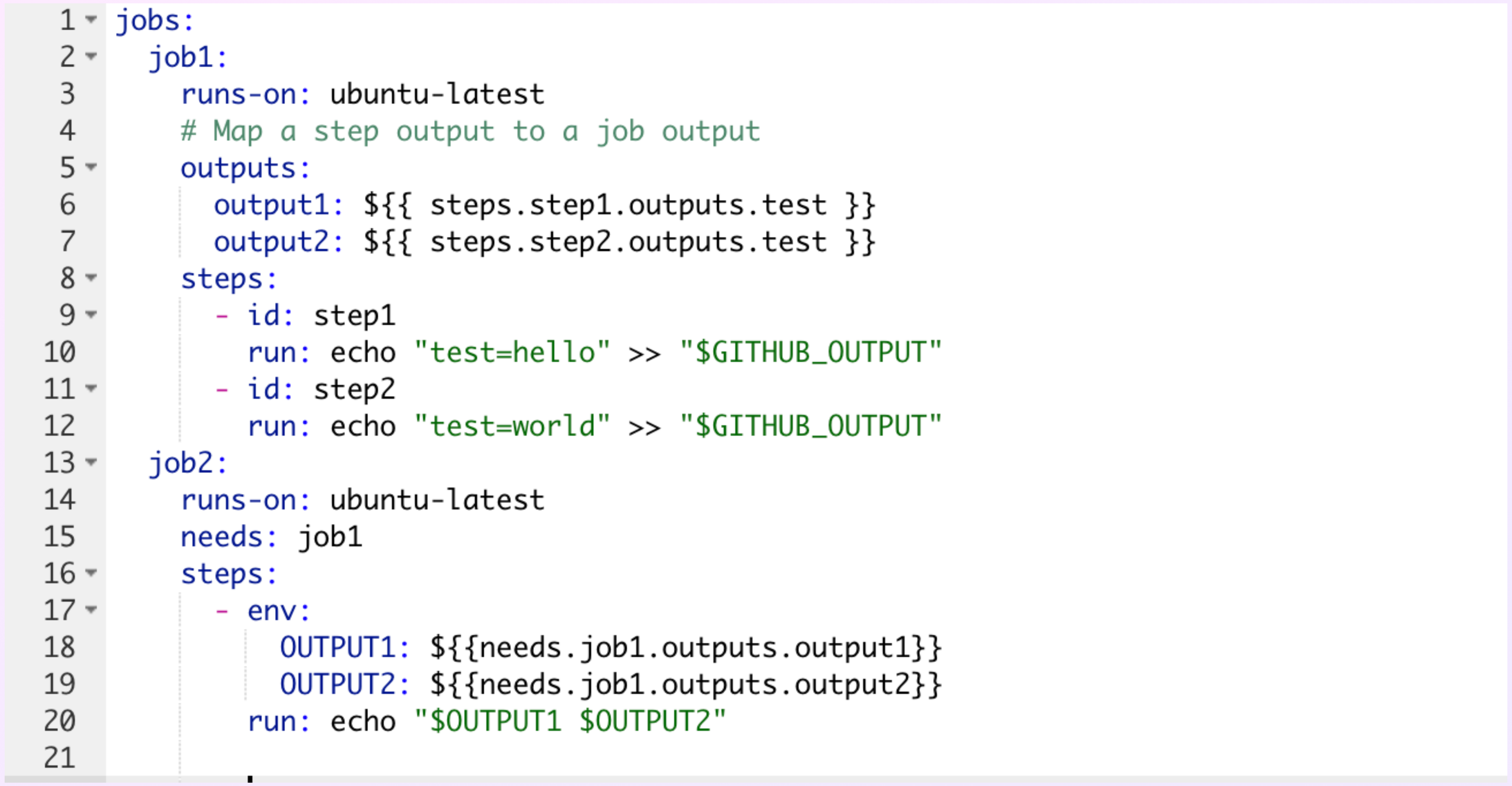
Outputs

Passing values between steps and jobs in a workflow

- ★ `jobs.<job_id>.outputs` >> create a **map** of outputs for a **job**.
- ★ Job outputs are **available** to all downstream jobs that **depend** on this job.

❗ `retention period : anywhere between 1 or 400 days`

Example: Defining outputs for a job



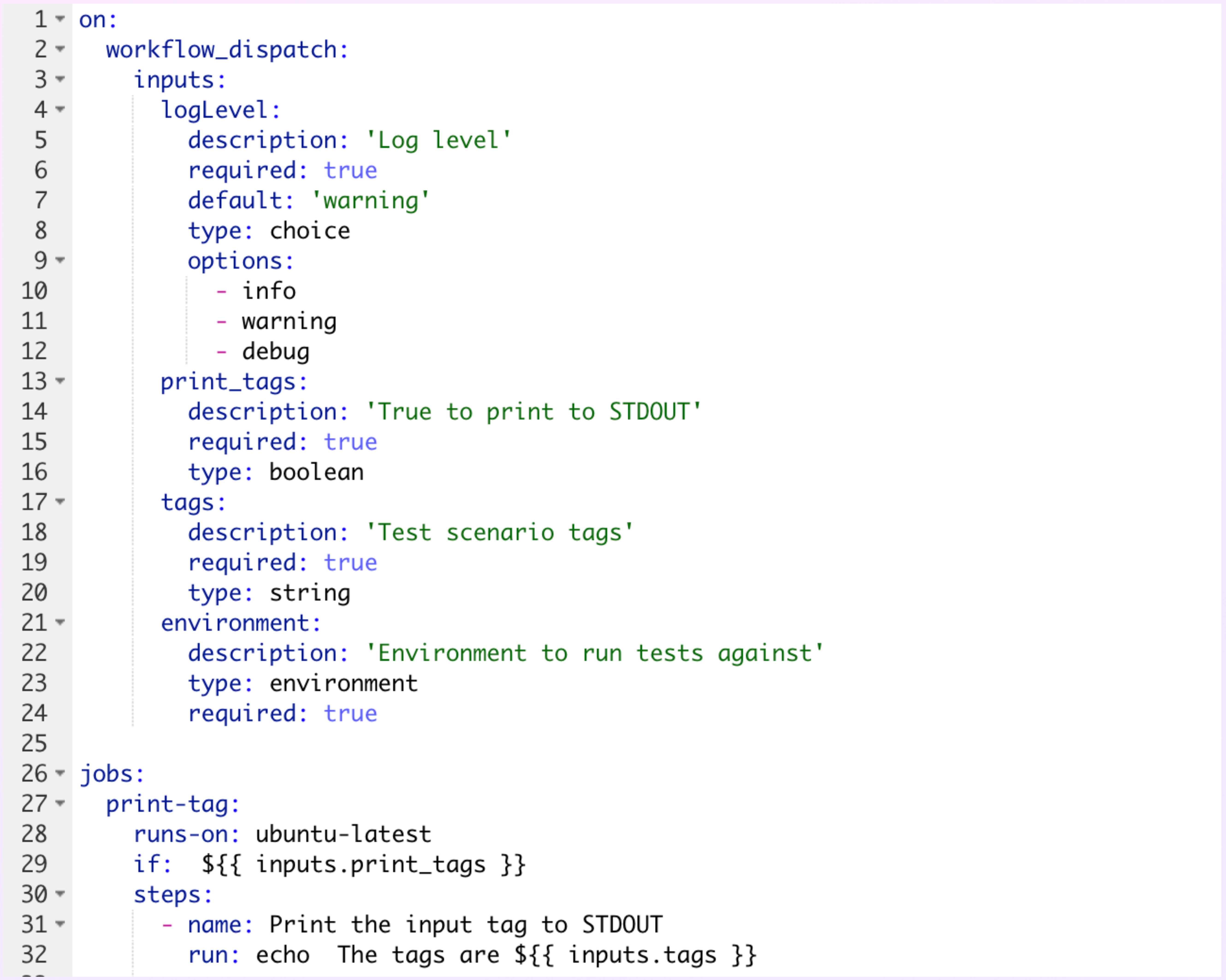
```yaml
jobs:
  job1:
    runs-on: ubuntu-latest
    # Map a step output to a job output
    outputs:
      output1: ${{ steps.step1.outputs.test }}
      output2: ${{ steps.step2.outputs.test }}
    steps:
      - id: step1
        run: echo "test=hello" >> "$GITHUB_OUTPUT"
      - id: step2
        run: echo "test=world" >> "$GITHUB_OUTPUT"
  job2:
    runs-on: ubuntu-latest
    needs: job1
    steps:
      - env:
          OUTPUT1: ${{needs.job1.outputs.output1}}
          OUTPUT2: ${{needs.job1.outputs.output2}}
        run: echo "$OUTPUT1 $OUTPUT2"

```

Inputs

Defining inputs for manually triggered workflows

The **maximum** number of **top-level** properties for inputs is **10.**

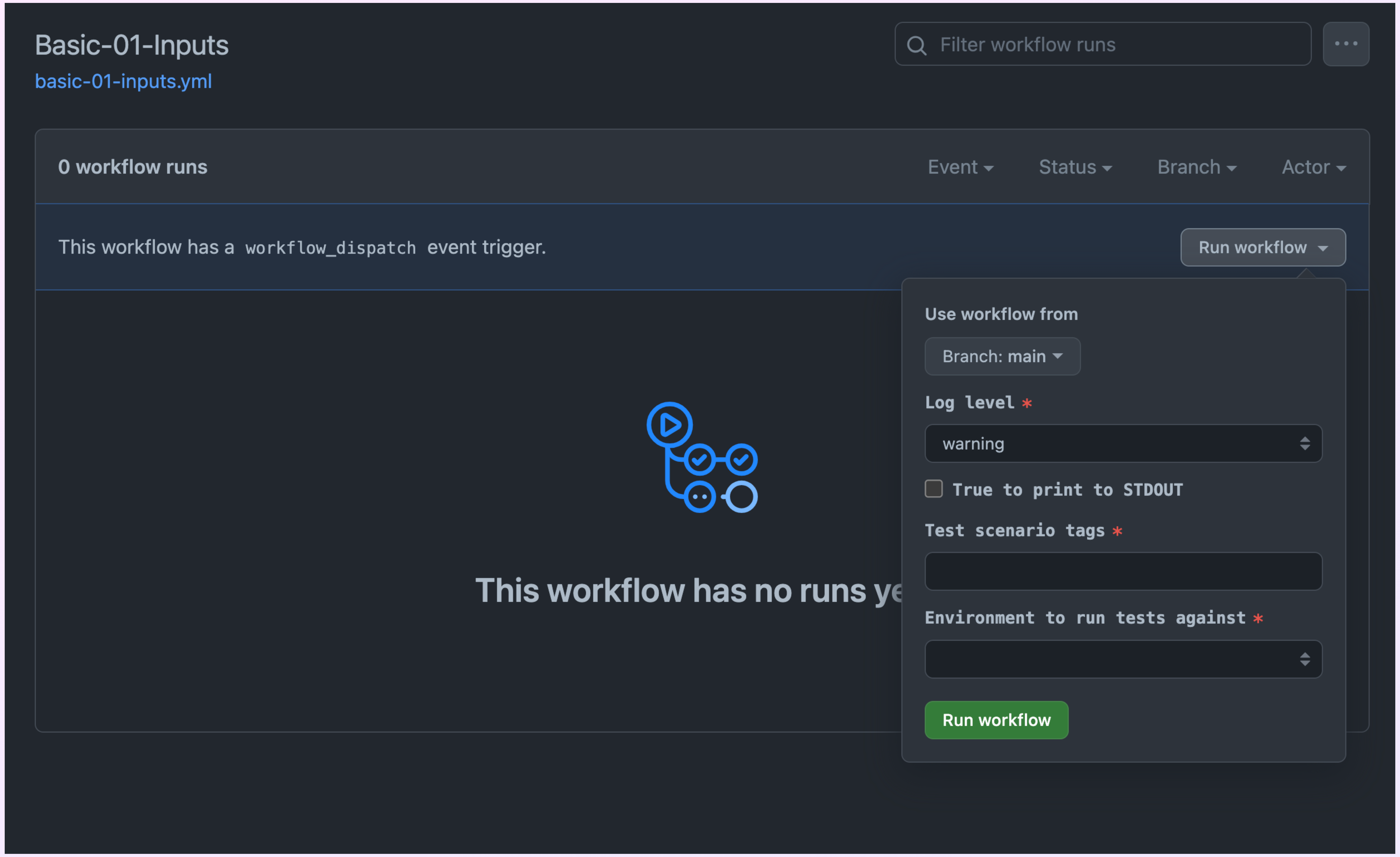
The maximum **payload** for inputs is **65,535 characters.**



```yaml
on:
  workflow_dispatch:
    inputs:
      logLevel:
        description: 'Log level'
        required: true
        default: 'warning'
        type: choice
        options:
          - info
          - warning
          - debug
      print_tags:
        description: 'True to print to STDOUT'
        required: true
        type: boolean
      tags:
        description: 'Test scenario tags'
        required: true
        type: string
      environment:
        description: 'Environment to run tests against'
        type: environment
        required: true

jobs:
  print-tag:
    runs-on: ubuntu-latest
    if:  ${{ inputs.print_tags }}
    steps:
      - name: Print the input tag to STDOUT
        run: echo  The tags are ${{ inputs.tags }}
```

Inputs

Defining inputs for manually triggered workflows



Basic-01-Inputs

basic-01-inputs.yml

This workflow has a `workflow_dispatch` event trigger.

Log level *

warning

☐ True to print to STDOUT

Test scenario tags *

Environment to run tests against *

Jobs

Using jobs

in a workflow

- A workflow run is made up of *one or more* jobs, which run in **parallel** by *default*.
- Each job runs in a *runner environment* specified by runs-on.
- You can run an unlimited number of jobs as long as you are within the workflow **usage limits**.

Using jobs

Limits

★ Job execution time	❗ **Can run for up to 6 hours — terminated & fail**
★ Workflow run time	❗ **Limited to 35 days — cancelled**
★ API requests	❗ **1,000 requests to the GitHub API in an hour across all actions within a repository**
★ Concurrent jobs	❗ **Free : 20 concurrent (5 max macOs)**

Jobs

Dependencies

```
jobs:
  job1:
  job2:
    needs: job1
  job3:
    needs: [job1, job2]
```

```
jobs:
  job1:
  job2:
    needs: job1
  job3:
    if: ${{ always() }}
    needs: [job1, job2]
```

needs

if: {{ always () }}

Conditionals

Conditionals

```
jobs:
  production-deploy:
    if: github.repository == 'octo-org/octo-repo-prod'
    runs-on: ubuntu-latest


  steps:
    name: My first step
    if: ${{ github.event_name == 'pull_request' && github.event.action == 'unassigned' }}
    run: echo This event is a pull request that had an assignee removed.
```

```
 if: ${{ ! startsWith(github.ref, 'refs/tags/') }}
```

```
 if: ${{ github.ref == 'refs/heads/main' }}
```

Concurrency

Concurrency

a single job or workflow using the same concurrency group will run at a time

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

concurrency:
  group: ${{ github.head_ref || github.run_id }}
  cancel-in-progress: true

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

Fallback value

! concurrency group **names** must be **unique** across workflows

Reusable workflows

Reusing workflows

- A workflow that uses another workflow is referred to as a "caller" workflow.
- The reusable workflow is a *"called"* workflow.
- One *"caller"* workflow can use **multiple** called workflows.
- Connect up to **four levels** of workflows.
- Call a *maximum of **20 reusable*** workflows from a single workflow file

Reusing workflows

- "Caller" *environment variables* ***NOT propagated*** to "Called".
- *"Called" environment variables* ***NOT accessible*** to "Caller".
  - Reuse variables in multiple workflows >> `vars` **context**
- Reusable workflows are called directly *within a* ***job****,* and **not** from within a job **step.**

Reusable Workflows

```yaml
name: Reusable workflow

on:
  workflow_call:
    # Map the workflow outputs to job outputs
    outputs:
      firstword:
        description: "The first output string"
        value: ${{ jobs.example_job.outputs.output1 }}
      secondword:
        description: "The second output string"
        value: ${{ jobs.example_job.outputs.output2 }}

jobs:
  example_job:
    name: Generate output
    runs-on: ubuntu-latest
    # Map the job outputs to step outputs
    outputs:
      output1: ${{ steps.step1.outputs.firstword }}
      output2: ${{ steps.step2.outputs.secondword }}
    steps:
      - id: step1
        run: echo "firstword=hello" >> $GITHUB_OUTPUT
      - id: step2
        run: echo "secondword=world" >> $GITHUB_OUTPUT
```

```yaml
name: Call a reusable workflow and use its outputs

on:
  workflow_dispatch:

jobs:
  job1:
    uses: octo-org/example-repo/.github/workflows/called-workflow.yml@v1

  job2:
    runs-on: ubuntu-latest
    needs: job1
    steps:
      - run: echo ${{ needs.job1.outputs.firstword }} $
{{ needs.job1.outputs.secondword }}
```

```
hello world
```

Reusable Workflows

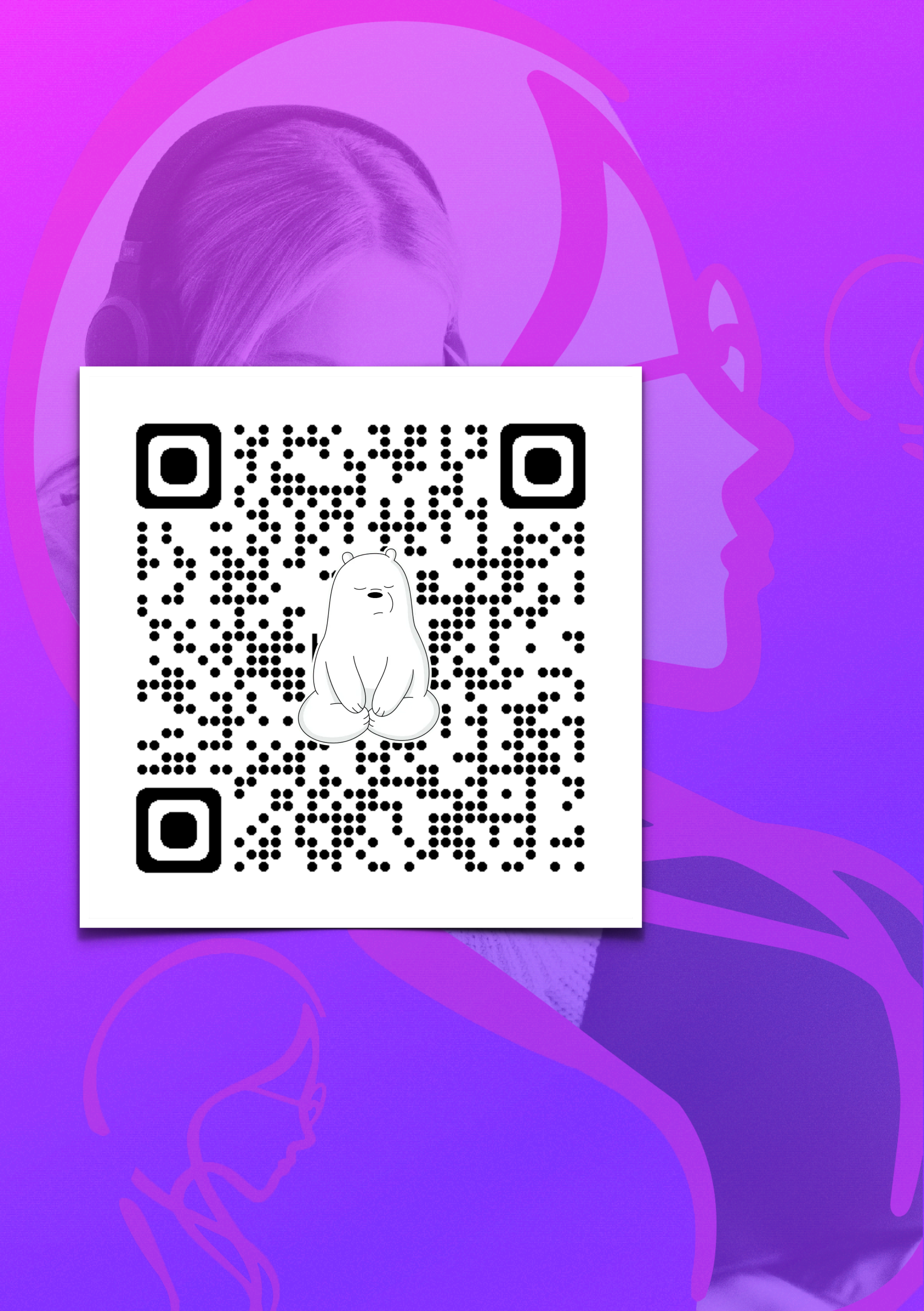
Secrets

```
jobs:
  workflowA-calls-workflowB:
    uses: octo-org/example-repo/.github/workflows/B.yml@main
    secrets: inherit # pass all secrets
```

! GITHUB_TOKEN permissions can only be the **same or more restrictive** in nested workflows

```
jobs:
  workflowB-calls-workflowC:
    uses: different-org/example-repo/.github/workflows/C.yml@main
    secrets:
      envPAT: ${{ secrets.envPAT }} # pass just this secret
```

DEMOS



WRITE US!

@ixchelruiz@mastodon.social

linkedin/in/ixchelruiz

github/ixchelruiz

Dad.. That kid said my artwork sucked!
Oh yeah... Well I'm going to teach him a lesson!
How To GIVE GOOD FEEDBACK
① Be Positive
② Be Specific
③ Give a Next STEP.
Good use of color, but I think it needs more detail.
I'll definitely consider that. Thanks.
www.lunarbaboon.com