GitHub Actions

Pre-requisites

- Git 2.35.x
- Github account

Activities

Activity 1a: Simple Workflow

Steps

1. Create a new git repository and push to github
    a. Name: `gh-actions-demo`
2. Create the github actions directory '`.github/workflows`'
3. Create a workflow at '`.github/workflows/demo.yml`'

```yaml
name: Demo Workflow
on: [push]
jobs:
 Demo-Job:
   runs-on: ubuntu-latest
   steps:
     - name: Check out repository code
       uses: actions/checkout@v4
     - name: List files in the repository
       run: |
         echo "Listing files in the repository(${{ github.workspace }}).."
         ls ${{ github.workspace }}
     - run: echo "Job's status is ${{ job.status }}"
```

4. Commit, push, and check the workflow execution status

Hints

- Reference to understand the syntax
  https://docs.github.com/en/actions/using-workflows/workflow-syntax-for-github-actions

Prompts

1. Where is the runner running at? In whose cloud environment is your workflow running?
2. What is the operating system your workflow is running on?
3. Which step in the workflow cloned your repository into the runner?

Activity 1b: Errored Workflow

Observe that when an error occurs (e.g. non-zero exit by a program), the workflow would halt. We shall attempt to simulate an error, then navigate the github page to find where and what caused the workflow to fail.

Steps

1. Add `Faking an error` as a step within the last workflow

```
    - name: Faking an error
      run: |
        echo "Before faking an error"
        exit 1
        echo "After faking an error"
```

2. Commit, push, and check the workflow execution status

Prompts

1. Which lines got printed in the console?

Activity 2a: Another Workflow and Another Branch

Create a second workflow that shows the contents of your readme

Steps

1. Create a workflow at '`.github/workflows/demo-2.yml`'
    a. Copy the full contents of '`demo1.yml`'
    b. Change the name, in line 1, to '`Demo Workflow 2`'
2. Modify the workflow such that it will print (e.g. `cat` command) the contents of the '`README.md`' file
3. Commit, push, and check the workflow execution status

Prompts

1. How many workflow runs have been triggered, does it correspond to the number of pushes you have made to the remote repository?
2. Would the workflow still get triggered when you introduce a change to another branch? Attempt to work on another branch and see the workflow execution status.

Activity 2b: Narrowing the Scope of Workflow Triggers

Filters can be used to control when your workflow should run

Setup

1. Back to the '`main`' branch
2. Modify the workflow at '`.github/workflows/demo2.yml`' such that it gets triggered upon a push to the '`main`' branch only
3. Commit, push, and check the workflow execution status
   Validate that your filters work

Prompts

1. What are other events that can trigger workflows?

Hints

- Reference to filters
  https://docs.github.com/en/actions/using-workflows/triggering-a-workflow#using-filters

Activity 3a: Manual Workflow Triggers

Steps

1. Create a workflow at '`.github/workflows/demo3.yml`'
    a. Copy the full contents of '`demo1.yml`'
    b. Change the name, in line 1, to '`Demo Workflow 3`'
2. Modify the workflow such that it requires a manual trigger
3. Commit and push
4. Attempt to trigger your workflow

Hints

- Reference to workflow dispatch event https://docs.github.com/en/actions/using-workflows/triggering-a-workflow#defining-inputs-for-manually-triggered-workflows
- Workflow dispatch without arguments https://github.com/orgs/community/discussions/26098

Activity 3b: Repository Secrets and Variables

Steps

1. On the repository page on github navigate to
   `Settings > Security > Secrets and variables > Actions`
2. Create a repository variable `EXAMPLE_VARIABLE`
3. Create a new step that includes the following run statement

```
echo "repository variable : $vars.EXAMPLE_VARIABLE"
```

4. Commit, push, and check the workflow execution status
   Ensure that the variable is printed
5. Create a repository secret `EXAMPLE_SECRET`
6. Modify the step to include the following run statement

```
echo "repository secret : $secrets.EXAMPLE_SECRET"
```

7. Commit, push, and check the workflow execution status
   Ensure that the secret is printed and not exposed

Hints

- Configuring Repository Variables
  https://docs.github.com/en/actions/learn-github-actions/variables#creating-configuration-variables-for-a-repository
- Using Variables
  https://docs.github.com/en/actions/learn-github-actions/variables#using-the-vars-context-to-access-configuration-variable-values
- Configuring Repository Secrets
  https://docs.github.com/en/actions/security-guides/using-secrets-in-github-actions#creating-secrets-for-a-repository
- Using Secrets
  https://docs.github.com/en/actions/security-guides/using-secrets-in-github-actions#example-using-bash

(Bonus) Activity 4a: Checkout Action

Instead of cloning the main branch, clone a different branch

Steps

1. Provide an additional argument to the `checkout` step so it will switch to a different branch after cloning

Hint

- Documentation to Checkout Action
  https://github.com/actions/checkout

(Bonus) Activity 4b: Thought Exercise

How can we make use of Github Actions to deploy infrastructure and build container images?

What executable binaries are needed to execute your build / deploy commands?

Are there other *actions* available that can perform terraform or docker related actions?