Inputs in GitHub Actions

Inputs are a valuable feature in GitHub Actions, enabling workflows and custom actions to accept and utilize input parameters. It can be used in custom actions, reusable workflows, and manually triggered workflows.

Overview

- The `inputs` object becomes accessible in a workflow when triggered, allowing users to define input parameters.
- Each entry `inputs.<name>` represent individual input values passed from an external workflow or trigger event.

- **Reusable Workflows:** Inputs are defined under the `workflow_call` key in the `on` definition of the workflow.
- **Manually Triggered Workflows:** Inputs are defined under the `workflow_dispatch` key in the `on` definition of the workflow.
- **Custom Actions:** Inputs are defined in the `action.yaml` file in the custom action definition.

Testing Inputs in Workflow

1. Create Workflow File:

- Create a new file named `13-01-Inputs.yml` in the `.github/workflows` directory of your repository.

```yaml
name: 13-01-Inputs

on:
  workflow_dispatch:
    inputs:
      dry-run:
        type: boolean
        description: Skip deployment and only print build output
        default: false
      target:
        type: environment
        required: true
        description: Which environment the workflow will target
      tag:
        type: choice
        options:
          - v1
          - v2
          - v3
        default: v3
        description: Release from which to build and deploy
```

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Build
        run: echo "Building from tag ${{ inputs.tag }}"
  deploy:
    runs-on: ubuntu-latest
    if: ${{ !inputs.dry-run }}
    environment: ${{ inputs.target }}
    needs: build
    steps:
      - name: Deploy
        run: echo "Deploying to ${{ inputs.target }}"
```

- Copy and paste the provided YAML configuration into this file.

2. Understanding the Workflow:

- This workflow is triggered manually via the GitHub UI (`workflow_dispatch`).
- It includes two input parameters: `dry-run`, `target`, and `tag`, allowing users to specify options such as skipping deployment (`dry-run`), selecting the target environment (`target`), and choosing the release tag (`tag`).
- The `dry-run` input is a boolean type that defaults to `false`, allowing users to toggle whether deployment should be skipped and only the build output should be printed.
- The `target` input is of type `environment`, which requires users to specify the target environment where the deployment will occur. It is a required input.
- The `tag` input is of type `choice` with predefined options (`v1`, `v2`, `v3`), allowing users to select the release tag from which to build and deploy. It defaults to `v3`.
- The workflow consists of two jobs: `build` and `deploy`.
- The `build` job runs on an Ubuntu latest runner and prints a message indicating the tag from which it is building.
- The `deploy` job runs on an Ubuntu latest runner if the `dry-run` input is not set to `true`. It is conditional on the value of `dry-run`.
- The `deploy` job specifies the `target` environment using the input value provided by the user.

3. Testing the Workflow:

- Commit and push the workflow file (`13-01-Inputs.yml`) to your repository.
- Navigate to the "Actions" tab in your GitHub repository.
- Manually trigger the workflow by clicking on the "Run workflow" button for the `13-01-Inputs` workflow.
- Provide input values for the `dry-run`, `target`, and `tag` parameters as prompted.
- Monitor the workflow run and verify that the specified inputs are correctly processed and applied to the workflow execution.
- Pay attention to the conditional execution of the `deploy` job based on the value of the `dry-run` input.

4. Observing Output:

- Review the logs of each job and step to ensure that the workflow behaves as expected based on the provided inputs.
- Verify that the correct messages are printed for the `build` and `deploy` jobs, considering the input values provided.