Working with Outputs in GitHub Actions

In GitHub Actions, outputs allow you to pass data between different jobs within the same workflow. This can be incredibly useful for sharing results or data produced in one job with others that depend on it. Let's explore how to work with job outputs in GitHub Actions.

Overview

- You can use `jobs.<job_id>.outputs` to define outputs for a specific job.
- Job outputs containing expressions are evaluated at the end of each job.
- Outputs that contain secrets are redacted on the runner and not exposed in GitHub Actions.

Testing Outputs in Workflow

Using Outputs for a Job

1. Create Workflow File:

- Create a new file named `14-01-Outputs.yml` in the `.github/workflows` directory of your repository.

```yaml
name: 14-01-Outputs

on:
  workflow_dispatch

jobs:
  build:
    runs-on: ubuntu-latest
    outputs:
      result: ${{ steps.build.outputs.result }}
    steps:
      - name: Build
        id: build
        run: echo "result=success" >> "$GITHUB_OUTPUT"

  deploy:
    runs-on: ubuntu-latest
    needs: build
    steps:
      - name: Deployment
        env:
          BUILD_STATUS: ${{ needs.build.outputs.result }}
        run: echo "Build Status: $BUILD_STATUS"
```

- 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 defines two jobs: `build` and `deploy`.
- The `build` job runs on an `Ubuntu latest` runner and is responsible for building the project.
- The `deploy` job runs on an Ubuntu latest runner and is conditional on the successful completion of the `build` job (needs: `build`).
- The `build` job outputs a variable named `result` with the value `"success"`.
- The `deploy` job retrieves the output value of the `result` variable from the `build` job and assigns it to the `BUILD_STATUS` environment variable.
- The `deploy` job then prints the value of the `BUILD_STATUS` environment variable.

3. Testing the Workflow:

- Commit and push the workflow file (`14-01-Outputs.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 `14-01-Outputs` workflow.
- Monitor the workflow run and verify that the `build` job completes successfully and outputs the `result` variable with the value `"success"`.
- Ensure that the `deploy` job is triggered after the `build` job and that it correctly retrieves the `result` output from the `build` job.
- Verify that the `deploy` job prints the correct value of the `BUILD_STATUS` environment variable, which should be `"success"`.

4. Observing Output:

- Review the logs of each job and step to ensure that the workflow behaves as expected.
- Verify that the value of the `BUILD_STATUS` environment variable matches the output value of the `result` variable from the `build` job.

Avoiding the Mistake of Overwritting the Output File

1. Create Workflow File:

- Create a new file named `14-02-Outputs.yml` in the `.github/workflows` directory of your repository.

```yaml
name: 14-02-Outputs

on:
  workflow_dispatch:
    inputs:
      build-status:
        type: choice
        options:
          - success
          - failure
```

```
        default: success

jobs:
  build:
    runs-on: ubuntu-latest
    outputs:
      build-status: ${{ steps.build.outputs.status }}
      output1: ${{ steps.build.outputs.output1 }}
    steps:
      - name: Print GITHUB_OUTPUT path
        run: echo "$GITHUB_OUTPUT"
      - name: Build
        id: build
        run: |
          echo "$GITHUB_OUTPUT"
          echo "status=${{ inputs.build-status }}" >> "$GITHUB_OUTPUT"
          echo "output1=value1" >> "$GITHUB_OUTPUT"
          echo "output2=value2" >> "$GITHUB_OUTPUT"
          cat "$GITHUB_OUTPUT"
      - name: Step with mistake
        run: |
          echo "mistake=true" > "$GITHUB_OUTPUT"
          cat "$GITHUB_OUTPUT"
  deploy:
    runs-on: ubuntu-latest
    needs: build
    if: ${{ needs.build.outputs.build-status == 'success' }}
    steps:
      - name: Deploy
        run: echo "Deploying"
      - name: Print Outputs
        run: |
          echo "Output 1: ${{ needs.build.outputs.output1 }}"
```

- 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 accepts an input parameter named `build-status`, which is a choice between "success" and "failure" with a default value of "success".
- The `build` job runs on an Ubuntu latest runner and is responsible for building the project.
- The `build` job outputs two variables: `build-status` and `output1`.
- The `deploy` job runs on an Ubuntu latest runner and is conditional on the successful completion of the `build` job (`needs: build`).
- The `deploy` job checks the `build-status` output from the `build` job and only executes if the status is "success".
- The `deploy` job then prints the value of the `output1` variable from the `build` job.

3. Testing the Workflow:

- Commit and push the workflow file (`14-02-Outputs.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 `14-02-Outputs` workflow.
- Monitor the workflow run and verify that the `build` job completes successfully and outputs the `build-status` and `output1` variables.
- Ensure that the `deploy` job is triggered after the `build` job and that it correctly checks the `build-status` output.
- If the `build` job status is "success", verify that the `deploy` job prints the correct value of the `output1` variable.

4. Observing Output:

- Review the logs of each job and step to ensure that the workflow behaves as expected.
- Verify that the `deploy` job is only executed when the `build-status` output is "success".
- Confirm that the value of the `output1` variable in the `deploy` job matches the output value from the `build` job.