Enabling Direct Data Sharing Between GitHub Actions Jobs Using a Self-Hosted Runner with Persistent Volume

Problem Statement

In GitHub Actions and many other CI/CD pipeline solutions, workflows often consist of multiple jobs that need to share data or variables with each other. However, each job typically runs in its own isolated environment (such as a separate container or virtual machine), which prevents direct sharing of data between jobs.

The current standard approach to overcome this limitation is to use artifacts: one job uploads data as an artifact, and subsequent jobs download it. While effective, this method introduces additional configuration overhead for each job, increases pipeline complexity, and can significantly slow down execution due to artifact upload/download steps.

Proposed Solution

To address this challenge, a self-hosted runner can be used with a persistent volume attached to the pipeline. This shared volume remains accessible throughout the workflow execution, allowing jobs to read from and write to the same storage location. By enabling direct data sharing between jobs, this approach eliminates the need for artifact upload/download steps, reducing configuration complexity and improving pipeline performance and efficiency.

Implementation Steps

1. Provision a Self-Hosted Runner

- Choose a Linux VM (AWS EC2, GCP, or on-prem).
- Install GitHub Actions runner:
```
mkdir actions-runner && cd actions-runner
curl -o actions-runner-linux-x64-2.313.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.313.0/actions-
runner-linux-x64-2.313.0.tar.gz
tar xzf ./actions-runner-linux-x64-2.313.0.tar.gz
./config.sh --url https://github.com/your-org/your-repo --token
YOUR_TOKEN
./run.sh
```

2. Attach a Persistent Volume

- Option 1: Use a **Local Volume** (e.g., `/mnt/shared` on the runner).
- Option 2: Mount a **NFS Share** to `/mnt/shared`.
- Ensure read/write access for the `runner` user.

3. Update GitHub Workflow to Use Self-Hosted Runner

Example `.github/workflows/pipeline.yml`:

```yaml
jobs:
  job1:
    runs-on: self-hosted
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Generate data
        run: |
          echo "Hello from Job1" > /mnt/shared/data.txt

  job2:
    runs-on: self-hosted
    needs: job1
    steps:
      - name: Read data
        run: |
          cat /mnt/shared/data.txt
```

4. Ensure Shared Path is Writable

- Validate both jobs run on the **same runner** or share the same mounted path.
- Ensure `/mnt/shared` has appropriate permissions (`chmod 777` if needed during testing).

Benefits

- **Faster pipeline execution** by avoiding artifact upload/download delays.
- **Simplified configuration** since jobs can directly access shared files or data.
- **Reduced complexity** in managing inter-job dependencies and data flow.

Downsides

- **Limited Scalability -** One runner limits concurrent job capacity and load handling.
- **Manual Maintenance -** Self-hosted runners require patching, monitoring, and lifecycle management.
- **No Built-in Isolation -** Jobs on the same runner can accidentally interfere with each other's files.
- **Persistent Volume Setup Overhead -** Extra setup needed to manage and secure shared file storage. Need NFS
- **Reduced Portability -** Workflow becomes tied to local file paths, affecting flexibility and migration.