Matrices in GitHub Actions

A matrix strategy in GitHub Actions allows you to define variations for each job by using variables. This feature is incredibly useful when you need to test your code across multiple versions of a language or on different operating systems.

About Matrix Strategies

Matrix strategies enable you to use variables in a single job definition to create multiple job runs based on combinations of those variables. For instance, you can use a matrix to test your code in multiple versions of a language (e.g., Node.js) or on various operating systems.

Using a Matrix Strategy

To create a matrix in your GitHub Actions workflow, use `jobs.<job_id>.strategy.matrix`. Within your matrix, define one or more variables followed by an array of values. For example:

```
jobs:
  example_matrix:
    strategy:
      matrix:
        version: ['11', '14', '17']
        os: [ubuntu-latest, windows-latest]
```

In this example, your workflow will run six jobs, one for each combination of the version and os variables.

The variables defined in your matrix become properties in the `matrix` context. You can reference these properties in other areas of your workflow file. For instance, you can use `matrix.version` and `matrix.os` to access the current values of version and os that the job is using.

Adding or Removing Matrix Configurations

You can expand existing matrix configurations or add new ones using `jobs.<job_id>.strategy.matrix.include`. This allows you to include additional key-value pairs for matrix combinations without overwriting any original matrix values.

To remove specific configurations from the matrix, use `jobs.<job_id>.strategy.matrix.exclude`. Excluded configurations do not run.

Handling Failures

You can control how job failures are handled with jobs.`<job_id>.strategy.fail-fast` and `jobs.<job_id>.continue-on-error`. These settings enable you to decide whether a job failure should affect the entire matrix and whether specific jobs should continue running after a failure.

Testing Matrices in Workflow

1. Create Workflow File:

- Create a new file named `18-01-Matrices.yml` in the `.github/workflows` directory of your repository.

```yaml
# Workflow for Testing applications on different environments in a single workflow

name: 18-01-Matrices

on:
  workflow_dispatch

jobs:
  backwards-compatibility:
    name: ${{ matrix.os }}-${{ matrix.java-version }}
    runs-on: ${{ matrix.os }}
    strategy:
      matrix:
        java-version: [11,14,17]
        os:
          - ubuntu-latest
          - windows-latest

    steps:
      - name: Setup Java
        uses: actions/setup-java@v4
        with:
          distribution: 'adopt'
          java-version: ${{ matrix.java-version }}
      - name: Perform some tests
        run: echo "Running tests on OS ${{ matrix.os }} and Java ${{ matrix.java-version }}"
```

- 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 a job named `backwards-compatibility`.
- The job runs on two different operating systems: `ubuntu-latest` and `windows-latest`.
- It uses a matrix strategy to test the application with multiple Java versions: 11, 14, and 17.
- For each combination of OS and Java version, it sets up the corresponding Java environment and performs some tests.

3. Testing the Workflow:

- Commit and push the workflow file (`18-01-Matrices.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 `18-01-Matrices` workflow.
- Monitor the workflow run and verify that the tests run successfully on each combination of OS and Java version specified in the matrix.