GitHub Contexts

GitHub Actions provides access to a variety of GitHub contexts that allow you to access information related to the workflow, the event that triggered it, and the repository itself. This README provides an overview of different GitHub contexts, including the `github` context, `env` context, `inputs` context, and `vars` context, and how to work with them within your workflows.

GitHub contexts are predefined variables that contain information about the current workflow run, the event that triggered the workflow, and the repository where the workflow is hosted. These contexts are accessible as variables and can be used to customize and parameterize your workflows.

Common GitHub Contexts

Here are some common GitHub contexts that you can use within your workflows:

- **`github` Context**: Contains various properties related to the workflow run, event, and repository. For example, `github.event_name`, `github.ref`, and `github.sha`.

- **`env` Context**: Allows you to access environment variables and secrets securely. For example, `env.MY_SECRET`.

- **`inputs` Context**: Provides access to input variables defined in your workflow file. For example, `inputs.my_input`.

- **`vars` Context**: Provides a way to define and access workflow-level variables. For example, `vars.my_var`.

Testing GitHub Context

Github Context

1. Create Workflow File:

- Create a new file named `08-01-Contexts-GitHub.yml` in the `.github/workflows` directory of your repository.

```
name: 08-01-Contexts - github

on:
  workflow_dispatch:
```

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: Display Event Information
      run: |
        echo "Event Name: ${{ github.event_name }}"
        echo "Ref: ${{ github.ref }}"
        echo "SHA: ${{ github.sha }}"
        echo "Actor: ${{ github.actor }}"
        echo "Workflow: ${{ github.workflow }}"
        echo "Run ID: ${{ github.run_id }}"
        echo "Run number: ${{ github.run_number }}"
```

- 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 contains a single job named `build` that runs on the `ubuntu-latest` environment.
- The job includes a step named `Display Event Information` which runs a shell script to echo various GitHub context variables such as event name, ref, SHA, actor, workflow, run ID, and run number.

3. Testing the Workflow:

- Commit and push the workflow file (`08-01-Contexts-GitHub.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 `08-01-Contexts-GitHub` workflow.
- Once the workflow run completes, click on the job to view its details.
- Check the logs of the `Display Event Information` step to verify that the GitHub context variables are correctly displayed.

4. Reviewing Workflow Output:

- Review the output of each echoed GitHub context variable to ensure that it provides the expected information.
- Confirm that the event name, ref, SHA, actor, workflow, run ID, and run number are accurately displayed based on the triggering event and workflow execution.

ENV Context

1. Create Workflow File:

- Create a new file named `08-02-Contexts-Env.yml` in the `.github/workflows` directory of your repository.

```yaml
name: 08-02-Contexts - env
```

```yaml
on:
  workflow_dispatch:

env:
  MY_WORKFLOW_VAR: 'workflow'
  MY_OVERWRITTEN_VAR: 'workflow'

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Print Env Variables
        env:
          MY_OVERWRITTEN_VAR: 'step'
        run: |
          echo "Workflow env: ${{ env.MY_WORKFLOW_VAR }}"
          echo "Overwritten env: ${{ env.MY_OVERWRITTEN_VAR }}"
      - name: Print Env Variables
        run: |
          echo "Workflow env: ${{ env.MY_WORKFLOW_VAR }}"
          echo "Overwritten env: ${{ env.MY_OVERWRITTEN_VAR }}"
```

- 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 environment variables using the `env` key. Two variables, `MY_WORKFLOW_VAR` and `MY_OVERWRITTEN_VAR`, are set at the workflow level.
- The `build` job runs on the `ubuntu-latest` environment and consists of two steps.
- The first step (`Print Env Variables`) overrides the value of `MY_OVERWRITTEN_VAR` at the step level and prints both workflow and overwritten environment variables.
- The second step (`Print Env Variables`) prints the values of the environment variables without overriding them at the step level.

3. Testing the Workflow:

- Commit and push the workflow file (`08-02-Contexts-Env.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 `08-02-Contexts-Env` workflow.
- Once the workflow run completes, click on the job to view its details.
- Check the logs of both steps to verify that the environment variables are correctly printed with their respective values.

4. Reviewing Workflow Output:

- Confirm that the values of `MY_WORKFLOW_VAR` and `MY_OVERWRITTEN_VAR` are displayed as expected in the workflow logs.
- Verify that the overwritten value of `MY_OVERWRITTEN_VAR` in the first step reflects the step-level override, while the value in the second step remains unchanged from the workflow-level definition.

Inputs Context

1. Create Workflow File:

- Create a new file named `08-03-Contexts-Inputs.yml` in the `.github/workflows` directory of your repository.

```
name: 08-03-Contexts - Inputs

on:
  workflow_dispatch:
    inputs:
      debug:
        type: boolean
        default: false

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Use Workflow Input
        run: |
          echo "DEBUG: ${{ inputs.debug }}"
```

- 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 workflow input named `debug` of type `boolean` with a default value of `false`.
- The `build` job runs on the `ubuntu-latest` environment and consists of one step.
- The step (`Use Workflow Input`) simply prints the value of the debug input.

3. Testing the Workflow:

- Commit and push the workflow file (`08-03-Contexts-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 `08-03-Contexts-Inputs` workflow.
- When prompted, provide input values for the workflow. For the `debug` input, you can choose to either enable or disable debugging.
- Once the workflow run completes, click on the job to view its details.
- Check the logs of the step to verify that the input value is correctly printed.

4. Reviewing Workflow Output:

- Confirm that the value of the `debug` input matches the input provided during workflow execution.
- Verify that the step output reflects the debugging status based on the input value provided.

Output Context

1. **Create Workflow File:**

   - Create a new file named `08-04-Contexts-Outputs.yml` in the `.github/workflows` directory of your repository.

   ```yaml
   name: 08-04-Contexts - Outputs

   on:
     workflow_dispatch:

   jobs:
     build:
       runs-on: ubuntu-latest
       steps:
         - name: Define Workflow Variable
           id: set_var
           run: |
             echo "Setting Workflow Variable"
             echo "name=Max" >> "$GITHUB_OUTPUT"
         - name: Use Workflow Variable
           run: |
             echo "Workflow Variable Value: ${{ steps.set_var.outputs.name }}"
   ```

   - 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`).
   - The `build` job runs on the `ubuntu-latest` environment and consists of two steps.
   - The first step (`Define Workflow Variable`) defines a workflow output named `name` with the value Max.
   - The second step (`Use Workflow Variable`) retrieves and uses the value of the `name` output from the previous step.

3. **Testing the Workflow:**

   - Commit and push the workflow file (`08-04-Contexts-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 `08-04-Contexts-Outputs` workflow.
   - Once the workflow run completes, click on the job to view its details.
   - Check the logs of the second step (`Use Workflow Variable`) to verify that the workflow output value is correctly retrieved and displayed.

4. **Reviewing Workflow Output:**

   - Confirm that the workflow output value (`Max`) is correctly printed in the logs of the `Use Workflow Variable` step.