Steps to build Databricks assets bundling with GitHub actions

To create a GitHub Actions workflow for building Databricks assets, you can use the following YAML configuration. This example assumes you have a Databricks CLI configured and you want to deploy notebooks or other assets to your Databricks workspace.

name: Build and Deploy Databricks Assets

on:

  push:

    branches:

      - main

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

    - name: Checkout repository

      uses: actions/checkout@v2

    - name: Set up Python

      uses: actions/setup-python@v2

      with:

        python-version: '3.8'

    - name: Install Databricks CLI

      run: pip install databricks-cli

    - name: Configure Databricks CLI

      env:

        DATABRICKS\_HOST: ${{ secrets.DATABRICKS\_HOST }}

        DATABRICKS\_TOKEN: ${{ secrets.DATABRICKS\_TOKEN }}

      run: |

        databricks configure --token <<EOF

        $DATABRICKS\_HOST

        $DATABRICKS\_TOKEN

        EOF

    - name: Deploy Notebooks

      run: |

        databricks workspace import\_dir /local/path/to/notebooks /databricks/path/to/notebooks -o

**Explanation:**

1. **Trigger**: The workflow triggers on a push to the main branch.
2. **Jobs**: The build job runs on the latest Ubuntu runner.
3. **Steps**:
   * **Checkout repository**: Uses the actions/checkout action to pull the repository code.
   * **Set up Python**: Uses the actions/setup-python action to set up Python 3.8.
   * **Install Databricks CLI**: Installs the Databricks CLI using pip.
   * **Configure Databricks CLI**: Configures the Databricks CLI using secrets stored in GitHub.
   * **Deploy Notebooks**: Uses the Databricks CLI to import notebooks from a local directory to the Databricks workspace.

Make sure to add DATABRICKS\_HOST and DATABRICKS\_TOKEN to your GitHub repository secrets.

What assets should I include?

/databricks

/notebooks

notebook1.py

notebook2.py

/libraries

library1.py

library2.py

/jobs

job1.json

job2.json

/clusters

cluster1.json

cluster2.json

/dbfs

file1.csv

file2.csv

You can then modify the GitHub Actions workflow to deploy these assets to your Databricks workspace. Here is an updated example:

name: Build and Deploy Databricks Assets

on:

  push:

    branches:

      - main

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

    - name: Checkout repository

      uses: actions/checkout@v2

    - name: Set up Python

      uses: actions/setup-python@v2

      with:

        python-version: '3.8'

    - name: Install Databricks CLI

      run: pip install databricks-cli

    - name: Configure Databricks CLI

      env:

        DATABRICKS\_HOST: ${{ secrets.DATABRICKS\_HOST }}

        DATABRICKS\_TOKEN: ${{ secrets.DATABRICKS\_TOKEN }}

      run: |

        databricks configure --token <<EOF

        $DATABRICKS\_HOST

        $DATABRICKS\_TOKEN

        EOF

    - name: Deploy Notebooks

      run: |

        databricks workspace import\_dir databricks/notebooks /Workspace/Notebooks -o

    - name: Deploy Libraries

      run: |

        databricks fs cp --recursive databricks/libraries dbfs:/FileStore/libraries

    - name: Deploy Jobs

      run: |

        for job in databricks/jobs/\*.json; do

          databricks jobs create --json-file "$job"

        done

    - name: Deploy Clusters

      run: |

        for cluster in databricks/clusters/\*.json; do

          databricks clusters create --json-file "$cluster"

        done

    - name: Deploy DBFS Files

      run: |

        databricks fs cp --recursive databricks/dbfs dbfs:/FileStore/dbfs

**Explanation:**

1. **Deploy Notebooks**: Imports notebooks from the local databricks/notebooks directory to the Databricks workspace.
2. **Deploy Libraries**: Copies custom libraries to DBFS.
3. **Deploy Jobs**: Creates jobs using JSON configurations from the databricks/jobs directory.
4. **Deploy Clusters**: Creates clusters using JSON configurations from the databricks/clusters directory.
5. **Deploy DBFS Files**: Copies files to DBFS.

name: Build and Deploy Databricks Assets

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout repository

uses: actions/checkout@v2

- name: Set up Python

uses: actions/setup-python@v2

with:

python-version: '3.8'

- name: Install Databricks CLI

run: pip install databricks-cli

- name: Configure Databricks CLI

env:

DATABRICKS\_HOST: ${{ secrets.DATABRICKS\_HOST }}

DATABRICKS\_TOKEN: ${{ secrets.DATABRICKS\_TOKEN }}

run: |

databricks configure --token <<EOF

$DATABRICKS\_HOST

$DATABRICKS\_TOKEN

EOF

- name: Deploy Notebooks

run: |

databricks workspace import\_dir databricks/notebooks /Workspace/Notebooks -o

- name: Deploy Libraries

run: |

databricks fs cp --recursive databricks/libraries dbfs:/FileStore/libraries

- name: Deploy Jobs

run: |

for job in databricks/jobs/\*.json; do

databricks jobs create --json-file "$job"

done

- name: Deploy Clusters

run: |

for cluster in databricks/clusters/\*.json; do

databricks clusters create --json-file "$cluster"

done

- name: Deploy DBFS Files

run: |

databricks fs cp --recursive databricks/dbfs dbfs:/FileStore/dbfs

how toto add DATABRICKS\_HOST and DATABRICKS\_TOKEN to your GitHub repository secrets.

To add DATABRICKS\_HOST and DATABRICKS\_TOKEN to your GitHub repository secrets, follow these steps:

1. **Navigate to your repository on GitHub**:
   * Go to the main page of your repository.
2. **Go to the Settings**:
   * Click on the Settings tab at the top of the repository page.
3. **Access Secrets**:
   * In the left sidebar, click on Secrets and variables and then Actions.
4. **Add a new secret**:
   * Click the New repository secret button.
5. **Add DATABRICKS\_HOST**:
   * In the Name field, enter DATABRICKS\_HOST.
   * In the Value field, enter your Databricks host URL (e.g., https://<databricks-instance>.<region>.databricks.com).
   * Click the Add secret button.
6. **Add DATABRICKS\_TOKEN**:
   * Click the New repository secret button again.
   * In the Name field, enter DATABRICKS\_TOKEN.
   * In the Value field, enter your Databricks personal access token.
   * Click the Add secret button.

After adding these secrets, they will be available for use in your GitHub Actions workflows.

How to add in yaml file

To use the DATABRICKS\_HOST and DATABRICKS\_TOKEN secrets in your GitHub Actions YAML file, you need to reference them using the secrets context. Here is an example of how to include these secrets in your YAML file:

name: Build and Deploy Databricks Assets

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout repository

uses: actions/checkout@v2

- name: Set up Python

uses: actions/setup-python@v2

with:

python-version: '3.8'

- name: Install Databricks CLI

run: pip install databricks-cli

- name: Configure Databricks CLI

env:

DATABRICKS\_HOST: ${{ secrets.DATABRICKS\_HOST }}

DATABRICKS\_TOKEN: ${{ secrets.DATABRICKS\_TOKEN }}

run: |

databricks configure --token <<EOF

$DATABRICKS\_HOST

$DATABRICKS\_TOKEN

EOF

- name: Deploy Notebooks

run: |

databricks workspace import\_dir databricks/notebooks /Workspace/Notebooks -o

- name: Deploy Libraries

run: |

databricks fs cp --recursive databricks/libraries dbfs:/FileStore/libraries

- name: Deploy Jobs

run: |

for job in databricks/jobs/\*.json; do

databricks jobs create --json-file "$job"

done

- name: Deploy Clusters

run: |

for cluster in databricks/clusters/\*.json; do

databricks clusters create --json-file "$cluster"

done

- name: Deploy DBFS Files

run: |

databricks fs cp --recursive databricks/dbfs dbfs:/FileStore/dbfs

**Explanation:**

* **Secrets Reference**: The DATABRICKS\_HOST and DATABRICKS\_TOKEN secrets are referenced using ${{ secrets.DATABRICKS\_HOST }} and ${{ secrets.DATABRICKS\_TOKEN }} respectively.
* **Environment Variables**: These secrets are set as environment variables for the Configure Databricks CLI step, allowing the Databricks CLI to use them for authentication.

How to add Environment Variables: These secrets are set as environment variables for the Configure Databricks CLI step, allowing the Databricks CLI to use them for authentication.

To set environment variables using secrets in the GitHub Actions YAML file, you can use the env keyword within the specific step. Here is an example of how to configure the Databricks CLI step with environment variables for authentication:

name: Build and Deploy Databricks Assets

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout repository

uses: actions/checkout@v2

- name: Set up Python

uses: actions/setup-python@v2

with:

python-version: '3.8'

- name: Install Databricks CLI

run: pip install databricks-cli

- name: Configure Databricks CLI

env:

DATABRICKS\_HOST: ${{ secrets.DATABRICKS\_HOST }}

DATABRICKS\_TOKEN: ${{ secrets.DATABRICKS\_TOKEN }}

run: |

databricks configure --token <<EOF

$DATABRICKS\_HOST

$DATABRICKS\_TOKEN

EOF

- name: Deploy Notebooks

run: |

databricks workspace import\_dir databricks/notebooks /Workspace/Notebooks -o

- name: Deploy Libraries

run: |

databricks fs cp --recursive databricks/libraries dbfs:/FileStore/libraries

- name: Deploy Jobs

run: |

for job in databricks/jobs/\*.json; do

databricks jobs create --json-file "$job"

done

- name: Deploy Clusters

run: |

for cluster in databricks/clusters/\*.json; do

databricks clusters create --json-file "$cluster"

done

- name: Deploy DBFS Files

run: |

databricks fs cp --recursive databricks/dbfs dbfs:/FileStore/dbfs

**Explanation:**

* **Environment Variables**: The env keyword is used within the Configure Databricks CLI step to set DATABRICKS\_HOST and DATABRICKS\_TOKEN as environment variables.
* **Secrets Reference**: The secrets are referenced using ${{ secrets.DATABRICKS\_HOST }} and ${{ secrets.DATABRICKS\_TOKEN }} to securely pass the values to the environment variables.

How to test the yaml file in local machine

To test a GitHub Actions YAML file on a local Windows machine, you can use the act tool. Here are the steps to set it up and run your workflow locally:

1. **Install Docker**:
   * Ensure Docker is installed and running on your Windows machine. You can download Docker Desktop from [here](vscode-file://vscode-app/c:/Users/gujji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).
2. **Install act**:
   * Download the act binary for Windows from the [act GitHub repository](vscode-file://vscode-app/c:/Users/gujji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html).
   * Extract the downloaded ZIP file and add the act executable to your system's PATH.
3. **Prepare your repository**:
   * Ensure your repository has a .github/workflows directory with your YAML file inside it.
4. **Create a .secrets file** (optional):
   * If your workflow requires secrets, create a .secrets file in the root of your repository with the necessary secrets.

Example .secrets file:

DATABRICKS\_HOST=https://<databricks-instance>.<region>.databricks.com

DATABRICKS\_TOKEN=your-databricks-token

1. **Run act**:
   * Open a Command Prompt or PowerShell window.
   * Navigate to your repository's root directory.
   * Run act to execute the workflow. You can specify the event to trigger, such as push or pull\_request.

act push

* + If you created a .secrets file, act will automatically use it. Alternatively, you can pass secrets directly in the command.

act -s DATABRICKS\_HOST=https://<databricks-instance>.<region>.databricks.com -s DATABRICKS\_TOKEN=your-databricks-token

**Example command with secrets:**

act -s DATABRICKS\_HOST=https://<databricks-instance>.<region>.databricks.com -s DATABRICKS\_TOKEN=your-databricks-token

1 vulnerability

**Explanation:**

* **Docker**: Required to run the workflows in isolated containers.
* **act**: A tool to run GitHub Actions locally.
* **Secrets**: Provide necessary secrets either through a .secrets file or directly in the command.

By following these steps, you can test your GitHub Actions YAML file locally on your Windows machine before pushing it to your repository.

How to test the yaml file in githuib repository

To test a GitHub Actions YAML file in your GitHub repository, you need to push the YAML file to the repository and trigger the workflow. Here are the steps to do this:

1. **Create or Update the Workflow YAML File**:
   * Ensure your workflow YAML file is located in the .github/workflows directory of your repository.
   * For example, create a file named deploy-databricks.yml in the .github/workflows directory.
2. **Add the Workflow YAML File to Your Repository**:
   * If you haven't already, add the workflow YAML file to your repository.

mkdir -p .github/workflows

touch .github/workflows/deploy-databricks.yml

* + Copy your workflow YAML content into this file.

1. **Commit and Push the Changes**:
   * Commit the new or updated workflow file to your repository and push the changes to GitHub.

git add .github/workflows/deploy-databricks.yml

git commit -m "Add GitHub Actions workflow for Databricks deployment"

git push origin main

1. **Trigger the Workflow**:
   * The workflow will automatically trigger based on the events defined in the YAML file (e.g., push to the main branch).
2. **Check the Workflow Run**:
   * Go to your repository on GitHub.
   * Click on the Actions tab to see the list of workflow runs.
   * Click on the latest workflow run to see the details and logs.

**Example Workflow YAML File (deploy-databricks.yml):**

name: Build and Deploy Databricks Assets

on:

  push:

    branches:

      - main

jobs:

  build:

    runs-on: ubuntu-latest

    steps:

    - name: Checkout repository

      uses: actions/checkout@v2

    - name: Set up Python

      uses: actions/setup-python@v2

      with:

        python-version: '3.8'

    - name: Install Databricks CLI

      run: pip install databricks-cli

    - name: Configure Databricks CLI

      env:

        DATABRICKS\_HOST: ${{ secrets.DATABRICKS\_HOST }}

        DATABRICKS\_TOKEN: ${{ secrets.DATABRICKS\_TOKEN }}

      run: |

        databricks configure --token <<EOF

        $DATABRICKS\_HOST

        $DATABRICKS\_TOKEN

        EOF

    - name: Deploy Notebooks

      run: |

        databricks workspace import\_dir databricks/notebooks /Workspace/Notebooks -o

    - name: Deploy Libraries

      run: |

        databricks fs cp --recursive databricks/libraries dbfs:/FileStore/libraries

    - name: Deploy Jobs

      run: |

        for job in databricks/jobs/\*.json; do

          databricks jobs create --json-file "$job"

        done

    - name: Deploy Clusters

      run: |

        for cluster in databricks/clusters/\*.json; do

          databricks clusters create --json-file "$cluster"

        done

    - name: Deploy DBFS Files

      run: |

        databricks fs cp --recursive databricks/dbfs dbfs:/FileStore/dbfs

**Explanation:**

* **Workflow File Location**: The workflow YAML file must be in the .github/workflows directory.
* **Trigger**: The workflow triggers on a push to the main branch.
* **Actions Tab**: You can monitor the workflow runs and logs in the Actions tab of your GitHub repository.

By following these steps, you can test and run your GitHub Actions workflow directly in your GitHub repository.

Similar code found with 1 license type -

View matches