Configuring OpenID Connect in cloud providers

In this article

Use OpenID Connect within your workflows to authenticate with cloud providers.

Overview

OpenID Connect (OIDC) allows your GitHub Actions workflows to access resources in your cloud provider, without having to store any credentials as long-lived GitHub secrets.

To use OIDC, you will first need to configure your cloud provider to trust GitHub's OIDC as a federated identity, and must then update your workflows to authenticate using tokens.

Prerequisites

- To learn the basic concepts of how GitHub uses OpenID Connect (OIDC), and its architecture and benefits, see "About security hardening with OpenID Connect."
- Before proceeding, you must plan your security strategy to ensure that access tokens are only allocated in a predictable way. To control how your cloud provider issues access tokens, you **must** define at least one condition, so that untrusted repositories can’t request access tokens for your cloud resources. For more information, see "About security hardening with OpenID Connect."

Updating your GitHub Actions workflow

To update your workflows for OIDC, you will need to make two changes to your YAML:

1. Add permissions settings for the token.
2. Use the official action from your cloud provider to exchange the OIDC token (JWT) for a cloud access token.

If your cloud provider doesn't yet offer an official action, you can update your workflows to perform these steps manually.

Adding permissions settings

The job or workflow run requires a `permissions` setting with `id-token: write`. You won't be able to request the OIDC JWT ID token if the `permissions` setting for `id-token` is set to `read` or `none`.

The `id-token: write` setting allows the JWT to be requested from GitHub's OIDC provider using one of these approaches:

- Using environment variables on the runner (`ACTIONS_ID_TOKEN_REQUEST_URL` and `ACTIONS_ID_TOKEN_REQUEST_TOKEN`).
- Using `getIDToken()` from the Actions toolkit.

If you need to fetch an OIDC token for a workflow, then the permission can be set at the workflow level. For example:

YAML

```yaml
permissions:
  id-token: write # This is required for requesting the JWT
  contents: read  # This is required for actions/checkout
```

If you only need to fetch an OIDC token for a single job, then this permission can be set within that job. For example:

YAML

```yaml
permissions:
  id-token: write # This is required for requesting the JWT
```

You may need to specify additional permissions here, depending on your workflow's requirements.

For reusable workflows that are owned by the same user, organization, or enterprise as the caller workflow, the OIDC token generated in the reusable workflow can be accessed from the caller's context. For reusable workflows outside your enterprise or organization, the `permissions` setting for `id-token` should be explicitly set to `write` at the caller workflow level or in the specific job that calls the reusable workflow. This ensures that the OIDC token generated in the reusable workflow is only allowed to be consumed in the caller workflows when intended.

For more information, see "Reusing workflows."

Using official actions

If your cloud provider has created an official action for using OIDC with GitHub Actions, it will allow you to easily exchange the OIDC token for an access token. You can then update your workflows to use this token when accessing cloud resources.

Using custom actions

If your cloud provider doesn't have an official action, or if you prefer to create custom scripts, you can manually request the JSON Web Token (JWT) from GitHub's OIDC provider.

If you're not using an official action, then GitHub recommends that you use the Actions core toolkit. Alternatively, you can use the following environment variables to retrieve the token: `ACTIONS_RUNTIME_TOKEN`, `ACTIONS_ID_TOKEN_REQUEST_URL`.

To update your workflows using this approach, you will need to make three changes to your YAML:

1. Add permissions settings for the token.

2. Add code that requests the OIDC token from GitHub's OIDC provider.

3. Add code that exchanges the OIDC token with your cloud provider for an access token.

Requesting the JWT using the Actions core toolkit

The following example demonstrates how to use `actions/github-script` with the `core` toolkit to request the JWT from GitHub's OIDC provider. For more information, see "Creating a JavaScript action."

```
jobs:
  job:
    environment: Production
    runs-on: ubuntu-latest
    steps:
    - name: Install OIDC Client from Core Package
      run: npm install @actions/core@1.6.0 @actions/http-client
    - name: Get Id Token
      uses: actions/github-script@v6
      id: idtoken
      with:
        script: |
          const coredemo = require('@actions/core')
          let id_token = await coredemo.getIDToken()
          coredemo.setOutput('id_token', id_token)
```

Requesting the JWT using environment variables

The following example demonstrates how to use environment variables to request a JSON Web Token.

For your deployment job, you will need to define the token settings, using `actions/github-script` with the `core` toolkit. For more information, see "Creating a JavaScript action."

For example:

```
jobs:
  job:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/github-script@v6
      id: script
      timeout-minutes: 10
      with:
        debug: true
        script: |
          const token = process.env['ACTIONS_RUNTIME_TOKEN']
          const runtimeUrl = process.env['ACTIONS_ID_TOKEN_REQUEST_URL']
          core.setOutput('TOKEN', token.trim())
          core.setOutput('IDTOKENURL', runtimeUrl.trim())
```

You can then use `curl` to retrieve a JWT from the GitHub OIDC provider. For example:

```
    - run: |
        IDTOKEN=$(curl -H "Authorization: bearer
${{steps.script.outputs.TOKEN}}" ${{steps.script.outputs.IDTOKENURL}}  -H
"Accept: application/json; api-version=2.0" -H "Content-Type: application/json" -
d "{}" | jq -r '.value')
        echo $IDTOKEN
        jwtd() {
            if [[ -x $(command -v jq) ]]; then
                jq -R 'split(".") | .[0],.[1] | @base64d | fromjson' <<< "${1}"
                echo "Signature: $(echo "${1}" | awk -F'.' '{print $3}')"
            fi
        }
        jwtd $IDTOKEN
        echo "idToken=${IDTOKEN}" >> $GITHUB_OUTPUT
      id: tokenid
```

Getting the access token from the cloud provider

You will need to present the OIDC JSON web token to your cloud provider in order to obtain an access token.

For each deployment, your workflows must use cloud login actions (or custom scripts) that fetch the OIDC token and present it to your cloud provider. The cloud provider then validates the claims in the token; if successful, it provides a cloud access token that is available only to that job run. The provided access token can then be used by subsequent actions in the job to connect to the cloud and deploy to its resources.

The steps for exchanging the OIDC token for an access token will vary for each cloud provider.

Accessing resources in your cloud provider

Once you've obtained the access token, you can use specific cloud actions or scripts to authenticate to the cloud provider and deploy to its resources. These steps could differ for each cloud provider. In addition, the default expiration time of this access token could vary between each cloud and can be configurable at the cloud provider's side.

Further reading

- Using OpenID Connect with reusable workflows
- About self-hosted runners