Configuring OpenID Connect in Google Cloud Platform

In this article

Overview
Prerequisites
Adding a Google Cloud Workload Identity Provider
Updating your GitHub Actions workflow
Further reading

Use OpenID Connect within your workflows to authenticate with Google Cloud Platform.

Note: GitHub-hosted runners are not currently supported on GitHub Enterprise Server. You can see more information about planned future support on the GitHub public roadmap.

Overview

OpenID Connect (OIDC) allows your GitHub Actions workflows to access resources in Google Cloud Platform (GCP), without needing to store the GCP credentials as long-lived GitHub secrets.

This guide gives an overview of how to configure GCP to trust GitHub's OIDC as a federated identity, and includes a workflow example for the `google-github-actions/auth` action that uses tokens to authenticate to GCP and access resources.

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."

Adding a Google Cloud Workload Identity Provider

To configure the OIDC identity provider in GCP, you will need to perform the following configuration. For instructions on making these changes, refer to the GCP documentation.

1. Create a new identity pool.
2. Configure the mapping and add conditions.

3. Connect the new pool to a service account.

Additional guidance for configuring the identity provider:

- For security hardening, make sure you've reviewed "About security hardening with OpenID Connect." For an example, see "About security hardening with OpenID Connect."
- For the service account to be available for configuration, it needs to be assigned to the `roles/iam.workloadIdentityUser` role. For more information, see the GCP documentation.
- The Issuer URL to use: `https://HOSTNAME/_services/token`

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 `google-github-actions/auth` action to exchange the OIDC token (JWT) for a cloud access token.

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."

Requesting the access token

The `google-github-actions/auth` action receives a JWT from the GitHub OIDC provider, and then requests an access token from GCP. For more information, see the GCP documentation.

This example has a job called `Get_OIDC_ID_token` that uses actions to request a list of services from GCP.

- `<example-workload-identity-provider>`: Replace this with the path to your identity provider in GCP. For example, `projects/<example-project-id>/locations/global/workloadIdentityPools/<name-of-pool/providers/<name-of-provider>`
- `<example-service-account>`: Replace this with the name of your service account in GCP.
- `<project-id>`: Replace this with the ID of your GCP project.

This action exchanges a GitHub OIDC token for a Google Cloud access token, using Workload Identity Federation.

YAML

```yaml
name: List services in GCP
on:
  pull_request:
    branches:
      - main

permissions:
  id-token: write

jobs:
  Get_OIDC_ID_token:
    runs-on: ubuntu-latest
    steps:
    - id: 'auth'
      name: 'Authenticate to GCP'
      uses: 'google-github-actions/auth@v0.3.1'
      with:
          create_credentials_file: 'true'
          workload_identity_provider: '<example-workload-identity-provider>'
          service_account: '<example-service-account>'
    - id: 'gcloud'
      name: 'gcloud'
      run: |-
        gcloud auth login --brief --cred-file="${{
steps.auth.outputs.credentials_file_path }}"
        gcloud services list
```

Further reading

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