Configuring OpenID Connect in Amazon Web Services

In this article

Use OpenID Connect within your workflows to authenticate with Amazon Web Services.

Overview

OpenID Connect (OIDC) allows your GitHub Actions workflows to access resources in Amazon Web Services (AWS), without needing to store the AWS credentials as long-lived GitHub secrets.

This guide explains how to configure AWS to trust GitHub's OIDC as a federated identity, and includes a workflow example for the `aws-actions/configure-aws-credentials` that uses tokens to authenticate to AWS 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 the identity provider to AWS

To add the GitHub OIDC provider to IAM, see the AWS documentation.

- For the provider URL: Use `https://token.actions.githubusercontent.com`
- For the "Audience": Use `sts.amazonaws.com` if you are using the official action.

Configuring the role and trust policy

To configure the role and trust in IAM, see the AWS documentation "Configure AWS Credentials for GitHub Actions" and "Configuring a role for GitHub OIDC identity provider."

> **Note**: AWS Identity and Access Management (IAM) recommends that users evaluate the IAM condition key, `token.actions.githubusercontent.com:sub`, in the trust policy of any role that trusts GitHub’s OIDC identity provider (IdP). Evaluating this condition key in the role trust policy limits which GitHub actions are able to assume the role.

Edit the trust policy, adding the `sub` field to the validation conditions. For example:

JSON

```json
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
    "token.actions.githubusercontent.com:sub": "repo:octo-org/octo-
repo:ref:refs/heads/octo-branch"
  }
}
```

If you use a workflow with an environment, the `sub` field must reference the environment name: `repo:OWNER/REPOSITORY:environment:NAME`. For more information, see "About security hardening with OpenID Connect."

JSON

```json
"Condition": {
  "StringEquals": {
    "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
    "token.actions.githubusercontent.com:sub": "repo:octo-org/octo-
repo:environment:prod"
  }
}
```

In the following example, `StringLike` is used with a wildcard operator (`*`) to allow any branch, pull request merge branch, or environment from the `octo-org/octo-repo` organization and repository to assume a role in AWS.

JSON

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::123456123456:oidc-
provider/token.actions.githubusercontent.com"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringLike": {
                    "token.actions.githubusercontent.com:sub": "repo:octo-
org/octo-repo:*"
                },
                "StringEquals": {
                    "token.actions.githubusercontent.com:aud":
"sts.amazonaws.com"
                }
            }
        }
    ]
}
```

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 `aws-actions/configure-aws-credentials` 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 `aws-actions/configure-aws-credentials` action receives a JWT from the GitHub OIDC provider, and then requests an access token from AWS. For more information, see the AWS documentation.

- `<example-bucket-name>`: Add the name of your S3 bucket here.
- `<role-to-assume>`: Replace the example with your AWS role.
- `<example-aws-region>`: Add the name of your AWS region here.

YAML

```yaml
# Sample workflow to access AWS resources when workflow is tied to branch
# The workflow Creates static website using aws s3
name: AWS example workflow
on:
  push
env:
  BUCKET_NAME : "<example-bucket-name>"
  AWS_REGION : "<example-aws-region>"
# permission can be added at job level or workflow level
permissions:
  id-token: write   # This is required for requesting the JWT
  contents: read    # This is required for actions/checkout
jobs:
  S3PackageUpload:
    runs-on: ubuntu-latest
    steps:
      - name: Git clone the repository
        uses: actions/checkout@v4
      - name: configure aws credentials
        uses: aws-actions/configure-aws-credentials@v3
        with:
          role-to-assume: arn:aws:iam::1234567890:role/example-role
          role-session-name: samplerolesession
          aws-region: ${{ env.AWS_REGION }}
      # Upload a file to AWS s3
      - name:  Copy index.html to s3
        run: |
          aws s3 cp ./index.html s3://${{ env.BUCKET_NAME }}/
```

Further reading

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