Configure Workload Identity Federation with deployment pipelines

This guide describes how to use Workload Identity Federation to let deployment pipelines authenticate to Google Cloud.

Depending on the CI/CD system you're using, your deployment pipelines might have access to ambient, environment-specific credentials. For example:

- Azure DevOps pipelines can use a Microsoft Entra workload identity federation service connection (https://learn.microsoft.com/en-au/azure/devops/pipelines/library/connect-to-azure?view=azure-devops#create-an-azure-resource-manager-service-connection-that-uses-workload-identity-federation) to obtain an ID token that uniquely identifies the Azure DevOps project.
- GitHub Actions workflows can obtain a GitHub OIDC token (https://docs.github.com/en/actions/deployment/security-hardening-your-deployments/about-security-hardening-with-openid-connect) that uniquely identifies the workflow and its repository.
- GitLab SaaS lets CI/CD jobs access an ID token (https://docs.gitlab.com/ee/ci/secrets/id_token_authentication.html) that uniquely identifies the job and its project, environment, and repository.
- Terraform Cloud can provide an OIDC token (https://discuss.hashicorp.com/t/tfe-release-v202208-1-647/43164#httpswwwterraformioenterprisereleases2022v202208-1featuresfeatures-2) to your your Terraform configuration that uniquely identifies the workspace and environment.

You can configure your deployment pipelines to use these credentials to authenticate to Google Cloud by using Workload Identity Federation. This approach eliminates the maintenance and security burden associated with service account keys (/iam/docs/service-account-creds#key-types).

Before you begin

Set up authentication

Select the tab for how you plan to use the samples on this page:

Console (#console) gcloud (#gcloud) Python (#python)

When you use the Google Cloud console to access Google Cloud services and APIs, you don't need to set up authentication.

Required roles

To get the permissions that you need to configure Workload Identity Federation, ask your administrator to grant you the Workload Identity Pool Admin (https://cloud.google.com/iam/docs/understanding-roles#iam.workloadIdentityPoolAdmin) (`roles/iam.workloadIdentityPoolAdmin`) IAM role on the project. For more information about granting roles, see Manage access to projects, folders, and organizations (/iam/docs/granting-changing-revoking-access).

You might also be able to get the required permissions through custom roles (/iam/docs/creating-custom-roles) or other predefined roles (/iam/docs/understanding-roles).

Alternatively, the IAM Owner (`roles/owner`) basic role also includes permissions to configure identity federation. You should not grant basic roles in a production environment, but you can grant them in a development or test environment.

Prepare your external IdP

Azure DevOps (#azure-devops) GitHub Actions ... GitLab SaaS (#gitlab-saas) Terraform Cloud ...

To let an Azure DevOps pipeline authenticate to Google Cloud, you first configure a service connection for Azure Resource Manager. This connection lets the pipeline obtain an ID token, which it can then exchange for Google Cloud credentials.

To create a service connection for Azure Resource Manager, do the following:

1. In Azure DevOps, open your project and go to **Project Settings**.
2. Go to **Pipelines > Service connections**.
3. Click **Create service connection**.
4. Select **Azure Resource Manager**.
5. Click **Next**.
6. Select **Workload Identity federation (automatic)**
7. Click **Next**.
8. Configure the following settings:
   - **Scope level**: Select a subscription.

     You must select a subscription even if you're not planning to use the service connection to access Azure resources.
   - **Service connection name**: Enter a name such as `google-cloud`.
9. Click **Save**.

In a later step, you'll need the issuer and subject identifier of the service connection. To look up these details, do the following:

1. Click the service connection you just created.
2. Click **Manage Service Principal**.
3. Go to **Certificate & secrets > Federated credentials**.
4. Click the federated credential.
5. On the **Edit a credential** page, find the following identifiers:
   - **Issuer**: uniquely identifies your Azure DevOps organization
   - **Subject identifier**: uniquely identifies the service connection

Azure DevOps automatically grants access on the subscription that you selected as scope to the service principal associated with your new service connection. Because you're not planning to use the service connection to access Azure resources, you can revoke this access by doing the following:

1. In the Azure portal, open the subscription that you selected as scope.
2. Go to **Access control (IAM) > Role assignments**.
3. Find the role assignment for the service connection and remove it.

Configure Workload Identity Federation

You must perform these steps for each GitHub organization, GitLab group, or Terraform Cloud organization.

To start configuring Workload Identity Federation, do the following:

1. In the Google Cloud console, on the project selector page, select or create a Google Cloud project.

   Go to project selector (https://console.cloud.google.com/projectselector2/home/dashboard)

   It's best to use a dedicated project to manage workload identity pools and providers (/iam/docs/best-practices-for-using-workload-identity-federation#dedicated-project).

2. Make sure that billing is enabled for your Google Cloud project (/billing/docs/how-to/verify-billing-enabled#console).

   Enable the IAM, Resource Manager, Service Account Credentials, and Security Token Service APIs.

   Enable the APIs (https://console.cloud.google.com/flows/enableapi?apiid=iam.googleapis.com,cloudresourcemanager.googleapis.com,iamcredentials.googleapis.com,sts.googleapis.com&redirect=https://console.cloud.google.com)

Define an attribute mapping

The environment-specific credentials of your deployment pipeline can contain multiple attributes, and you must decide which attribute you want to use as subject identifier (`google.subject`) in Google Cloud.

Optionally, you can map additional attributes (/iam/docs/workload-identity-federation#mapping). You can then refer to these additional attributes when you grant access to resources.

Azure DevOps (#azure-devops) GitHub Actions ... GitLab SaaS (#gitlab-saas) Terraform Cloud ...

The Azure DevOps ID token includes a `sub` claim that contains the subject identifier of your service connection. The subject identifier uses the following format:

```
sc://ORGANIZATION/PROJECT/CONNECTION
```

Use the following attribute mapping to map this identifier to `google.subject`:

```
google.subject=assertion.sub
```

Define an attribute condition

Attribute conditions (/iam/docs/workload-identity-federation#conditions) are CEL expressions that can check assertion attributes and target attributes. If the attribute condition evaluates to `true` for a given credential, the credential is accepted. Otherwise, the credential is rejected. You must have an attribute mapping for all attribute condition fields.

Warning: GitHub, GitLab SaaS, and Terraform Cloud use a single issuer URL across all organizations and some of the claims embedded in OIDC tokens might not be unique to your organization. To help protect against spoofing threats (/iam/docs/best-practices-for-using-workload-identity-federation#protecting_against_spoofing_threats), you must use an attribute condition that restricts access to tokens issued by your GitHub organization, GitLab group, or Terraform Cloud organization.

Azure DevOps (#azure-devops) GitHub Actions ... GitLab SaaS (#gitlab-saas) Terraform Cloud ...

Optionally, use an attribute condition to restrict access to certain service connections. For example, the following condition limits access to connections in a certain Azure DevOps project:

```
assertion.sub.startsWith('sc://ORGANIZATION/PROJECT/')
```

Replace the following:

- *`ORGANIZATION`*: the name of your Azure DevOps organization.
- *`PROJECT`*: the name of your Azure DevOps project.

Create the workload identity pool and provider

You've now collected all the information you need to create a workload identity pool and provider:

Console (#console) gcloud (#gcloud)

1. In the Google Cloud console, go to the **New workload provider and pool** page.

   Go to New workload provider and pool (https://console.cloud.google.com/iam-admin/workload-identity-pools/create)

2. Under **Create an identity pool**, enter the following:
   - **Name**: Name for the pool. The name is also used as the pool ID. You can't change the pool ID later.
   - **Description**: Text that describes the purpose of the pool.
3. Click **Continue**.
4. Configure provider settings:

   Azure DevOps (#azure-devops) GitHub Actions ... GitLab SaaS (#gitlab-saas) Terraform Cloud ...

   - **Select a provider**: **OpenID Connect (OIDC)**.
   - **Provider name**: the name of the Azure DevOps project, or a custom name.
   - **Provider ID**: the name of the Azure DevOps project, or a custom ID. You cannot change the provider ID later.
   - **Issuer URL**: the service connection issuer that you've looked up previously (#prepare).
   - **Audiences**: Select **Allowed audiences** and paste the following value

   ```
   api://AzureADTokenExchange
   ```
5. Click **Continue**.

6. Under **Configure provider attributes**, add the attribute mappings that you've identified previously (#mappings-and-conditions).
7. Under **Attribute conditions**, enter the attribute condition that you've identified previously (#mappings-and-conditions).
8. Click **Save** to create the workload identity pool and provider.

Note: The prefix `gcp-` is reserved and can't be used in a pool or provider ID.

Update attribute condition on a workload identity provider

This section describes how you can update the attribute condition on an existing workload identity pool provider to restrict access to tokens issued by your GitHub organization, GitLab group, or Terraform Cloud organization.

To find the recommended attribute condition for your pipeline, see Define an attribute condition (#conditions).

Consolegcloud (#gcloud)
(#console)

1. In the Google Cloud console, go to the **Workload Identity Pools** page.

   Go to Workload Identity Pools (https://console.cloud.google.com/iam-admin/workload-identity-pools)

2. Find the workload identity pool that contains the provider, and then click the **Expand node** icon for the pool.
3. Find the workload identity pool provider that you want to edit and click **Edit**.
4. In **Attribute conditions**, enter the attribute condition that you've identified previously (#mappings-and-conditions).
5. To update the workload identity pool and provider, click **Save**.

Authenticate a deployment pipeline

You must perform these steps for each GitHub Actions workflow or Terraform Cloud workspace.

Allow your external workload to access Google Cloud resources

To complete the instructions later in this guide, you must configure service account impersonation as described in this section.

To provide your workload with access to Google Cloud resources, we recommend that you grant direct resource access to the principal. In this case, the principal is the federated user. Some Google Cloud products have Google Cloud API limitations (/iam/docs/federated-identity-supported-services). If your workload calls an API endpoint that has a limitation, you can instead use service account impersonation. In this case, the principal is the Google Cloud service account, which acts as the identity. You grant access to the service account on the resource.

Direct resource access Service account impersonation
(#direct-resource-access)

You can grant access to a federated identity directly on resources by using the Google Cloud console or the gcloud CLI.

Consolegcloud (#gcloud)
(#console)

To use the Google Cloud console to grant IAM roles directly on a resource, you must go to the resource's page, and then grant the role. The following example shows you how to go to the Cloud Storage page and grant the role Storage Object Viewer (`roles/storage.objectViewer`) to a federated identity directly on a Cloud Storage bucket.

1. In the Google Cloud console, go to the Cloud Storage **Buckets** page.

   Go to Buckets (https://console.cloud.google.com/storage/browser)

2. In the list of buckets, click the name of the bucket for which you want to grant the role.
3. Select the **Permissions** tab near the top of the page.
4. Click the **Grant access** button.

   The **Add principals** dialog appears.

5. In the **New principals** field, enter one or more identities that need access to your bucket.

   By subjectBy group (#by-group)By attribute (#by-attribute)
   (#by-subject)

   ```
   principal://iam.googleapis.com/projects/PROJECT_NUMBER/locations/global/workloadIdentityPools/POOL_ID/subject/SUBJECT
   ```

   Replace the following:

   - *`PROJECT_NUMBER`*: the project number
   - *`POOL_ID`*: the workload pool ID
   - *`SUBJECT`*: the individual subject mapped from your IdP—for example, `administrator@example.com`

6. Select a role (or roles) from the **Select a role** drop-down menu. The roles you select appear in the pane with a short description of the permissions they grant.
7. Click **Save**.

Configure the deployment pipeline

This section describes how to use Workload Identity Federation in your deployment pipeline. The instructions in this section assume that your workloads use service account impersonation (#access) to access Google Cloud resources.

Azure DevOpsGitHub Actions ... GitLab SaaS (#gitlab-saas)Terraform Cloud ...
(#azure-devops)

Edit your `azure-pipelines.yml` file and add the following to your job configuration:

```
variables:
- name: Azure.WorkloadIdentity.Connection
  value: CONNECTION
- name: GoogleCloud.WorkloadIdentity.ProjectNumber
```

```
    value: PROJECT_NUMBER
  - name: GoogleCloud.WorkloadIdentity.Pool
    value: POOL_ID
  - name: GoogleCloud.WorkloadIdentity.Provider
    value: PROVIDER_ID
  - name: GoogleCloud.WorkloadIdentity.ServiceAccount
    value: SERVICE_ACCOUNT_EMAIL
  - name: GOOGLE_APPLICATION_CREDENTIALS
    value: $(Pipeline.Workspace)/.workload_identity.wlconfig

steps:
  - task: AzureCLI@2
    inputs:
      connectedServiceNameARM: $(Azure.WorkloadIdentity.Connection)
      addSpnToEnvironment: true
      scriptType: 'bash'
      scriptLocation: 'inlineScript'
      inlineScript: |
        echo $idToken > $(Pipeline.Workspace)/.workload_identity.jwt
        cat << EOF > $GOOGLE_APPLICATION_CREDENTIALS
        {
          "type": "external_account",
          "audience": "//iam.googleapis.com/projects/$(GoogleCloud.WorkloadIdentity.ProjectNumber)/locations/global/workloadIdentityPools/$(GoogleCloud.WorkloadIdentity.Pool
          "subject_token_type": "urn:ietf:params:oauth:token-type:jwt",
          "token_url": "https://sts.googleapis.com/v1/token",
          "credential_source": {
            "file": "$(Pipeline.Workspace)/.workload_identity.jwt"
          },
          "service_account_impersonation_url": "https://iamcredentials.googleapis.com/v1/projects/-/serviceAccounts/$(GoogleCloud.WorkloadIdentity.ServiceAccount):generateAc
        }
        EOF
```

Replace the following values:

- *`CONNECTION`*: the name of your service connection
- *`PROJECT_NUMBER`*: the project number of the project that contains the workload identity pool
- *`POOL_ID`*: the ID of the workload identity pool
- *`PROVIDER_ID`*: the ID of the workload identity pool provider
- *`SERVICE_ACCOUNT_EMAIL`*: the email address of the service account

The configuration does the following:

1. Uses the `AzureCLI` task (https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/reference/azure-cli-v2?view=azure-pipelines) to obtain an ID token for the service connection, and makes it available in a variable named `idToken`.
2. Saves the ID token to a temporary file named `.workload_identity.jwt`.
3. Creates a credential configuration file that instructs client libraries to read the ID token from `.workload_identity.jwt` and uses it to impersonate a service account.
4. Sets the environment variable `GOOGLE_APPLICATION_CREDENTIALS` to point to the credential configuration file.

What's next

- Read more about Workload Identity Federation (/iam/docs/workload-identity-federation).
- Learn about best practices for using Workload Identity Federation in deployment pipelines (/iam/docs/best-practices-for-using-service-accounts-in-deployment-pipelines).
- See how you can manage workload identity pools and providers (/iam/docs/manage-workload-identity-pools-providers).

Last updated 2024-09-13 UTC.