

How to implement WIF for Gitlab Pipeline with GCP

- [What is WIF?](#)
- [Why do I need to use WIF to integrate Gitlab Pipeline with GCP?](#)
- [What do I need to take note of when I am using WIF for my pipeline?](#)
- [Who is responsible for the implementation of WIF for my project's pipeline?](#)
- [Script template to be used within the Gitlab's pipeline](#)
- [What is the required to be done on the GCP project ?](#)
- [Request Template for Cloudops](#)
- [Troubleshooting - If pipeline is failing with WIF](#)
- [List of Google APIs required for the GCP Projects](#)

What is WIF?

WIF is short for Workload Identity Federation for GCP.

Traditionally, applications running outside Google Cloud can use service account keys to access Google Cloud resources. However, service account keys are powerful credentials, and can present a security risk if they are not managed correctly.

Workload Identity Federation eliminates the maintenance and security burden associated with service account keys.

Visit the link below for detail explanation.

<https://cloud.google.com/iam/docs/workload-identity-federation>

Why do I need to use WIF to integrate Gitlab Pipeline with GCP?

Implementing WIF will avoid exposing the GSA's keys, which is a security burden.

This helps to reduce the following:

- manual effort to rotate the GSA keys annually.
- risk of the GSA keys being used for non-authorization purposes.

What do I need to take note of when I am using WIF for my pipeline?

The implementation of the WIF is intended with certain security measures:

- The access from Gitlab will be restricted to the repository's path and branches.
Any moving of the gitlab repository to another project's location will require the access to be defined in the GCP.

Who is responsible for the implementation of WIF for my project's pipeline?

The team for WIF implementation are defined as the following:

- Pipeline and CI/CD configuration within the Gitlab's repository:
 - The application team who is responsible for the pipeline within the repository. (Responsible)
- The script template that the application team can refer to:
 - This wiki page (Refer to "Script template to be used within the Gitlab's pipeline" section).
- The GCP project's configuration to enable WIF:
 - For **Landing Zone** GCP projects:
 - **CloudOps**. (Refer to the "Request Template for Cloudops" section to be requested with Service One)
 - For **non-Landing Zone** GCP projects:
 - the team with Owner/Editor permission (primary)
 - If no one, CloudOps.
 - You can find in this page the guide on how to configure on the GCP.

Script template to be used within the Gitlab's pipeline

The code below here is a reusable YAML Anchors and Aliases for majority of the pipeline.

The code is simply to perform the following:

1. Obtain the JWT token for the Gitlab instance.
2. Exchange JWT token with GCP project to obtain the impersonated GCP credential.

3. Login to the GCP with the impersonated GCP credential.

Authentication with Gitlab

```
.gcloud_auth: &gcloud_auth
# Replace the GCP service account name that needs to perform the deployment.
- export GCP_SERVICE_ACCOUNT=<GSA_email>@${PROJECT_ID}.iam.gserviceaccount.com
- echo ${GCP_TOKEN} > .ci_job_jwt_file
- gcloud iam workload-identity-pools create-cred-config ${GCP_WORKLOAD_IDENTITY_PROVIDER}
  --service-account=${GCP_SERVICE_ACCOUNT}
  --output-file=.gcp_temp_cred.json
  --credential-source-file=.ci_job_jwt_file
- gcloud auth login --cred-file=`pwd`/.gcp_temp_cred.json
- gcloud config set project ${PROJECT_ID}
- export GOOGLE_APPLICATION_CREDENTIALS=`pwd`/.gcp_temp_cred.json
```

To use the Anchor defined above, define the [PROJECT_ID](#) and [GCP_WORKLOAD_IDENTITY_PROVIDER](#) by getting the values from the Gitlab project's CI/CD variables.

1. PROJECT_ID - the ID of the GCP project ID that the pipeline is interacting with.
2. GCP_WORKLOAD_IDENTITY_PROVIDER - the value of the GCP Workload Identity Provider.
The format is usually like this:
projects/<project id>/locations/global/workloadIdentityPools/<ID pool name>/providers/<provider name>

Within the deploy step, include the id_tokens declaration and set the aud to <https://gitlab.syensqo.com>.

Usage of the Authentication

```
deploy-test:
  stage: deploy
  only:
    - develop
  id_tokens:
    GCP_TOKEN:
      aud: https://gitlab.syensqo.com
  script:
    - export PROJECT_ID=$G_CLOUD_TARGET_PROJECT_TEST
    - export GCP_WORKLOAD_IDENTITY_PROVIDER=$GCP_WORKLOAD_IDENTITY_PROVIDER_TEST
    - *gcloud_auth
    - mvn package appengine:deploy -DskipTests
```

What is the required to be done on the GCP project ?

The following will be required to be configured within the targeted GCP Project:

1. Create the ID pool within the "Workload Identity Federation".

Workload Identity Pools

[+ ADD PROVIDER](#) [+ CREATE POOL](#)

Workload Identity Federation usage

Count of successful token exchanges using Workload Identity Federation.

Time	Count of successful token exchanges
3:40	1
3:45	0
3:50	0

Use pools to organize and manage external identities. Create a pool for each environment that needs access to Google Cloud resources. [Learn more.](#)

☐ Show deleted pools and providers

Filter Enter property name or value			
Display Name	ID	Providers	Status
idp-gitlab	idp-gitlab	1	✓

2. Create the OIDC provider for the above ID Pool.

ID
gitlab-ce

Issuer (URL) *
https://gitlab.solvay.com
Issuer URL must start with https://

JWK file (JSON) BROWSE
The JWK file should comply with [JWK specification](#). [↗](#) The max size of acceptable file is 80kB.

Audiences
Acceptable values for the aud field in the OIDC token.

☐ Default audience

☒ Allowed audiences
Note when setting an allowed audience, the default is no longer accepted.

Audience 1 *
https://gitlab.solvay.com
Each audience may be at most 256 characters.

+ ADD ALLOWED AUDIENCE

☒ Enabled provider

Attribute Mapping
Credentials can include attributes that provide information about an identity. You can use attribute mapping to grant access to a subset of identities. [Learn more](#). [↗](#)

Google 1
google.subject ?
Supported keys: "google.subject",
"google.groups",
"attribute.custom_attribute".

OIDC 1 *
assertion.sub ?
Value must be a [CEL expression](#). [↗](#)
for example, "assertion.sub".

3. Grant Access for the impersonated GSA and define the subject value to be "project_path:<repository path>:ref_type:branch:ref:<name of the branch: develop|master>".
Actual example: "project_path:cas/winter-clothes-luggage-claim-singapore/WCLC-FrontEnd:ref_type:branch:ref:develop"
In doing so, you are only granting permission to repository with the repository's path and only for the develop branch to access this GCP project.

ch (/) for resources, docs, products, and

+ GRANT ACCESS
DELETE PO

7 days 14 days 30 days Custom

eration.

ected time frame.

4:15 4:20 4:25 4:30 4:35

cts/113988787134/locations/global/workload

BUTE_VALUE

Grant access to service account

Grant access to your workload pools using federated identities or service account impersonation. Federated identities only work with certain [supported services](#).

☐ Grant access using federated identities (Recommended)

Download the Application Default Credential (ADC) file and make it available to your workload. The file lets your workload access Cloud Client Libraries and Google API Client Libraries.

DOWNLOAD CONFIG

☒ Grant access using Service Account impersonation

This option allows your workload to impersonate a service account. After enabling the connection, download the credential configuration file and make it available to your workloads and client libraries.

To access resources, pool identities must be granted access to a service account. Once added, these identities will have permission to access any Google Cloud services that the service account can access. [Learn more](#).

Select service account

Principals will have the same resource permissions as this service accounts.

Service account
sa-gitlab-pipeline

Select principals (identities that can access the service account)

subject

Attribute name. Limited to the keywords from provider attribute mapping.

Please enter an attribute value.

4. You can verify if the grant is successful by checking the permission of the GSA. See the example of 2 successful permission grants for 2 different repositories.

sa-gitlab-pipeline

DETAILS PERMISSIONS KEYS METRICS LOGS

You can use Policy Analyzer to view which resources this service account has access to.

VIEW ACCESS

Principals with access to this service account

You can allow specific users to have ownership and access to service accounts and their settings. Users with the Owner or Editor basic role on the project can already modify service accounts, but you might want to restrict access for some users so that they can take only specific actions against service account resources. [Learn how to grant users access to service accounts](#)

☐ Include Google-provided roles

VIEW BY PRINCIPALS VIEW BY ROLES

GRANT ACCESS REMOVE ACCESS

Filter principal Enter property name or value

Type	Principal	Name	Role	Inheritance
☐	principal://iam.googleapis.com/projects/1047848773320/locations/global/workloadidentityPools/identity-provider/subject/project_path:cas/winter-clothes-luggage-claim-singapore/WCLC-FrontEnd.ref_type:branch.ref:develop		Workload Identity User	
☐	principal://iam.googleapis.com/projects/1047848773320/locations/global/workloadidentityPools/identity-provider/subject/project_path:cas/winter-clothes-luggage-claim-singapore/WCLC-BackEnd.ref_type:branch.ref:develop		Workload Identity User	

Request Template for Cloudops

Request Title: Implement WIF for <gcp project name>

Within description, use the text below and replace with the value required.

Target GCP project: <GCP project ID>

Impersonation GSA: <Service account name used by the gitlab pipeline to perform deployment>

Provider: <<https://gitlab.syensqo.com>>

Repository Path: <example path url of the repository: cas/sinequa/bigquery-connector>

Branch name for Dev: <develop | other name | NA>

Branch name for Test: <develop | other name | NA>
Branch name for Pre-prod: <develop | other name | NA>
Branch name for Prod: master

Troubleshooting - If pipeline is failing with WIF

If the pipeline is failing, please check on the following:

1. Check on Gitlab's pipeline (.gitlab-ci.yml) for the following:
 - a. Make sure the "id_tokens" is defined.
 - b. Make sure the "GCP_SERVICE_ACCOUNT" is using the correct GSA's email.
 - c. Make sure the "GCP_WORKLOAD_IDENTITY_PROVIDER" and "PROJECT_ID" is using the correct values.
 - d. Make sure the "&gcloud_auth" is referring to the \${GCP_TOKEN} variable.

```
echo ${GCP_TOKEN} > .ci_job_jwt_file
```

2. Check on the GCP for the following:
 - a. Make sure the WIF's subject is using the correct "project_path".

List of Google APIs required for the GCP Projects

Here is a list of Google APIs required for WIF:

1. Identity and Access Management (IAM) API
2. IAM Service Account Credentials API