

# GCP Dataflow usage and IAM



I want you to act as a GCP SOLUTION architect in a tech startup. Imagine you're working on a greenfield deployment of GCP. Can you explain gcp dataflow capability and how it can be used to transfer data from other GCP projects.

Go in detail and explain above use case, best practices and IAM roles required to enable dataflow service account to read from Bigquery in another GCP project.

Nice — I'll act as your GCP Solutions Architect and walk through Dataflow's capabilities, a concrete cross-project BigQuery → Dataflow usage pattern, exact IAM requirements (what to grant where and why), plus best practices and step-by-step commands you can copy-paste. I'll be explicit about who (the launcher vs the workers) needs access and call out gotchas (org policy, Storage Read API, least-privilege). Sources are linked inline so you can validate. [Google Cloud Documentation+4](#)

## 1) Quick Dataflow recap (what it is & core capabilities)

- **Managed stream & batch processing** service (Apache Beam SDK runners) for ETL, enrichment, streaming analytics, CDC, windowing, joins, ML inference, etc. Dataflow manages worker VMs, autoscaling, job updates, and execution. [Google Cloud Documentation](#)
- **Connectors:** built-in connectors (BigQueryIO / BigQuery Storage API, Pub/Sub, GCS, Cloud Spanner, JDBC, etc.). For BigQuery, Dataflow supports both the high-level Managed I/O and the lower-level BigQueryIO (and can leverage the BigQuery Storage Read API for high throughput reads). [Google Cloud Documentation+1](#)
- **Templates:** you can package pipelines as Classic/Flex templates and run them from other projects or orchestration systems (careful with cross-project service account usage). [Google Cloud Documentation](#)

## 2) Typical cross-project use case: Dataflow in Project-A reads BigQuery in Project-B

Actors and identities to consider:

1. **Job launcher principal** — the identity (user or service account) that starts the Dataflow job (often from CI/CD, Cloud Composer, Cloud Build, or your laptop).
2. **Dataflow worker service account** — the service account that Dataflow worker VMs assume to perform reads/writes at runtime. (This is the principal that actually accesses BigQuery data from the worker VMs.) [Google Cloud Documentation](#)

For cross-project reads (Dataflow in Project-A reading a BigQuery dataset in Project-B) both of these principals need the right permissions in the correct places:

- The **worker service account** must have BigQuery permissions on the *source dataset* (Project-B).
- The **launcher** must be allowed to *act as* the worker service account when starting the job (usually `iam.serviceAccounts.actAs / roles/iam.serviceAccountUser` on the service account).
- If you create a Dataflow template in one project and run it in another, watch org policy `iam.disableCrossProjectServiceAccountUsage` — it can block cross-project SA usage. [Google Developer forums](#)

### 3) Exactly which IAM roles to grant (recommended minimums)

#### On the Dataflow side (Project-A)

- `roles/dataflow.worker` — **on Project-A** assigned to the worker service account so workers can run Dataflow tasks and interact with other GCP services. (This is the standard role for Dataflow worker VMs.) [Google Cloud](#)

#### On the BigQuery source side (Project-B)

Grant the worker service account least-privilege BigQuery roles on the dataset (preferred) or project (if you need many datasets):

- `roles/bigquery.dataViewer` — allows reading table metadata and table data (basic read). Use **dataset-level** binding where possible. [Google Cloud Documentation](#)
- `roles/bigquery.readSessionUser` — necessary **if** the pipeline will use the BigQuery Storage Read API (recommended for high throughput reads). Grants ability to create read sessions. [Google Cloud Documentation+1](#)
- `roles/bigquery.jobUser` — grant **only if** Dataflow will execute BigQuery jobs (e.g., submit queries/export jobs) as part of the pipeline. Not required for direct table reads via Storage API. [Google Cloud Documentation](#)

#### For the job launcher (the entity that starts the Dataflow job)

- `roles/iam.serviceAccountUser` (or equivalent permission `iam.serviceAccounts.actAs`) on the worker service account — so the launcher can “impersonate” / attach that service account to the Dataflow job. Without this, you’ll see `actAs` errors when creating jobs or templates. Also ensure cross-project SA-use org policy is not blocking. [Google Developer forums](#)

Optional / operational roles:

- `roles/logging.logWriter`, `roles/monitoring.metricWriter` on worker SA if you need explicit logging/monitoring ingestion from workers (Dataflow often adds these automatically but double-check).
- If writing output to BigQuery in Project-C, give the worker SA the appropriate BigQuery writer roles there (e.g., `roles/bigquery.dataEditor` or `dataset-writer`).

### 4) Step-by-step example (create SA, grant roles, run Dataflow)

Assume:

- Dataflow project: **proj-dataflow**
- Source BigQuery project: **proj-source**
- Worker service account: **df-worker@proj-dataflow.iam.gserviceaccount.com**

1. **Create a dedicated worker service account** in **proj-dataflow** (recommended — don't use project default SA):

```
bash

gcloud iam service-accounts create df-worker \
  --project=proj-dataflow \
  --display-name="Dataflow worker SA (prod)"
```

2. **Grant Dataflow worker role in Dataflow project** (so workers can run):

```
bash

gcloud projects add-iam-policy-binding proj-dataflow \
  --member="serviceAccount:df-worker@proj-dataflow.iam.gserviceaccount.com" \
  --role="roles/dataflow.worker"
```

3. **Grant the worker SA read access to BigQuery dataset(s) in proj-source** (preferred: dataset-level).  
If you choose project-level:

```
bash

gcloud projects add-iam-policy-binding proj-source \
  --member="serviceAccount:df-worker@proj-dataflow.iam.gserviceaccount.com" \
  --role="roles/bigquery.dataViewer"
```

For dataset-level (console UI recommended), or create a policy JSON and apply with **bq**:

```
bash

# Example approach: create a policy.json that grants a member and apply it to dataset
bq --project_id=proj-source update --dataset --set_iam_policy=policy.json proj-
source:dataset_name
```

(Using dataset-level bindings is best practice to limit scope.) [Google Cloud Documentation](#)

4. **If using BigQuery Storage API for high throughput reads**, add:

```
bash

gcloud projects add-iam-policy-binding proj-source \
  --member="serviceAccount:df-worker@proj-dataflow.iam.gserviceaccount.com" \
  --role="roles/bigquery.readSessionUser"
```

This enables Dataflow to create read sessions (Storage Read API). [Google Cloud Documentation](#)

5. **If Dataflow will run BigQuery jobs (queries/export jobs)**, add **bigquery.jobUser** on the project where jobs run:

```
bash

gcloud projects add-iam-policy-binding proj-source \
  --member="serviceAccount:df-worker@proj-dataflow.iam.gserviceaccount.com" \
  --role="roles/bigquery.jobUser"
```

6. **Allow the launcher to act as the worker SA** (the user/CI that launches the Dataflow job must have `iam.serviceAccounts.actAs`):

```
bash

gcloud iam service-accounts add-iam-policy-binding \
  --project=proj-dataflow \
  df-worker@proj-dataflow.iam.gserviceaccount.com \
  --member="user:ci-user@yourcompany.com" \
  --role="roles/iam.serviceAccountUser"
```

If you launch from another service (Cloud Composer, Cloud Build), grant the corresponding service account this role. Failure to do this typically shows **Current user cannot act as service account** errors.

[Google Developer forums](#)

7. **Run Dataflow pipeline and attach the worker SA** (example Apache Beam pipeline option):

```
bash

# Java/Beam example flag
--serviceAccount=df-worker@proj-dataflow.iam.gserviceaccount.com
```

Or via `gcloud dataflow jobs run` specify `--service-account-email`.

## 5) Best practices & operational guidance (practical rules)

- **Least privilege:** grant dataset-level BigQuery roles instead of project-level. Prefer `bigquery.dataViewer` + `bigquery.readSessionUser` over broad Editor/Owner roles. [Google Cloud Documentation+1](#)
- **Dedicated SAs per environment:** one SA for dev, one for staging, one for prod to prevent blast radius and allow auditability.
- **Avoid JSON keys:** do *not* distribute long-lived JSON keys for SAs; use workload identity/short-lived tokens or let Dataflow attach the SA.
- **Use BigQuery Storage Read API** for large table reads — it's faster and scales; grant `bigquery.readSessionUser`. [Google Cloud Documentation](#)
- **Monitor & audit:** enable audit logs for BigQuery and Dataflow; review who is actingAs SAs and who launches jobs.
- **Network security:** if sensitive data, run workers in private subnets (VPC) and use Private Google Access or serverless VPC access, and evaluate VPC Service Controls for perimeter protection.
- **Template & CI/CD model:** build Flex templates and store them in GCS; your orchestration (Cloud Build, Composer, Airflow) should have only the minimum rights to *launch* templates and actAs the pipeline SA.

- **Check org policies:** `iam.disableCrossProjectServiceAccountUsage` can block using an SA from another project — work with infra/security to ensure the intended cross-project SA usage is allowed. [Google Developer forums](#)

## 6) Common gotchas & troubleshooting

- **“Current user cannot act as service account”** → missing `roles/iam.serviceAccountUser` for the launcher on the SA or org policy blocking cross-project SA usage. [Google Developer forums](#)
- **Permission errors on BigQuery reads** → you granted project-level roles but dataset has more restrictive bindings or the dataset-level policy overrides. Always test with `gcloud/bq` using the worker SA identity or run a small job. [Google Cloud Documentation](#)
- **Poor performance** → not using BigQuery Storage Read API. Add `bigquery.readSessionUser` and use Storage API via Beam’s connectors. [Google Cloud Documentation](#)

## 7) Short checklist to implement now

1. Create dedicated worker SA in Dataflow project.
2. Grant `roles/dataflow.worker` in Dataflow project to that SA. [Google Cloud](#)
3. Grant `roles/bigquery.dataViewer` (dataset level) in source project to that SA. [Google Cloud Documentation](#)
4. Add `roles/bigquery.readSessionUser` if you’ll use the BigQuery Storage Read API. [Google Cloud Documentation](#)
5. Give job-launcher identity `roles/iam.serviceAccountUser` on the worker SA. Check org policy for cross-project SA usage. [Google Developer forums](#)
6. Test with a small Dataflow job that reads a small table — verify audit logs, job runs, and that workers can read data.

If you want, I can:

- produce the exact `gcloud` + `bq` commands tailored to your real project IDs and dataset names, or
- draft Terraform snippets (IAM bindings + service account + Dataflow job template creation) so you can apply infra-as-code.

Tell me which one you want and I’ll produce it right away (with exact commands/snippets for your project names).