

Implementing Least Privilege IAM Policy Bindings in Cloud Run

Overview

The principle of least privilege states that a resource should only have access to the exact set of resources it needs in order to function. For example, if a service is performing an automated database backup, the service should be restricted to read-only permissions on exactly one database. Similarly, if a service is only responsible for encrypting data, it should not have permissions for decrypting data.

In Cloud Run, if a service is deployed without specifying a service account, a default service account is used. The default service account used is the Compute Engine service account which has the broad Editor role on the project. Because of policy

binding inheritance, this service account has read and write permissions on most resources in your project. While convenient, it's an inherent security risk as resources can be created, modified, or deleted with this service account.

To mitigate this risk and implement the principle of least privilege, you should create a service account that serves as the service's identity, and grant the minimum set of permissions to the account that are required for the service's functionality.

Objectives

In this lab, you learn to:

- Configure your environment and enable the Cloud Run API.
- Create and deploy a public Cloud Run service.
- Test the service with unauthenticated requests.
- Create a service account with minimum permissions.
- Use the gcloud CLI to authenticate with the service account, and invoke a Cloud Run service.
- Implement least privilege by granting the minimum set of permissions required to invoke a service on Cloud Run.

Prerequisites

These labs are based on intermediate knowledge of Google Cloud. While the steps required are covered in the content, it would be helpful to have familiarity with any of the following products:

- IAM
- Cloud Run

Task 1. Configure the environment

Set up environment variables in Cloud Shell to make the provisioning process more flexible.

1. Enable Cloud Run API:
2. `gcloud services enable run.googleapis.com`
3. Copied!
4. `content_copy`
5. Create a LOCATION environment variable:
6. `LOCATION=us-west1`
7. Copied!
8. `content_copy`
9. Set the default Cloud Run region:
10. `gcloud config set run/region $LOCATION`
11. Copied!
12. `content_copy`

Task 2. Create and deploy a public service

Requirements

Quickway parking has a Cloud Run billing service that they would like to be made more secure. In this task, you:

- Deploy the *billing service* from an image.
- Test the service by invoking it without any authentication.

Deploying with Cloud Run

The Quickway development team already has an image of the billing application available on Google Cloud.

1. Deploy the billing application image to Cloud Run:

```
gcloud run deploy billing-service \
  --image gcr.io/qwiklabs-resources/gsp723-parking-service \
  --region $LOCATION \
  2.    --allow-unauthenticated
```

3. Copied!

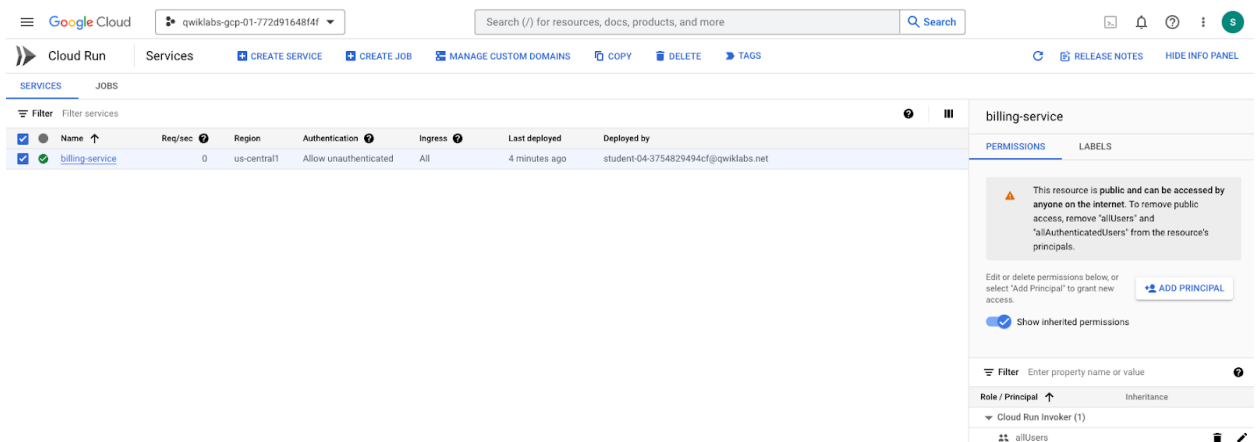
4. content_copy

5. Assign the URL of the new service to an environment variable:

```
BILLING_SERVICE_URL=$(gcloud run services list \
  --format='value(URL)' \
  6.    --filter="billing-service")
```

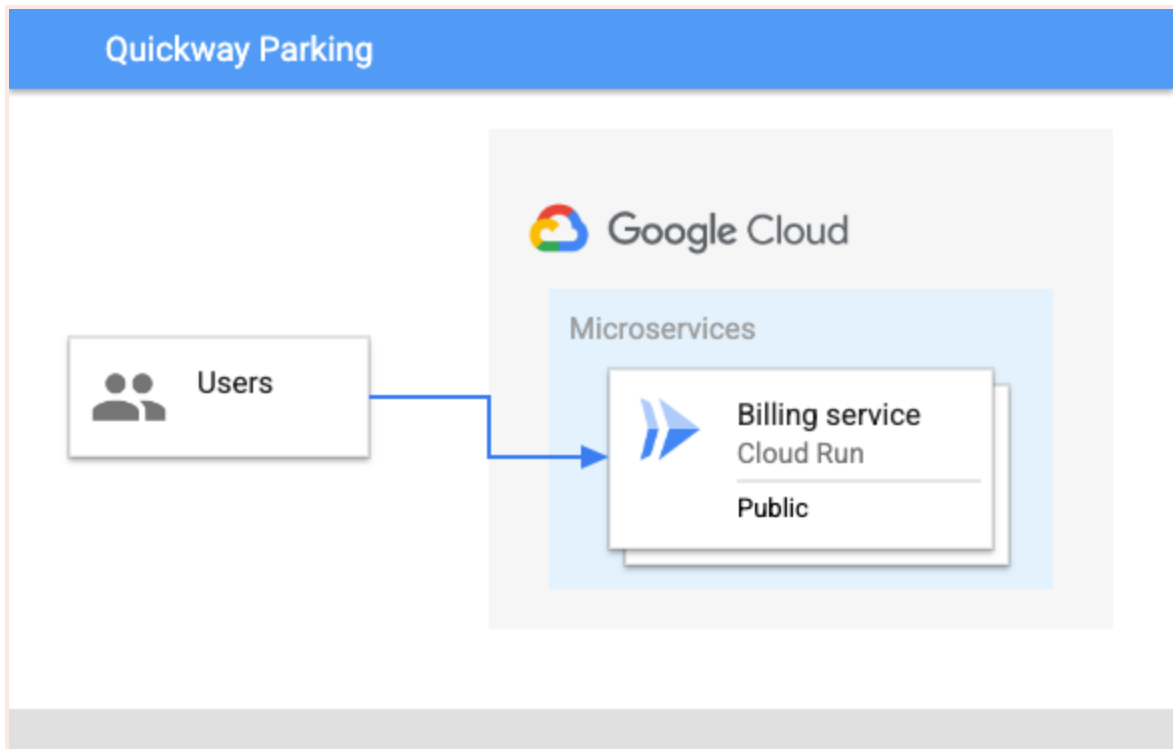
7. Copied!
8. content_copy
9. Invoke the service without any authorization:
10.

```
curl -X POST -H "Content-Type: application/json"
    $BILLING_SERVICE_URL -d '{"userid": "1234", "minBalance": 100}'
```
11. Copied!
12. content_copy
13. The service does not generate any output when invoked.
14. In the Google Cloud console **Navigation menu** (**≡**), click **Cloud Run**.
15. Click the link to the *billing-service*.
16. To view the service logs, click **Logs**.
17. Add the log filter *minBalance* to view the minimum balance received in the request made to the service.
18. To go back to the service details page, click **<- Service details**.
19. Select the *billing-service* by checking the box to the left of the green check mark.



The Security team has spotted something in the security settings. Can you see what part of the above configuration has them so concerned?

20. Take a closer look at the authentication applied. Currently **anyone on the internet** can call the billing service. This is indicated by the *allUsers* identity that has the *Cloud Run Invoker* role.



21.

When the Billing service was originally deployed, it used the `--allow-unauthenticated` permission, which means that the service is publicly accessible, and can be invoked without any authentication.

Type	Permission	Description
URL Access s	<code>--allow-unauthenticated</code>	Make the service publicly accessible (Unauthenticated users can access it).
Invoking Princi pal	<code>allUsers</code>	Allow the service to be invoked/triggered by anyone.

22.

By removing the `--allow-unauthenticated` permission you can use the Cloud Run

default permissions to secure the service, or you can explicitly specify the `no-allow-unauthenticated` permission.

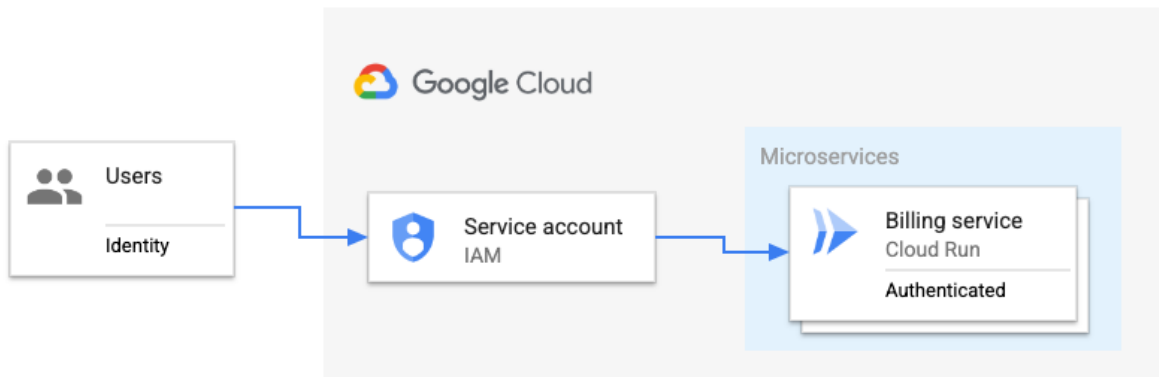
Type	Permission	Description
URL Access	<code>--no-allow-unauthenticated</code>	Secure the service with authentication (Only authenticated users can access it).
Invoking Principal	none	Do not allow the service to be invoked/triggered by anyone.

23. **Note:** Remember, with Google Cloud, always try to use least privilege permissions in your solution.

By making these changes, the Security team will be a lot happier with the overall design.

Task 3. Authenticating service requests

The team updates the application design to show how the changes will work:



The main changes are:

- Remove unauthenticated public access to the *billing service*.
- Create a new service account with appropriate permissions to invoke the *billing service*.

Update the service to require authentication

Now that you understand more about the permissions used with Cloud Run, correct the authentication permissions applied to the Billing service:


1. Delete the existing deployed service:
2. `gcloud run services delete billing-service`
3. Copied!
4. `content_copy`
5. If prompted, type **Y**, and then press **Enter**.
6. Redeploy the *billing service* with the `--no-allow-unauthenticated` permission:

- ```
gcloud run deploy billing-service \
--image gcr.io/qwiklabs-resources/gsp723-parking-service \
--region $LOCATION \
7. --no-allow-unauthenticated
8. Copied!
9. content_copy
10. Redeploying the service means it no longer allows unauthenticated access at its
 service URL. In addition, the access permission to invoke the service has been
 removed.
11. Wait a few seconds, and then invoke the billing service again as before:
12. curl -X POST -H "Content-Type: application/json"
 $BILLING_SERVICE_URL -d '{"userid": "1234", "minBalance": 100}'
13. Copied!
14. content_copy
15. As expected, the output is a permissions error since the service now requires
 authentication.
```

## Create a service account

To invoke the *billing service* you will need an identity or service account with appropriate permissions, and bind that identity to the service.

This can be done in the Google Cloud console, or with the gcloud command line interface. In this lab, you use the Google Cloud console to create the service account and set up the new policy binding for the *billing service*.

1. In the Google Cloud console **Navigation menu** () , select **IAM & Admin > Service Accounts**.

2. To create a new service account that will provide authenticated access, click **Create Service Account** near the top.
3. Name the service account: **Billing Initiator**.

## Create service account

1

### Service account details

Service account name

Billing Initiator

Display name for this service account

Service account I...

billing-initiator @qwiklabs-gcp-02-d1d935e202be.iam.gservicea X ↺

Service account description

Describe what this service account will do

CREATE AND CONTINUE

4. To create the account, click **Create and Continue**, and then advance to the **Grant Access** step.
5. To give the *Billing Initiator* service account permissions to invoke the *billing service*, select the **Role** drop-down, scroll the left side to **Cloud Run**, and then

select the role **Cloud Run Invoker**.

## Create service account

### ✓ Service account details

### 2 Grant this service account access to project (optional)

Grant this service account access to qwiklabs-gcp-03-18845966d94a so that it has permission to complete specific actions on the resources in your project. [Learn more](#)

Role

Cloud Run Invoker ▼

Condition

[Add condition](#)

Can invoke a Cloud Run service.

[+ ADD ANOTHER ROLE](#)

**CONTINUE**

### 3 Grant users access to this service account (optional)

**DONE**

CANCEL

6. To complete the setup of the service account, click **Continue**, and then click **Done**.






You will see the new service account at the top of the list of service accounts in

the console.

## Service accounts for project "qwiklabs-gcp-01-e9163f39c5e1"

A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine

Organization policies can be used to secure service accounts and block risky service account features, such as automatic creation. For more information, see [about service account organization policies](#).

|  Filter Enter property name or value |                                                                                                                                                          |                                                                                    |                                |                                                     |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|--------------------------------|-----------------------------------------------------|
| <input type="checkbox"/>                                                                                              | Email                                                                                                                                                    | Status                                                                             | Name ↑                         | Description                                         |
| <input type="checkbox"/>                                                                                              |  billing-initiator@qwiklabs-gcp-01-e9163f39c5e1.iam.gserviceaccount.com |  | Billing Initiator              | Trigger the Billing Service job via Cloud Scheduler |
| <input type="checkbox"/>                                                                                              |  923411142380-compute@developer.gserviceaccount.com                     |  | Compute Engine default service |                                                     |

The service account **Billing Initiator** has been created with the authorization to invoke a Cloud Run service, using an IAM policy binding on your project.


## Task 4. Invoke the service with authentication

Now that you have a service account with the appropriate permission, you can use it to invoke your Cloud Run service.

# Authenticate with gcloud

The first step is to set the service account in gcloud so it can be used to authenticate with the service.

1. In the Cloud Shell terminal menu, open a new shell in a separate tab by clicking

**Add** (  ).

Execute the remaining commands of this task in this Cloud Shell window.

2. Get the service account identity email and save it in an environment variable:
3. `BILLING_INITIATOR_EMAIL=$(gcloud iam service-accounts list --filter="Billing Initiator" --format="value(EMAIL)"); echo $BILLING_INITIATOR_EMAIL`
4. In this Cloud Shell terminal, assign the URL of the *billing service* to an environment variable:

```
BILLING_SERVICE_URL=$(gcloud run services list \
--format='value(URL)' \
5. --filter="billing-service")
```

6. To authenticate gcloud using the service account, generate a key file:
7. `gcloud iam service-accounts keys create key.json --iam-account=${BILLING_INITIATOR_EMAIL}`
8. Authorize access to Cloud Run with a service account:
9. `gcloud auth activate-service-account --key-file=key.json`

## Invoke the service

1. Invoke the Cloud Run *billing* service with an identity token generated from the service account:

```
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-identity-token)" \
2. $BILLING_SERVICE_URL -d '{"userid": "1234", "minBalance": 500}'
```

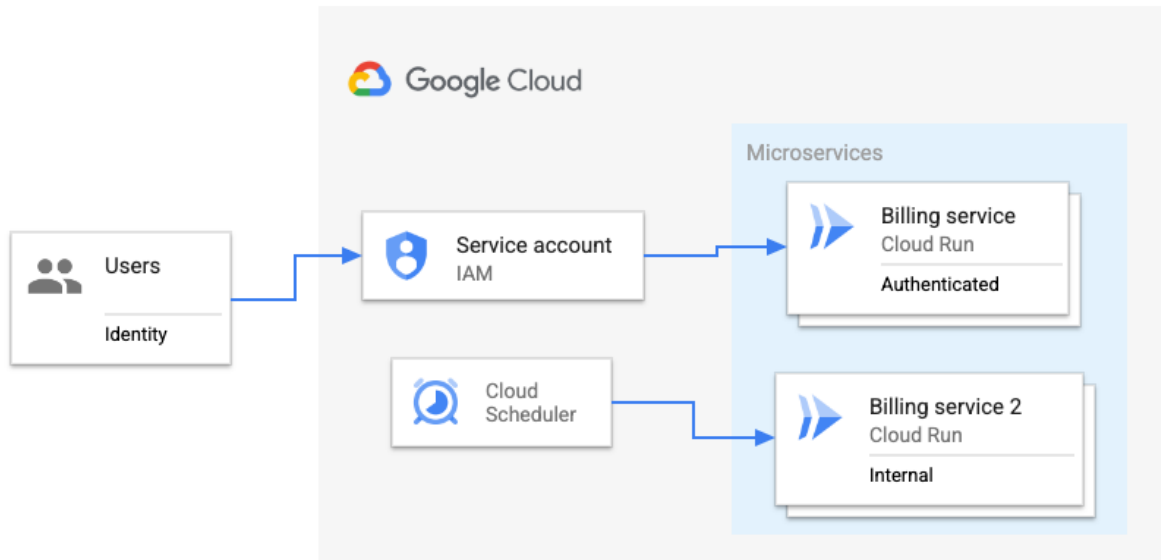
3. In the Google Cloud console **Navigation menu** (≡), click **Cloud Run**.
4. Click the link to the *billing-service*.
5. To view the service logs, click **Logs**.
6. Add the log filter *minBalance* to view the updated minimum balance received in the request made to the service.

## Task 5. Implement least privilege

You've used a service account with the appropriate permissions to invoke a Cloud Run service that was previously accessible by anyone. But, have you used the absolute minimum privileges needed to call this specific service?

To determine if this is true, deploy a second billing service which we will assume should be accessible only by other internal private services, such as Cloud Scheduler.

Here's a diagram of this requirement:



## Deploy a second service

1. Open a third Cloud Shell terminal window or tab.
2. Create a LOCATION environment variable:
3. LOCATION=us-west1
4. To simulate a second service, deploy the billing application image to Cloud Run:

```
gcloud run deploy billing-service-2 \
 --image gcr.io/qwiklabs-resources/gsp723-parking-service \
 --region $LOCATION \
```

5. --no-allow-unauthenticated

6. Assign the URL of the new service to an environment variable:

```
BILLING_SERVICE_2_URL=$(gcloud run services list \
 --format='value(URL)' \
```

7. `--filter="billing-service-2")`

## Invoke the second service with the service account identity

1. In this third Cloud Shell terminal, authorize access to Cloud Run with the same service account:
2. `gcloud auth activate-service-account --key-file=key.json`
3. Invoke the second Cloud Run service with an identity token generated from the service account:

```
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-identity-token)" \
4. $BILLING_SERVICE_2_URL -d '{"userid": "1234", "minBalance": 900}'
```

Why was this successful!? It's because when you created the service account, the Cloud Run Invoker permissions were granted to this account on the project. Because of inheritance, resources in the project such as the two Cloud Run services inherit those permissions, and as a result, the service account can be used to invoke the services.

## Restrict service account permissions

To fully implement least privilege, the service account should only be granted permissions on the service that it needs.



In this subtask, you remove the permission previously granted to the service account on the project, and then add the appropriate permissions required to invoke the original billing service.

1. Switch to the first Cloud Shell terminal window.
2. In this Cloud Shell terminal, get the service account identity email and save it in an environment variable:
3. `BILLING_INITIATOR_EMAIL=$(gcloud iam service-accounts list --filter="Billing Initiator" --format="value(EMAIL)"); echo $BILLING_INITIATOR_EMAIL`
4. Remove the permission on the service account for the project:

```
gcloud projects remove-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
--member=serviceAccount:${BILLING_INITIATOR_EMAIL} \
5. --role=roles/run.invoker
```

6. Add the permission to the service account on the *billing service*:

```
gcloud run services add-iam-policy-binding billing-service --region
$LOCATION \
--member=serviceAccount:${BILLING_INITIATOR_EMAIL} \
7. --role=roles/run.invoker --platform managed
```

## Invoke the services

1. Switch back to the second Cloud Shell terminal window or tab.
2. Wait a few seconds, and then invoke the first Cloud Run *billing service* with an identity token generated from the service account:

```
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-identity-token)" \
```

3. `$BILLING_SERVICE_URL -d '{"userid": "1234", "minBalance": 700}'`

It takes a few seconds for the updated permissions to propagate, after which this invocation should be successful.

1. Switch to the third Cloud Shell terminal window.
2. Try to invoke the second Cloud Run service with an identity token generated from the same service account:

```
curl -X POST -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-identity-token)" \
3. $BILLING_SERVICE_2_URL -d '{"userid": "1234", "minBalance": 500}'
```

4. You should now receive a permissions error indicating that the service account has only the minimum set of permissions required to invoke the first service.

## Congratulations!

In this lab, you have seen how to reconfigure a deployed service to secure access to it, and implemented the principle of least privilege when granting permissions to access resources on Google Cloud. You:

- Deployed a service to Cloud Run.
- Used the gcloud CLI to update the service to require authentication.
- Created a service account with the required permissions to invoke the service.
- Set the minimum permissions required to invoke a specific service on Cloud Run, implementing least privilege.