Executing asynchronous tasks

You can use Cloud Tasks to securely enqueue a task to be asynchronously processed by a Cloud Run service. Typical use cases include:

- Preserving requests through unexpected production incidents
- Smoothing traffic spikes by delaying work that is not user-facing
- Speeding user response time by delegating slow background operations to be handled by another service, such as database updates or batch processing
- Limiting the call rate to backing services like databases and third-party APIs

This page shows how to enqueue tasks that are securely pushed via the HTTPS protocol to a private Cloud Run service. It describes required behavior for the private Cloud Run service, required service account permissions, task queue creation, and task creation.

Before you start

ENABLE THE CLOUD TASKS API (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/LIBRARY/CLOUDTASKS.G
on the project you are using.

Deploying a Cloud Run service to handle tasks

To deploy a service that accepts tasks sent to the task queue, deploy the service (https://cloud.google.com/run/docs/deploying) in the same way as any other Cloud Run service. The Cloud Run service must return an HTTP `200` code to confirm proper processing of the task.

Tasks will be pushed to this Cloud Run service as HTTPS requests by Cloud Tasks.

Note: If you are using Cloud Run for Anthos on Google Cloud, you must verify the identity of the received task within the container. See the IAP sample code

(https://cloud.google.com/iap/docs/authentication-howto#authenticating_from_a_service_account) that demonstrates this.

Creating a task queue

To create a task queue, use the command

```
gcloud tasks queues create QUEUE-ID
```

replacing ***QUEUE-ID*** with the name you want to give to your task queue: it must be unique in your project. If you are prompted to create an App Engine app in your project, respond `y` to create it. Cloud Tasks uses this for the queue: make sure you choose the same location as you are using for your Cloud Run service.

The default task queue configuration should work in most cases. However, you can optionally set different rate limits (https://cloud.google.com/tasks/docs/configuring-queues#rate) and retry parameters (https://cloud.google.com/tasks/docs/configuring-queues#retry) if you want.

Creating a service account to associate with the tasks

You must create a service account that will be associated with the enqueued tasks. This service account must have the Cloud Run Invoker IAM role to allow the task queue to push tasks to the Cloud Run (fully managed) service. .

CONSOLE COMMAND LINE

1. Visit the *Create service account key* page in the Cloud Console.

   CREATE SERVICE ACCOUNT PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/CREDENTIALS/

2. From the *Service account list*, select **New service account**.
3. In the *Service account name* field, enter the name you want to use for the service account.
4. Click **Create**.
5. Copy the service account email to use in the following steps.
6. Click **Continue** if prompted to specify permissions.

7. Visit the *Cloud Run Services* page in the Cloud Console.

   GO TO THE SERVICES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/RUN)

8. Select your service in the displayed list.
9. If necessary, click the **Show Info Panel**/**Hide Info Panel** toggle in the far right of the page to show information.
10. Locate the *Permissions* tab, and in that tab, click **Add Member**.
11. Paste your service account email into the **New members** field.
12. From the Role dropdown menu, select **Cloud Run** > **Cloud Run Invoker**.
13. Click **Save**.

Creating HTTP tasks with authentication tokens

When you create a task to send to the task queue, you specify the project, the location, queue name, the email of the previously created service account (https://cloud.google.com/run/docs/triggering/service-account-invoker) to associate with tasks, the URL of the private Cloud Run service that will run the task, and any other data you need to send.

Refer to the Cloud Tasks API documentation for details on the task request body (https://cloud.google.com/tasks/docs/reference/rest/v2/projects.locations.queues.tasks#Task). Note that requests that contain data payloads must use the HTTP `PUT` or `POST` method.

The code that enqueues the tasks must have the necessary IAM permissions to do so, such as the Cloud Tasks Enqueuer role. Your code will have the necessary IAM permissions if you use the default service account on Cloud Run (fully managed).

The following examples create task requests that also include the creation of a header token. OIDC tokens are used in the examples. To use an OAuth token, replace the OIDC parameter with the language appropriate OAuth parameter in constructing the request.

PYTHON JAVA GO NODE.JS

tasks/create_http_task_with_token.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/tasks/create_http_task_with_token.py)

UDPLATFORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/TASKS/CREATE_HTTP_TASK_WITH_TOKEN.PY)

```
"""Create a task for a given queue with an arbitrary payload."""

from google.cloud import tasks_v2
from google.protobuf import timestamp_pb2

# Create a client.
client = tasks_v2.CloudTasksClient()

# TODO(developer): Uncomment these lines and replace with your values.
# project = 'my-project-id'
# queue = 'my-queue'
# location = 'us-central1'
# url = 'https://example.com/task_handler'
# payload = 'hello'

# Construct the fully qualified queue name.
parent = client.queue_path(project, location, queue)

# Construct the request body.
task = {
        'http_request': {  # Specify the type of request.
            'http_method': 'POST',
            'url': url,  # The full url path that the task will be sent to.
            'oidc_token': {
                'service_account_email': service_account_email
            }
        }
}

if payload is not None:
    # The API expects a payload of type bytes.
    converted_payload = payload.encode()

    # Add the payload to the request.
    task['http_request']['body'] = converted_payload

if in_seconds is not None:
    # Convert "seconds from now" into an rfc3339 datetime string.
    d = datetime.datetime.utcnow() + datetime.timedelta(seconds=in_seconds)

    # Create Timestamp protobuf.
    timestamp = timestamp_pb2.Timestamp()
    timestamp.FromDatetime(d)

    # Add the timestamp to the tasks.
```

```
    task['schedule_time'] = timestamp

# Use the client to build and send the task.
response = client.create_task(parent, task)

print('Created task {}'.format(response.name))
return response
```

Note the `requirements.txt` file:

tasks/requirements.txt (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/tasks/requirements.txt)

OM/GOOGLECLOUDPLATFORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/TASKS/REQUIREMENTS.TXT)

```
google-cloud-tasks==1.3.0
googleapis-common-protos==1.6.0
```

What's next

- Logging and viewing logs (https://cloud.google.com/run/docs/logging)
- Monitoring health and performance (https://cloud.google.com/run/docs/monitoring)
- Triggering from Pub/Sub (https://cloud.google.com/run/docs/events/pubsub)
- Invoking with HTTPS (https://cloud.google.com/run/docs/events/https-request)
- Running services on a schedule (https://cloud.google.com/run/docs/events/using-scheduler)