Google Cloud Pub/Sub Triggers

Cloud Functions uses event-driven functions (/functions/docs/writing#event-driven_functions) to handle events from your Cloud infrastructure. For example, Cloud Functions can be triggered by messages published to Pub/Sub topics (/pubsub/docs) in the same Cloud project as the function. Pub/Sub is a globally distributed message bus that automatically scales as you need it and provides a foundation for building your own robust, global services.

this document is **not** applicable to HTTP-triggered functions invoked by Cloud Pub/Sub HTTP push subscriptions (/pubsub/docs/push). Functions that proces
ub messages forwarded by HTTP push subscriptions have different (/functions/docs/calling/http) function arguments and event structures.

Event types

There is a single Pub/Sub event used by Cloud Functions, and it has the trigger type value `google.pubsub.topic.publish`.

This event is sent when a message is published to a Pub/Sub topic that is specified when a function is deployed (#deploying_your_function). Every message published to this topic will trigger function execution with message contents passed as input data.

Event structure

Cloud Functions triggered from a Pub/Sub (/pubsub/docs) topic will be sent events conforming to the `PubsubMessage` (/pubsub/docs/reference/rest/v1/PubsubMessage) type, with the caveat that `publishTime` and `messageId` are not directly available in the

`PubsubMessage`. Instead, you can access `publishTime` and `messageId` via the event ID and timestamp properties of the event metadata. This metadata is accessible via the context object (/functions/docs/writing/background#function_parameters) that is passed to your function when it is invoked.

The payload of the `PubsubMessage` object, the data published to the topic, is stored as a base64-encoded string in the `data` attribute of the `PubsubMessage`. To extract the payload of the `PubsubMessage` object, you may need to decode the data attribute as shown in the examples below.

Sample code

Node.js (#node.js) Python (#python) Go (#go) Java (#java) Kotlin (#kotlin) C# (#c) Ruby (#ruby) PHP (#php)

functions/helloworld/main.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/HEAD/functions/helloworld/main.py)

View on GitHub (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/HEAD/functions/helloworld/main.py)

```
def hello_pubsub(event, context):
    """Background Cloud Function to be triggered by Pub/Sub.
    Args:
         event (dict):  The dictionary with data specific to this type of
                        event. The `@type` field maps to
                         `type.googleapis.com/google.pubsub.v1.PubsubMessage`.
                        The `data` field maps to the PubsubMessage data
                        in a base64-encoded string. The `attributes` field maps
                        to the PubsubMessage attributes if any is present.
         context (google.cloud.functions.Context): Metadata of triggering event
                        including `event_id` which maps to the PubsubMessage
                        messageId, `timestamp` which maps to the PubsubMessage
```

```
                    publishTime, `event_type` which maps to
                    `google.pubsub.topic.publish`, and `resource` which is
                    a dictionary that describes the service API endpoint
                    pubsub.googleapis.com, the triggering topic's name, and
                    the triggering event type
                    `type.googleapis.com/google.pubsub.v1.PubsubMessage`.
Returns:
    None. The output is written to Cloud Logging.
"""
import base64

print("""This Function was triggered by messageId {} published at {} to {}
""".format(context.event_id, context.timestamp, context.resource["name"]))

if 'data' in event:
    name = base64.b64decode(event['data']).decode('utf-8')
else:
    name = 'World'
print('Hello {}!'.format(name))
```

Unless an exception is thrown **and** automatic retrying is enabled, Cloud Functions **ack**s incoming Pub/Sub messages internally upon function invocation. See t
ng Background Functions (/functions/docs/bestpractices/retries) documentation for more information on how to automatically retry invocations when an exce
wn.

Publishing a message from within a function

You can also publish a message to a Pub/Sub topic from within a function. This lets you trigger subsequent Cloud Function invocations using Cloud Pub/Sub messages. You can use this technique to:

- Chain sequential function calls together.
- Distribute (or "fan out") groups of tasks in parallel across multiple Cloud Function instances.

In the following example, an HTTP `publish` function sends a message to a Pub/Sub topic, and that in turn triggers a `subscribe` function.

This snippet shows the `publish` function that publishes a message to a Pub/Sub topic.

the code below assumes you have set the **`GOOGLE_CLOUD_PROJECT`** environment variable to the name of your Google Cloud project.

Node.js (#node.js) Python (#python) Go (#go) Java (#java)

functions/pubsub/main.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/HEAD/functions/pubsub/main.py)

View on GitHub (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/HEAD/functions/pubsub/main.py)

```
import base64
import json
import os

from google.cloud import pubsub_v1


# Instantiates a Pub/Sub client
publisher = pubsub_v1.PublisherClient()
PROJECT_ID = os.getenv('GOOGLE_CLOUD_PROJECT')
```

```
# Publishes a message to a Cloud Pub/Sub topic.
def publish(request):
    request_json = request.get_json(silent=True)

    topic_name = request_json.get("topic")
    message = request_json.get("message")

    if not topic_name or not message:
        return ('Missing "topic" and/or "message" parameter.', 400)

    print(f'Publishing message to topic {topic_name}')

    # References an existing topic
    topic_path = publisher.topic_path(PROJECT_ID, topic_name)

    message_json = json.dumps({
        'data': {'message': message},
    })
    message_bytes = message_json.encode('utf-8')

    # Publishes a message
    try:
        publish_future = publisher.publish(topic_path, data=message_bytes)
        publish_future.result()  # Verify the publish succeeded
        return 'Message published.'
    except Exception as e:
        print(e)
        return (e, 500)
```

This snippet shows the `subscribe` function that is triggered when the message is published to the Pub/Sub topic:

Node.js (#node.js) Python Java (#java) (#python)

functions/pubsub/main.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/HEAD/functions/pubsub/main.py)

View on GitHub (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/HEAD/functions/pubsub/main.py)

```
# Triggered from a message on a Cloud Pub/Sub topic.
def subscribe(event, context):
    # Print out the data from Pub/Sub, to prove that it worked
    print(base64.b64decode(event['data']))
```

In production, you might use the cURL command-line utility to invoke the `publish` function, as follows:

```
curl https://GCF_REGION-GCP_PROJECT_ID.cloudfunctions.net/publish -X POST  -d "{\"topic\": \"PUBSUB_TOPIC\", \"messag
```

But for testing and debugging, you could instead use the `gcloud functions call` command to call the function directly (/functions/docs/calling/direct#using_the_gcloud_command-line_interface). The following steps describe how to run the above example using the `gcloud functions call` command:

1. Create a Pub/Sub topic, where `MY_TOPIC` is the name of the new topic you are creating:

```
gcloud pubsub topics create MY_TOPIC
```

2. Deploy the `publish` function, where ***`RUNTIME`*** is the name of the runtime you are using, such as `nodejs8`:

```
gcloud functions deploy publish --trigger-http --runtime RUNTIME
```

3. Deploy the `subscribe` function:

```
gcloud functions deploy subscribe --trigger-topic MY_TOPIC --runtime RUNTIME
```

4. Directly invoke the `publish` function using the `gcloud functions call` command, and supply the required data as JSON in the `--data` argument:

```
gcloud functions call publish --data '{"topic":"MY_TOPIC","message":"Hello World!"}'
```

5. Check the logs for the `subscribe` function. Note that it may take a few minutes for your results to appear in the log:

```
gcloud functions logs read subscribe
```

You should see output resembling the following:

```
D    ...Function execution started
I    ...{"data":{"message":"Hello World!"}}
D    ...Function execution took 753 ms, finished with status: 'ok'
```

Deploying your function

The following `gcloud` command deploys a function that will be triggered when a message is published to a Pub/Sub topic:

```
gcloud functions deploy FUNCTION_NAME --entry-point ENTRY_POINT --trigger-topic TOPIC_NAME FLAGS...
```

When you deploy a function using the **`gcloud`** command-line tool, the command must include the name of the function contained in your code that you want th y command to execute. You can specify that function using either **`FUNCTION_NAME`** or the optional **`--entry-point`** flag, depending on the needs of your nentation. See Deploy using the **`gcloud`** tool (/functions/docs/deploying/filesystem#deploy_using_the_gcloud_tool) for more discussion of this topic.

Argument	Description
`FUNCTION_NAME`	The registered name of the Cloud Function you are deploying. This can either be the name of a function in your source code, or an arbitrary string. If ***`FUNCTION_NAME`*** is an arbitrary string, then you must include the `--entry-point` flag.
`--entry-point` ***`ENTRY_POINT`***	The name of a function or class in your source code. Optional, unless you did not use ***`FUNCTION_NAME`*** to specify the function in your source code to be executed during deployment. In that case, you must use `--entry-point` to supply the name of the executable function.
`--trigger-topic` ***`TOPIC_NAME`***	The name of the Pub/Sub topic to which the function is subscribed. If the topic doesn't exist, it is created during deployment.
`FLAGS...`	Additional flags you must specify during deployment, such as `--runtime`. For a full reference, see the **`gcloud`**

`functions deploy` documentation (/sdk/gcloud/reference/functions/deploy).

See the Pub/Sub Tutorial (/functions/docs/tutorials/pubsub) for a complete example of how to use Pub/Sub triggers.

Legacy Cloud Pub/Sub triggers

The `gcloud` command below deploys a function that is triggered by legacy Pub/Sub notifications on a specific topic. These notifications are supported for legacy functions already consuming these events. However, we recommend using the `--trigger-topic` flag instead, as the legacy notifications might be removed at a future date.

```
gcloud functions deploy FUNCTION_NAME \
--trigger-resource TOPIC_NAME \
--trigger-event providers/cloud.pubsub/eventTypes/topic.publish \
FLAGS...
```

f you have an existing function using the legacy event, and you want to use the new event type instead, you must first delete the function that uses the legacy e

Next steps

See the Pub/Sub Tutorial (/functions/docs/tutorials/pubsub) for an example of how to implement an event-driven function that is triggered by Pub/Sub.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the Apache 2.0 License (https://www.apache.org/licenses/LICENSE-2.0). For details, see the Google Developers Site Policies (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2021-08-19 UTC.