Processing images from Cloud Storage tutorial

This tutorial demonstrates using Cloud Run, Cloud Vision API, and ImageMagick (https://imagemagick.org/index.php) to detect and blur offensive images uploaded to a Cloud Storage bucket. This tutorial builds on the tutorial Using Pub/Sub with Cloud Run (https://cloud.google.com/run/docs/tutorials/pubsub).

This tutorial walks through modifying an existing sample app. You can also download the completed sample (#downloading) if you want.

You can use this tutorial with Cloud Run (fully managed) or Cloud Run for Anthos on Google Cloud.

Objectives

- Write, build, and deploy an asynchronous data processing service to Cloud Run (fully managed) or Cloud Run for Anthos on Google Cloud
- Invoke the service by uploading a file to Cloud Storage, creating a Pub/Sub message.
- Use the Cloud Vision API to detect violent or adult content.
- Use ImageMagick to blur offensive images.
- Test the service by uploading an image of a flesh-eating zombie.

Costs

This tutorial uses billable components of Cloud Platform, including:

- Cloud Build (https://cloud.google.com/cloud-build/)
- Container Registry (https://cloud.google.com/container-registry/)
- Pub/Sub (https://cloud.google.com/pubsub/)
- Cloud Storage (https://cloud.google.com/storage/)

- Cloud Vision API (https://cloud.google.com/vision/)
- Cloud Run or Cloud Run for Anthos on Google Cloud (https://cloud.google.com/run/)

Use the Pricing Calculator (https://cloud.google.com/products/calculator/) to generate a cost estimate based on your projected usage.

New Cloud Platform users might be eligible for a free trial (https://cloud.google.com/free/).

Before you begin

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Google Cloud project.

   Note: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

   GO TO THE PROJECT SELECTOR PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECT

3. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (https://cloud.google.com/billing/docs/how-to/modify-project).

4. Enable the Cloud Run and Cloud Vision APIs.

   ENABLE THE APIS (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=RUN.GOO

5. Install and initialize (https://cloud.google.com/sdk/docs/) the Cloud SDK.

6. For Cloud Run for Anthos on Google Cloud install the gcloud kubectl component:

   ```
   gcloud components install kubectl
   ```

7. Update components:

   ```
   gcloud components update
   ```

8. Using Cloud Run for Anthos on Google Cloud, create a new cluster using the instructions in Setting up Cloud Run for Anthos on Google Cloud (https://cloud.google.com/run/docs/gke/setup).
9. Set up a Pub/Sub topic, a secure push subscription, and an initial Cloud Run service to handle messages by following the Using Pub/Sub Tutorial (https://cloud.google.com/run/docs/tutorials/pubsub)

Setting up gcloud defaults

To configure gcloud with defaults for your Cloud Run service:

1. Set your default project:

```
gcloud config set project PROJECT-ID
```

   Replace ***PROJECT-ID*** with the name of the project you created for this tutorial.
2. If you are using Cloud Run (fully managed), configure gcloud for your chosen region:

```
gcloud config set run/region REGION
```

   Replace ***REGION*** with the supported Cloud Run region (#follow-cloud-run) of your choice.
3. If you are using Cloud Run for Anthos on Google Cloud, configure gcloud for your cluster:

```
gcloud config set run/cluster CLUSTER-NAME
gcloud config set run/cluster_location REGION
```

   Replace

   - ***CLUSTER-NAME*** with the name you used for your cluster,
   - ***REGION*** with the supported cluster location of your choice.

Understanding the sequence of operations

The flow of data in this tutorial follows these steps:

1. A user uploads an image to a Cloud Storage bucket.

2. Cloud Storage publishes a message about the new file to Pub/Sub.
3. Pub/Sub pushes the message to the Cloud Run service.
4. The Cloud Run service retrieves the image file referenced in the Pub/Sub message.
5. The Cloud Run service uses the Cloud Vision API to analyze the image.
6. If violent or adult content is detected, the Cloud Run service uses ImageMagick to blur the image.
7. The Cloud Run service uploads the blurred image to another Cloud Storage bucket for use.

Subsequent use of the blurred image is left as an exercise for the reader.

Setting up Cloud Storage buckets

1. Create a Cloud Storage bucket for uploading images, where ***INPUT_BUCKET_NAME*** is a globally unique bucket name:

   ```
   gsutil mb gs://INPUT_BUCKET_NAME
   ```

   The Cloud Run service only reads from this bucket.
2. Create a second Cloud Storage bucket to receive blurred images, where ***BLURRED_BUCKET_NAME*** is a globally unique bucket name:

   ```
   gsutil mb gs://BLURRED_BUCKET_NAME
   ```

   The Cloud Run service uploads blurred images to this bucket. Using a separate bucket prevents processed images from re-triggering the service.

In the following steps, you create and deploy a service that processes notification of file uploads to the ***INPUT_BUCKET_NAME***. You turn on notification delivery after you deploy and test the service, to avoid premature invocation of the new service.

Modifying the Pub/Sub tutorial sample code

This tutorial builds on the code assembled in the Using Pub/Sub tutorial (https://cloud.google.com/run/docs/tutorials/pubsub). If you have not yet completed that tutorial, do so now, skipping the cleanup steps, then return here to add image processing behavior.

Adding image processing code

The image processing code is separated from request handling for readability and ease of testing. To add image processing code:

1. Change to the directory of the Pub/Sub tutorial sample code.
2. Add code to import the image processing dependencies, including libraries to integrate with Google Cloud services, ImageMagick, and the file system.

NODE.JS PYTHON GO JAVA

Open a new `image.js` file in your editor, and copy in the following:

run/image-processing/image.js (https://github.com/GoogleCloudPlatform/nodejs-docs-samples/blob/master/run/image-processing/image.js)

GLECLOUDPLATFORM/NODEJS-DOCS-SAMPLES/BLOB/MASTER/RUN/IMAGE-PROCESSING/IMAGE.JS)

FEEDBACK (#)

```
const gm = require('gm').subClass({imageMagick: true});
const fs = require('fs');
const {promisify} = require('util');
const path = require('path');
const vision = require('@google-cloud/vision');

const {Storage} = require('@google-cloud/storage');
const storage = new Storage();
const client = new vision.ImageAnnotatorClient();

const {BLURRED_BUCKET_NAME} = process.env;
```

3. Add code to receives a Pub/Sub message as an event object and control the image processing.

   The event contains data about the originally uploaded image. This code determines if the image needs be blurred by checking the results of a Cloud Vision analysis for violent or adult content.

NODE.JS PYTHON GO JAVA

run/image-processing/image.js (https://github.com/GoogleCloudPlatform/nodejs-docs-samples/blob/master/run/image-processing/image.js)

)GLECLOUDPLATFORM/NODEJS-DOCS-SAMPLES/BLOB/MASTER/RUN/IMAGE-PROCESSING/IMAGE.JS)

FEEDBACK (#)

```
// Blurs uploaded images that are flagged as Adult or Violence.
exports.blurOffensiveImages = async event => {
  // This event represents the triggering Cloud Storage object.
  const object = event;

  const file = storage.bucket(object.bucket).file(object.name);
  const filePath = `gs://${object.bucket}/${object.name}`;

  console.log(`Analyzing ${file.name}.`);

  try {
    const [result] = await client.safeSearchDetection(filePath);
    const detections = result.safeSearchAnnotation || {};

    if (
      // Levels are defined in https://cloud.google.com/vision/docs/reference
      detections.adult === 'VERY_LIKELY' ||
      detections.violence === 'VERY_LIKELY'
    ) {
      console.log(`Detected ${file.name} as inappropriate.`);
      return blurImage(file, BLURRED_BUCKET_NAME);
    } else {
      console.log(`Detected ${file.name} as OK.`);
    }
  } catch (err) {
    console.error(`Failed to analyze ${file.name}.`, err);
    throw err;
  }
};
```

4. Retrieve the referenced image from the Cloud Storage input bucket created above, use ImageMagick to transform the image with a blur effect, and upload the result to the output bucket.

NODE.JS PYTHON GO JAVA

run/image-processing/image.js (https://github.com/GoogleCloudPlatform/nodejs-docs-samples/blob/master/run/image-processing/image.js)

)GLECLOUDPLATFORM/NODEJS-DOCS-SAMPLES/BLOB/MASTER/RUN/IMAGE-PROCESSING/IMAGE.JS)

FEEDBACK (#)

```
// Blurs the given file using ImageMagick, and uploads it to another bucket.
const blurImage = async (file, blurredBucketName) => {
  const tempLocalPath = `/tmp/${path.parse(file.name).base}`;

  // Download file from bucket.
  try {
    await file.download({destination: tempLocalPath});

    console.log(`Downloaded ${file.name} to ${tempLocalPath}.`);
  } catch (err) {
    throw new Error(`File download failed: ${err}`);
  }

  await new Promise((resolve, reject) => {
    gm(tempLocalPath)
      .blur(0, 16)
      .write(tempLocalPath, (err, stdout) => {
        if (err) {
          console.error('Failed to blur image.', err);
          reject(err);
        } else {
          console.log(`Blurred image: ${file.name}`);
          resolve(stdout);
        }
      });
  });

  // Upload result to a different bucket, to avoid re-triggering this functio
  const blurredBucket = storage.bucket(blurredBucketName);
  // Upload the Blurred image back into the bucket.
  const gcsPath = `gs://${blurredBucketName}/${file.name}`;
  try {
    await blurredBucket.upload(tempLocalPath, {destination: file.name});
    console.log(`Uploaded blurred image to: ${gcsPath}`);
  } catch (err) {
    throw new Error(`Unable to upload blurred image to ${gcsPath}: ${err}`);
  }

  // Delete the temporary file.
  const unlink = promisify(fs.unlink);
  return unlink(tempLocalPath);
};
```

Integrating image processing into the Pub/Sub sample code

To modify the existing service to incorporate the image processing code:

1. Add new dependencies for your service, including the Cloud Vision and Cloud Storage client libraries:

   NODE.JS PYTHON GO JAVA

   ```
   npm install --save gm @google-cloud/storage @google-cloud/vision
   ```

2. Add the ImageMagick system package to your container by modifying the `Dockerfile` below the `FROM` statement. If using a "multi-stage" Dockerfile, place this in the final stage.

   DEBIAN/UBUNTU ALPINE

   ```
   # Install Imagemagick into the container image.
   # For more on system packages review the system packages tutorial.
   # https://cloud.google.com/run/docs/tutorials/system-packages#dockerfile
   RUN set -ex; \
     apt-get -y update; \
     apt-get -y install imagemagick; \
     rm -rf /var/lib/apt/lists/*
   ```

Read more about working with system packages in your Cloud Run service in the Using system packages tutorial (https://cloud.google.com/run/docs/tutorials/system-packages).

3. Replace the existing Pub/Sub message handling code with a function call to our new blurring logic.

NODE.JS PYTHON GO JAVA

The `app.js` file defines the Express.js app and prepares received Pub/Sub messages for use. Make the following changes:

- Add code to import the new `image.js` file
- Remove the existing "Hello World" code from the route
- Add code to further validate the Pub/Sub message
- Add code to call the new image processing function

  When you are finished, the code will look like this:

run/image-processing/app.js (https://github.com/GoogleCloudPlatform/nodejs-docs-samples/blob/master/run/image-processing/app.js)

OOGLECLOUDPLATFORM/NODEJS-DOCS-SAMPLES/BLOB/MASTER/RUN/IMAGE-PROCESSING/APP.JS)

FEEDBACK (#)

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.json());

const image = require('./image');

app.post('/', async (req, res) => {
  if (!req.body) {
    const msg = 'no Pub/Sub message received';
    console.error(`error: ${msg}`);
    res.status(400).send(`Bad Request: ${msg}`);
    return;
  }
  if (!req.body.message || !req.body.message.data) {
    const msg = 'invalid Pub/Sub message format';
    console.error(`error: ${msg}`);
    res.status(400).send(`Bad Request: ${msg}`);
    return;
  }

  // Decode the Pub/Sub message.
  const pubSubMessage = req.body.message;
  let data;
  try {
    data = Buffer.from(pubSubMessage.data, 'base64')
      .toString()
      .trim();
    data = JSON.parse(data);
  } catch (err) {
    const msg =
      'Invalid Pub/Sub message: data property is not valid base64 encoded JSO
    console.error(`error: ${msg}: ${err}`);
    res.status(400).send(`Bad Request: ${msg}`);
    return;
  }

  // Validate the message is a Cloud Storage event.
  if (!data.name || !data.bucket) {
    const msg =
      'invalid Cloud Storage notification: expected name and bucket propertie
    console.error(`error: ${msg}`);
    res.status(400).send(`Bad Request: ${msg}`);
    return;
  }

  try {
    await image.blurOffensiveImages(data);
    res.status(204).send();
  } catch (err) {
    console.error(`error: Blurring image: ${err}`);
    res.status(500).send();
  }
});
```

Downloading the complete sample

To retrieve the complete Image Processing code sample for use:

1. Clone the sample app repository to your local machine:

NODE.JS PYTHON GO JAVA

```
git clone https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git
```

Alternatively, you can download the sample (https://github.com/GoogleCloudPlatform/nodejs-docs-samples/archive/master.zip) as a zip file and extract it.

2. Change to the directory that contains the Cloud Run sample code:

NODE.JS PYTHON GO JAVA

```
cd nodejs-docs-samples/run/image-processing/
```

Shipping the code

Shipping code consists of three steps: building a container image with Cloud Build, uploading the container image to Container Registry, and deploying the container image to Cloud Run or Cloud Run for Anthos on Google Cloud.

To ship your code:

1. Build your container and publish on Container Registry:

```
gcloud builds submit --tag gcr.io/PROJECT-ID/pubsub
```

Where ***PROJECT-ID*** is your GCP project ID, and `pubsub` is the name you want to give your service.

Upon success, you should see a SUCCESS message containing the ID, creation time, and image name. The image is stored in Container Registry and can be re-used if desired.

2. Run the following command to deploy your service, using the same service name you used in the Using Cloud Pub/Sub Tutorial (https://cloud.google.com/run/docs/tutorials/pubsub):

NODE.JS PYTHON GO JAVA

```
gcloud run deploy pubsub-tutorial --image gcr.io/PROJECT-ID/pubsub --set-env-v
```

Replace ***PROJECT-ID*** with your GCP project ID. `pubsub` is the container name and `pubsub-tutorial` is the name of the service. Notice that the container image is deployed to the service and region (Cloud Run) or cluster (Cloud Run for Anthos on Google Cloud) that you configured previously under Setting up gcloud defaults (#setting-up-gcloud).

Replace ***BLURRED_BUCKET_NAME*** with your Cloud Storage bucket you created earlier to receive blurred images to set the environment variable.

If deploying to Cloud Run, respond `n`, "No", to the "allow unauthenticated" prompt. By keeping the service private you can rely Cloud Run's automatic Pub/Sub integration to authenticate requests. See Integrating with Pub/Sub (#integrating-pubsub) for more details on how this is configured. See Managing Access (https://cloud.google.com/run/docs/securing/managing-access) for more details on IAM-based authentication.

Wait until the deployment is complete: this can take about half a minute. On success, the command line displays the service URL.

Turning on notifications from Cloud Storage

Configure Cloud Storage to publish a message to a Pub/Sub topic whenever a file (known as an object), is uploaded or changed. Send the notification to the previously created topic so any new file upload will invoke the service.

```
gsutil notification create -t myRunTopic -f json gs://INPUT_BUCKET_NAME
```

The gsutil command is installed as part of the Cloud SDK. `myRunTopic` is the topic you created in the previous tutorial.

Replace ***INPUT_BUCKET_NAME*** with the name you used when you created the buckets (#storage-buckets).

For more details about storage bucket notifications, read object change notifications (https://cloud.google.com/storage/docs/reporting-changes).

Trying it out

1. Upload an offensive image, such as this image of a flesh-eating zombie (https://cdn.pixabay.com/photo/2015/09/21/14/24/zombie-949916_960_720.jpg):

   ```
   gsutil cp zombie.jpg gs://INPUT_BUCKET_NAME
   ```

   where ***INPUT_BUCKET_NAME*** is the Cloud Storage bucket you created earlier for uploading images.

2. Navigate to the service logs:

   a. Navigate to the Cloud Run page in the Google Cloud Console (https://console.cloud.google.com/run)

   b. Click the `pubsub-tutorial` service.

   c. Select the **Logs** tab. Logs might take a few moments to appear. If you don't see them immediately, check again after a few moments.

3. Look for the `Blurred image: zombie.png` message.

4. You can view the blurred images in the ***BLURRED_BUCKET_NAME*** Cloud Storage bucket you created earlier: locate the bucket in the Cloud Storage page in the Google Cloud Console (https://console.cloud.google.com/storage)

Cleaning up

If you created a new project for this tutorial, delete the project (#delete-project). If you used an existing project and wish to keep it without the changes added in this tutorial, delete resources created for the tutorial (#delete-resources).

Deleting the project

The easiest way to eliminate billing is to delete the project that you created for the tutorial.

To delete the project:

Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.

- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an `appspot.com` URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

   GO TO THE MANAGE RESOURCES PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/PRC

2. In the project list, select the project you want to delete and click **Delete** 🗑.
3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

Deleting tutorial resources

1. Delete the Cloud Run service you deployed in this tutorial:

   ```
   gcloud run services delete SERVICE-NAME
   ```

   Where ***SERVICE-NAME*** is your chosen service name.

   You can also delete Cloud Run services from the Google Cloud Console (https://console.cloud.google.com/run).

2. Remove the gcloud default configurations you added during tutorial setup.

   If you use Cloud Run (fully managed), remove the region setting:

   ```
   gcloud config unset run/region
   ```

   If you use Cloud Run for Anthos on Google Cloud, remove the cluster configuration:

   ```
   gcloud config unset run/cluster run/cluster
   gcloud config unset run/cluster run/cluster_location
   ```

3. Remove the project configuration:

   ```
   gcloud config unset project
   ```

4. Delete other Google Cloud resources created in this tutorial:
   - Delete the Pub/Sub topic `myRunTopic` (https://cloud.google.com/pubsub/docs/admin#pubsub-delete-topic-gcloud)
   - Delete the Pub/Sub subscription `myRunSubscription` (https://cloud.google.com/pubsub/docs/admin#delete_subscription)
   - Delete the container image (https://cloud.google.com/container-registry/docs/managing#deleting_images) named `gcr.io/<var>PROJECT-ID</var>/pubsub` from Container Registry
   - Delete the invoker service account `cloud-run-pubsub-invoker@PROJECT-ID.iam.gserviceaccount.com` (https://cloud.google.com/iam/docs/creating-managing-service-accounts#iam-service-accounts-delete-gcloud)
   - Delete the Cloud Storage buckets (https://cloud.google.com/storage/docs/deleting-buckets) created for the placeholders `INPUT_BUCKET_NAME` and `BLURRED_BUCKET_NAME`

What's next

- Learn more about persisting data with Cloud Run via Cloud Storage (https://cloud.google.com/storage/docs)
- Learn how to use Cloud Vision API (https://cloud.google.com/vision/docs/how-to) to detect things besides explicit content.
- Try out other Google Cloud Platform features for yourself. Have a look at our tutorials (https://cloud.google.com/docs/tutorials).

Last updated December 11, 2019.