Cloud IoT

using

Pylon Camera



# 

# Overview

This document explains how to build an end-to-end IoT solution using Pylon Camera on the Google Cloud Platform. The architectural diagrams and sample project are included to help developers get started on similar project. Further down in this document I will be talking about how to set up the project step by step.

Audience

This document is intended for developers who will be working on connecting the Pylon Camera to the Google Cloud through the Internet using IoT solution.

# Introduction to IoT

IoT is the network of physical devices, vehicles, cameras, alarms, home appliances and other items embedded with electronics, software, sensors, actuators, and network connectivity which enables these objects to connect and exchange data through the Internet.

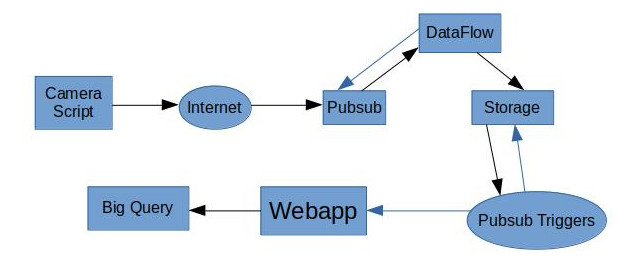
# Components:

1. Pylon Camera
2. Google Cloud Platform

Programing Languages:

Python, Java, C/C++, Flask, HTML, CSS

Architectural Diagram



IoT Solution With Pylon Camera

In this section, I will be talking about how we can create a solution for the Pylon Camera and Google Cloud Platform Services to work together. For this particular IoT solution, it starts with the camera capturing a live image every 2 seconds and storing this image data in a temporary storage buffer. The image data is immediately taken out of the buffer and then get resized to below 256 KB due to each message getting sent by IoT devices being restricted to the maximum size of 256 KB. After the shrinkage of the size of an image, the image data get converted to the base64 data and then get forwarded to the Pub/Sub queue through the Internet.

Meanwhile a Dataflow cron job is running and listening in the background constantly. When this cron job found new messages in the the Pub/Sub queue, it collects a group of base64 messages from the Pub/Sub queue and concatenates these messages and saves them in a file stored in the bucket of Storage on a 5-minute window basis. Dataflow cron job continues to repeat the steps of picking up the data from the Pub/Sub queue and dumping the data to a new file stored in the bucket of Storage when the 5-minute window mark is passed. In other words, a file stores messages published within a 5-minute window. After a 5-minute window mark is passed, a new file is created and store all messages published within the next 5-minute window.

As soon as the Pub/Sub Triggers function detects that a new image file in the bucket of the Storage is being created, it makes a REST call to the webapp application running on the Compute Engine VM. Once the webapp receives the rest call, it uses Vision APIs to analyze the newly created image file in the bucket. If it finds that a person is found in the image, it will persist a record in the Bigquery table called “table\_image\_info” with information about the image in the image file.

Setting up the Projects

Pre-requisite

Before setting up the project, an account with Google is needed to log into the Google Cloud Platform. Please register an account with Google if you don’t have it. Once an account is created successfully, log into the Google Cloud Platform and create a service account, a public and private key pair, a Pub/Sub topic, a Pub/Sub subscription, a device registry, a device, a VM instance, a Storage bucket, and a Dataflow cron job and enable the IoT Core APIs.

Create Service account

1) Open the Service Accounts page in the GCP Console.

2) Click Select a project.

3) Select your project and click Open.

4) Click Create Service Account.

5) Enter a service account name, select a role you wish to grant to the service account, and choose the “Furnish a new private key” option to generate a service account JSON file. Later on this file will be used in connecting to the Google Cloud for authentication so that the project can use various Google Cloud API sets. Make sure you place this file in the folder “PemFiles”.

6) Click on Create to complete. After you create a service account, grant one or more roles to the service account to access the Google Cloud Platform.

Generate RSA Public and Private keys

Log into a box that can run shell scrips and use the commands below to create the private and self-signed public key and put the the keys in the folder “PemFiles”.

mkdir PemFiles

cd PemFiles

openssl req -x509 -newkey rsa:2048 -keyout rsa\_private.pem -nodes -out rsa\_cert.pem -subj "/CN=unused"

Create a Topic

1) Go to the Google Cloud Pub/Sub topics page in the GCP Console.

2) Click Create a topic.

3) Enter a unique Name for your topic.

Create a Subscription

1) Go to the Google Cloud Pub/Sub Subscriptions page in the GCP Console

2) Click New subscription.

3) Type a name for the subscription. Check the Pull box on the delivery type.

4) Click Create.

Create a Device Registry

1) Go to the Device registries page in GCP Console.

2) At the top of the page, click Create device registry.

3) Enter a Registry ID and select a cloud region.

Select both MQTT and HTTP protocols that devices in this registry will use to connect to Cloud IoT Core.

4) Select a Telemetry topic or create a new one. All device telemetry (the event data sent from devices to the cloud) will be published to the Cloud Pub/Sub topic you specify in this field.

Selecting the Device state topic or create a new one is optional. You can leave this one out.

Click Create to continue.

Create a Device

1) On the Registry Details page, click Add device.

2) Enter my-device for the Device ID.

3) Select Allow for Device communication.

4) On the Authentication section, click Add and select public key format RS256\_X509. Copy the data from the public key file rsa\_cert.pem in the PemFiles folder and paste onto the **Public key value** box.

5) Click Add to complete.

Create VM Instance

1) Go to the VM instances page.

2) Select your project and click Continue.

3) Click the Create instance button.

In the Boot disk section, click Change to begin configuring your boot disk.

Create a boot disk no larger than 2 TB to account for the limitations of MBR partitions.

In the OS images tab, choose an image. Click Select.

If desired, change the zone for this instance.

4) Under “Firewall”, check both Allow HTTP traffic and Allow HTTPS traffic boxes. This will external access to make call to REST apis.

5) Click the Create button to create and start the instance.

Create Storage Bucket

1. Open the Cloud Storage browser in the Google Cloud Platform Console.

2. Click Create bucket.

3. In the Create bucket dialog, specify:

A Name subject to the bucket name requirements.

The default Storage class for the bucket.

A Location where the bucket data will be stored

4. Click Create.

Create Dataflow Cron Job

**Create dataflow template with google provided “Cloud Pub/Sub to to Cloud Storage Text” template.**

1. Go to the Cloud Dataflow page in the Cloud Platform Console.
2. Click CREATE JOB FROM TEMPLATE.
3. Select the Google-provided template that you want to execute from the Cloud Dataflow template drop-down menu.
4. Choose “Cloud PubSub to GCS text” for Dataflow template
5. Enter a job name in the Job Name field. Your job name must match the regular expression [a-z]([-a-z0-9]{0,38}[a-z0-9])? to be valid.
6. Enter your parameter values in the provided parameter fields. You should not need the Additional Parameters section when you use a Google-provided template.
7. Click Run Job.

**Create a custom dataflow template**

1) Have google cloud sdk install on your computer

2) Download and install maven compiler or eclipse

3) Run a cd command to the dataflowtemplate folder

4) Replace all the values of the parameters stagingLocation, project, outputDirectory, topic, outputFilenamePrefix, outputFilenameSuffix, windowDuration, numShards, dataflowJobFile and gcpTempLocation with yours and run the command below

/path\_to\_maven\_folder/bin/mvn compile exec:java -Dexec.mainClass=com.google.dataflowtemplates.SubPubToGCSTemplate -Dexec.args="--project=cloud-iot-testing-185623 --stagingLocation=path\_to\_storage\_bucket/staging --runner=DataflowRunner --outputDirectory=path\_to\_storage\_bucket/output/ --topic=projects/cloud-iot-testing-185623/topics/cloud-iot-topic1 --outputFilenamePrefix=base64 --outputFilenameSuffix=-txt --windowDuration=10s --numShards=10 --dataflowJobFile=path\_to\_storage\_bucket/template/pubsubtogcs --gcpTempLocation=path\_to\_storage\_bucket/tmp --numWorkers=1 --jobName=PubSubtoGCS"

Example:

/home/khanhl/apache-maven-3.5.2/bin/mvn compile exec:java -Dexec.mainClass=com.google.dataflowtemplates.SubPubToGCSTemplate -Dexec.args="--project=cloud-iot-testing-185623 --stagingLocation=gs://dataflow-cloud-iot-testing-185623/staging --runner=DataflowRunner --outputDirectory=gs://dataflow-cloud-iot-testing-185623/output/ --topic=projects/cloud-iot-testing-185623/topics/cloud-iot-topic1 --outputFilenamePrefix=base64 --outputFilenameSuffix=-txt --windowDuration=10s --numShards=10 --dataflowJobFile=gs://dataflow-cloud-iot-testing-185623/template/pubsubtogcs --gcpTempLocation=gs://dataflow-cloud-iot-testing-185623/tmp --numWorkers=1 --jobName=PubSubtoGCS"

Setting up the Camera Project on the UpBoard

1) Go to<https://emutex.com/products/ubilinux> and download the UBILINUX 4 iso image file for the UP BOARD. Use Etcher or other tool to burn the image to the USB drive.

2) Plug the monitor, USB drive, USB keyboard and USB mouse, RJ 45 network cable with Internet access, and power cable into the Up Board. Make sure the Up Board is connected to the internet.

3) Follow the instructions that can be found on the USB drive on how to boot up from the USB drive and install the UBILINUX 4 OS.

4) After the UBILINUX 4 OS is installed, run the commands below to install the packages.

sudo apt-get install python-pip python-dev build-essential python-setuptools python-tk python-qt4 git

sudo pip install google-cloud

sudo pip install oauth2client

sudo pip install apiclient

sudo pip install google-api-python-client

sudo pip install cryptography

sudo pip install requests

sudo pip install google-gax

sudo pip install flask-login

sudo pip install Flask-JWT

sudo pip install python-dateutil

sudo pip install pyjwt

sudo pip install Flask

sudo pip install flask

sudo pip install Pillow

5) Download the camera\_script project below from github.com

git clone https://github.com/khanhhale/IoTProjectWithPylonCamera.git

6) From the camera\_script project folder, install Pylon SDK, protobuf3-to-dict-0.1.5 API, and PylonCamera scripts using command below. The pylon SDK is available for download at<https://www.baslerweb.com/en/support/downloads/software-downloads/>.

a) download the SDK compressed file at<https://www.baslerweb.com/en/support/downloads/software-downloads/> and extract the pylon5 folder to /opt.

sudo tar -C /opt -xzf pylonSDK\*.tar.gz

b) cd /path\_to\_camera\_script\_folder/protobuf3-to-dict-0.1.5

python ./setup.py install

c) cd /path\_to\_camera\_script\_folder/PylonCamera

python ./setup.py install

7) Copy the PemFile folder containing the Service JSON file and Public and Private keys

into the camera project folder.

8) Add environment variable for service account JSON file to the linux startup file or shell login file.

export GOOGLE\_APPLICATION\_CREDENTIALS="/path\_to\_camera\_script\_folder/PemFiles/cloud\_iot\_service\_account\_json\_file"

Note:

1) Replace cloud\_iot\_service\_account\_json\_file with your service account json file

2) Replace path\_to\_camera\_script\_folder with your path to camera\_script folder

9) Add the script file camera\_startup within the camera folder to the /etc/init.d folder. This will start up the script next time the Up Board boots up. Please also remember to replace the value for the parameters --project\_id, --registry\_id,- -device\_id, --private\_key\_file, --public\_key\_file, --credential, --cloud\_region, send\_message\_interval, and --registry\_id that comes after the equal sign and change the path to the camera\_script folder for the code below that appears within the camera\_startup file.

python /path\_to\_camera\_script\_folder/Main.py --project\_id=cloud-iot-testing-185623 --registry\_id=cloud-iot-registry1 --device\_id=iot-device1 --algorithm=RS256 --private\_key\_file=/path\_to\_camera\_script\_folder/PemFiles/rsa\_private.pem --public\_key\_file=/path\_to\_camera\_script\_folder/PemFiles/rsa\_cert.pem --credential=/path\_to\_camera\_script\_folder/PemFiles/cloud\_iot\_service\_account\_json\_file --cloud\_region=us-central1 --message\_type=event --message\_data\_type=image\_string --send\_message\_interval=2 &

Note:

1) Replace cloud\_iot\_service\_account\_json\_file with your service account json file

2) Replace path\_to\_camera\_script\_folder with your path to camera\_script folder

For convenience, please use the codes below. What the code below will do is it copies the camera\_startup file to the /etc/init.d folder and add default startup and stop levels.

sudo cp /path\_to\_camera\_script\_folder/camera\_startup /etc/init.d

sudo chmod +x /etc/init.d/camera\_startup

sudo update-rc.d camera\_startup defaults

Setting up the webapp Project on the VM Instance

1) Run the commands below to install the packages.

sudo apt-get install python-pip python-dev build-essential python-setuptools python-tk python-qt4 git

sudo pip install google-cloud

sudo pip install oauth2client

sudo pip install apiclient

sudo pip install google-api-python-client

sudo pip install cryptography

sudo pip install requests

sudo pip install google-gax

sudo pip install flask-login

sudo pip install Flask-JWT

sudo pip install python-dateutil

sudo pip install pyjwt

sudo pip install Flask

sudo pip install flask

sudo pip install Pillow

sudo pip install --upgrade pyasn1-modules

2) Download the webapp project below from github.com

git clone https://github.com/khanhhale/IoTProjectWithPylonCamera.git

3) Copy the PemFile folder containing the Service JSON file and Public and Private keys

into the webapp project folder.

4) Add environment variable for service account JSON file to the linux startup file or shell login file.

export GOOGLE\_APPLICATION\_CREDENTIALS="/path\_to\_webapp\_folder/PemFiles/cloud\_iot\_service\_account\_json\_file"

Note:

1) Replace cloud\_iot\_service\_account\_json\_file with your service account json file

2) Replace path\_to\_webapp with your path to webapp folder

5) Add the webapp startup code to the startup file on the VM instance. Please also note to replace the value for the parameters --project\_id, --registry\_id,- -device\_id, --private\_key\_file, --public\_key\_file, --credential, --pubsub\_topic, and --cloud\_region that comes after the equal sign.

Add the script file webapp\_startup within the camera folder to the /etc/init.d folder. This will start up the script next time the Up Board boots up. Please also remember to replace the value for the parameters --project\_id, --registry\_id,- -device\_id, --private\_key\_file, --public\_key\_file, --credential, --cloud\_region, --pubsub\_topic that comes after the equal sign and change the path to the webapp folder for the code below that appears within the webapp\_startup file.

python /path\_to\_webapp\_folder/Main.py --project\_id=cloud-iot-testing-185623 --registry\_id=cloud-iot-registry1 --device\_id=iot-device1 --algorithm=RS256 --private\_key\_file=/path\_to\_webapp\_folder/PemFiles/rsa\_private.pem --public\_key\_file=/path\_to\_webapp\_folder/PemFiles/rsa\_cert.pem --credential=/path\_to\_webapp\_folder/PemFiles/cloud\_iot\_service\_account\_json\_file --cloud\_region=us-central1 --pubsub\_topic=cloud-iot-topic1 --message\_type=event --message\_data\_type=image\_string &

Note:

1) Replace cloud\_iot\_service\_account\_json\_file with your service account json file

2) Replace path\_to\_webapp with your path to webapp folder

For convenience, please use the codes below. What the code below will do is it copies the camera\_startup file to the /etc/init.d folder and add default startup and stop levels.

sudo cp /path\_to\_webapp\_folder/webapp\_startup /etc/init.d

sudo chmod +x /etc/init.d/webapp\_startup

sudo update-rc.d webapp\_startup enable

sudo update-rc.d webapp\_startup defaults

Create a bigquery data table

Create a bigquery table called table\_image\_info with the schema below:

[

{

"name": "fileName",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "filePath",

"type": "STRING",

"mode": "NULLABLE"

},

{

"name": "dateCreated",

"type": "DATETIME",

"mode": "NULLABLE"

},

{

"name": "dateDeleted",

"type": "DATETIME",

"mode": "NULLABLE"

},

{

"name": "personDetected",

"type": "BOOLEAN",

"mode": "NULLABLE"

}

]

Setting up the Pub/Sub triggers function

1) Open the Cloud Function page in the GCP Console

2) Create Pub/Sub triggers function and update the index.js and package.json with the two files in the cloud functions folder.

a) Update the options object with your own configurations.

b) Deploy the Pubsub triggers function.