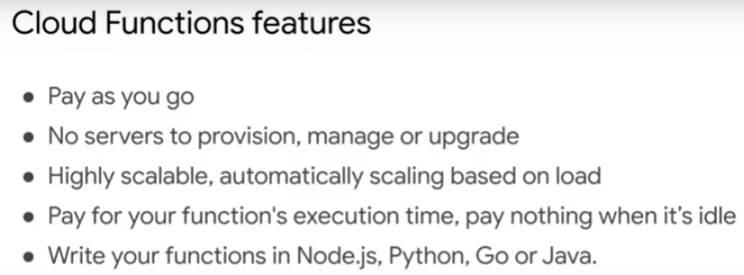
Course: Securing and Integrating Components of your Application

Module 4: Using Cloud Functions for Event-driven Processing

**Cloud Functions Concepts**

Characteristics

* serverless environment
* microservices architecture. Integrates application components and data sources.
* small piece of logic executed on demand, in response to an event
* scalable
* can be glue between applications
* pay-as-you-go. Priced according to how long function runs (to nearest 100ms), the number of times invoked, and resources provisioned for fn. Pay nothing when fn service not used.
* executed in language-specific runtimes



To learn:

* triggers of Cloud Functions
  + asynchronous events and synchronous HTTP invocations
* develop and deploy Cloud Functions for event-driven processing
* how to handle logging, monitoring and error reporting

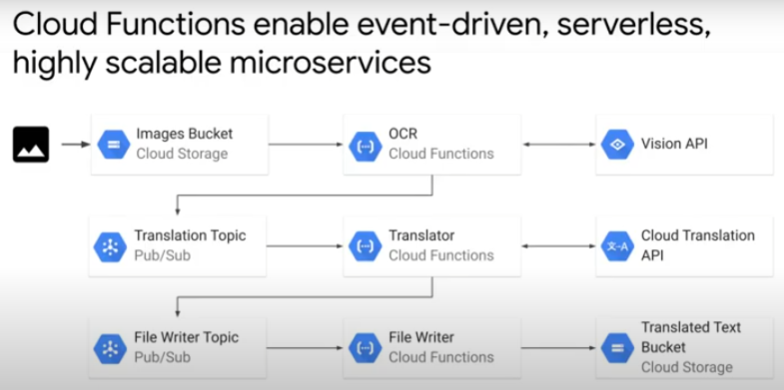
What are events?

* Activities with Google Cloud services, e.g. upload file to Cloud Storage bucket
* messages published to Pub/Sub topic
* events in external services, e.g. commits to GitHub repository

What can Cloud Functions do?

* invoke other APIs
* write to cloud database

Example

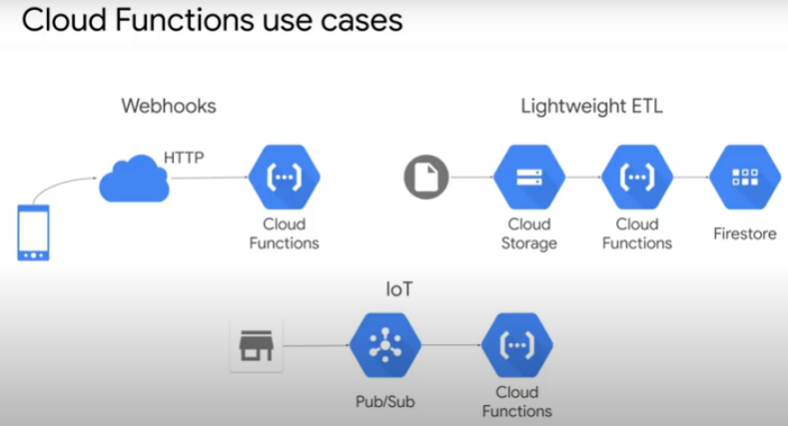


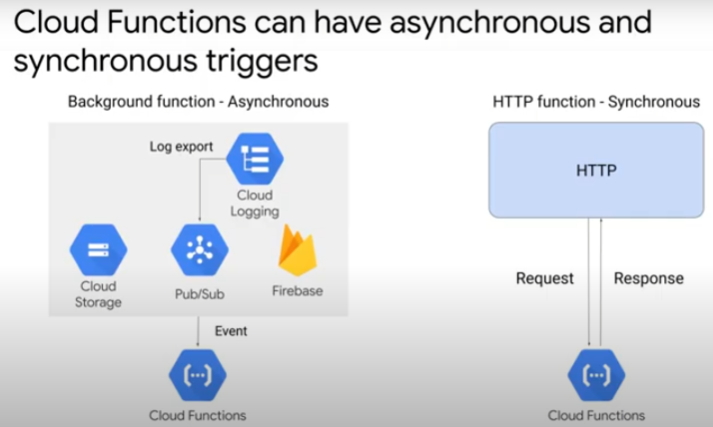
1. Upload image containing text to bucket in Google Cloud Storage (event)
2. OCR Cloud Function triggered
   * invokes Vision API to identify and extract text from image
   * publishes/queues information (text) to Pub/Sub Translation topic in the form of a notification message
3. Translator Cloud Function subscribes to Translation topic
   * Pub/Sub topic pushes message to subscriber
   * triggers Translator Cloud Function
   * Cloud Function invokes Cloud Translation API to translate text
4. translated text queued up in File Writer Pub/Sub topic
5. Fie Writer Cloud Function subscribes to File Writer topic
   * Pub/Sub pushes message to Cloud Function
   * Cloud Function writes translated text as seperate files to bucket in Cloud Storage

* All components of applications use fully managed services and APIs.
  + Cloud Storage, Cloud Function, Pub/Sub
  + Vision API, Translation API
  + these services scale automatically according to volume of incoming data
  + architecture is scalable, reliable. Developer focus more on application code.

Use cases for Cloud Functions

1. Serve as webhooks
   * webhooks are automated messages (containing payload) pushed to unique URL when something happens.
   * only when invoked by direct HTTP request to Cloud Function
   * e.g. when there is a commit to a Git repository
   * similarly, external/internal clients make direct HTTP calls to invoke microservices in Cloud Function
2. For light-weight ETL (extract, transform, load) operations
   * e.g. file uploaded to Cloud Storage -> Cloud Function triggered to transform and upload data into Firestore
3. Process IoT streaming data or other application messages that have been published to a Pub/Sub topic





Triggers – asynchronous and synchronous

Asynchronous

* triggered in response to events, e.g. those in Cloud Storage, Pub/Sub, Firebase
* called background functions

Synchronous

* invoked directly by HTTP request. Response by Cloud Function.
* Called HTTP functions
* note: default timeout of 60s for Cloud Functions
  + need to ensure function will complete execution before timeout
  + minimise execution time for better user experience

**Writing, Deploying and Monitoring Cloud Functions**

Recap: There are 2 types of Cloud Functions

1. Asynchronous: background functions
2. Synchronous: HTTP functions

Writing Cloud Function

Using node.js example

* write the function in **index.js** file (main file)
  + can include other files and reference them in index.js
* For background function: function xxx (**event, callback**) { ...}
  + event and callback functions as input parameters
* For HTTP function: function yyy (**request, response**) { ... }
  + request and response objects as input parameters
* declare dependencies in **package.json**. Do not upload dependencies as zip files.
  + Cloud Function service automatically install dependencies before running codes
* Each Cloud Function has access to local disk mount point (/temp)
  + writing to /temp will use memory allocated to the fn
  + for temporary file processing
  + make sure to code for clean up of any temporary files
  + To inspect memory usage, check Cloud Console > Cloud Functions > function name

Deploying Cloud Function



* deploy from local machine using Cloud Console or gCloud command
* gCloud command automatically zips codes and uploads to staging bucket you specify
  + gcloud functions deploy [function name] –source=. –other flags
  + https://cloud.google.com/functions/docs/create-deploy-gcloud#functions-deploy-command-nodejs
* staging bucket can be Cloud Storage bucket, other Cloud source repository, GitHub or Bitbucket repository

Logging, Error Reporting and Monitoring

Cloud Logging

* view outputs for
  + info log (console.log)
  + error log (console.error)
  + debug log (internal system messages)

Error Reporting

* Cloud Functions automatically captures errors and uncaught errors in Error Reporting
* If do not want to return error to caller in code, can output to Error Reporting

Cloud Function in Cloud Console

* see metrics for fns
* number of invocations, execution time and memory usage

