## Perform Foundational Infrastructure Tasks in Google Cloud

* Learn how to build and connect storage-centric cloud infrastructure using the basic capabilities of the of the following technologies: Cloud Storage, Identity and Access Management, Cloud Functions, and Pub/Sub.
* Badge for Perform Foundational Infrastructure Tasks in Google Cloud
* When you complete this activity, you can earn the badge displayed above! View all the badges you have earned by visiting your profile page. Boost your cloud career by showing the world the skills you have developed!
* 1.0 Cloud Storage: Qwik Start - Cloud Console
  + This hands-on lab shows you how to perform basic tasks in Cloud Storage using the Google Cloud Console. Watch the short video Google Cloud Storage: Massive Scalability Plus More.
* 2.0 Cloud Storage: Qwik Start - CLI/SDK
  + This hands-on lab shows you how to perform basic tasks in Cloud Storage using the gsutil command-line tool. Watch the short video Google Cloud Storage: Massive Scalability Plus More.
* 3.0 Cloud IAM: Qwik Start
  + Google Cloud IAM unifies access control for Cloud Platform services into a single system to present a consistent set of operations. Watch the short video Manage Access Control with Google Cloud IAM.
* 4.0 Cloud Monitoring: Qwik Start
  + This lab shows you how to monitor a Google Compute Engine virtual machine (VM) instance with Cloud Monitoring. Watch the short videos Monitor Health of All Your Cloud Apps with Google Cloud monitoring and Monitor a VM Instance with Cloud monitoring, GCP Essentials.
* 5.0 Cloud Functions: Qwik Start - Console
  + This hands-on lab shows you how to create and deploy a Cloud Function using the Cloud Platform Console. Watch the short video Connect & Extend GCP Services with Google Cloud Functions.
* 6.0 Cloud Functions: Qwik Start - Command Line
  + This hands-on lab shows you how to create and deploy a Cloud Function using the Cloud Platform Command Line. Watch the short video Connect & Extend GCP Services with Google Cloud Functions.
* 7.0 Google Cloud Pub/Sub: Qwik Start - Console
  + This hands-on lab shows you how to publish and consume messages with a pull subscriber, using the C
  + GC Platform Console. Watch the short video Simplify Event Driven Processing with Cloud Pub/Sub.
* 8.0 Google Cloud Pub/Sub: Qwik Start - Command Line
  + This hands-on lab shows you how to publish and consume messages with a pull subscriber, using the Google Cloud command line. Watch the short video Simplify Event Driven Processing with Cloud Pub/Sub.
* 9.0 Google Cloud Pub/Sub: Qwik Start - Python
  + learn about Pub/Sub and how to create a topic and subscriber with a Python script, and then publish and view messages. Watch the short video Simplify Event Driven Processing with Cloud Pub/Sub.
* 10.0 Perform Foundational Infrastructure Tasks in Google Cloud: Challenge Lab
  + This challenge lab tests your skills and knowledge from the labs in the Baseline: Infrastructure quest. You should be familiar with the content of labs before attempting this lab.

## 1.0 Cloud Storage: Qwik Start - Cloud Console - https://www.cloudskillsboost.google/focuses/1760?parent=catalog

* Overview
  + Cloud Storage allows world-wide storage and retrieval of any amount of data at any time. You can use Cloud Storage for a range of scenarios including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download. learn how to use the Cloud Console to create a storage bucket, then upload objects, create folders and subfolders, and make those objects publicly accessible.
* Create a Bucket
  + Buckets are the basic containers that hold your data in Cloud Storage. To create a bucket:
    - In the Cloud Console, go to Navigation menu > Cloud Storage > Browser. Click Create Bucket:
    - Enter your bucket information and click Continue to complete each step:
      * Name your bucket: Enter a unique name for your bucket. For this lab, you can use your Project ID as the bucket name because it will always be unique.
      * Choose Region for Location type and us-east1 (South Carolina) for Location.
      * Choose Standard for default storage class. Choose Uniform for Access control.
      * Click Create.
  + Test your Understanding
    - Every bucket must have a unique name across the entire Cloud Storage namespace. True
    - Cloud Storage offers which storage classes: Standard, Nearline, Coldline, Archive
* Upload an object into the Bucket
  + In the Cloud Storage browser page, click the name of the bucket that you created.
  + In the Objects tab, click Upload files.
  + After the upload completes, you should see the file name and information about the file, such as its size and type.
  + Object names must be unique only within a given bucket. True
* Share a Bucket Publicly
  + To allow public access to the bucket and create a publicly accessible URL for the image:
  + Click the Permissions tab above the list of files.
  + Ensure the view is set to Principals. Click Add to view the Add principals pane.
  + In the New principals box, enter allUsers.
  + In the Select a role drop-down, select Cloud Storage > Storage Object Viewer.
  + Click Save. In the Are you sure you want to make this resource public? window, click Allow public access.
  + To verify, click the Objects tab to return to the list of objects. Your object's Public access column should read Public to internet.
  + https://storage.googleapis.com/YOUR\_BUCKET\_NAME/kitten.png
* Create Folders
  + In the Objects tab, click Create folder. Enter folder1 for Name and click Create.
  + You should see the folder in the bucket with an image of a folder icon to distinguish it from objects.
  + Create a subfolder and upload a file to it: Click folder1. Click Create folder.
  + Enter folder2 for Name and click Create. Click folder2. Click Upload files.
* Delete a folder
  + Click the arrow next to Bucket details to return to the buckets level.
  + Select the bucket. Select the checkbox next to folder1. Click on the Delete button.
  + In the window that opens, type DELETE to confirm the deletion of the folder.
  + Click Delete to permanently delete the folder and all objects and subfolders in it.

## 2.0 Cloud Storage: Qwik Start - CLI/SDK - https://www.cloudskillsboost.google/focuses/569?parent=catalog

* Create a bucket
  + Name your bucket: Enter a unique name for your bucket.
  + Location type: Multi-region Location: us (multiple regions in United States)
  + Default Storage class: Standard
  + Uncheck Enforce public access prevention on this bucket checkbox under Prevent public access.
  + Choose Fine-grained under Access Control.
  + Test your Understanding
    - Each bucket has a default storage class, which you can specify when you create your bucket. True
* Upload an object into your bucket
  + First, download this image to a temporary instance (ada.jpg) in Cloud Shell:
    - curl https://upload.wikimedia.org/wikipedia/commons/thumb/a/a4/Ada\_Lovelace\_portrait.jpg/800px-Ada\_Lovelace\_portrait.jpg --output ada.jpg
  + Use the gsutil cp command to upload the image from the location where you saved it to the bucket you created: gsutil cp ada.jpg gs://YOUR-BUCKET-NAME
  + Now remove the downloaded image: rm ada.jpg
* Download an object from your bucket
  + Use the gsutil cp command to download the image you stored in your bucket to Cloud Shell:
    - gsutil cp -r gs://YOUR-BUCKET-NAME/ada.jpg .
* Copy an object to a folder in the bucket
  + Use the gsutil cp command to create a folder called image-folder and copy the image (ada.jpg) into it:
    - gsutil cp gs://YOUR-BUCKET-NAME/ada.jpg gs://YOUR-BUCKET-NAME/image-folder/
* List contents of a bucket or folder
  + Use the gsutil ls command to list the contents of the bucket: gsutil ls gs://YOUR-BUCKET-NAME
* List details for an object
  + Use the gsutil ls command, with the -l flag to get some details about the image file you uploaded to your bucket: gsutil ls -l gs://YOUR-BUCKET-NAME/ada.jpg
* Make your object publicly accessible
  + Use the gsutil acl ch command to grant all users read permission for the object stored in your bucket:
    - gsutil acl ch -u AllUsers:R gs://YOUR-BUCKET-NAME/ada.jpg
  + An access control list (ACL) is a mechanism you can use to define who has access to your buckets and objects. True
* Remove public access gsutil acl ch -d AllUsers gs://YOUR-BUCKET-NAME/ada.jpg
  + You can stop publicly sharing an object by removing permission entry that have: allUsers
* Delete objects
  + Use the gsutil rm command to delete an object - the image file in your bucket:
    - gsutil rm gs://YOUR-BUCKET-NAME/ada.jpg

## 3.0 Cloud IAM: Qwik Start - https://www.cloudskillsboost.google/focuses/551?parent=catalog

* Overview
* Google Cloud's Identity and Access Management (IAM) service lets you create and manage permissions for Google Cloud resources. Cloud IAM unifies access control for Google Cloud services into a single system and provides a consistent set of operations. In this hands-on lab you learn how to assign a role to a second user and remove assigned roles associated with Cloud IAM. More specifically, you sign in with 2 different sets of credentials to experience how granting and revoking permissions works from Google Cloud Project Owner and Viewer roles.
* Explore editor roles
  + Now switch to the Username 2 console.
  + Navigate to the IAM & Admin console, select Navigation menu > IAM & Admin > IAM.
  + Search through the table to find Username 1 and Username 2 and examine the roles they are granted. You should see something like this:
  + Graphical user interface, text, application, email

    Description automatically generated
    - Username 2 has the "Viewer" role granted to it.
    - The +ADD button at the top is grayed out—if you try to click on it you get the following message:
* Prepare a resource for access testing
  + Ensure that you are in the Username 1 Cloud Console.
  + Create a bucket
    - Name: globally unique name (create it yourself!) and click CONTINUE.
    - Location Type: Multi-Region
    - Click CREATE.
  + Upload a sample file
  + Verify project viewer access
    - Switch to the Username 2 console.
    - From the Console, select Navigation menu > Cloud Storage > Browser. Verify that this user can see the bucket.
* Remove project access
  + Switch to the Username 1 console.
  + Remove Project Viewer for Username 2
    - Select Navigation menu > IAM & Admin > IAM. Then click the pencil icon next to Username 2.
    - Remove Project Viewer access for Username 2 by clicking the trashcan icon next to the role name. Then click SAVE.
  + Verify that Username 2 has lost access
    - Switch to Username 2 Cloud Console. Ensure that you are still signed in with Username 2's credentials and that you haven't been signed out of the project after permissions were revoked. If signed out, sign in back with the proper credentials.
    - Navigate back to Cloud Storage by selecting Navigation menu > Cloud Storage > Browser.
    - You should see a permission error.
* Add Storage permissions
  + Copy Username 2 name from the Qwiklabs "Connection Details" panel.
  + Switch to Username 1 console. Ensure that you are still signed in with Username 1's credentials. If you are signed out, sign in back with the proper credentials.
  + In the Console, select Navigation menu > IAM & Admin > IAM.
  + Click + ADD button and paste the Username 2 name into the New principals field.
  + In the Select a role field, select Cloud Storage > Storage Object Viewer from the drop-down menu.
  + Click SAVE.
  + Verify access
    - Switch to the Username 2 console. You'll still be on the Storage page.
* Click the Activate Cloud Shell
  + gsutil ls gs://[YOUR\_BUCKET\_NAME]
  + gs://[YOUR\_BUCKET\_NAME]/sample.txt
    - As you can see, you gave Username 2 view access to the Cloud Storage bucket.

## 4.0 Cloud Monitoring: Qwik Start - https://www.cloudskillsboost.google/focuses/10599?parent=catalog

* Overview
  + Cloud Monitoring provides visibility into the performance, uptime, and overall health of cloud-powered applications. Cloud Monitoring collects metrics, events, and metadata from Google Cloud, Amazon Web Services, hosted uptime probes, application instrumentation, and a variety of common application components including Cassandra, Nginx, Apache Web Server, Elasticsearch, and many others. Cloud Monitoring ingests that data and generates insights via dashboards, charts, and alerts. Cloud Monitoring alerting helps you collaborate by integrating with Slack, PagerDuty, HipChat, Campfire, and more.
  + This hands-on lab shows you how to monitor a Compute Engine virtual machine (VM) instance with Cloud Monitoring. You'll also install monitoring and logging agents for your VM which collects more information from your instance, which could include metrics and logs from 3rd party apps.
* Create a Compute Engine instance
  + Field Value
  + Name lamp-1-vm
  + Region us-central1 (Iowa)
  + Zone us-central1-a
  + Series N1
  + Machine type n1-standard-2
  + Firewall check Allow HTTP traffic
  + Click Create.
* Add Apache2 HTTP Server to your instance
  + In the Cloud Console, click SSH to open a terminal to your instance.
  + Run the following commands in the SSH window to set up Apache2 HTTP Server:
    - sudo apt-get update sudo apt-get install apache2 php7.0
      * When asked if you want to continue, enter Y.
    - sudo service apache2 restart
* Create a Monitoring Metrics Scope
  + In the Cloud Console, click Navigation menu > Monitoring.
  + When the Monitoring Overview page opens, your metrics scope project is ready.
* Install the Monitoring and Logging agents
  + Agents collect data and then send or stream info to Cloud Monitoring in the Cloud Console.
  + The Cloud Monitoring agent is a collectd-based daemon that gathers system and application metrics from virtual machine instances and sends them to Monitoring. By default, the Monitoring agent collects disk, CPU, network, and process metrics. Configuring the Monitoring agent allows third-party applications to get the full list of agent metrics. See Cloud Monitoring agent overview for more information.
  + In this section, you install the Cloud Logging agent to stream logs from your VM instances to Cloud Logging. Later in this lab, you see what logs are generated when you stop and start your VM.
  + It is best practice to run the Cloud Logging agent on all your VM instances.
  + Install agents on the VM:
    - Run the Monitoring agent install script command in the SSH terminal of your VM instance to install the Cloud Monitoring agent.
      * curl -sSO https://dl.google.com/cloudagents/add-monitoring-agent-repo.sh
      * sudo bash add-monitoring-agent-repo.sh
      * sudo apt-get update
      * sudo apt-get install stackdriver-agent
      * When asked if you want to continue, enter Y.
    - Run the Logging agent install script command in the SSH terminal of your VM instance to install the Cloud Logging agent
      * curl -sSO https://dl.google.com/cloudagents/add-logging-agent-repo.sh
      * sudo bash add-logging-agent-repo.sh
      * sudo apt-get update
      * sudo apt-get install google-fluentd
* Create an uptime check
  + Uptime checks verify that a resource is always accessible. For practice, create an uptime check to verify your VM is up.
  + In the Cloud Console, in the left menu, click Uptime checks, and then click Create Uptime Check.
  + Set the following fields:
    - Title: Lamp Uptime Check, then click Next.
    - Protocol: HTTP
    - Resource Type: Instance
    - Applies to: Single, lamp-1-vm
    - Path: leave at default
    - Check Frequency: 1 min
  + Click on Next to leave the other details to default and click Test to verify that your uptime check can connect to the resource.
  + When you see a green check mark everything can connect. Click Create.
* Create an alerting policy
  + Use Cloud Monitoring to create one or more alerting policies.
  + In the left menu, click Alerting, and then click Create Policy. Click Add Condition.
  + Set the following in the panel that opens, leave all other fields at the default value.
    - Target: Start typing "VM" in the resource type and metric field, and then select:
    - Resource Type: VM Instance (gce\_instance)
    - Metric: Type "network", and then select Network traffic (gce\_instance+1). Be sure to choose the Network traffic resource with agent.googleapis.com/interface/traffic:
    - Configuration
      * Condition: is above
      * Threshold: 500
      * For: 1 minute
  + Click ADD. Click on Next.
  + Click on drop down arrow next to Notification Channels, then click on Manage Notification Channels.
    - A Notification channels page will open in new tab.
  + Scroll down the page and click on ADD NEW for Email.
  + In Create Email Channel dialog box, enter your personal email address in the Email Address field and a Display name.
  + Click on Save. Go back to the previous Create alerting policy tab.
  + Click on Notification Channels again, then click on the Refresh icon to get the display name you mentioned in the previous step.
  + Now, select your Display name and click OK. Click Next.
  + Mention the Alert name as Inbound Traffic Alert.
  + Add a message in documentation, which will be included in the emailed alert. Click on Save.
* Create a dashboard and chart
  + You can display the metrics collected by Cloud Monitoring in your own charts and dashboards. In this section you create the charts for the lab metrics and a custom dashboard.
  + In the left menu select Dashboards, and then Create Dashboard.
  + Name the dashboard Cloud Monitoring LAMP Qwik Start Dashboard.
  + Add the first chart
    - Click Line option in Chart library.
    - Name the chart title CPU Load.
    - Set the Resource type to VM Instance.
    - Set the Metric CPU load (1m) (You may need to uncheck the only show active box). Refresh the tab to view the graph.
  + Add the second chart
    - Click + Add Chart and select Line option in Chart library.
    - Name this chart Received Packets.
    - Set the resource type to VM Instance.
    - Set the Metric Received packets (gce\_instance). Refresh the tab to view the graph.
    - Leave the other fields at their default values. You see the chart data.
* View your logs
  + Cloud Monitoring and Cloud Logging are closely integrated. Check out the logs for your lab.
  + Select Navigation menu > Logging > Logs Explorer.
  + Select the logs you want to see, in this case, you select the logs for the lamp-1-vm instance you created at the start of this lab: Click on Resource.
  + Select VM Instance > lamp-1-vm in the Resource drop-down menu. Click Add.
  + Leave the other fields with their default values. Click the Stream logs.
    - You see the logs for your VM instance:
* Check out what happens when you start and stop the VM instance.
  + To best see how Cloud Monitoring and Cloud Logging reflect VM instance changes, make changes to your instance in one browser window and then see what happens in the Cloud Monitoring, and then Cloud Logging windows.
  + Open the Compute Engine window in a new browser window. Select Navigation menu > Compute Engine, right-click VM instances > Open link in new window.
  + Move the Logs Viewer browser window next to the Compute Engine window. This makes it easier to view how changes to the VM are reflected in the logs.
  + In the Compute Engine window, select the lamp-1-vm instance, click the three vertical dots at the top of the screen and then click Stop, and then confirm to stop the instance.
    - It takes a few minutes for the instance to stop.
  + Watch in the Logs View tab for when the VM is stopped.
  + In the VM instance details window, click the three vertical dots at the top of the screen and then click Start/resume, and then confirm. It will take a few minutes for the instance to re-start. Watch the log messages to monitor the start up.
* Check the uptime check results and triggered alerts
  + In the Cloud Logging window, select Navigation menu > Monitoring > Uptime checks. This view provides a list of all active uptime checks, and the status of each in different locations.
    - You will see Lamp Uptime Check listed. Since you have just restarted your instance, the regions are in a failed status. It may take up to 5 minutes for the regions to become active. Reload your browser window as necessary until the regions are active.
  + Click the name of the uptime check, Lamp Uptime Check.
    - Since you have just restarted your instance, it may take some minutes for the regions to become active. Reload your browser window as necessary.
  + Check if alerts have been triggered
    - In the left menu, click Alerting. You see incidents and events listed in the Alerting window.
    - Check your email account. You should see Cloud Monitoring Alerts.
      * Note: Remove the email notification from your alerting policy. The resources for the lab may be active for a while after you finish, and this may result in a few more email notifications getting sent out.

## 5.0 Cloud Functions: Qwik Start – Console - https://www.cloudskillsboost.google/focuses/1763?parent=catalog

* Overview
  + Cloud Functions is a serverless execution environment for building and connecting cloud services. With Cloud Functions you write simple, single-purpose functions that are attached to events emitted from your cloud infrastructure and services. Your Cloud Function is triggered when an event being watched is fired. Your code executes in a fully managed environment. There is no need to provision any infrastructure or worry about managing any servers.
  + Cloud Functions are written in Javascript and execute in a Node.js environment on Google Cloud. You can take your Cloud Function and run it in any standard Node.js runtime which makes both portability and local testing a breeze.
* Connect and Extend Cloud Services
  + Cloud Functions provides a connective layer of logic that lets you write code to connect and extend cloud services. Listen and respond to a file upload to Cloud Storage, a log change, or an incoming message on a Cloud Pub/Sub topic. Cloud Functions augments existing cloud services and allows you to address an increasing number of use cases with arbitrary programming logic. Cloud Functions have access to the Google Service Account credential and are thus seamlessly authenticated with the majority of Google Cloud services such as Datastore, Cloud Spanner, Cloud Translation API, Cloud Vision API, as well as many others. In addition, Cloud Functions are supported by numerous Node.js client libraries, which further simplify these integrations.
* Events and Triggers
  + Cloud events are things that happen in your cloud environment.These might be things like changes to data in a database, files added to a storage system, or a new virtual machine instance being created.
  + Events occur whether or not you choose to respond to them. You create a response to an event with a trigger. A trigger is a declaration that you are interested in a certain event or set of events. Binding a function to a trigger allows you to capture and act on events. For more information on creating triggers and associating them with your functions, see Events and Triggers.
* Serverless
  + Cloud Functions removes the work of managing servers, configuring software, updating frameworks, and patching operating systems. The software and infrastructure are fully managed by Google so that you just add code. Furthermore, provisioning of resources happens automatically in response to events. This means that a function can scale from a few invocations a day to many millions of invocations without any work from you.
* Use Cases
  + Asynchronous workloads like lightweight ETL, or cloud automations like triggering application builds now no longer need their own server and a developer to wire it up. You simply deploy a Cloud Function bound to the event you want and you're done.
  + The fine-grained, on-demand nature of Cloud Functions also makes it a perfect candidate for lightweight APIs and webhooks. In addition, the automatic provisioning of HTTP endpoints when you deploy an HTTP Function means there is no complicated configuration required as there is with some other services. See the following table for additional common Cloud Functions use cases:

|  |  |
| --- | --- |
| **Use Case** | **Description** |
| **Data Processing / ETL** | **Listen and respond to**[**Cloud Storage**](https://cloud.google.com/storage)**events such as when a file is created, changed, or removed. Process images, perform video transcoding, validate and transform data, and invoke any service on the Internet from your Cloud Function.** |
| **Webhooks** | **Via a simple**[**HTTP trigger**](https://cloud.google.com/functions/docs/calling/http)**, respond to events originating from 3rd party systems like GitHub, Slack, Stripe, or from anywhere that can send HTTP requests.** |
| **Lightweight APIs** | **Compose applications from lightweight, loosely coupled bits of logic that are quick to build and that scale instantly. Your functions can be event-driven or invoked directly over HTTP/S.** |
| **Mobile Backend** | **Use Google's mobile platform for app developers,**[**Firebase**](https://firebase.google.com/docs/functions/)**, and write your mobile backend in Cloud Functions. Listen and respond to events from Firebase Analytics, Realtime Database, Authentication, and Storage.** |
| **IoT** | **Imagine tens or hundreds of thousands of devices streaming data into Cloud Pub/Sub, thereby launching Cloud Functions to process, transform and store data. Cloud Functions lets you do in a way that's completely serverless.** |

* What you'll do
  + Create a cloud function
  + Deploy and test the function
  + View logs
* Create a function
  + In the console, click the Navigation menu > Cloud Functions. Click Create function.
  + In the Create function dialog, enter the following values:

|  |  |
| --- | --- |
| **Field** | **Value** |
| Function name | GCFunction |
| Trigger | Select **HTTP** and click **Save** |
| Memory allocated (In Runtime, Build, Connections and Security Settings) | Keep it default |
| Autoscaling | Set the **Maximum number of instance** to **5** and then click **Ne** |

* Deploy the function
  + While the function is being deployed, the icon next to it is a small spinner. When it's deployed, the spinner is a green check mark.
* Test the function
  + In the Cloud Functions Overview page, display the menu for your function, and click Test function.
  + In the Triggering event field, enter the following text between the brackets {} and click Test the function.
    - "message":"Hello World!"
    - In the Output field, you should see the message Success: Hello World!
    - In the Logs field, a status code of 200 indicates success. (It may take a minute for the logs to appear.)
* View logs
  + Click the blue arrow to go back to the Cloud Functions Overview page.
  + Display the menu for your function, and click View logs.
    - Example of the log history that displays in Query results:
* Cloud Functions is a serverless execution environment for building and connecting cloud services. True
* Which type of trigger is bound while creating Cloud Function in the lab? HTTP

## 6.0 Cloud Functions: Qwik Start - Command Line - https://www.cloudskillsboost.google/focuses/916?parent=catalog

* Create a function
  + First, you're going to create a simple function named helloWorld. This function writes a message to the Cloud Functions logs. It is triggered by cloud function events and accepts a callback function used to signal completion of the function.
  + For this lab the cloud function event is a cloud pub/sub topic event. A pub/sub is a messaging service where the senders of messages are decoupled from the receivers of messages. When a message is sent or posted, a subscription is required for a receiver to be alerted and receive the message. For more information about pub/subs, see Google Cloud Pub/Sub: A Google-Scale Messaging Service.
  + To create a cloud function:
  + In the Cloud Shell command line, create a directory for the function code.
    - mkdir gcf\_hello\_world
  + Move to the gcf\_hello\_world directory.
    - cd gcf\_hello\_world
  + Create and open index.js to edit.
    - nano index.js
  + Copy the following into the index.js file
    - /\*\*
    - \* Background Cloud Function to be triggered by Pub/Sub.
    - \* This function is exported by index.js, and executed when
    - \* the trigger topic receives a message.
    - \*
    - \* @param {object} data The event payload.
    - \* @param {object} context The event metadata.
    - \*/
    - exports.helloWorld = (data, context) => {
    - const pubSubMessage = data;
    - const name = pubSubMessage.data
    - ? Buffer.from(pubSubMessage.data, 'base64').toString() : "Hello World";
    - console.log(`My Cloud Function: ${name}`);
    - };
  + Exit nano (Ctrl+x) and save (Y) the file.
* Create a cloud storage bucket
  + Use the following command to create a new cloud storage bucket for your function:
  + gsutil mb -p [PROJECT\_ID] gs://[BUCKET\_NAME]
    - PROJECT\_ID is the Project ID in the connection details of this lab
    - BUCKET\_NAME is the name you give to the bucket. It must be a globally unique name. For more information, see Bucket naming guidelines.
* Deploy your function
  + When deploying a new function, you must specify --trigger-topic, --trigger-bucket, or --trigger-http. When deploying an update to an existing function, the function keeps the existing trigger unless otherwise specified.
  + For this lab, you'll set the --trigger-topic as hello\_world.
  + Deploy the function to a pub/sub topic named hello\_world, replacing [BUCKET\_NAME] with the name of your bucket:
    - gcloud functions deploy helloWorld \
    - --stage-bucket [BUCKET\_NAME] \
    - --trigger-topic hello\_world \
    - --runtime nodejs8
  + If you get OperationError, ignore warning and re-run the command.
  + If prompted, enter Y to allow unauthenticated invocations of a new function.
  + Verify the status of the function.
    - gcloud functions describe helloWorld
    - An ACTIVE status indicates that the function has been deployed.
  + Every message published in the topic triggers function execution, the message contents are passed as input data.
* Test the function
  + After you deploy the function and know that it's active, test that the function writes a message to the cloud log after detecting an event.
  + Enter this command to create a message test of the function.
    - DATA=$(printf 'Hello World!'|base64) && gcloud functions call helloWorld --data '{"data":"'$DATA'"}'
  + The cloud tool returns the execution ID for the function, which means a message has been written in the log.
  + View logs to confirm that there are log messages with that execution ID.
* View logs
  + Check the logs to see your messages in the log history.
    - gcloud functions logs read helloWorld
  + Note: The logs will take around 10 mins to appear. Also, the alternative way to view the logs is, go to Logging > Logs Explorer.
* Serverless lets you write and deploy code without the hassle of managing the underlying infrastructure. True

## 7.0 Google Cloud Pub/Sub: Qwik Start – Console - https://www.cloudskillsboost.google/focuses/3719?parent=catalog

* Overview
  + Google Cloud Pub/Sub is a messaging service for exchanging event data among applications and services. A producer of data publishes messages to a Cloud Pub/Sub topic. A consumer creates a subscription to that topic. Subscribers either pull messages from a subscription or are configured as webhooks for push subscriptions. Every subscriber must acknowledge each message within a configurable window of time.
* What you'll do
  + Set up a topic to hold data.
  + Subscribe to a topic to access the data.
  + Publish and then consume messages with a pull subscriber.
* Setting up Pub/Sub
  + You can use the Google Cloud Shell console to perform operations in Google Cloud Pub/Sub.
  + To use a Pub/Sub, you create a topic to hold data and a subscription to access data .published to the topic.
  + Click Navigation menu > Pub/Sub > Topics. Click Create topic.
  + The topic must have a unique name. For this lab, name your topic MyTopic. In the Create a topic dialog:
    - For Topic ID, type MyTopic.
    - Leave Encryption at the default value.
    - Click CREATE TOPIC.
* Add a subscription
  + Now you'll make a subscription to access the topic.
  + Click Topics in the left panel to return to the Topics page. For the topic you just made click the three dot icon > Create subscription.
  + In the Add subscription to topic dialog:
    - Type a name for the subscription, such as MySub
    - Set the Delivery Type to Pull.
    - Leave all other options at the default values.
  + Click Create.
  + Your subscription is listed in the Subscription list.
* A publisher application creates and sends messages to a \_\_\_\_. Subscriber applications create a \_\_\_\_ to a topic to receive messages from it. topic, subscription
* Cloud Pub/Sub is an asynchronous messaging service designed to be highly reliable and scalable. True
* Publish a message to the topic
  + At the bottom of the Topics details page, click MESSAGES tab and then click PUBLISH MESSAGE.
  + Enter Hello World in the Message field and click Publish.
* View the message
  + To view the message you'll use the subscription (MySub) to pull the message (Hello World) from the topic (MyTopic).
  + Enter the following command in command line.
    - gcloud pubsub subscriptions pull --auto-ack MySub
  + The message appears in the DATA field of the command output.

## 8.0 Google Cloud Pub/Sub: Qwik Start - Command Line - https://www.cloudskillsboost.google/focuses/925?parent=catalog

* Overview
  + Google Cloud Pub/Sub is a messaging service for exchanging event data among applications and services. By decoupling senders and receivers, it allows for secure and highly available communication between independently written applications. Google Cloud Pub/Sub delivers low-latency/durable messaging, and is commonly used by developers in implementing asynchronous workflows, distributing event notifications, and streaming data from various processes or devices.
  + In this lab, you will do the following:
    - Learn the basics of Pub/Sub.
    - Create, delete, and list Pub/Sub topics.
    - Create, delete, and list Pub/Sub subscriptions.
    - Publish messages to a topic.
    - Use a pull subscriber to output individual topic messages.
    - Use a pull subscriber with a flag to output multiple messages.
* The Pub/Sub basics
  + As stated earlier, Google Cloud Pub/Sub is an asynchronous global messaging service. There are three terms in Pub/Sub that appear often: topics, publishing, and subscribing.
    - A topic is a shared string that allows applications to connect with one another through a common thread.
    - Publishers push (or publish) a message to a Cloud Pub/Sub topic.
    - Subscribers make a "subscription" to a topic where they will either pull messages from the subscription or configure webhooks for push subscriptions. Every subscriber must acknowledge each message within a configurable window of time.
  + To sum it up, a producer publishes messages to a topic and a consumer creates a subscription to a topic to receive messages from it.
* Pub/Sub topics
  + Pub/Sub comes preinstalled in the Google Cloud Shell, so there are no installations or configurations required to get started with this service.
  + Run the following command to create a topic called myTopic: gcloud pubsub topics create myTopic
  + For good measure, create two more topics; one called Test1 and the other called Test2:
    - gcloud pubsub topics create Test1
    - gcloud pubsub topics create Test2
    - To see the three topics you just created, run the following command: gcloud pubsub topics list
  + Time to cleanup. Delete Test1 and Test2 by running the following commands:
    - gcloud pubsub topics delete Test1
    - gcloud pubsub topics delete Test2
  + Run the gcloud pubsub topics list command one more time to verify the topics were deleted:
    - gcloud pubsub topics list
* Pub/Sub subscriptions
  + Now that you're comfortable creating, viewing, and deleting topics, time to work with subscriptions.
  + Run the following command to create a subscription called mySubscription to topic myTopic:
    - gcloud pubsub subscriptions create --topic myTopic mySubscription
  + Add another two subscriptions to myTopic. Run the following commands to make Test1 and Test2 subscriptions:
    - gcloud pubsub subscriptions create --topic myTopic Test1
    - gcloud pubsub subscriptions create --topic myTopic Test2
  + Run the following command to list the subscriptions to myTopic:
    - gcloud pubsub topics list-subscriptions myTopic
  + To receive messages published to a topic, you must create a subscription to that topic. True
  + Now delete the Test1 and Test2 subscriptions. Run the following commands:
    - gcloud pubsub subscriptions delete Test1
    - gcloud pubsub subscriptions delete Test2
  + See if the Test1 and Test2 subscriptions were deleted. Run the list-subscriptions command one more time:
    - gcloud pubsub topics list-subscriptions myTopic
* Pub/Sub Publishing and Pulling a Single Message
  + Next you'll learn how to publish a message to a Pub/Sub topic.
  + Run the following command to publish the message "hello" to the topic you created previously (myTopic):
    - gcloud pubsub topics publish myTopic --message "Hello"
  + Publish a few more messages to myTopic. Run the following commands (replacing <YOUR NAME> with your name and <FOOD> with a food you like to eat):
    - gcloud pubsub topics publish myTopic --message "Publisher's name is <YOUR NAME>"
    - gcloud pubsub topics publish myTopic --message "Publisher likes to eat <FOOD>"
    - gcloud pubsub topics publish myTopic --message "Publisher thinks Pub/Sub is awesome"
  + next, use the pull command to get the messages from your topic. The pull command is subscription based, meaning it should work because earlier you set up the subscription mySubscription to the topic myTopic.
  + Use the following command to pull the messages you just published from the Pub/Sub topic:
    - gcloud pubsub subscriptions pull mySubscription --auto-ack
  + What's going on here? You published 4 messages to your topic, but only 1 was outputted.
  + Now is an important time to note a couple features of the pull command that often trip developers up:
    - Using the pull command without any flags will output only one message, even if you are subscribed to a topic that has more held in it.
    - Once an individual message has been outputted from a particular subscription-based pull command, you cannot access that message again with the pull command.
    - To see what the second bullet is talking about, run the last command three more times. You will see that it will output the other messages you published before.
  + Now, run the command a 4th time. You'll get the following output (since there were none left to return):
* Pub/Sub pulling all messages from subscriptions
  + Since you pulled all of the messages from your topic in hte last example, populate myTopic with a few more messages.
  + Run the following commands:
    - gcloud pubsub topics publish myTopic --message "Publisher is starting to get the hang of Pub/Sub"
    - gcloud pubsub topics publish myTopic --message "Publisher wonders if all messages will be pulled"
    - gcloud pubsub topics publish myTopic --message "Publisher will have to test to find out"
  + Add a flag to your command so you can output all three messages in one request. You may have not noticed, but you have actually been using a flag this entire time: the --auto-ack part of the pull command is a flag that has been formatting your messages into the neat boxes that you see your pulled messages in.
  + limit is another flag that sets an upper limit on the number of messages to pull.
  + Wait a minute to let the topics get created. Run the pull command with the limit flag:
    - gcloud pubsub subscriptions pull mySubscription --auto-ack --limit=3

## 9.0 Google Cloud Pub/Sub: Qwik Start – Python - https://www.cloudskillsboost.google/focuses/2775?parent=catalog

* Create a virtual environment
  + Execute the following command to download and update the packages list.
    - sudo apt-get update
  + Python virtual environments are used to isolate package installation from the system.
    - sudo apt-get install -y virtualenv
      * If prompted [Y/n], press Y and then Enter.
    - virtualenv -p python3 venv
  + Activate the virtual environment.
    - source venv/bin/activate
* Install the client library
  + Run the following to install the client library:
    - pip install --upgrade google-cloud-pubsub
  + Get the sample code by cloning a GitHub repository:
    - git clone https://github.com/googleapis/python-pubsub.git
  + Navigate to the directory:
    - cd python-pubsub/samples/snippets
* Pub/Sub in Google CLoud
  + Pub/Sub comes preinstalled in the Cloud Shell, so there are no installations or configurations required to get started with this service. In this lab you use Python to create the topic, subscriber, and then view the message. You use a gcloud command to publish the message to the topic.
* Create a topic
  + To publish data to Cloud Pub/Sub you create a topic and then configure a publisher to the topic.
  + In Cloud Shell, your Project ID should automatically be stored in the environment variable GOOGLE\_CLOUD\_PROJECT:
    - echo $GOOGLE\_CLOUD\_PROJECT
      * Ensure the output is the same as the Project ID in your CONNECTION DETAILS.
  + publisher.py is a script that demonstrates how to perform basic operations on topics with the Cloud Pub/Sub API. View the content of publisher script:
    - cat publisher.py
      * Note: Alternatively, you can use the shell editors that are installed on Cloud Shell, such as nano or vim or use the Cloud Shell code editor to view python-pubsub/samples/snippets/publisher.py
  + For information about the publisher script:
    - python publisher.py -h
  + Run the publisher script to create Pub/Sub Topic:
    - python publisher.py $GOOGLE\_CLOUD\_PROJECT create MyTopic
  + This command returns a list of all Pub/Sub topics in a given project:
    - python publisher.py $GOOGLE\_CLOUD\_PROJECT list
  + You can also view the topic you just made in the Cloud Console.
    - Navigate to Navigation menu > Pub/Sub > Topics.
* Create a subscription
  + Create a Pub/Sub subscription for topic with subscriber.py script:
    - python subscriber.py $GOOGLE\_CLOUD\_PROJECT create MyTopic MySub
  + This command returns a list of subscribers in given project:
    - python subscriber.py $GOOGLE\_CLOUD\_PROJECT list-in-project
  + Check out the subscription you just made in the console. In the left pane, click Subscriptions. You should see the subscription name and other details.
  + For information about the subscriber script:
    - python subscriber.py -h
* Publish messages
  + Now that you've set up MyTopic (the topic), a subscription to MyTopic (MySub), see if you can use gcloud commands to publish a message to MyTopic.
  + Publish the message "Hello" to MyTopic:
    - gcloud pubsub topics publish MyTopic --message "Hello"
  + Publish a few more messages to MyTopic—run the following commands (replacing <YOUR NAME> with your name and <FOOD> with a food you like to eat):
    - gcloud pubsub topics publish MyTopic --message "Publisher's name is <YOUR NAME>"
    - gcloud pubsub topics publish MyTopic --message "Publisher likes to eat <FOOD>"
    - gcloud pubsub topics publish MyTopic --message "Publisher thinks Pub/Sub is awesome"
* View messages
  + Now that you've published messages to MyTopic, pull and view the messages using MySub.
  + Use MySub to pull the message from MyTopic:
    - python subscriber.py $GOOGLE\_CLOUD\_PROJECT receive MySub
  + Click Ctrl+c to stop listening.
* Google Cloud Pub/Sub service allows applications to exchange messages reliably, quickly, and asynchronously.
  + True
* A \_\_\_\_\_ is a shared string that allows applications to connect with one another. topic

## 10.0 Perform Foundational Infrastructure Tasks in Google Cloud: Challenge Lab - https://www.cloudskillsboost.google/focuses/10379?parent=catalog

* Overview
  + In a challenge lab you’re given a scenario and a set of tasks. Instead of following step-by-step instructions, you will use the skills learned from the labs in the quest to figure out how to complete the tasks on your own! An automated scoring system (shown on this page) will provide feedback on whether you have completed your tasks correctly.
* Challenge scenario
  + You are just starting your junior cloud engineer role with Jooli inc. So far you have been helping teams create and manage Google Cloud resources.
  + You are expected to have the skills and knowledge for these tasks so don’t expect step-by-step guides.
  + Your challenge
    - You are now asked to help a newly formed development team with some of their initial work on a new project around storing and organizing photographs, called memories. You have been asked to assist the memories team with initial configuration for their application development environment; you receive the following request to complete the following tasks:
      * Create a bucket for storing the photographs.
      * Create a Pub/Sub topic that will be used by a Cloud Function you create.
      * Create a Cloud Function.
      * Remove the previous cloud engineer’s access from the memories project.
  + Some Jooli Inc. standards you should follow:
    - Create all resources in the us-east1 region and us-east1-b zone, unless otherwise directed.
    - Use the project VPCs.
    - Naming is normally team-resource, e.g. an instance could be named kraken-webserver1
    - Allocate cost effective resource sizes. Projects are monitored and excessive resource use will result in the containing project's termination (and possibly yours), so beware. This is the guidance the monitoring team is willing to share; unless directed, use f1-micro for small Linux VMs and n1-standard-1 for Windows or other applications such as Kubernetes nodes.
* Task 1: Create a bucket
  + You need to create a bucket called Bucket Name for the storage of the photographs.
* Task 2: Create a Pub/Sub topic
  + Create a Pub/Sub topic called Topic Name for the Cloud Function to send messages.
* Task 3: Create the thumbnail Cloud Function
  + Create a Cloud Function called Cloud Function Name that executes every time an object is created in the bucket Bucket Name you created in task 1. The function is written in Node.js 14. Make sure you set the Entry point (Function to execute) to thumbnail and Trigger to Cloud Storage.
  + In line 15 of index.js replace the text REPLACE\_WITH\_YOUR\_TOPIC ID with the Topic Name you created in task 2.
* index.js:
* /\* globals exports, require \*/
* //jshint strict: false
* //jshint esversion: 6
* "use strict";
* const crc32 = require("fast-crc32c");
* const { Storage } = require('@google-cloud/storage');
* const gcs = new Storage();
* const { PubSub } = require('@google-cloud/pubsub');
* const imagemagick = require("imagemagick-stream");
* exports.thumbnail = (event, context) => {
* const fileName = event.name;
* const bucketName = event.bucket;
* const size = "64x64"
* const bucket = gcs.bucket(bucketName);
* const topicName = "REPLACE\_WITH\_YOUR\_TOPIC ID";
* const pubsub = new PubSub();
* if ( fileName.search("64x64\_thumbnail") == -1 ){
* // doesn't have a thumbnail, get the filename extension
* var filename\_split = fileName.split('.');
* var filename\_ext = filename\_split[filename\_split.length - 1];
* var filename\_without\_ext = fileName.substring(0, fileName.length - filename\_ext.length );
* if (filename\_ext.toLowerCase() == 'png' || filename\_ext.toLowerCase() == 'jpg'){
* // only support png and jpg at this point
* console.log(`Processing Original: gs://${bucketName}/${fileName}`);
* const gcsObject = bucket.file(fileName);
* let newFilename = filename\_without\_ext + size + '\_thumbnail.' + filename\_ext;
* let gcsNewObject = bucket.file(newFilename);
* let srcStream = gcsObject.createReadStream();
* let dstStream = gcsNewObject.createWriteStream();
* let resize = imagemagick().resize(size).quality(90);
* srcStream.pipe(resize).pipe(dstStream);
* return new Promise((resolve, reject) => {
* dstStream
* .on("error", (err) => {
* console.log(`Error: ${err}`);
* reject(err);
* })
* .on("finish", () => {
* console.log(`Success: ${fileName} → ${newFilename}`);
* // set the content-type
* gcsNewObject.setMetadata(
* {
* contentType: 'image/'+ filename\_ext.toLowerCase()
* }, function(err, apiResponse) {});
* pubsub
* .topic(topicName)
* .publisher()
* .publish(Buffer.from(newFilename))
* .then(messageId => {
* console.log(`Message ${messageId} published.`);
* })
* .catch(err => {
* console.error('ERROR:', err);
* });
* });
* });
* }
* else {
* console.log(`gs://${bucketName}/${fileName} is not an image I can handle`);
* }
* }
* else {
* console.log(`gs://${bucketName}/${fileName} already has a thumbnail`);
* }
* };
* Copied!
* package.json:
* {
* "name": "thumbnails",
* "version": "1.0.0",
* "description": "Create Thumbnail of uploaded image",
* "scripts": {
* "start": "node index.js"
* },
* "dependencies": {
* "@google-cloud/pubsub": "^2.0.0",
* "@google-cloud/storage": "^5.0.0",
* "fast-crc32c": "1.0.4",
* "imagemagick-stream": "4.1.1"
* },
* "devDependencies": {},
* "engines": {
* "node": ">=4.3.2"
* }
  + }
* You must upload one JPG or PNG image into the bucket, we will verify the thumbnail was created (after creating the function successfully). Use any JPG or PNG image, or use this image https://storage.googleapis.com/cloud-training/gsp315/map.jpg; download the image to your machine and then upload that file to your bucket. You will see a thumbnail image appear shortly afterwards (use REFRESH in the bucket details).
* Task 4: Remove the previous cloud engineer
  + You will see that there are two users, one is your account (with the role of Owner) and the other is the previous cloud engineer ( Username 2 with the role of Viewer). We like to keep our security tight, so please remove the previous cloud engineer’s access to the project.