Essential Google Cloud Infrastructure: Core Services

* + Identity and Access Management (IAM)
  + Identity and Access Management 1 minute - https://youtu.be/JL1lUhTmfEg
  + Organization 4 minutes - https://youtu.be/q4KYl1ze8vs
  + Roles 4 minutes - https://youtu.be/LV2lmA-LTCc
  + Demo: Custom roles 3 minutes - https://youtu.be/VDMoI0WhdEg
  + Members 7 minutes - https://youtu.be/pYVydwOZ0Iw
  + Service Accounts 7 minutes - https://youtu.be/BoFJjivIWNM
  + IAM best practices 2 minutes - https://youtu.be/zc0eQp6hXkg
  + Cloud IAM 30 minutes
    - Overview
      * In this lab, you learn how to use the Service Account User role and how to grant roles.
    - Objectives
      * Use Cloud IAM to implement access control
      * Restrict access to specific features or resources
      * Use the Service Account User role
    - Task 1: Setup for two users
      * Sign in to the Cloud Console as the first user Username 1
      * Click on the user icon in the top-right corner of the screen, and then click Add account.
      * Sign in to the Cloud Console with the Username 2 provided in Qwiklabs.
    - Task 3: Prepare a resource for access testing
      * Create a bucket and upload a sample file
      * Specify the following, and leave the remaining settings as their defaults:
        + Property Value (type value or select option as specified)
        + Name Enter a globally unique name
        + Location type Multi-region
      * Upload any sample file from your local machine.
      * Rename the file to sample.txt, and click RENAME.
      * Verify project viewer access
        + Switch to the Username 2 Cloud Console tab.
        + In the Console, navigate to Navigation menu > Cloud Storage > Browser.
        + Verify that Username 2 can see the bucket.
    - Task 4: Remove project access
      * Select Username 2 and click REMOVE.
      * Verify that Username 2 has lost access
        + Switch to the Username 2 Cloud Console tab.
        + Cloud Storage > Browser. An error will be displayed.
    - Task 5: Add storage access
      * Switch to the Username 1 Cloud Console tab.
      * ADD Username 2 user, Role > Storage Object Viewer. Click SAVE.
      * Verify that Username 2 has storage access
        + Switch to the Username 2 Cloud Console tab.

Username 2 doesn't have Project Viewer roles, so that user can't see the project or any of its resources in the Console. However, the user has specific access to Cloud Storage.

* + - * click Activate Cloud Shell
        + gsutil ls gs://[YOUR\_BUCKET\_NAME]
      * As you can see, Username 2 has limited access to Cloud Storage.
    - Task 6: Set up the Service Account User
      * Create a service account
        + Specify the Service account name as read-bucket-objects .
        + Click CREATE AND CONTINUE.
        + Specify the Role as Cloud Storage > Storage Object Viewer .
        + Click CONTINUE. Click DONE.
      * Add the user to the service account
        + Select the read-bucket-objects service account.
        + click on Manage permissions

You will grant the user the role of Service Account User, which allows that person to use a service account on a VM, if they have access to the VM.

You could perform this activity for a specific user, group, or domain.

For training purposes, you will grant the Service Account User role to everyone at a company called Altostrat.com. Altostrat.com is a fake company used for demonstration and training.

* + - * + Click on the GRANT ACCESS button. Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

New members altostrat.com

Select a role Service Accounts > Service Account User

Click SAVE.

* + - * Grant Compute Engine access
        + You now give the entire organization at Altostrat the Compute Engine Admin role.
        + On the Navigation menu (7a91d354499ac9f1.png), click IAM & admin > IAM.
        + Click ADD.

Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

New members altostrat.com

Select a role Compute Engine > Compute Instance Admin (v1)

* + - * + Click SAVE.
        + This step is a rehearsal of the activity you would perform for a specific user.
        + This action gives the user limited abilities with a VM instance. The user will be able to connect via SSH to a VM and perform some administration tasks.
      * Create a VM with the Service Account User
        + On the Navigation menu (7a91d354499ac9f1.png), click Compute Engine > VM instances.
        + Click CREATE INSTANCE.
        + Specify the following, and leave the remaining settings as their defaults:
        + Property Value (type value or select option as specified)
        + Name demoiam
        + Region us-central1
        + Zone us-centra1-c
        + Series N1
        + Machine Type f1-micro
        + Boot disk Debian GNU/Linux 10 (buster)
        + Service account read-bucket-objects
        + Click Create.
        + Click Check my progress to verify the objective.
        + Set up the Service Account User and create a VM
    - Task 7: Explore the Service Account User role
      * At this point, you might have the user test access by connecting via SSH to the VM and performing the next actions. As the owner of the project, you already possess the Service Account User role. So you can simulate what the user would experience by just using SSH to access the VM from the Cloud Console.
      * The actions you perform and results will be the same as if you were the target user.
      * Use the Service Account User
        + For demoiam, click SSH to launch a terminal and connect.

gcloud compute instances list

* + - * + Result (do not copy; this is example output):

ERROR: (gcloud.compute.instances.list) Some requests did not succeed:

* + - * + What happened? Why?
        + Copy the sample.txt file from the bucket you created earlier. Note that the trailing period is part of the command below. It means copy to "this location":

gsutil cp gs://[YOUR\_BUCKET\_NAME]/sample.txt .

* + - * + To rename the file you copied, run the following command:

mv sample.txt sample2.txt

* + - * + To copy the renamed file back to the bucket, run the following command:

gsutil cp sample2.txt gs://[YOUR\_BUCKET\_NAME]

* + - * + Result (do not copy; this is example output):

AccessDeniedException: 403 Caller does not have storage.objects.create access to bucket train-test-iam.

* + - * + What happened?

Because you connected via SSH to the instance, you can "act as the service account," essentially assuming the same permissions.The service account the instance was started with had the Storage Viewer role, which permits downloading objects from GCS buckets in the project.To list instances in a project, you need to grant the compute.instance.list permission. Because the service account did not have this permission, you could not list instances running in the project. Because the service account did have permission to download objects, it could download an object from the bucket. It did not have permission to write objects, so you got a "403 access denied" message.

* + - * + On the Navigation menu (7a91d354499ac9f1.png), click IAM & admin > IAM.

Browse the list for the lines with read-bucket-objects, click on the pencil icon. read-bucket-objects currently has the Storage Object Viewer role. Alter the Role to Cloud Storage > Storage Object Creator .

Click Save

Return to the SSH window for demoiam

To copy the renamed file back to the bucket, run the following command:

gsutil cp sample2.txt gs://[YOUR\_BUCKET\_NAME]

This time the command succeeds as the service account has the correct permissions

* + Lab Review: Cloud IAM 14 minutes - https://youtu.be/XGyvJjXI1Yc
  + Quiz: Identity and Access Management
    - What abstraction is primarily used to administer user access in IAM ?
      * Roles, an abstraction of job roles.
      * IAM administration uses pre-defined roles for administration of user access. The roles are defined by more granular permissions. But permissions are not applied to users directly, only through the roles that are assigned to them.
    - Which of the following is not a type of IAM role?
      * Advanced
      * That's correct! There are three types of roles in IAM: basic roles, predefined roles, and custom roles. There are no "advanced" roles in IAM.
    - Which of the following is not a type of IAM member?
      * Organization Account
      * That's correct! There are five different types of members: Google Accounts, Service Accounts, Google Groups, Google Workspace domain, and Cloud Identity domains. There are no "Organization Accounts" in IAM.
  + Storage and Database Services
  + Cloud Storage 9 minutes - https://youtu.be/3l10nuX-dWA
  + Cloud Storage Features 6 minutes - https://youtu.be/FYwzdFmw6lo
  + Choosing a storage class 1 minute - https://youtu.be/RVrdgAzl3q0
  + Filestore 3 minutes - https://youtu.be/9tUKB08hY14
  + Cloud Storage 1 hour - <https://www.cloudskillsboost.google/course_sessions/838017/labs/112302>
    - Overview
      * Cloud Storage is a fundamental resource in Google Cloud, with many advanced features. In this lab, you exercise many Cloud Storage features that could be useful in your designs. You explore Cloud Storage using both the console and the gsutil tool.
    - Objectives
      * Create and use buckets
      * Set access control lists to restrict access
      * Use your own encryption keys
      * Implement version controls
      * Use directory synchronization
      * Share a bucket across projects using IAM
    - Task 1: Preparation
      * Create a Cloud Storage bucket
      * Cloud Storage > Browser > Create bucket > Name: BUCKET\_NAME\_1, Location type: Multi-region
        + Enforce public access prevention on this bucket unchecked
        + Access control Fine-grained (object-level permission in addition to your bucket-level permissions)
      * Download a sample file using CURL and make two copies
        + In the Cloud Console, click Activate Cloud Shell (Cloud Shell).
        + Store [BUCKET\_NAME\_1] in an environment variable:

export BUCKET\_NAME\_1=<enter bucket name 1 here>

curl https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html > setup.html

cp setup.html setup2.html

cp setup.html setup3.html

* + - Task 2: Access control lists (ACLs)
      * Copy the file to the bucket and configure the access control list
        + gsutil cp setup.html gs://$BUCKET\_NAME\_1/
      * To get the default access list that's been assigned to setup.html, run the following command:
        + gsutil acl get gs://$BUCKET\_NAME\_1/setup.html > acl.txt
        + cat acl.txt
      * To set the access list to private and verify the results, run the following commands:
        + gsutil acl set private gs://$BUCKET\_NAME\_1/setup.html
        + gsutil acl get gs://$BUCKET\_NAME\_1/setup.html > acl2.txt
        + cat acl2.txt
      * To update the access list to make the file publicly readable, run the following commands:
        + gsutil acl ch -u AllUsers:R gs://$BUCKET\_NAME\_1/setup.html
        + gsutil acl get gs://$BUCKET\_NAME\_1/setup.html > acl3.txt
        + cat acl3.txt
      * Delete the local file and copy back from Cloud Storage

rm setup.html

* + - * + To copy the file from the bucket again, run the following command:

gsutil cp gs://$BUCKET\_NAME\_1/setup.html setup.html

* + - Task 3: Customer-supplied encryption keys (CSEK)
      * Generate a CSEK key
        + For the next step, you need an AES-256 base-64 key.
        + Run the following command to create a key:

python3 -c 'import base64; import os; print(base64.encodebytes(os.urandom(32)))'

* + - * + Result (do not copy; this is example output):

b'tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=\n'

* + - * + Copy the value of the generated key excluding b' and \n' from the command output. Key should be in form of tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=.
      * Modify the boto file
        + The encryption controls are contained in a gsutil configuration file named .boto.
        + To view and open the boto file, run the following commands:

ls -al

nano .boto

* + - * + Note: if the .boto file is empty, close the nano editor with Ctrl+X and generate a new .boto file using the gsutil config -n command. Then, try opening the file again with the above commands.
        + If the .boto file is still empty, you might have to locate it using the gsutil version -l command.
        + Locate the line with "#encryption\_key="
        + Note: the bottom of the nano editor provides you with shortcuts to quickly navigate files. Use the Where Is shortcut to quickly locate the line with the #encryption\_key=.
        + Uncomment the line by removing the # character, and paste the key you generated earlier at the end.
        + Example (do not copy; this is an example):

Before:

# encryption\_key=

After:

encryption\_key=tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=

* + - * + Press Ctrl+O, ENTER to save the boto file, and then press Ctrl+X to exit nano.
      * Upload the remaining setup files (encrypted) and verify in the Cloud Console
        + To upload the remaining setup.html files, run the following commands:

gsutil cp setup2.html gs://$BUCKET\_NAME\_1/

gsutil cp setup3.html gs://$BUCKET\_NAME\_1/

* + - * + Click [BUCKET\_NAME\_1]. Both setup2.html and setup3.html files show that they are customer-encrypted.
      * Delete local files, copy new files, and verify encryption
        + To delete your local files, run the following command in Cloud Shell:

rm setup\*

* + - * To copy the files from the bucket again, run the following command:
        + gsutil cp gs://$BUCKET\_NAME\_1/setup\* ./
      * To cat the encrypted files to see whether they made it back, run the following commands:
        + cat setup.html
        + cat setup2.html
        + cat setup3.html
    - Task 4: Rotate CSEK keys
      * Move the current CSEK encrypt key to decrypt key
        + Run the following command to open the .boto file:

nano .boto

* + - * + Comment out the current encryption\_key line by adding the # character to the beginning of the line.
        + Uncomment decryption\_key1 by removing the # character, and copy the current key from the encryption\_key line to the decryption\_key1 line.
        + Result (do not copy; this is example output):

Before:

encryption\_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

# decryption\_key1=

After:

# encryption\_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

decryption\_key1=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

* + - * + Press Ctrl+O, ENTER to save the boto file, and then press Ctrl+X to exit nano.
        + Note: In practice, you would delete the old CSEK key from the encryption\_key line.
      * Generate another CSEK key and add to the boto file
        + Run the following command to generate a new key:

python3 -c 'import base64; import os; print(base64.encodebytes(os.urandom(32)))'

* + - * + Copy the value of the generated key excluding b' and \n' from the command output. Key should be in form of tmxElCaabWvJqR7uXEWQF39DhWTcDvChzuCmpHe6sb0=.
        + To open the boto file, run the following command:

nano .boto

* + - * + Uncomment encryption and paste the new key value for encryption\_key=.

Before:

# encryption\_key=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

After:

encryption\_key=HbFK4I8CaStcvKKIx6aNpdTse0kTsfZNUjFpM+YUEjY=

* + - * + Press Ctrl+O, ENTER to save the boto file, and then press Ctrl+X to exit nano.
      * Rewrite the key for file 1 and comment out the old decrypt key
        + When a file is encrypted, rewriting the file decrypts it using the decryption\_key1 that you previously set, and encrypts the file with the new encryption\_key.
        + You are rewriting the key for setup2.html, but not for setup3.html, so that you can see what happens if you don't rotate the keys properly.
        + Run the following command:

gsutil rewrite -k gs://$BUCKET\_NAME\_1/setup2.html

* + - * + To open the boto file, run the following command:

nano .boto

* + - * + Comment out the current decryption\_key1 line by adding the # character back in.

Before:

decryption\_key1=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

After:

# decryption\_key1=2dFWQGnKhjOcz4h0CudPdVHLG2g+OoxP8FQOIKKTzsg=

* + - * + Press Ctrl+O, ENTER to save the boto file, and then press Ctrl+X to exit nano.
        + Note: In practice, you would delete the old CSEK key from the decryption\_key1 line.
      * Download setup 2 and setup3
        + To download setup2.html, run the following command:

gsutil cp gs://$BUCKET\_NAME\_1/setup2.html recover2.html

gsutil cp gs://$BUCKET\_NAME\_1/setup3.html recover3.html

* + - * + Note: What happened? setup3.html was not rewritten with the new key, so it can no longer be decrypted, and the copy will fail.
        + You have successfully rotated the CSEK keys.
    - Task 5: Enable lifecycle management
      * View the current lifecycle policy for the bucket
        + Run the following command to view the current lifecycle policy:

gsutil lifecycle get gs://$BUCKET\_NAME\_1

* + - * + Note: there is no lifecycle configuration. You create one in the next steps.
      * Create a JSON lifecycle policy file
        + To create a file named life.json, run the following command:

nano life.json

* + - * + Paste the following value into the life.json file:

{

"rule":

[

{

"action": {"type": "Delete"},

"condition": {"age": 31}

}

]

}

* + - * + Note: these instructions tell Cloud Storage to delete the object after 31 days.
        + Press Ctrl+O, ENTER to save the file, and then press Ctrl+X to exit nano.
      * Set the policy and verify
        + To set the policy, run the following command:

gsutil lifecycle set life.json gs://$BUCKET\_NAME\_1

* + - * + To verify the policy, run the following command:

gsutil lifecycle get gs://$BUCKET\_NAME\_1

* + - Task 6: Enable versioning
      * View the versioning status for the bucket and enable versioning
        + Run the following command to view the current versioning status for the bucket:

gsutil versioning get gs://$BUCKET\_NAME\_1

* + - * + Note: the Suspended policy means that it is not enabled.
        + To enable versioning, run the following command:

gsutil versioning set on gs://$BUCKET\_NAME\_1

* + - * + To verify that versioning was enabled, run the following command:

gsutil versioning get gs://$BUCKET\_NAME\_1

* + - * Create several versions of the sample file in the bucket
        + Check the size of the sample file:

ls -al setup.html

* + - * + Open the setup.html file:

nano setup.html

* + - * + Delete any 5 lines from setup.html to change the size of the file.

Press Ctrl+O, ENTER to save the file, and then press Ctrl+X to exit nano.

* + - * + Copy the file to the bucket with the -v versioning option:

gsutil cp -v setup.html gs://$BUCKET\_NAME\_1

* + - * + Open the setup.html file:

nano setup.html

* + - * + Delete another 5 lines from setup.html to change the size of the file.
        + Press Ctrl+O, ENTER to save the file, and then press Ctrl+X to exit nano.
        + Copy the file to the bucket with the -v versioning option:

gsutil cp -v setup.html gs://$BUCKET\_NAME\_1

* + - * List all versions of the file
        + To list all versions of the file, run the following command:

gsutil ls -a gs://$BUCKET\_NAME\_1/setup.html

* + - * + Highlight and copy the name of the oldest version of the file (the first listed), referred to as [VERSION\_NAME] in the next step.
        + Note: make sure to copy the full path of the file, starting with gs://
        + Store the version value in the environment variable [VERSION\_NAME].

export VERSION\_NAME=<Enter VERSION name here>

* + - * + Result (do not copy; this is example output):

gs://BUCKET\_NAME\_1/setup.html#1584457872853517

* + - * Download the oldest, original version of the file and verify recovery
        + Download the original version of the file:

gsutil cp $VERSION\_NAME recovered.txt

* + - * + To verify recovery, run the following commands:

ls -al setup.html

ls -al recovered.txt

* + - * + Note: you have recovered the original file from the backup version. Notice that the original is bigger than the current version because you deleted lines.
    - Task 7: Synchronize a directory to a bucket
      * Make a nested directory and sync with a bucket
        + Make a nested directory structure so that you can examine what happens when it is recursively copied to a bucket.
        + Run the following commands:

mkdir firstlevel

mkdir ./firstlevel/secondlevel

cp setup.html firstlevel

cp setup.html firstlevel/secondlevel

* + - * + To sync the firstlevel directory on the VM with your bucket, run the following command:

gsutil rsync -r ./firstlevel gs://$BUCKET\_NAME\_1/firstlevel

* + - * Examine the results
        + Cloud Storage > Browser > Click [BUCKET\_NAME\_1]
        + Click on /firstlevel and then on /secondlevel.
        + Compare what you see in the Cloud Console with the results of the following cmd:

gsutil ls -r gs://$BUCKET\_NAME\_1/firstlevel

* + - Task 8: Cross-project sharing
      * Switch to the second project Click the project selector dropdown in the title bar.
        + Click All, select [PROJECT\_ID\_2].
      * Prepare the bucket
      * In the Cloud Console, on the Navigation menu (Navigation menu icon), click Cloud Storage > Browser.
      * Click Create bucket.
        + Cloud Storage > Browser > Create bucket > Name: BUCKET\_NAME\_2, Location type: Multi-region

Enforce public access prevention on this bucket unchecked

Access control Fine-grained (object-level permission in addition to your bucket-level permissions)

* + - * Upload a text file to the bucket
        + Upload a file to [BUCKET\_NAME\_2]. Any small example file or text file will do.
      * Create an IAM Service Account
        + IAM & admin > Service accounts > Click Create service account.
        + On Service account details page, specify the Service account name as cross-project-storage.
        + Click Create and Continue.
        + On the Service account permissions page, specify the role as Cloud Storage > Storage Object Viewer.
        + Click Continue and then Done.
        + Click the cross-project-storage service account to add the JSON key.
        + In Keys tab, click Add Key dropdown and select Create new key.
        + Select JSON as the key type and click Create. A JSON key file will be downloaded. You will need to find this key file and upload it in into the VM in a later step.
        + Click Close.
        + On your hard drive, rename the JSON key file to credentials.json.
        + In the upper pane, switch back to [PROJECT\_ID\_1].
      * Create a VM
        + Compute Engine > VM instances > Click Create Instance.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name crossproject

Region europe-west1

Zone europe-west1-d

Series N1

Machine type n1-standard-1

Boot disk Debian GNU/Linux 10 (buster)

* + - * + Click Create.
      * SSH to the VM
        + For crossproject, click SSH to launch a terminal and connect.
        + Note: if the message appears like Connection via Cloud Identity-Aware Proxy Failed then click Connect without Identity-Aware Proxy.
        + Store [BUCKET\_NAME\_2] in an environment variable:

export BUCKET\_NAME\_2=<enter bucket name 2 here>

* + - * + Store [FILE\_NAME] in an environment variable:

export FILE\_NAME=<enter FILE\_NAME here>

* + - * + List the files in [PROJECT\_ID\_2]:

gsutil ls gs://$BUCKET\_NAME\_2/

* + - * + Result (do not copy; this is example output):

AccessDeniedException: 403 404513585876-compute@developer.gserviceaccount.com does not have storage.objects.list access to the Google Cloud Storage bucket.

* + - * Authorize the VM
        + To upload credentials.json through the SSH VM terminal, click on the gear icon (Settings gear icon) in the upper-right corner, and then click Upload file.
        + Select credentials.json and upload it.
        + Click Close in the File Transfer window.
        + Enter the following command in the terminal to authorize the VM to use the Google Cloud API:

gcloud auth activate-service-account --key-file credentials.json

* + - * + Note: the image you are using has the Google Cloud SDK pre-installed; therefore, you don't need to initialize the Google Cloud SDK.
        + If you are attempting this lab in a different environment, make sure you have followed these procedures regarding installing the Google Cloud SDK.
      * Verify access
        + Retry this command:

gsutil ls gs://$BUCKET\_NAME\_2/

* + - * + Retry this command:

gsutil cat gs://$BUCKET\_NAME\_2/$FILE\_NAME

* + - * + Try to copy the credentials file to the bucket:

gsutil cp credentials.json gs://$BUCKET\_NAME\_2/

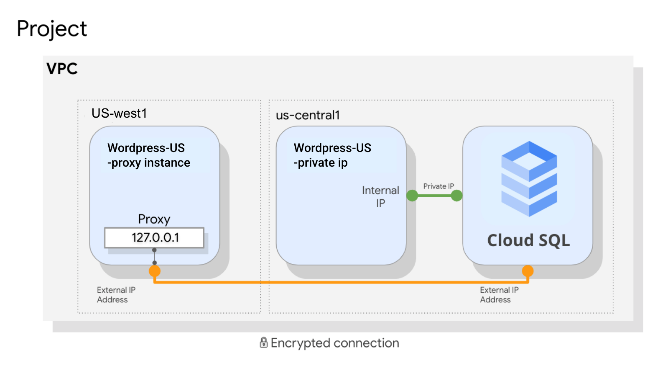
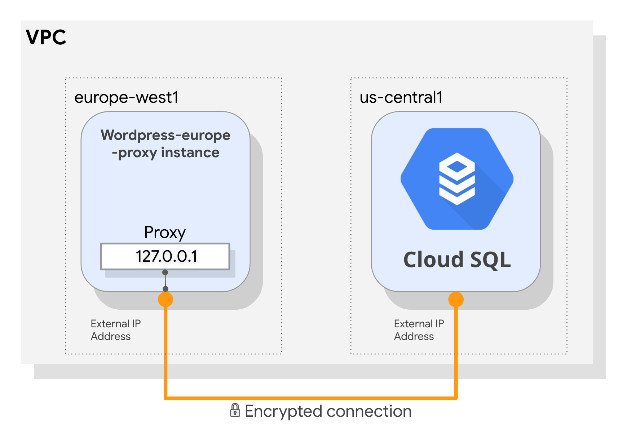
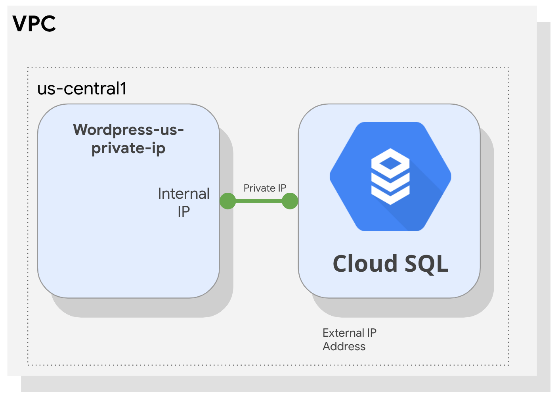
* + - * + Result (do not copy; this is example output):

Copying file://credentials.json [Content-Type=application/json]...

AccessDeniedException: 403 cross-project-storage@qwiklabs-gcp-02-

* + - * Modify role
        + In the upper pane, switch back to [PROJECT\_ID\_2].
        + click IAM & admin > IAM. > Edit cross-project-storage service account
        + Click on the Storage Object Viewer role, and then click Cloud Storage > Storage Object Admin.
        + Click Save. If you don't click Save, the change will not be made.
        + Click Check my progress to verify the objective.
        + Create and verify the resources in the first project
      * Verify changed access
        + Return to the SSH terminal for crossproject.
        + Copy the credentials file to the bucket:

gsutil cp credentials.json gs://$BUCKET\_NAME\_2/

* + - * + Note: in this example the VM in PROJECT\_ID\_1 can now upload files to Cloud Storage in a bucket that was created in another project.
        + Note that the project where the bucket was created is the billing project for this activity. That means if the VM uploads a ton of files, it will not be billed to PROJECT\_ID\_1, but instead to PROJECT\_ID\_2.
  + Lab Review: Cloud Storage 27 minutes - https://youtu.be/XUj31T3H9ic
  + Cloud SQL 5 minutes - <https://youtu.be/Z5xbUAh1ZU8>
  + Implementing Cloud SQL 48 minutes
    - Overview
      * you configure a Cloud SQL server and learn how to connect an application to it via a proxy over an external connection. You also configure a connection over a Private IP link that offers performance and security benefits. The app we chose to demonstrate in this lab is Wordpress, but the information and best practices are applicable to any application that needs SQL Server.
      * you will have 2 working instances of the Wordpress frontend connected over 2 different connection types to their SQL instance backend, as shown in this diagram:
        + 
    - Objectives
      * Create a Cloud SQL database
      * Configure a virtual machine to run a proxy
      * Create a connection between an application and Cloud SQL
      * Connect an application to Cloud SQL using Private IP address
    - Task 1: Create a Cloud SQL database
      * configure a SQL server according to GC best practices and create a Private IP connection.
      * Navigation > SQL > Create instance > Choose MySQL.
      * Specify the following, and leave the remaining settings as their defaults:
        + Property Value
        + Instance ID wordpress-db
        + Root password type a password
        + Region us-central1
        + Zone Any
        + Database Version MySQL 5.7
      * Note the root password; it will be used in a later step and referred to as [ROOT\_PASSWORD].
      * Expand Show configuration options. Expand the Machine type section.
      * Provision the right amount of vCPU and memory. To choose a Machine Type, click the dropdown menu, and then explore your options.
      * A few points to consider:
        + Shared-core machines are good for prototyping, and are not covered by Cloud SLA.
        + Each vCPU is subject to a 250 MB/s network throughput cap for peak performance. Each additional core increases the network cap, up to a theoretical maximum of 2000 MB/s.
        + For performance-sensitive workloads such as online transaction processing (OLTP), a general guideline is to ensure that your instance has enough memory to contain the entire working set and accommodate the number of active connections.
      * For this lab, select standard from the dropdown menu, and then select 1 vCPU, 3.75 GB.
      * Next, expand the Storage section and then choose Storage type and Storage capacity.
      * A few points to consider:
        + SSD (solid-state drive) is the best choice for most use cases. HDD (hard-disk drive) offers lower performance, but storage costs are significantly reduced, so HDD may be preferable for storing data that is infrequently accessed and does not require very low latency.
        + There is a direct relationship between the storage capacity and its throughput.
      * Click each of the capacity options to see how it affects the throughput. Reset the option to 10GB.
        + Setting your storage capacity too low without enabling an automatic storage increase can cause your instance to lose its SLA.
      * Expand the Connections section. Select Private IP. In the Network dropdown, select default.
      * Click the Set up Connection button that appears.
      * In the panel to the right, click Enable API, click Use an automatically allocated IP range, click Continue, and then click Create Connection.
      * Click Create Instance at the bottom of the page to create the database instance.
      * You might have to wait for the Private IP changes to propagate before the Create button becomes clickable.
    - Task 2: Configure a proxy on a virtual machine
      * When your application does not reside in the same VPC connected network and region as your Cloud SQL instance, use a proxy to secure its external connection.
      * In order to configure the proxy, you need the Cloud SQL instance connection name.
      * The lab comes with 2 virtual machines preconfigured with Wordpress and its dependencies. You can view the startup script and service account access by clicking on a virtual machine name. Notice that we used the principle of least privilege and only allow SQL access for that VM. There's also a network tag and a firewall preconfigured to allow port 80 from any host.
      * Navigation > Compute Engine > Click SSH next to wordpress-us-west1-proxy.
      * Download the Cloud SQL Proxy and make it executable:
        + wget https://dl.google.com/cloudsql/cloud\_sql\_proxy.linux.amd64 -O cloud\_sql\_proxy && chmod +x cloud\_sql\_proxy
      * In order to start the proxy, you need the connection name of the Cloud SQL instance. Keep your SSH window open and return to the Cloud Console.
      * On the Navigation menu (Navigation menu), click SQL.
      * Click on the wordpress-db instance and wait for a green checkmark next to its name, which indicates that it is operational (this could take a couple of minutes).
      * Note the Instance connection name; it will be used later and referred to as [SQL\_CONNECTION\_NAME].
      * In addition, for the application to work, you need to create a table. Click Databases.
      * Click Create database, type wordpress, which is the name the application expects, and then click Create.
      * Return to the SSH window and save the connection name in an environment variable, replacing [SQL\_CONNECTION\_NAME] with the unique name you copied in a previous step.
        + export SQL\_CONNECTION=[SQL\_CONNECTION\_NAME]
      * To activate the proxy connection to your Cloud SQL database and send the process to the background, run the following command:
        + ./cloud\_sql\_proxy -instances=$SQL\_CONNECTION=tcp:3306 &
      * The expected output is
        + Listening on 127.0.0.1:3306 for [SQL\_CONNECTION\_NAME]
        + Ready for new connections
      * Press ENTER.
        + The proxy will listen on 127.0.0.1:3306 (localhost) and proxy that connects securely to your Cloud SQL over a secure tunnel using the machine's external IP address.
    - Task 3: Connect an application to the Cloud SQL instance
      * In this task, you will connect a sample application to the Cloud SQL instance.
      * Configure the Wordpress application. To find the external IP address of your virtual machine, query its metadata:
        + curl -H "Metadata-Flavor: Google" http://169.254.169.254/computeMetadata/v1/instance/network-interfaces/0/access-configs/0/external-ip && echo
      * Go to the wordpress-us-west1-proxy external IP address in your browser and configure the Wordpress application. Click Let's Go.
      * Specify the following, replacing [ROOT\_PASSWORD] with the password you configured upon machine creation, and leave the remaining settings as their defaults:
        + Property Value
        + Username root
        + Password [ROOT\_PASSWORD]
        + Database Host 127.0.0.1
      * You are using 127.0.0.1, localhost as the Database IP because the proxy you initiated listens on this address and redirects that traffic to your SQL server securely.
        + 
      * Click Submit.
      * When a connection has been made, click Run the installation to instantiate Wordpress and its database in your Cloud SQL. This might take a few moments to complete.
      * Populate your demo site's information with random information and click Install Wordpress. You won't have to remember or use these details.
        + Installing Wordpress might take up to 3 minutes, because it propagates all its data to your SQL Server.
      * When a 'Success!' window appears, remove the text after the IP address in your web browser's address bar and press ENTER. You'll be presented with a working Wordpress Blog!
    - Task 4: Connect to Cloud SQL via internal IP
      * If you can host your application in the same region and VPC connected network as your Cloud SQL, you can leverage a more secure and performant configuration using Private IP.
      * By using Private IP, you will increase performance by reducing latency and minimize the attack surface of your Cloud SQL instance because you can communicate with it exclusively over internal IPs.
      * Navigation > SQL > Click wordpress-db.
        + Note the Private IP address of the Cloud SQL server; it will be referred to as [SQL\_PRIVATE\_IP].
      * On the Navigation menu, click Compute Engine.
        + Notice that wordpress-us-private-ip is located at us-central1, where your Cloud SQL is located, which enables you to leverage a more secure connection.
      * Copy the external IP address of wordpress-us-private-ip, paste it in a browser window, and press ENTER. Click Let's Go.
      * Specify the following, and leave the remaining settings as their defaults:
        + Property Value
        + Username root
        + Password type the [ROOT\_PASSWORD] configured when the Cloud SQL instance was created
        + Database Host [SQL\_PRIVATE\_IP]
      * Click Submit.
        + Notice that this time you are creating a direct connection to a Private IP, instead of configuring a proxy. That connection is private, which means that it doesn't egress to the internet and therefore benefits from better performance and security.
      * Click Run the installation. An 'Already Installed!' window is displayed, which means that your application is connected to the Cloud SQL server over private IP.
      * In your web browser's address bar, remove the text after the IP address and press ENTER. You'll be presented with a working Wordpress Blog!
        + 
  + Lab Review: Cloud SQL 11 minutes - https://youtu.be/pyZtoVLPof4
  + Cloud Spanner 2 minutes - https://youtu.be/ehE9D5VIc1g
  + Firestore 3 minutes - https://youtu.be/gtUen7\_J4Fo
  + Cloud Bigtable 4 minutes - https://youtu.be/N7\_SkPa2zPg
  + Memorystore 1 minute - https://youtu.be/Z1393KLrI7A
  + Quiz: Storage and Database Services
    - Which data storage service provides data warehouse services for storing data but also offers an interactive SQL interface for querying the data?
      * BigQuery
      * BigQuery is a data warehousing service that allows the storage of huge data sets while making them immediately processable without having to extract or run the processing in a separate service.
    - Which Google Cloud data storage service offers ACID transactions and can scale globally?
      * Cloud Spanner
      * Cloud Spanner provides ACID (Atomicity, Consistency, Isolation, Durability) properties that enable transactional reads and writes on the database. It can also scale globally.
    - What data storage service might you select if you just needed to migrate a standard relational database running on a single machine in a datacenter to the cloud?
      * Cloud SQL
      * Cloud SQL offers a PostgreSQL server or a MySQL server as a managed service.
  + Resource Management
  + Resource Manager 2 minutes - https://youtu.be/\_uiBDxOsqyg
  + Quotas 2 minutes - https://youtu.be/Xe2z7-ukGw4
  + Labels 2 minutes - https://youtu.be/erb6s\_mW1bE
  + Billing 2 minutes - https://youtu.be/oeRadY0PDyA
  + Demo: Billing Administration 4 minutes - https://youtu.be/axSbEuQ4BeI
  + Examining Billing data with BigQuery 30 minutes - <https://www.cloudskillsboost.google/course_sessions/838017/labs/112321>
    - Overview
      * In this lab, you learn how to use BigQuery to analyze billing data.
    - Objectives
      * Sign in to BigQuery from the Cloud Console
      * Create a dataset
      * Create a table
      * Import data from a billing CSV file stored in a bucket
      * Run complex queries on a larger dataset
    - Task 1: Use BigQuery to import data
      * Sign in to BigQuery and create a dataset
        + Navigation > BigQuery > If prompted, click Done.
        + Click on the View actions icon (:) next to your project ID & click Create dataset.

You can export billing data directly to BigQuery as outlined here. However, for the purposes of this lab, a sample CSV billing file has been prepared for you. It is located in a Cloud Storage bucket where it is accessible to your student account. You will import this billing information into a BigQuery table and examine it.

* + - * + Specify the following:

Property Value (type value or select option as specified)

Dataset ID: imported\_billing\_data

Data location: US

Default table expiration (check Enable table expiration): 1 days (Default maximum table age)

* + - * + Click Create Dataset. You should see imported\_billing\_data in the left pane.
      * Create a table and import
        + Click on the View actions icon next to your imported\_billing\_data dataset, and click Open and then click Create Table to create a new table.
        + For Source, specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Create table from: Google Cloud Storage

Select file from GCS bucket cloud-training/archinfra/export-billing-example.csv

File format CSV

* + - * + For Destination, specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Table name sampleinfotable

Table type Native table

* + - * + Under Schema check Auto detect.
        + Open Advanced options. Under Header rows to skip specify 1
        + Click Create Table. After the job is completed, the table appears below the dataset in the left pane.
    - Task 2: Examine the table
      * Click sampleinfotable.
        + This displays the schema that BigQuery automatically created based on the data it found in the imported CSV file. Notice that there are strings, integers, timestamps, and floating values.
        + Click Details. As you can see in Number of Rows, this is a relatively small table with 44 rows.
        + Click Preview.
        + Locate the row that has the Description: Network Internet Ingress from EMEA to Americas.

What was the total consumption and units consumed?

9,738,199 bytes

* + - * + Scroll to the Cost column.

The cost was 0.0, so with an ingress of 9.7 Mbytes, traffic from EMEA to the Americas had no charge.

* + - * + Locate the row that has the Description: Network Internet Egress from Americas to China.

Can you interpret the information?

5,542 bytes exited the Americas and was transferred to China at a charge of 1e-06.

* + - Task 3: Compose a simple query
      * When you reference a table in a query, both the dataset ID and table ID must be specified; the project ID is optional.
        + If the project ID is not specified, BigQuery will default to the current project.
      * All the information you need is available in the BigQuery interface. In the column on the left, you see the dataset ID (imported\_billing\_data) and table ID (sampleinfotable).
      * Recall that clicking on the table name brings up the Schema with all of the field names.
      * Now construct a simple query based on the Cost field.
      * Click Compose New Query.
      * Paste the following in Query Editor:
        + SELECT \* FROM `imported\_billing\_data.sampleinfotable` WHERE Cost > 0
      * Click Run. How many rows had cost greater than 0? 20 rows
      * How many rows involved non-zero charges?
        + The table shows 20 rows and they all have non-zero charges.
    - Task 4: Analyze a large billing dataset with SQL
      * In the next activity, you use BigQuery to analyze a sample dataset with 22,537 lines of billing data.
      * The cloud-training-prod-bucket.arch\_infra.billing\_data dataset used in this task is shared with the public. For more information on public datasets and how to share datasets with the public, refer to the documentation.
      * For New Query, paste the following in Query Editor:
        + SELECT

product, resource\_type, start\_time, end\_time, cost, project\_id, project\_name,

project\_labels\_key, currency, currency\_conversion\_rate, usage\_amount, usage\_unit

FROM

`cloud-training-prod-bucket.arch\_infra.billing\_data`

* + - * Click Run. Verify that the resulting table has 22,537 lines of billing data.
      * To find the latest 100 records where there were charges (cost > 0), for New Query:
        + WHERE Cost > 0 ORDER BY end\_time DESC LIMIT 100
      * To find all charges that were more than 3 dollars, for Compose New Query:
        + WHERE cost > 3
      * To find the product with the most records in the billing data, for New Query:
        + SELECT product, COUNT(\*) AS billing\_records
        + FROM `cloud-training-prod-bucket.arch\_infra.billing\_data`
        + GROUP BY product ORDER BY billing\_records DESC
      * Which product had the most billing records?
        + Cloud Pub/Sub has 10,271 records
      * To find the most frequently used product costing more than 1 dollar, for New Query:
        + Above qry + WHERE cost > 1
      * Which product had the most billing records of over $1
        + Compute Engine has 17 charges costing more than 1 dollar.
      * To find the most commonly charged unit of measure, for Compose New Query:
        + SELECT usage\_unit, COUNT(\*) AS billing\_records FROM

`cloud-training-prod-bucket.arch\_infra.billing\_data`

WHERE cost > 0 GROUP BY usage\_unit ORDER BY billing\_records DESC

* + - * What was the most commonly charged unit of measure?
        + Byte-seconds were the most commonly charged unit of measure with 2,937 requests.
      * To find the product with the highest aggregate cost, for New Query:
    - SELECT product, ROUND(SUM(cost),2) AS total\_cost

FROM `cloud-training-prod-bucket.arch\_infra.billing\_data`

GROUP BY product ORDER BY total\_cost DESC

* + - * Which product has the highest total cost?
        + Compute Engine has an aggregate cost of $112.02.
  + Lab Review: Examining Billing Data with BigQuery 7 minutes - https://youtu.be/34WHbA965nY
  + Quiz: Resource Management
    - A budget is set at $500 and an alert is set at 100%. What happens when the full amount is used?
      * A notification email is sent to the Billing Administrator.
      * Budgets in Google Cloud are not a way to prevent spending or stop resources. They are a tool for raising awareness about the consumption of resources so that a business can implement its own consumption management processes.
    - No resources in Google Cloud can be used without being associated with...
      * A project.
      * All resources in Google Cloud are tracked and their consumption is logged against a project. A project relates resources to a billing method.
    - How do quotas protect Google Cloud customers?
      * By preventing uncontrolled consumption of resources.
      * Quotas are established at reasonable defaults for common cloud usage and proof of concept activities. If you are planning to scale up a production cloud solution you may need to request that the quotas be raised. This is a reasonable checkpoint to verify that actions that might result in a large consumption of resources are reviewed.
  + Resource Monitoring
  + Google Cloud's Operations Suite 1 minute - https://youtu.be/Rx4Owb2hswU
  + Monitoring 7 minutes - https://youtu.be/L\_v0nlkIG5U
  + Resource Monitoring 33 minutes - <https://www.cloudskillsboost.google/course_sessions/838017/labs/112329>
    - Overview
      * learn how to use Cloud Monitoring to gain insight into applications that run on Google Cloud.
    - Objectives
      * Explore Cloud Monitoring
      * Add charts to dashboards
      * Create alerts with multiple conditions
      * Create resource groups
      * Create uptime checks
    - Task 1: Create a Cloud Monitoring workspace
      * Verify resources to monitor
        + Three VM instances have been created for you that you will monitor.
        + Navigation > Compute Engine > VM instances. Notice the nginxstack-1, nginxstack-2 and nginxstack-3 instances.
      * Create a Monitoring workspace
        + You will now setup a Monitoring workspace that's tied to your Qwiklabs GCP Project. The following steps create a new account that has a free trial of Monitoring.
        + In the Google Cloud Platform Console, click on Navigation menu > Monitoring.
        + Wait for your workspace to be provisioned.
        + When the Monitoring dashboard opens, your workspace is ready.
        + Why is monitoring important to Google?

It is at the base of site reliability which incorporates aspects of software engineering and applies that to operations whose goals are to create ultra-scalable and highly reliable software systems.

* + - Task 2: Custom dashboards
      * Create a dashboard.
        + In the left pane, click Dashboards. Click +Create Dashboard.
        + For New Dashboard Name, type My Dashboard.
      * Add a chart. From Chart library, Select Line.
        + For Title, give your chart a name (you can revise this before you save based on the selections you make).
        + For Resource type, select VM Instance.
        + For Metric, select a metric to chart for the Instance resource, such as CPU utilization or CPU usage. Click + Add Filter and explore the various options.
      * Metrics Explorer
        + The Metrics Explorer allows you to examine resources and metrics without having to create a chart on a dashboard. Try to recreate the chart you just created using the Metrics Explorer.
        + In the left pane, click Metrics explorer.
        + For Resource type and Metric, select a resource name and metric.
        + Explore the various options and try to recreate the chart you created earlier.

Not all metrics are currently available on the Metrics Explorer, so you might not be able to find the exact metric you used on the previous step.

* + - Task 3: Alerting policies
      * What is not a recommended best practice for alerts?
        + Report all noise to ensure all data points are presented.
      * Create an alert and add the first condition
        + In the left pane, click Alerting. Click + Create Policy. Click Add Condition.
        + For Find resource type and metric, select VM Instance.

If you cannot locate the VM Instance resource type, you might have to refresh the page.

* + - * + Select a metric you are interested in evaluating, such as CPU usage or CPU Utilization.
        + Under Configuration, for Condition, select is above.
        + Specify the threshold value and for how long the metric must cross this set value before the alert is triggered. For example, for THRESHOLD, type 20 and set FOR to 1 minute.
        + Click ADD.
        + In Policy Triggers, for Trigger when, click All conditions are met. Click Next.
      * Configure notifications and finish the alerting policy
        + Click on dropdown 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.
        + Enter your personal email in the Email Address field and a Display name. Click 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. Click Notification Channels again if needed.
        + Now, select your Display name and click OK. Click Next.
        + Enter a name of your choice in Alert name field. Click Save.
    - Task 4: Resource groups
      * In the left pane, click Groups. Click + Create Group.
      * Enter a name for the group. For example: VM instances
      * In the Criteria section, type nginx in the value field below Contains.
      * Click DONE. Click CREATE.
      * Review the dashboard Cloud Monitoring created for your group.
    - Task 5: Uptime monitoring
      * Select all valid targets for Cloud Monitoring uptime alert notifications.
        + email
        + webhook
        + SMS
        + Pub/sub
        + 3rd party service
      * In the Monitoring tab, click on Uptime Checks. Click + Create Uptime Check.
      * Specify the following, and leave the remaining settings as their defaults:
        + Property Value (type value or select option as specified)
        + Title Enter a title then click Next
        + Protocol HTTP
        + Resource Type Instance
        + Applies To Group
        + Group Select your group
        + Check Frequency 1 minute
      * Click on Next to leave the other details to default. Under Alert & Notification, select your Notification Channels from the dropdown.
      * 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.
      * The uptime check you configured takes a while for it to become active.
    - Task 6: Disable the alert
      * Disable the alert Alerting policies stay active for a while after a project is deleted, just in case it needs to be reinstalled. Since this is a lab, and you will not have access to this project again, remove the alerting policy you created.
      * You should still be in the Alerting section.
      * From you alert's Policy details page, click the Enabled link at the top of the page.
      * You will be asked to confirm that you want to disable the alerting policy - click Disable.
      * The link will now say Disabled.
  + Lab Review: Resource Monitoring 8 minutes - https://youtu.be/g8H50O2LLGE
  + Logging 2 minutes - https://youtu.be/s0pPc0SkUUo
  + Error Reporting 1 minute - https://youtu.be/ChW6MBFCCNI
  + Tracing 1 minute - https://youtu.be/4mA8tUt4Udk
  + Debugging 1 minute - https://youtu.be/IyyUCFRBFVI
  + Error Reporting and Debugging 30 minutes - https://www.cloudskillsboost.google/course\_sessions/838017/labs/112336
    - Overview
      * In this lab, you learn how to use Cloud Error Reporting and integrate Cloud Debugger.
    - Objectives
      * Launch a simple Google App Engine application
      * Introduce an error into the application
      * Explore Cloud Error Reporting
      * Use Cloud Debugger to identify the error in the code
      * Fix the bug and monitor in Cloud Operations
    - Task 1: Create an application
      * Get and test the application
      * Activate Cloud Shell
        + To create a local folder and get the App Engine Hello world application, run the following commands:

mkdir appengine-hello

cd appengine-hello

gsutil cp gs://cloud-training/archinfra/gae-hello/\* .

* + - * + To run the application using the local development server in Cloud Shell, run the following command:

dev\_appserver.py $(pwd)

* + - * + In Cloud Shell, click Web Preview > Preview on port 8080 to view the application. You may have to collapse the Navigation menu pane to access the Web Preview icon.
        + A new browser window opens to the localhost and displays the message Hello, World!
        + In Cloud Shell, press Ctrl+C to exit the development server.
      * Deploy the application to App Engine
        + To deploy the application to App Engine, run the following command:

gcloud app deploy app.yaml

* + - * + If prompted for a region, enter the number corresponding to a region.
        + When prompted, type Y to continue.
        + When the process is done, verify that the application is working by running the following command:

gcloud app browse

* + - * + If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app. You might have to refresh the page for the application to load.
        + If needed, press Ctrl+C to exit the development mode.
      * Introduce an error to break the application
        + To examine the main.py file, run the following command:

cat main.py

* + - * + Notice that the application imports webapp2.
        + You will break the configuration by replacing the import library with one that doesnt exist.
        + To use the sed stream editor to change the import library to the nonexistent webapp22, run the following command:

sed -i -e 's/webapp2/webapp22/' main.py

* + - * + Notice that the application now tries to import webapp22.
      * Re-deploy the application to App Engine
        + To re-deploy the application to App Engine, run the following command:

gcloud app deploy app.yaml --quiet

* + - * + The --quiet flag disables all interactive prompts when running gcloud commands. If input is required, defaults will be used. In this case, it avoids the need for you to type Y when prompted to continue the deployment.
        + When the process is done, verify that the application is broken by running the following command:

gcloud app browse

* + - * + If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.
        + If needed, press Ctrl+C to exit development mode.
        + Leave Cloud Shell open.
        + Which service requires a logging agent installed to collect and send logs to Cloud Operations?

Compute Engine

* + - Task 2: Explore Cloud Error Reporting
      * View Error Reporting and trigger additional errors
        + Navigation > Error Reporting.

You should see an error regarding the failed import of webapp22. Click Auto reload.

* + - * + In Cloud Shell, run the following command:

gcloud app browse

* + - * + If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.
        + Click the resulting link several times to generate more errors.
        + The number of errors is displayed in the Occurrences column. The graph shows the frequency of errors over time, and the number represents the sum of errors. This is a very handy visual indicator of the state of the error.
        + The First seen and Last seen columns show when the error was first seen and when it was last seen, respectively. This can help identify changes that might have triggered the error. In this case, it was the upload of the new version of app engine code.
        + Which service(s) are currently supported by Cloud Error Reporting?

App Engine Flexible

Compute Engine

Kubernetes

App Engine Standard

* + - * View details and identify the cause
        + Click the Error name: ImportError: No module named webapp22.
        + Now you can see a detailed graph of the errors. The Response Code field shows the explicit error: a 500 Internal Server Error.
        + For Stack trace sample, click Parsed. This opens the Cloud Debugger, showing the error in the code!
      * View the logs and fix the error
        + At the bottom of the Debug page, just below the code, find and open View logs in a new window or tab. Here you can find more detailed historical information about the error.
        + Introduce more errors by refreshing the page of your application. If you closed your application, use gcloud app browse and click the link to view the broken app.
        + On the Cloud Error Reporting page, ensure that Auto Reload is enabled to watch the addition of new errors.
        + In Cloud Shell, fix the error by running the following command:

sed -i -e 's/webapp22/webapp2/' main.py

* + - * + To re-deploy the application to App Engine, run the following command:

gcloud app deploy app.yaml --quiet

* + - * + When the process is done, to verify that the application is working again, run the following command:

gcloud app browse

* + - * + If Cloud Shell does not detect your browser, click the link in the Cloud Shell output to view your app.
        + On the Cloud Error Reporting page, ensure that Auto Reload is enabled to and see that no new errors are added.
        + What would not be considered a benefit of Cloud Operations?

Boosts all network performance

* + Lab Review: Error Reporting and Debugging 8 minutes - https://youtu.be/zqUQc6x1l3M
  + Quiz: Resource Monitoring
    - Google Cloud’s operations suite integrates several technologies, including monitoring, logging, error reporting, and debugging that are commonly implemented in other environments as separate solutions using separate products. What are key benefits of integration of these services?
      * Reduces overhead, reduces noise, streamlines use, and fixes problems faster
      * Integration with Google Cloud’s operations suite streamlines and unifies these traditionally independent services, making it much easier to establish procedures around them and to use them in continuous ways.
    - What is the foundational process at the base of Google's Site Reliability Engineering (SRE) ?
      * Monitoring.
      * Before you can take any of the other actions, you must first be monitoring the system.
    - What is the purpose of the Cloud Trace service?
      * Reporting on latency as part of managing performance.
      * Cloud Trace provides latency sampling and reporting for Google App Engine, Google HTTP(S) load balancers, and applications instrumented with the Cloud Trace SDKs. Reporting includes per-URL statistics and latency distributions.