CS530 F23 Week 6 Lab — Amelia Miner
Prof. Feng — 11/2/23

* Again, due to availability issues, all actions were performed in the us-west1-a zone instead of us-west1-b.
✅ indicates that a section was completed but did not request any screenshots or written answers.

TABLE OF CONTENTS:

Amelia Miner
11/2/23

I. Lab 6.1a - EB Guestbook

1. Elastic Beanstalk

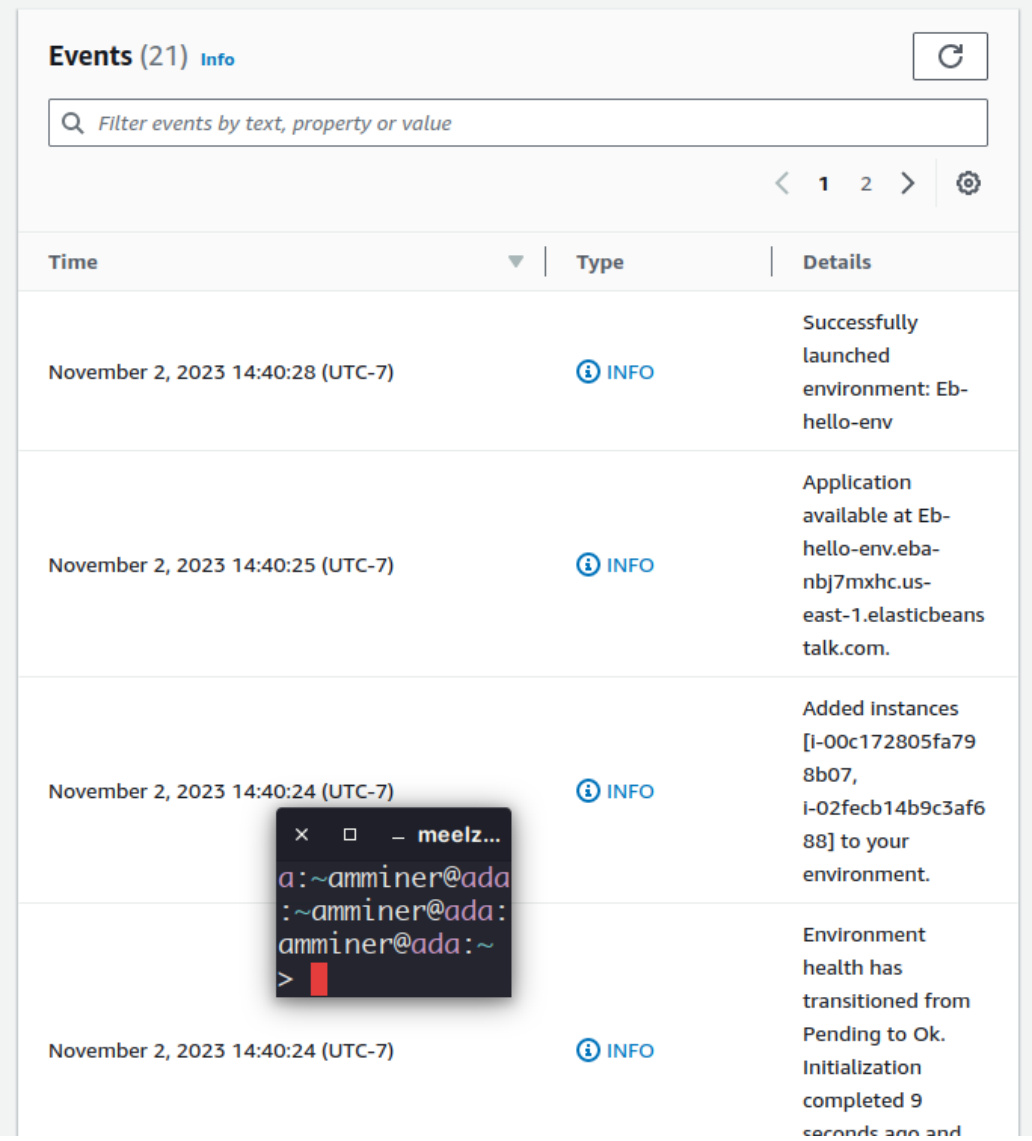
✅

2. Elastic beanstalk sample application

✅

3. Running the application

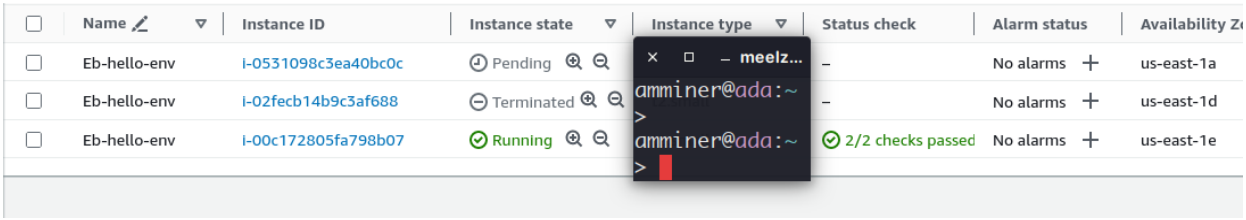
- observe the extreme convenience of the Elastic Beanstalk™.



Amelia Miner
11/2/23

4. Handling Failures Seamlessly

- Terminate one of the instances. Wait a while. Show the terminated instance alongside its replacement being started.



5. Clean Up

✅

6. Elastic Beanstalk Guestbook (CLI edition - CI/CD scripts do this)

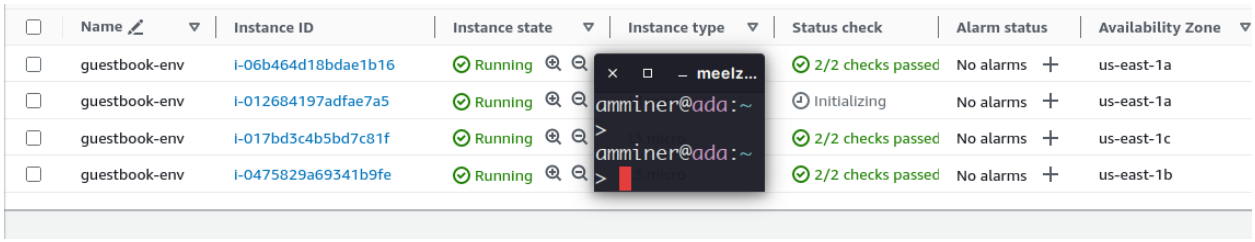
✅

7. Deploying the Guestbook

- Note how long this takes since we're creating ec2 instances instead of deploying container images to already-running machines.

- Sign the guestbook with “Hello Elastic Beanstalk!”.

- Show that the minimum number of instances are running.



8. Clean Up

✅

Amelia Miner
11/2/23

II. Lab 6.1g - App Engine Guestbook

1. App Engine

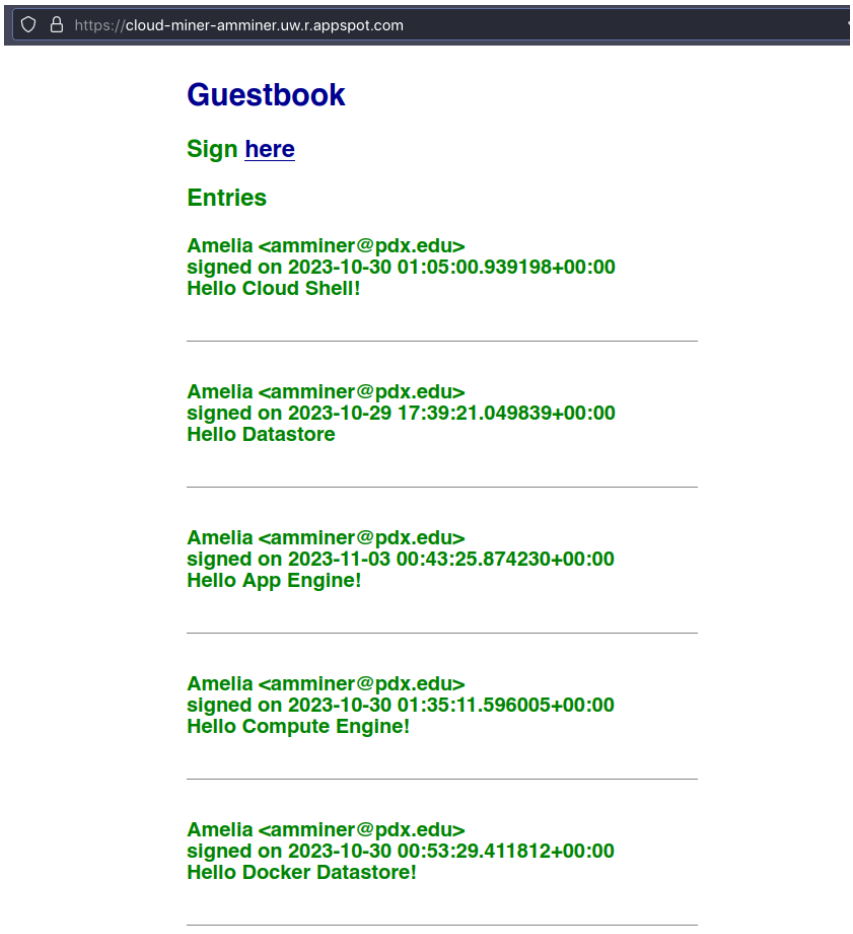
- `gcloud services enable appengine.googleapis.com`
  ✅

2. app.yaml

- Read `app.yaml`.
  ✅

3. Deploying the Guestbook

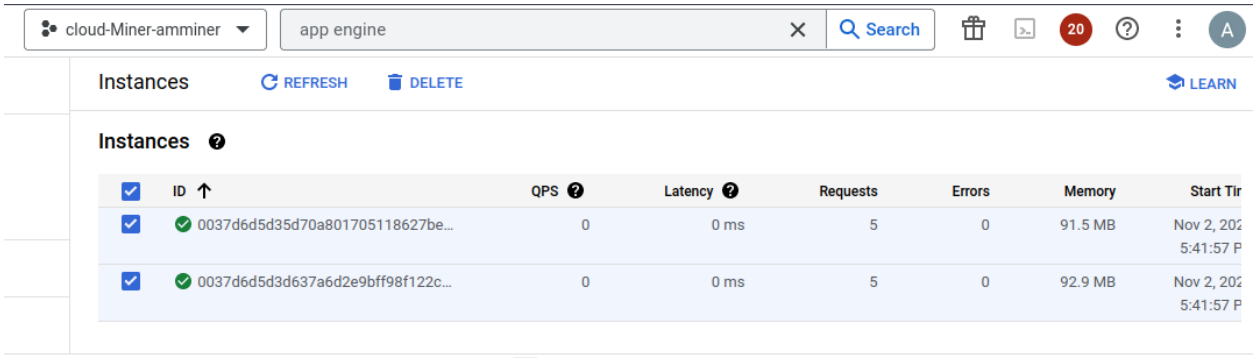
- Deploy with the guestbook service account from earlier.
  ✅
- Sign the guestbook with “Hello App Engine!”.



Amelia Miner
11/2/23

4. Handling Failures Seamlessly

- visit the App Engine home page and click on "Instances" to view the machines that have been brought up to serve your application. Show them running.



- Delete one and watch it be replaced.
  ✅

5. Clean Up

✅

Amelia Miner

III. Lab 6.2g - Cloud Run, Secret Manager (Web Proxy)

1. Cloud Run

   ✅

2. Application

   ✅

3. app.py

   ✅

4. templates/proxy.html

   ✅

5. requirements.txt

   ✅

6. Dockerfile

   ✅

7. Build and test in Cloud Shell

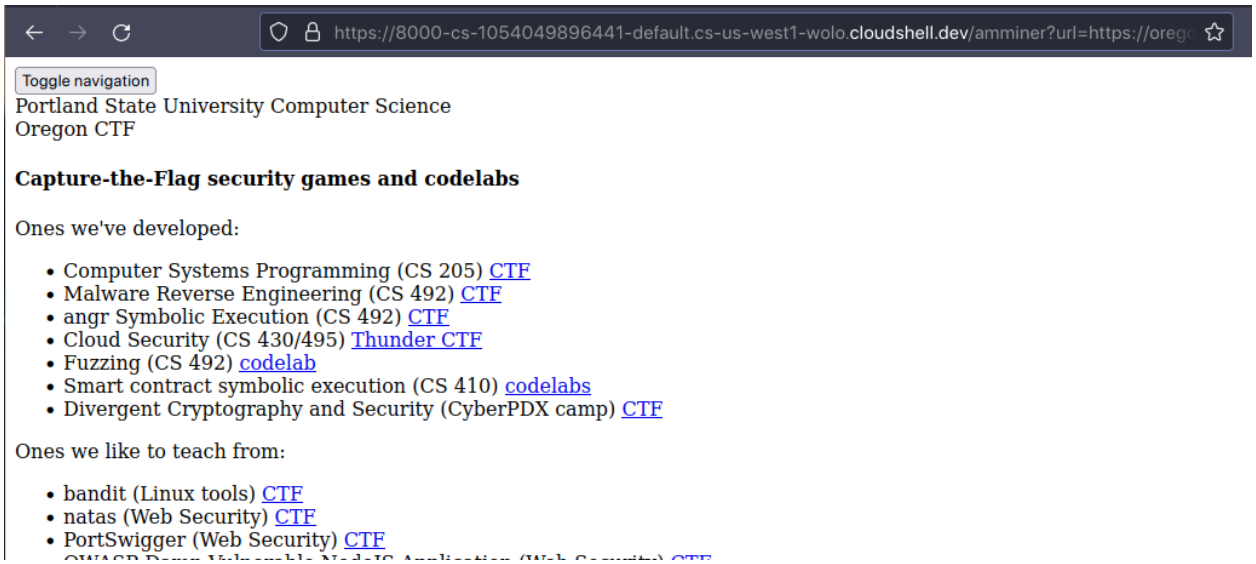
- build the image and run it, passing the port in as an env var and using the --rm flag to stop the container when you send a keyboard interrupt.

  ✅

Amelia Miner
11/2/23

8. Setup Secret Proxy

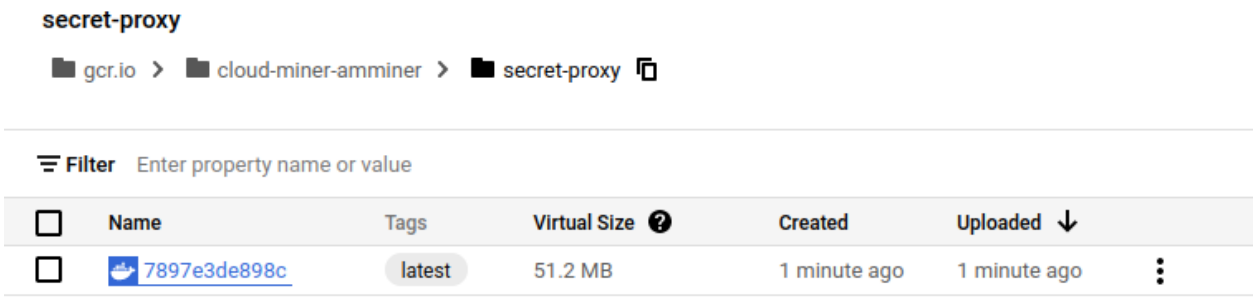
- Run it again, but with the SECRET_PROXY_ROUTE env var set to your Odin ID. Enter oregonctf.org.



- What is the security advantage of passing in the secret proxy route as an environment variable?
  Even with the source code public, if we run an instance of the application we can set the proxy route to whatever we want, sort of like a password for access to the application. I'm a little skeptical - as far as I'm aware my DNS traffic is not encrypted, so it seems like there's significant exposure of the secret URL there.

9. Cloud Build and Container Registry

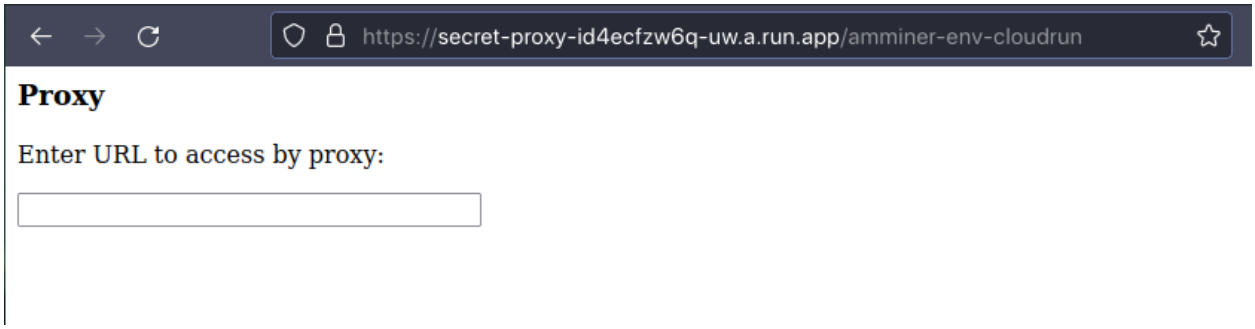
- Use gcloud to build and upload a container to gcr:
  gcloud builds submit --tag gcr.io/${GOOGLE_CLOUD_PROJECT}/secret-proxy
  ✅
- Show the image in the gcr GUI with its size.



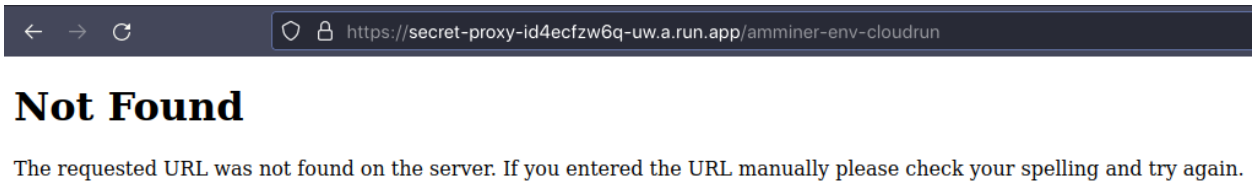
Amelia Miner

10. Deploy to Cloud Run

- `gcloud run deploy secret-proxy \`
  `--image gcr.io/$GOOGLE_CLOUD_PROJECT/secret-proxy \`
  `--platform=managed --region=us-west1 --allow-unauthenticated \`
  `--update-env-vars=SECRET_PROXY_ROUTE=/amminer-env-cloudrun`
  ✅
- Show the proxy page loaded with the URL



- `gcloud run services update secret-proxy \`
  `--remove-env-vars=SECRET_PROXY_ROUTE \`
  `--region=us-west1`
  ✅
- Show that the proxy page is no longer available



11. Secret Manager

- Enable the API and run these commands:

```
echo -n "/amminer-env-secretmanager" | gcloud secrets create
proxy-secret --data-file=-

COMPUTE_DSA=$(gcloud compute project-info describe
--format="value(defaultServiceAccount)")

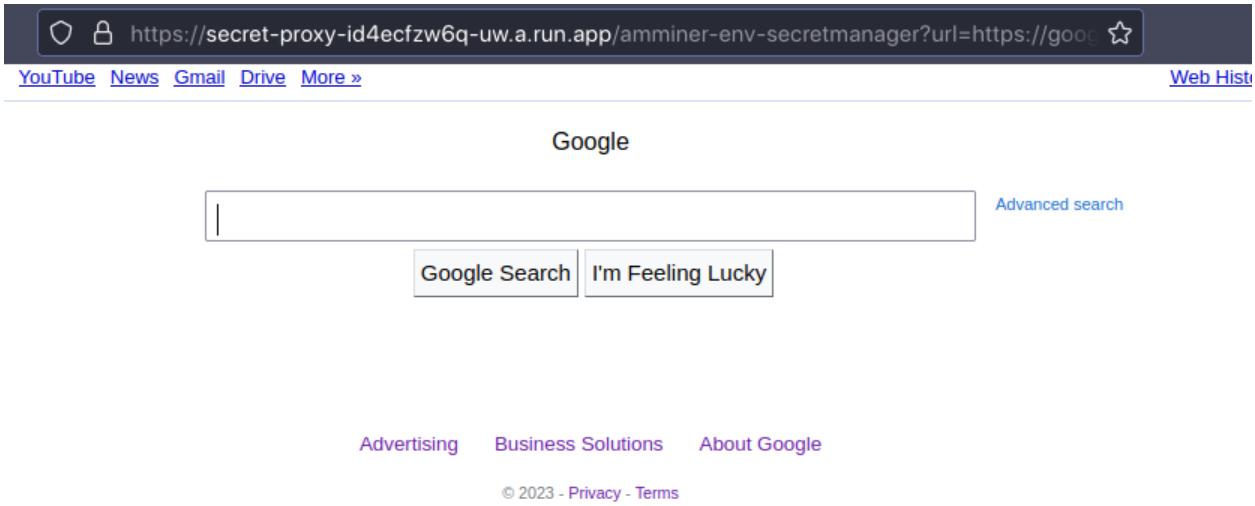
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member serviceAccount:$COMPUTE_DSA \
  --role roles/secretmanager.secretAccessor
```

✅

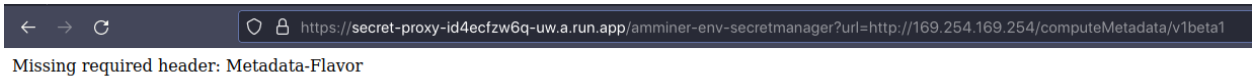
Amelia Miner
11/2/23

12. Deploy to Cloud Run with Secret Manager

- ```
  gcloud run services update secret-proxy \
    --update-secrets=SECRET_PROXY_ROUTE=proxy-secret:latest \
    --region=us-west1
  ```
  ✅
- Visit the proxy page and enter https://google.com/.



- Enter http://169.254.169.254/computeMetadata and http://169.254.169.254/computeMetadata/v1. Read the articles linked in the lab materials. Identify the vulnerability Google has prevented.
  I also tried 192.254.169.254/v1beta1, as discussed in the shopify link's initial report. It looks like Google has patched their systems so that user code running in their infrastructure doesn't have the ability to view sensitive project data anymore.



13. Clean Up

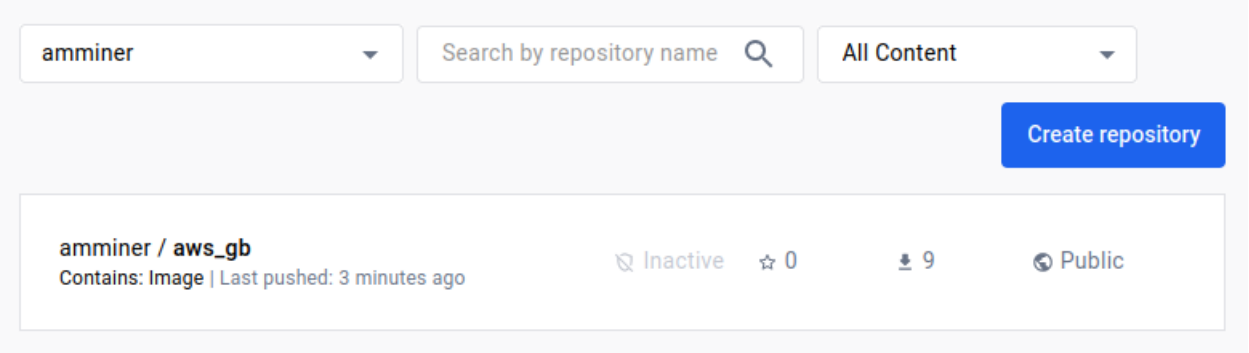
✅
  ```

Amelia Miner
11/2/23

IV. Lab 6.3a - ECS Guestbook

1. Prepare a Container Image

I feel concerned - this section says "Show that your image was uploaded to your account on Docker Hub." but not in bold. I was under the impression that any time we needed to include a screenshot it would be in bold. I hope I haven't missed anything like this in the preceding labs.



2. ECS Overview

ECS >
ECS Cluster (Compute etc. resources) (fargate) >
ECS Service (How to run it) >
ECS Task (What to run)

✅

3. ECS Task Definition

I'm sure this happens all the time, but just a heads up, the GUI in Create New Task Definition has re-ordered a little. Maybe that's just for me.

- Create a task definition for the guestbook app

✅

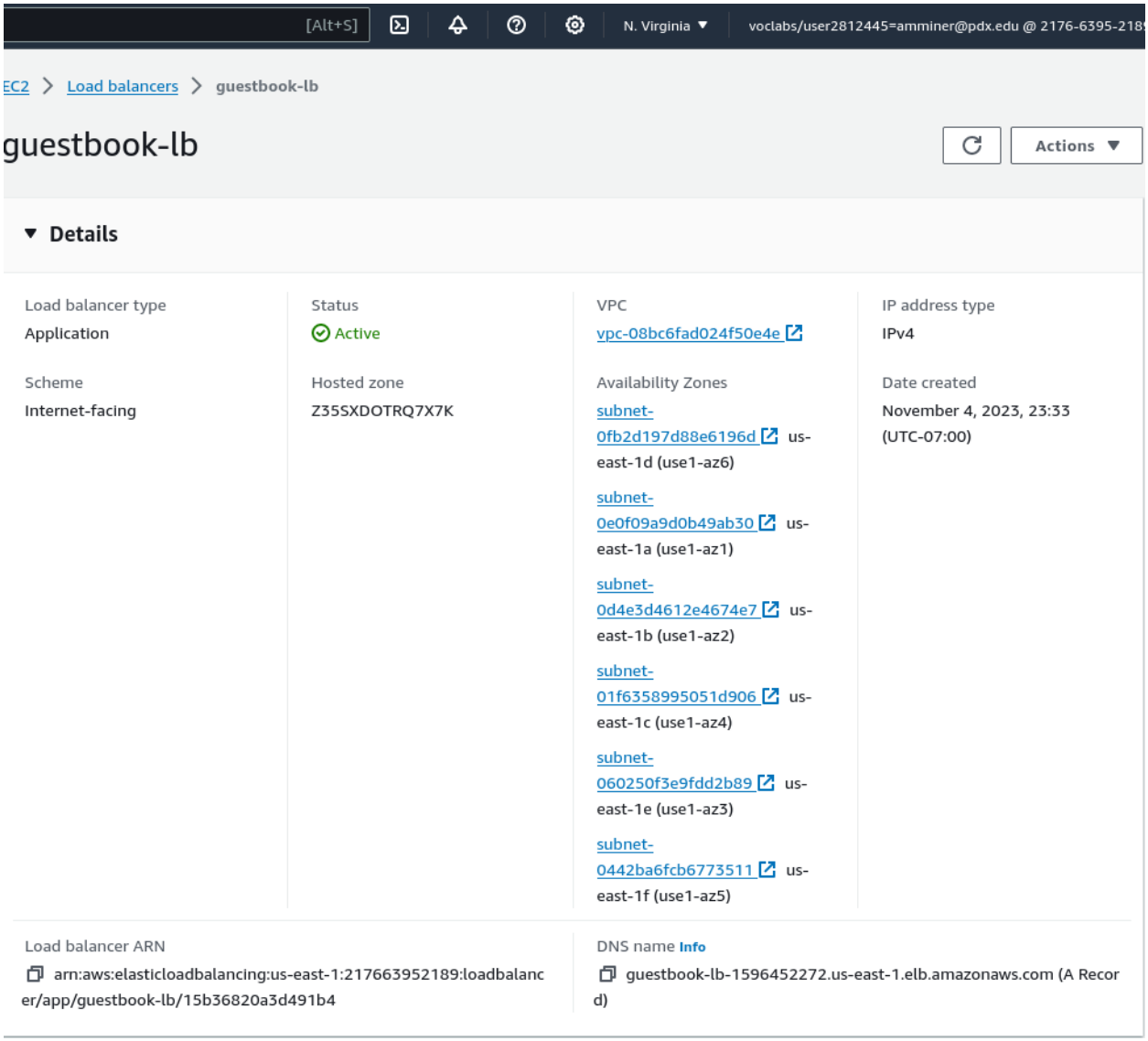
Amelia Miner
11/2/23

4. Create a Cluster and a Service

- Create a new cluster, then a new task in that cluster from the guestbook-config task definition.
  ✅

5. Examine the Service

- Show the DNS name of the guestbook-lb load balancer.



Amelia Miner
11/2/23

6. Visit the Site

- Navigate to the site via the DNS name of the frontend. Sign the guestbook “Hello ECS!”. Take a screenshot of the Guestbook app running in a browser that includes the DNS name of the site.

7. Clean Up

- Delete the guestbook service
- delete the cluster
- delete the task definition
  ✅

Amelia Miner
11/2/23

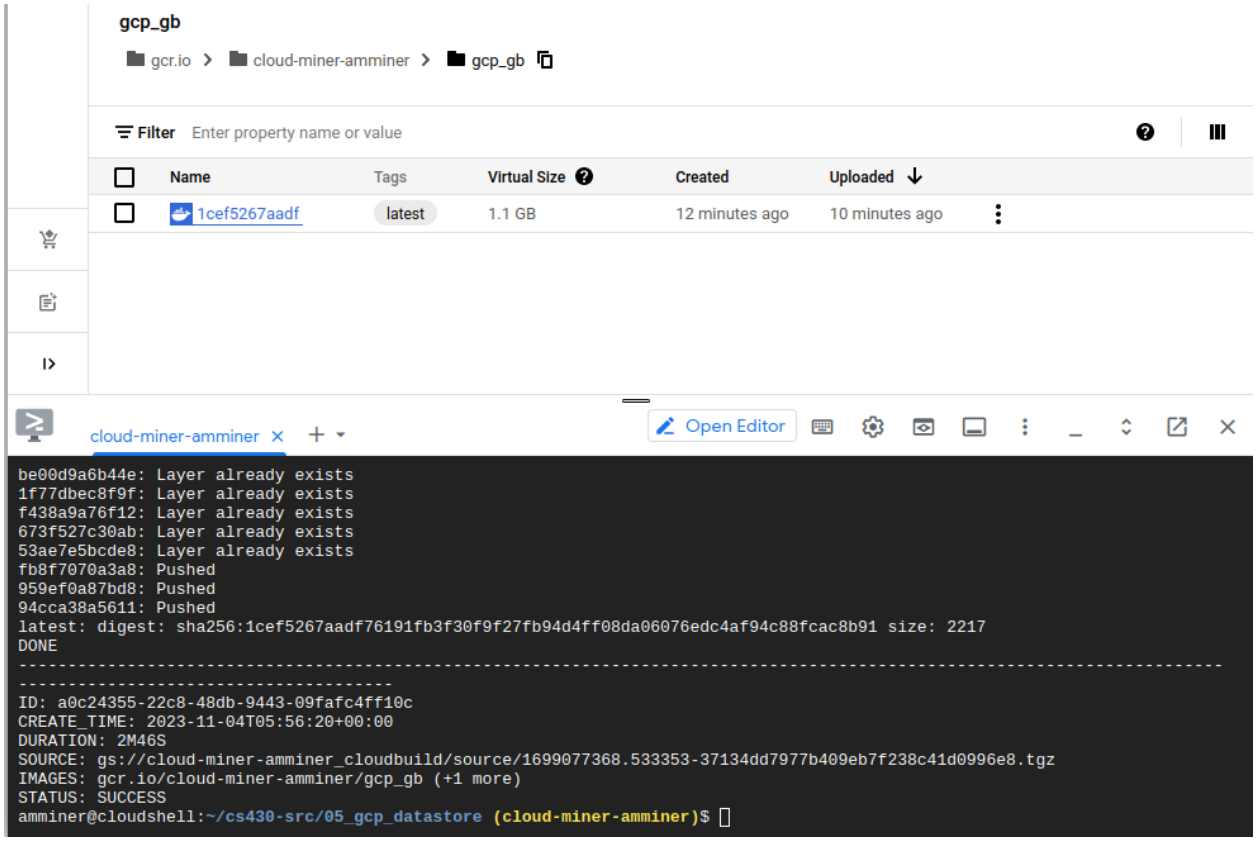
V. Lab 6.3g - Cloud Run Guestbook

1. Cloud Run

✅

2. Prepare a container image

- We're using Google's container registry service, gcr.io
- Build and push the container and show the output including the duration of the command's execution.



3. View Container Image

- Show the container and its virtual size in the GUI.
  See above
- View the Container Registry artifacts in their specially-named Cloud Storage Bucket.
  ✅

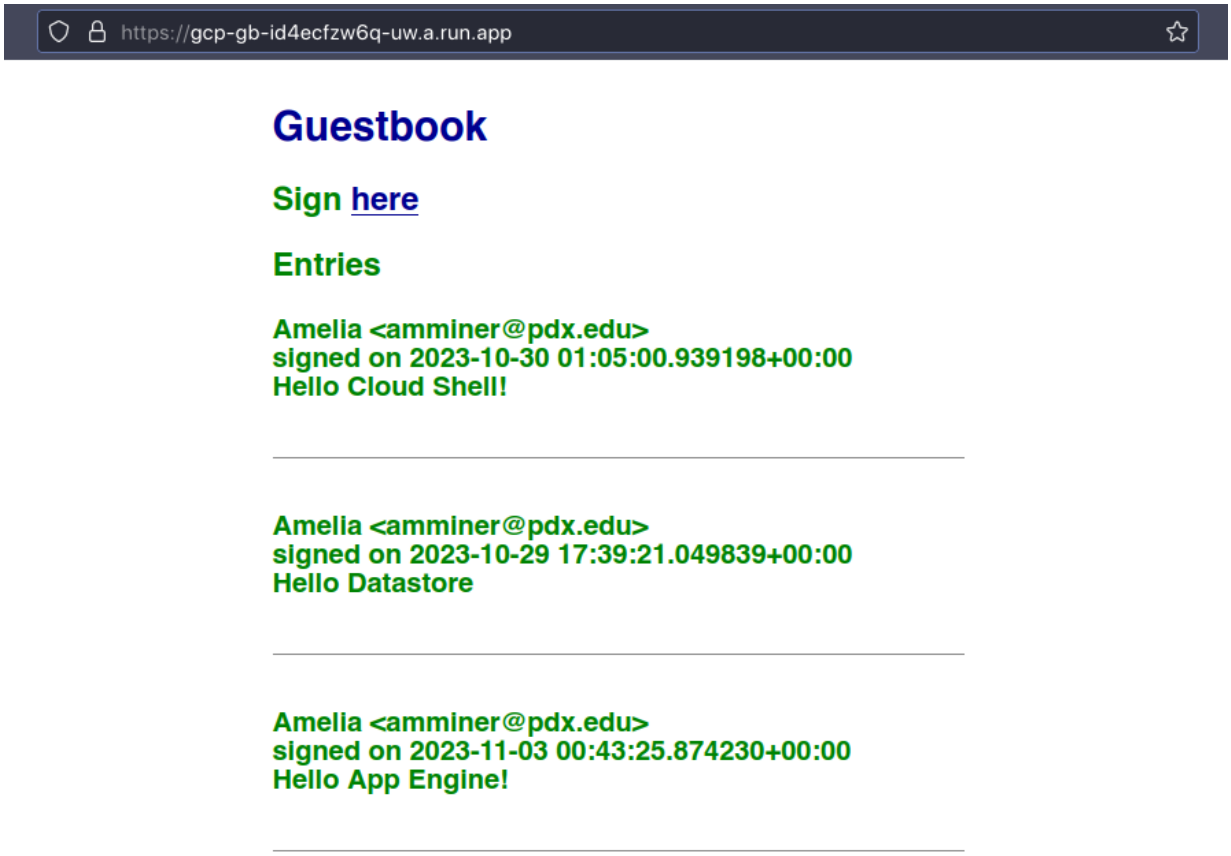
Amelia Miner
11/2/23

4. Deploy Container with Minimal Privileges

- We'll reuse the service account we made in a previous lab with the Cloud Datastore User role attached.
  ✅

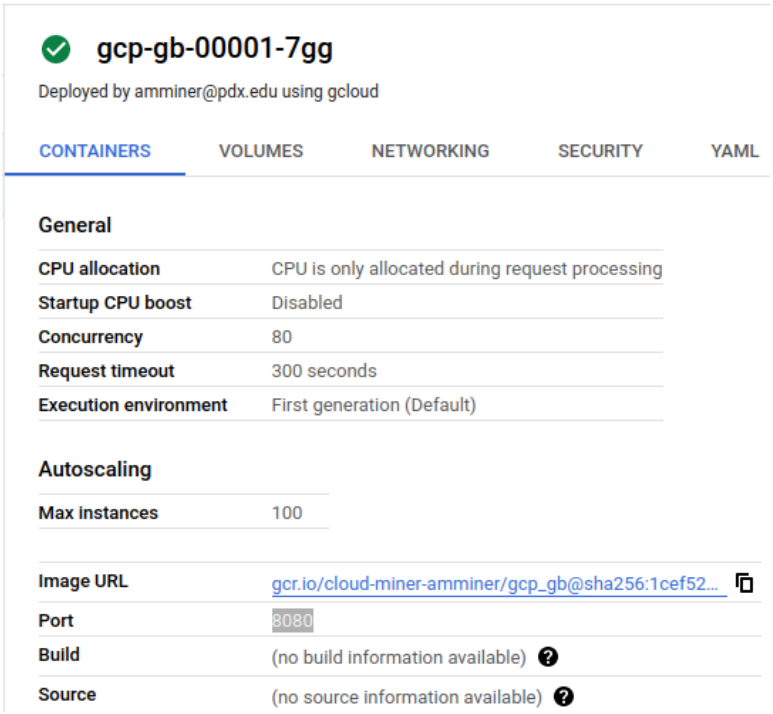
5. View the Guestbook

- Show the site signed "Hello Cloud Run!" with the URL bar in view.

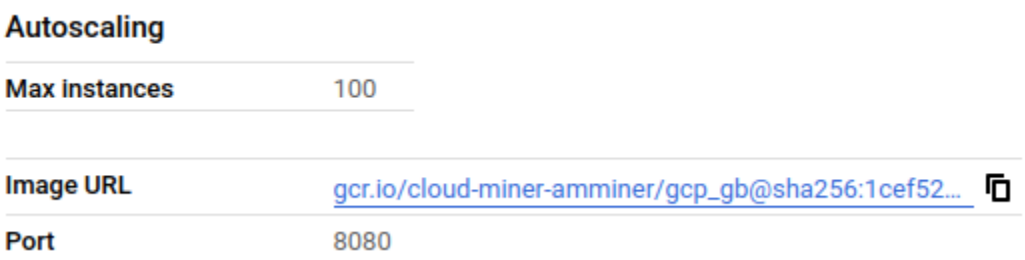


Amelia Miner
11/2/23

- From the web console, visit Cloud Run, click on the service you created, then click on "Revisions". As the UI shows, one can create multiple revisions of a service and manage user traffic between.
    - What port do container instances listen on?
      8080



    - What is the maximum number of instances Cloud Run will autoscale up to for your service?
      100



6. Clean Up

✅

Amelia Miner
11/2/23

VI. Lab 6.4g - Cloud Functions, PubSub

1. Cloud Functions Image Blurring

- clone the GCP python-docs-sample repository and cd to functions/imagemagick
- create a bucket with a unique name for input to the function, and another for output.
  ✅

2. Services Setup

- Enable the vision, cloudfunctions, and cloudbuild apis.
  ✅

3. Code

✅

4. -

- Examine the __blur_image function.
- After downloading the file from the bucket, where is it stored?
  A temporary local file. It seems like we could probably get away with just keeping it in memory. I find myself wondering why we use the disk for this.

```
# [START functions_imagemagick_blur]
# Blurs the given file using ImageMagick.
def __blur_image(current_blob):
    file_name = current_blob.name
    _, temp_local_filename = tempfile.mkstemp()

    # Download file from bucket.
    current_blob.download_to_filename(temp_local_filename)
    print(f"Image {file_name} was downloaded to {temp_local_filename}.")

    # Blur the image using ImageMagick.
    with Image(filename=temp_local_filename) as image:
        image.resize(*image.size, blur=16, filter="hamming")
        image.save(filename=temp_local_filename)

    print(f"Image {file_name} was blurred.")

    # Upload result to a second bucket, to avoid re-triggering the function.
    # You could instead re-upload it to the same bucket + tell your function
    # to ignore files marked as blurred (e.g. those with a "blurred" prefix)
    blur_bucket_name = os.getenv("BLURRED_BUCKET_NAME")
    blur_bucket = storage_client.bucket(blur_bucket_name)
    new_blob = blur_bucket.blob(file_name)
    new_blob.upload_from_filename(temp_local_filename)
    print(f"Blurred image uploaded to: gs://{blur_bucket_name}/{file_name}")

    # Delete the temporary file.
    os.remove(temp_local_filename)


# [END functions_imagemagick_blur]
amminer@cloudshell:~/python-docs-samples/functions/imagemagick (cloud-miner-amminer)$
```

Amelia Miner

- What class in the ImageMagick package is used to blur the file?
  Image

```
# [START functions_imagemagick_blur]
# Blurs the given file using ImageMagick.
def __blur_image(current_blob):
    file_name = current_blob.name
    _, temp_local_filename = tempfile.mkstemp()

    # Download file from bucket.
    current_blob.download_to_filename(temp_local_filename)
    print(f"Image {file_name} was downloaded to {temp_local_filename}.")

    # Blur the image using ImageMagick.
    with Image(filename=temp_local_filename) as image:
        image.resize(*image.size, blur=16, filter="hamming")
        image.save(filename=temp_local_filename)

    print(f"Image {file_name} was blurred.")

    # Upload result to a second bucket, to avoid re-triggering the function.
    # You could instead re-upload it to the same bucket + tell your function
    # to ignore files marked as blurred (e.g. those with a "blurred" prefix)
    blur_bucket_name = os.getenv("BLURRED_BUCKET_NAME")
    blur_bucket = storage_client.bucket(blur_bucket_name)
    new_blob = blur_bucket.blob(file_name)
    new_blob.upload_from_filename(temp_local_filename)
    print(f"Blurred image uploaded to: gs://{blur_bucket_name}/{file_name}")

    # Delete the temporary file.
    os.remove(temp_local_filename)


# [END functions_imagemagick_blur]
amminer@cloudshell:~/python-docs-samples/functions/imagemagick (cloud-miner-amminer)$
```

- What lines of code perform the blurring of the image and its storage back into the filesystem?
  Lines 73 and 74 respectively.

```
60
61  # [START functions_imagemagick_blur]
62  # Blurs the given file using ImageMagick.
63  def __blur_image(current_blob):
64      file_name = current_blob.name
65      _, temp_local_filename = tempfile.mkstemp()
66
67      # Download file from bucket.
68      current_blob.download_to_filename(temp_local_filename)
69      print(f"Image {file_name} was downloaded to {temp_local_filename}.")
70
71      # Blur the image using ImageMagick.
72      with Image(filename=temp_local_filename) as image:
73          image.resize(*image.size, blur=16, filter="hamming")
74          image.save(filename=temp_local_filename)
75
76      print(f"Image {file_name} was blurred.")
77
78      # Upload result to a second bucket, to avoid re-triggering the function.
79      # You could instead re-upload it to the same bucket + tell your function
80      # to ignore files marked as blurred (e.g. those with a "blurred" prefix)
81      blur_bucket_name = os.getenv("BLURRED_BUCKET_NAME")
82      blur_bucket = storage_client.bucket(blur_bucket_name)
83      new_blob = blur_bucket.blob(file_name)
84      new_blob.upload_from_filename(temp_local_filename)
85      print(f"Blurred image uploaded to: gs://{blur_bucket_name}/{file_name}")
86
87      # Delete the temporary file.
88      os.remove(temp_local_filename)
89
90
91  # [END functions_imagemagick_blur]
amminer@cloudshell:~/python-docs-samples/functions/imagemagick (cloud-miner-amminer)$
```

Amelia Miner
11/2/23

5. Set Up Service Account

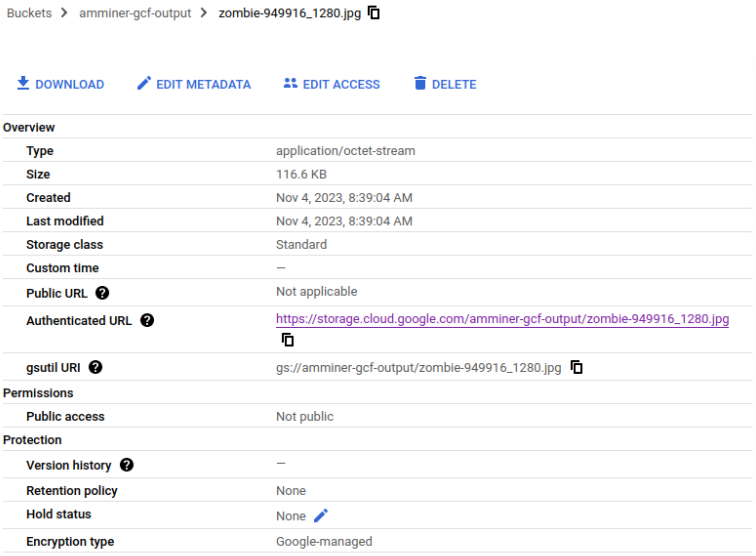
- By default, cloud functions are run using the app engine default service account. We will add the Storage Object Admin role to this account for this lab.
  ✅

6. Deploy the Function

- Deploy the function and set it to be triggered when a new file is uploaded to the input bucket.
  ✅

7. Test Function

- Upload the example image to the input bucket and show the blurred image in the output bucket.



Amelia Miner

- show the log entries for the invocation.

```
LEVEL: D
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:39:04.056
LOG: Function execution took 11960 ms, finished with status: 'ok'

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:39:04.054
LOG:

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:39:04.054
LOG: Blurred image uploaded to: gs://amminer-gcf-output/zombie-949916_1280.jpg

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:39:03.924
LOG:

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:39:03.924
LOG: Image zombie-949916_1280.jpg was blurred.

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:38:52.921
LOG:

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:38:52.921
LOG: Image zombie-949916_1280.jpg was downloaded to /tmp/tmpbq8bq_qs.

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:38:52.866
LOG:

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:38:52.866
LOG: The image zombie-949916_1280.jpg was detected as inappropriate.

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:38:52.481
LOG:

LEVEL: I
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:38:52.481
LOG: Analyzing zombie-949916_1280.jpg.

LEVEL: D
NAME: blur_offensive_images
EXECUTION_ID: a0vvg4faus6j
TIME_UTC: 2023-11-04 15:38:52.095
LOG: Function execution started
```

8. Clean Up

✅

Amelia Miner
11/2/23

9. PubSub

✅

10. Create VM for Subscriber

- Create a service account for the subscriber, attach the pubsub.editor role, create and launch a VM (see lab for specs), and ssh in.
  ✅

11. PubSub via CLI

- In cloud shell:
  - Create a pubsub topic named with your Odin ID.
  - Publish a message to the topic with the text “Message #1”
- In VM:
  - Create a subscription to the topic you just created in gcloud.
    - By default this will be pull-based
  - Attempt to pull a message
- Why are no items returned by the pull attempt?
  Because when we published the message the subscription didn’t exist, so the publisher just sort of dropped it.

```
amminer@pubsub:~$ gcloud pubsub subscriptions create sub-amminer --topic projects/cloud-miner-amminer/topics/top
c-amminer
Created subscription [projects/cloud-miner-amminer/subscriptions/sub-amminer].
amminer@pubsub:~$ gcloud pubsub subscriptions pull sub-amminer
Listed 0 items.
amminer@pubsub:~$
```

```
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud pubsub topics publish projects/cloud-miner-amminer/topics/topic-amminer --message="Message #1"
messageIds:
- '9571409574872084'
amminer@cloudshell:~ (cloud-miner-amminer)$
```

12. -

- In cloud shell, publish another message “Message #2”. What is the messageId?
  9571133818194729

```
amminer@cloudshell:~ (cloud-miner-amminer)$ gcloud pubsub topics publish projects/cloud-miner-amminer/topics/topic-amminer --message="Message #2"
messageIds:
- '9571133818194729'
amminer@cloudshell:~ (cloud-miner-amminer)$
```

- In the VM, pull the message. Show the message and its messageId.
  Wow - this output format is awful!

```
amminer@pubsub:~$ gcloud pubsub subscriptions pull sub-amminer
|    DATA    |     MESSAGE_ID     | ORDERING_KEY | ATTRIBUTES | DELIVERY_ATTEMPT |
                   ACK_ID
| Message #2 | 9571133818194729 |              |            |                  | RFAGFixdRkhRNxkIaFEOT14jPzUgKEUSBgNPAihdeTFNIEFadWhRDRlyfWB9OFNBCQMTAnZcURsH
E5tdR-EsrPoS0NVY18UAANGUHZeUxoEbFtefC-umtGj8MrFS0AvOdf6-eRpe5ibrp1uZiM9XhJLLD5-NSxFQV5AEkw-GURJUytDCypYEU4EISE-MD5FUw |
amminer@pubsub:~$
```

Amelia Miner
11/2/23

13. PubSub via Python

- On the VM
  - Install virtualenv and pip.
- In both
  - Create and activate a venv and install the pubsub package.
- In cloud shell
  - Copy-and-paste the code in from the lab instead of cloning into the class repo?...
  - Run the program and keep it running for the next step.

✅

14. -

- Copy the subscriber code in and run it.

✅

15. Test Programs and Clean Up

- In the cloud shell, send several messages into the publisher. Show them in the gcloud terminal after they've sent with their contents and messageIds.

```
(env) amminer@cloudshell:~ (cloud-miner-amminer)$ python3 publisher.py
Enter a message to send: message #1
Published 9571781272419420 to topic projects/cloud-miner-amminer/topics/my_topic
Enter a message to send: message #2
Published 9571111909732939 to topic projects/cloud-miner-amminer/topics/my_topic
Enter a message to send: message #3
Published 9570972968811604 to topic projects/cloud-miner-amminer/topics/my_topic
Enter a message to send:
```

- In the VM, show the same messages after receipt.

```
(env) amminer@pubsub:~$ python subscriber.py
Received message 9571781272419420: 2023-11-04 18:35:02 (projects/cloud-miner-amminer/topics/my_topic) : message #1
Received message 9571111909732939: 2023-11-04 18:35:05 (projects/cloud-miner-amminer/topics/my_topic) : message #2
Received message 9570972968811604: 2023-11-04 18:35:08 (projects/cloud-miner-amminer/topics/my_topic) : message #3
```

- Kill both programs, delete the subscription, topic, VM, and service account.

✅