

* 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:

I. Lab 6.1a - EB Guestbook.....	3
1. Elastic Beanstalk.....	3
2. Elastic beanstalk sample application.....	3
3. Running the application.....	3
4. Handling Failures Seamlessly.....	4
5. Clean Up.....	4
6. Elastic Beanstalk Guestbook (CLI edition - CI/CD scripts do this).....	4
7. Deploying the Guestbook.....	4
8. Clean Up.....	5
II. Lab 6.1g - App Engine Guestbook.....	6
1. App Engine.....	6
2. app.yaml.....	6
3. Deploying the Guestbook.....	6
4. Handling Failures Seamlessly.....	7
5. Clean Up.....	7
III. Lab 6.2g - Cloud Run, Secret Manager (Web Proxy).....	8
1. Cloud Run.....	8
2. Application.....	8
3. app.py.....	8
4. templates/proxy.html.....	8
5. requirements.txt.....	8
6. Dockerfile.....	8
7. Build and test in Cloud Shell.....	8
8. Setup Secret Proxy.....	9
9. Cloud Build and Container Registry.....	9
10. Deploy to Cloud Run.....	10
11. Secret Manager.....	10
12. Deploy to Cloud Run with Secret Manager.....	11
13. Clean Up.....	11
IV. Lab 6.3a - ECS Guestbook.....	12
1. Prepare a Container Image.....	12
2. ECS Overview.....	12
3. ECS Task Definition.....	12
4. Create a Cluster and a Service.....	13
5. Examine the Service.....	13
6. Visit the Site.....	14
7. Clean Up.....	14
V. Lab 6.3g - Cloud Run Guestbook.....	15

1. Cloud Run.....	15
2. Prepare a container image.....	15
3. View Container Image.....	15
4. Deploy Container with Minimal Privileges.....	16
5. View the Guestbook.....	16
6. Clean Up.....	17
VI. Lab 6.4g - Cloud Functions, PubSub.....	18
1. Cloud Functions Image Blurring.....	18
2. Services Setup.....	18
3. Code.....	18
4. -.....	18
5. Set Up Service Account.....	20
6. Deploy the Function.....	20
7. Test Function.....	20
8. Clean Up.....	21
9. PubSub.....	22
10. Create VM for Subscriber.....	22
11. PubSub via CLI.....	22
12. -.....	22
13. PubSub via Python.....	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™.

Events (21) [Info](#)

Filter events by text, property or value

< 1 2 > ⚙

Time	Type	Details
November 2, 2023 14:40:28 (UTC-7)	INFO	Successfully launched environment: Eb-hello-env
November 2, 2023 14:40:25 (UTC-7)	INFO	Application available at Eb-hello-env.eba-nbj7mxhc.us-east-1.elasticbeanstalk.com.
November 2, 2023 14:40:24 (UTC-7)	INFO	Added instances [i-00c172805fa798b07, i-02fecb14b9c3af688] to your environment.
November 2, 2023 14:40:24 (UTC-7)	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 9 seconds ago and

```
meelz...
a:~amminer@ada
:~amminer@ada:
amminer@ada:~
>
```

4. Handling Failures Seamlessly

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

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	Eb-hello-env	i-0531098c3ea40bc0c	Pending		-	No alarms	us-east-1a
<input type="checkbox"/>	Eb-hello-env	i-02fecb14b9c3af688	Terminated		-	No alarms	us-east-1d
<input type="checkbox"/>	Eb-hello-env	i-00c172805fa798b07	Running		2/2 checks passed	No alarms	us-east-1e

```
meelz...  
amminer@ada:~  
>  
amminer@ada:~  
>
```

5. Clean Up



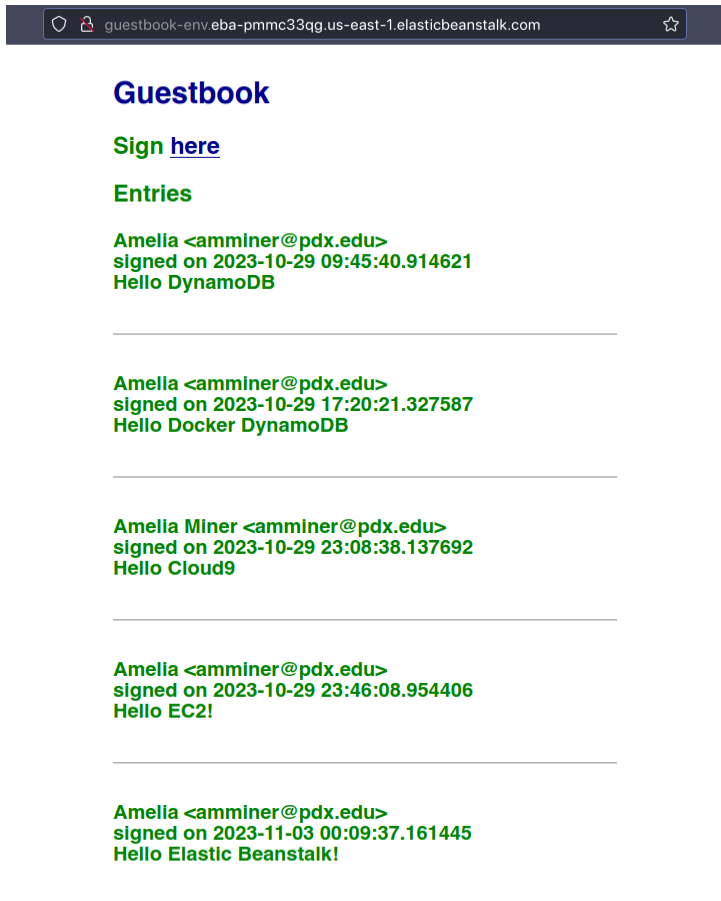
6. Elastic Beanstalk Guestbook (CLI edition - CI/CD scripts do this)



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!”.



The screenshot shows a web browser window with the URL `guestbook-env.eba-prmmc33qg.us-east-1.elasticbeanstalk.com`. The page has a title "Guestbook" and a link "Sign here". Below the link, there are five entries, each signed by "Amelia <amminer@pdx.edu>". The entries are:

- signed on 2023-10-29 09:45:40.914621
Hello DynamoDB
- signed on 2023-10-29 17:20:21.327587
Hello Docker DynamoDB
- signed on 2023-10-29 23:08:38.137692
Hello Cloud9
- signed on 2023-10-29 23:46:08.954406
Hello EC2!
- signed on 2023-11-03 00:09:37.161445
Hello Elastic Beanstalk!

- Show that the minimum number of instances are running.

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	guestbook-env	i-06b464d18bdae1b16	Running		2/2 checks passed	No alarms	us-east-1a
<input type="checkbox"/>	guestbook-env	i-012684197adfae7a5	Running		Initializing	No alarms	us-east-1a
<input type="checkbox"/>	guestbook-env	i-017bd3c4b5bd7c81f	Running		2/2 checks passed	No alarms	us-east-1c
<input type="checkbox"/>	guestbook-env	i-0475829a69341b9fe	Running		2/2 checks passed	No alarms	us-east-1b

8. Clean Up



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!”.

https://cloud-miner-amminer.uw.r.appspot.com

Guestbook

[Sign here](#)

Entries

Amelia <amminer@pdx.edu>
signed on 2023-10-30 01:05:00.939198+00:00
Hello Cloud Shell!

Amelia <amminer@pdx.edu>
signed on 2023-10-29 17:39:21.049839+00:00
Hello Datastore

Amelia <amminer@pdx.edu>
signed on 2023-11-03 00:43:25.874230+00:00
Hello App Engine!

Amelia <amminer@pdx.edu>
signed on 2023-10-30 01:35:11.596005+00:00
Hello Compute Engine!

Amelia <amminer@pdx.edu>
signed on 2023-10-30 00:53:29.411812+00:00
Hello Docker Datastore!

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.

cloud-Miner-amminer

app engine

×

Search

20

Instances

REFRESH

DELETE

LEARN

Instances

<input checked="" type="checkbox"/>	ID ↑	QPS	Latency	Requests	Errors	Memory	Start Time
<input checked="" type="checkbox"/>	0037d6d5d35d70a801705118627be...	0	0 ms	5	0	91.5 MB	Nov 2, 2023 5:41:57 PM
<input checked="" type="checkbox"/>	0037d6d5d3d637a6d2e9bff98f122c...	0	0 ms	5	0	92.9 MB	Nov 2, 2023 5:41:57 PM

- Delete one and watch it be replaced.



5. Clean Up



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



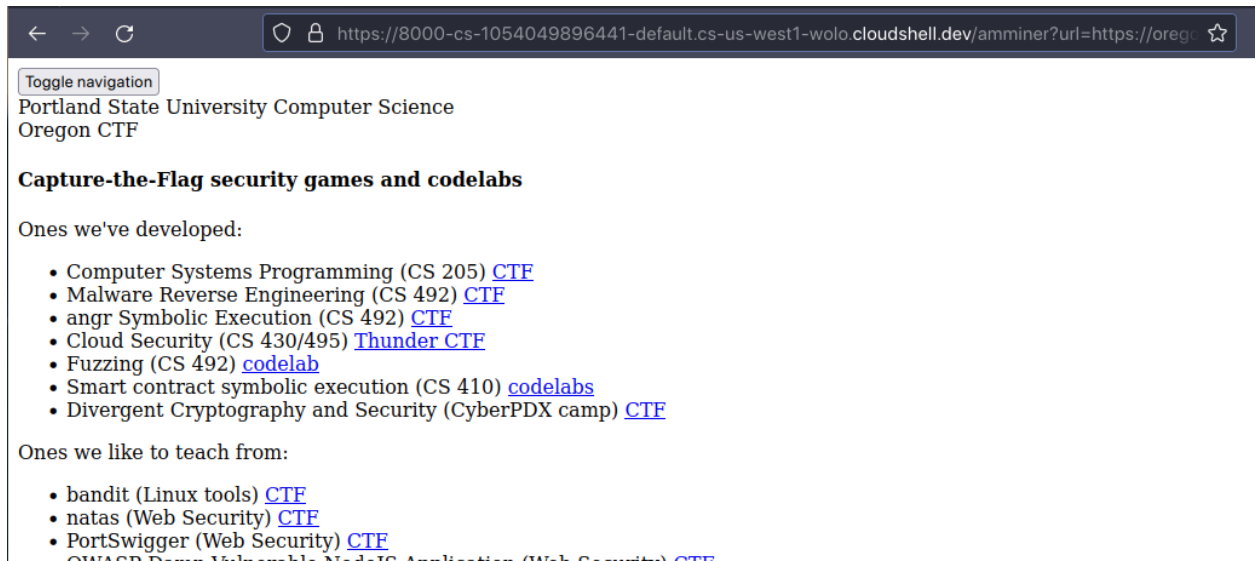
7. Build and test in Cloud Shell

- 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.



8. Setup Secret Proxy

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

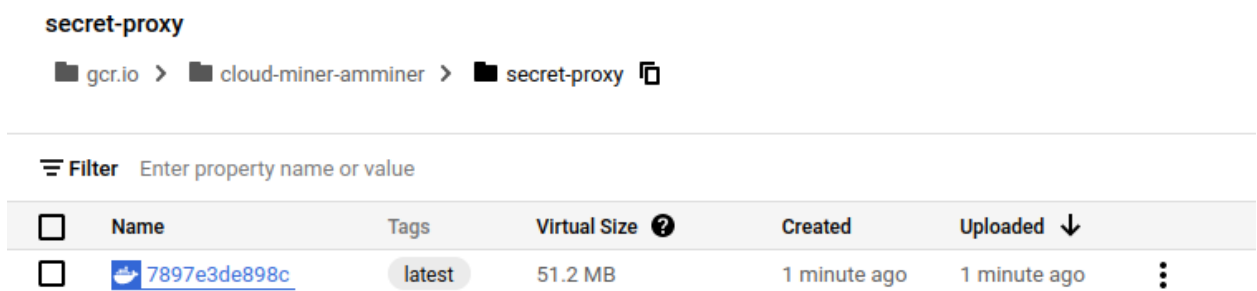


- What is the security advantage of passing in the secret proxy route as an environment variable?

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.

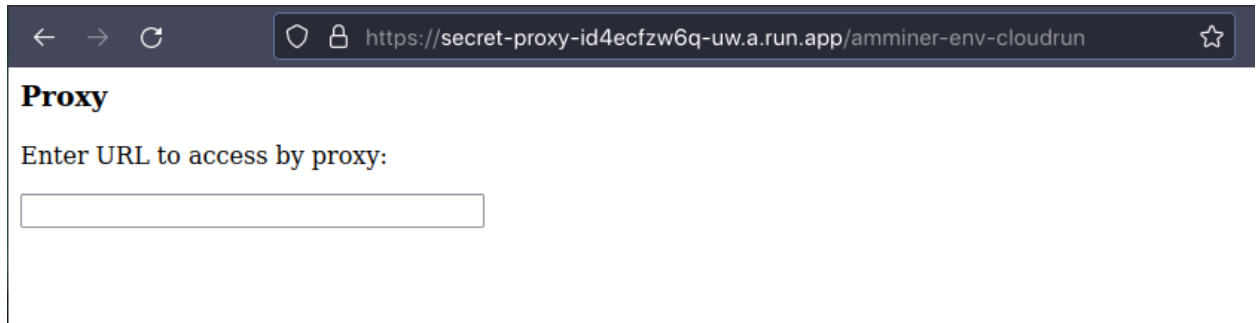


10. Deploy to Cloud Run

- ```
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



Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

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
```

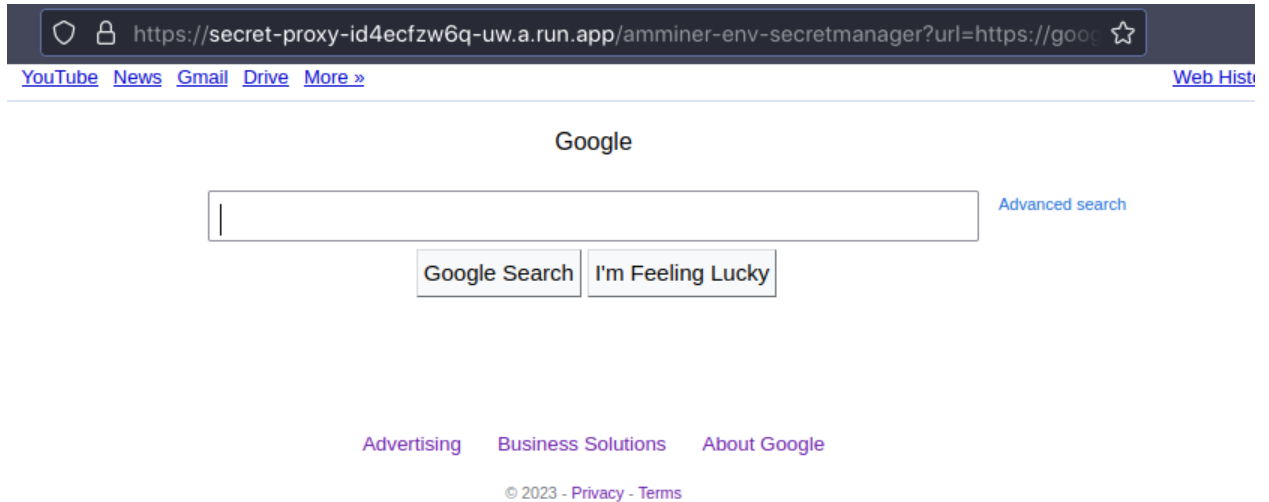


12. Deploy to Cloud Run with Secret Manager

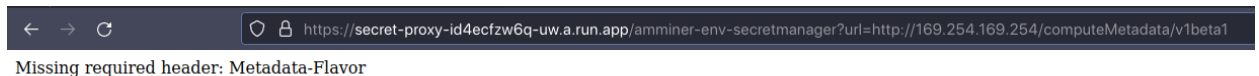
- `gcloud run services update secret-proxy \`
`--update-secrets=SECRET_PROXY_ROUTE=proxy-secret:latest \`
`--region=us-west1`



- 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.



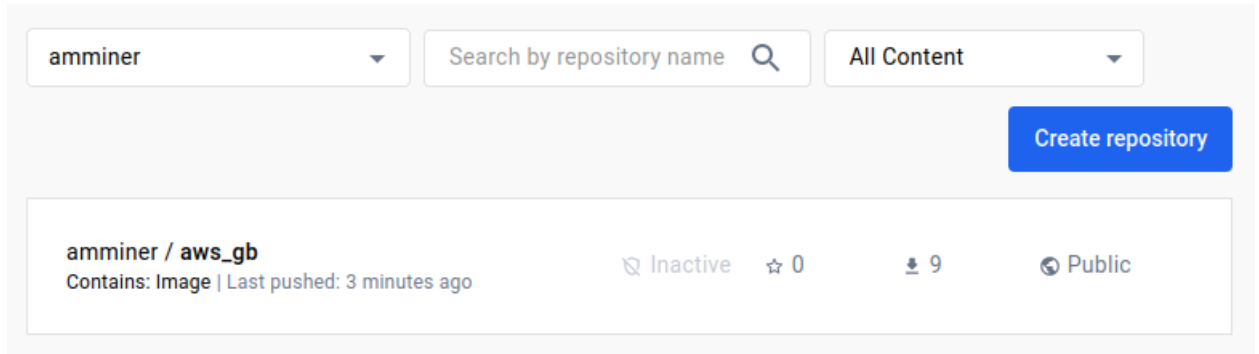
13. Clean Up



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



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.

[Alt+S] [Icons] N. Virginia ▼ voclabs/user2812445=amminer@pdx.edu @ 2176-6395-218

EC2 > Load balancers > guestbook-lb

guestbook-lb [Refresh] [Actions ▼]

▼ Details

Load balancer type Application	Status ✔ Active	VPC vpc-08bc6fad024f50e4e	IP address type IPv4
Scheme Internet-facing	Hosted zone Z35SXDOTRQ7X7K	Availability Zones subnet-0fb2d197d88e6196d us-east-1d (use1-az6) subnet-0e0f09a9d0b49ab30 us-east-1a (use1-az1) subnet-0d4e3d4612e4674e7 us-east-1b (use1-az2) subnet-01f6358995051d906 us-east-1c (use1-az4) subnet-060250f3e9fdd2b89 us-east-1e (use1-az3) subnet-0442ba6fcb6773511 us-east-1f (use1-az5)	Date created November 4, 2023, 23:33 (UTC-07:00)
Load balancer ARN arn:aws:elasticloadbalancing:us-east-1:217663952189:loadbalancer/app/guestbook-lb/15b36820a3d491b4		DNS name Info guestbook-lb-1596452272.us-east-1.elb.amazonaws.com (A Record)	

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.



Guestbook

[Sign here](#)

Entries

Amelia <amminer@pdx.edu>
signed on 2023-11-05 06:57:20.315270
Hello ECS!

7. Clean Up

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



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.

The screenshot shows the Google Cloud Platform console. At the top, the breadcrumb navigation is `gcr.io > cloud-miner-amminer > gcp_gb`. Below this is a table of container images. The table has columns: Name, Tags, Virtual Size, Created, and Uploaded. One image is listed with Name `1cef5267aadf`, Tag `latest`, and Virtual Size `1.1 GB`. Below the table is a terminal window titled `cloud-miner-amminer`. The terminal output shows the results of a build and push command, including a list of layers, their status, and the final digest and size of the image.

Name	Tags	Virtual Size	Created	Uploaded
1cef5267aadf	latest	1.1 GB	12 minutes ago	10 minutes ago

```
be00d9a6b44e: Layer already exists
1f77dbec8f9f: Layer already exists
f438a9a76f12: Layer already exists
673f527c30ab: Layer already exists
53ae7e5bcde8: Layer already exists
fb8f7070a3a8: Pushed
959ef0a87bd8: Pushed
94cca38a5611: Pushed
latest: digest: sha256:1cef5267aadf76191fb3f30f9f27fb94d4ff08da06076edc4af94c88fcac8b91 size: 2217
DONE
-----
ID: a0c24355-22c8-48db-9443-09fafc4ff10c
CREATE_TIME: 2023-11-04T05:56:20+00:00
DURATION: 2M46S
SOURCE: gs://cloud-miner-amminer_cloudbuild/source/1699077368.533353-37134dd7977b409eb7f238c41d0996e8.tgz
IMAGES: gcr.io/cloud-miner-amminer/gcp_gb (+1 more)
STATUS: SUCCESS
amminer@cloudshell:~/cs430-src/05_gcp_datastore (cloud-miner-amminer)$
```

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.



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.



Guestbook

[Sign here](#)

Entries


Amelia <amminer@pdx.edu>
signed on 2023-10-30 01:05:00.939198+00:00
Hello Cloud Shell!

Amelia <amminer@pdx.edu>
signed on 2023-10-29 17:39:21.049839+00:00
Hello Datastore

Amelia <amminer@pdx.edu>
signed on 2023-11-03 00:43:25.874230+00:00
Hello App Engine!

- 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

 **gcp-gb-00001-7gg**
Deployed by amminer@pdx.edu using gcloud


[CONTAINERS](#) [VOLUMES](#) [NETWORKING](#) [SECURITY](#) [YAML](#)

General

CPU allocation	CPU is only allocated during request processing
Startup CPU boost	Disabled
Concurrency	80
Request timeout	300 seconds
Execution environment	First generation (Default)

Autoscaling

Max instances	100
---------------	-----


Image URL	gcr.io/cloud-miner-amminer/gcp_gb@sha256:1cef52... 
Port	8080
Build	(no build information available) ?
Source	(no source information available) ?

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

100

Autoscaling

Max instances	100
---------------	-----

Image URL	gcr.io/cloud-miner-amminer/gcp_gb@sha256:1cef52... 
Port	8080

6. Clean Up



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)$
```

- 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)$
```

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

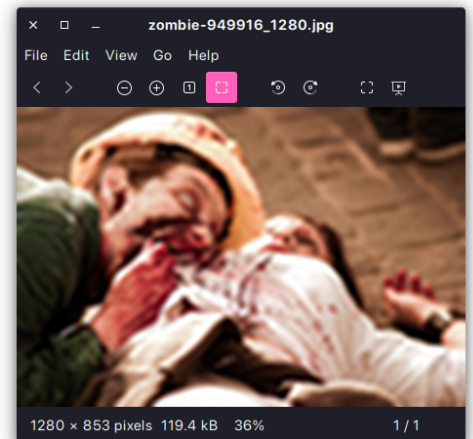
- Upload the example image to the input bucket and show the blurred image in the output bucket.

Buckets > amminer-gcf-output > zombie-949916_1280.jpg

[DOWNLOAD](#) [EDIT METADATA](#) [EDIT ACCESS](#) [DELETE](#)

Overview

Type	application/octet-stream
Size	116.6 KB
Created	Nov 4, 2023, 8:39:04 AM
Last modified	Nov 4, 2023, 8:39:04 AM
Storage class	Standard
Custom time	—
Public URL	Not applicable
Authenticated URL	https://storage.cloud.google.com/amminer-gcf-output/zombie-949916_1280.jpg
gsutil URI	gs://amminer-gcf-output/zombie-949916_1280.jpg
Permissions	
Public access	Not public
Protection	
Version history	—
Retention policy	None
Hold status	None
Encryption type	Google-managed



- 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



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 |
|-----+-----+-----+-----+-----+
| Message #2 | 9571133818194729 | | | |
|-----+-----+-----+-----+-----+
| EstdR-EsrPoS0NVY18UAANGUHZex0EbFtefC-umtGj8MrFS0Av0df6-eRpe5ibrp1uZ1M9XhJLLD5-NSxFQV5AEkw-GURJUytdCypYE04EISE-MD5FUw |
|-----+-----+-----+-----+-----+
amminer@pubsub:~$
```

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.

