Secure Systems & Network Engineering

Large Installation Administration

Lab 4 Google Cloud Platform

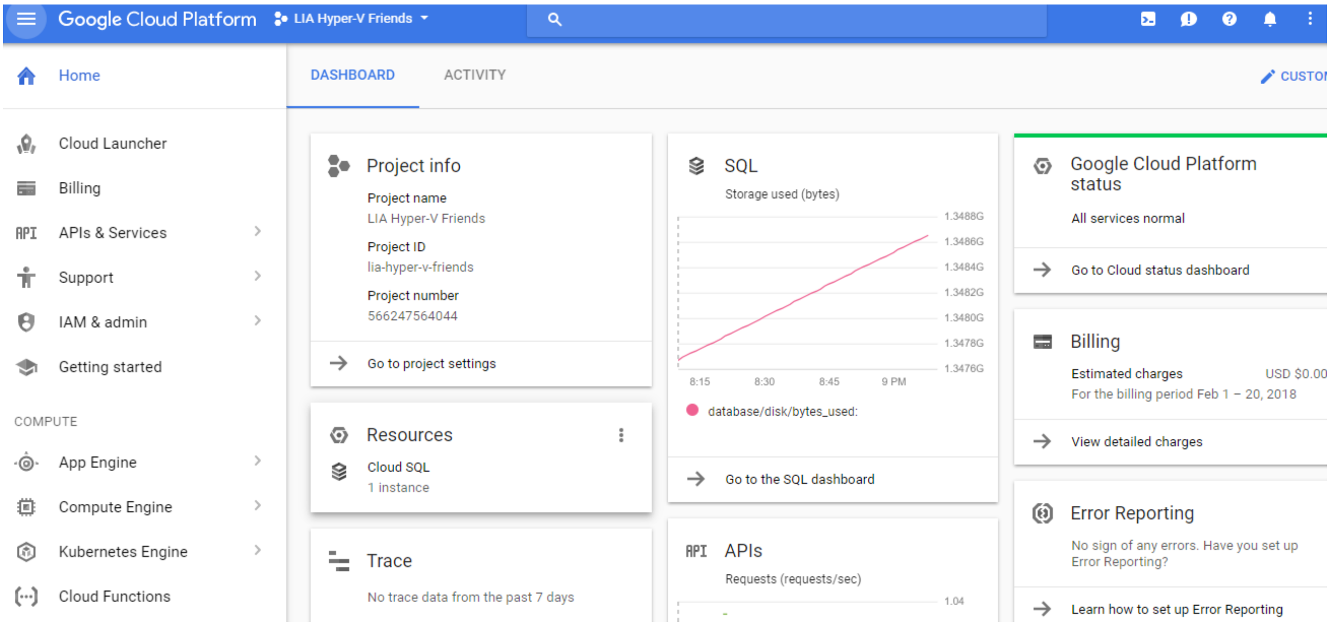
Grebennikov Sergey, Saad Saif,
Burikova Svetlana, Sharafitdinova Iuliia

February 20, 2018

1 Preparation

1.1 Create GCP account

The project `LIA Hyper-V friends` was created.



Project info

Project name
LIA Hyper-V Friends

Project ID
lia-hyper-v-friends

Project number
566247564044

Go to project settings

Google Cloud Platform status

All services normal

Go to Cloud status dashboard

Resources

Cloud SQL
1 instance

Billing

Estimated charges USD $0.00
For the billing period Feb 1 – 20, 2018

View detailed charges

Trace

No trace data from the past 7 days

Error Reporting

No sign of any errors. Have you set up Error Reporting?

Learn how to set up Error Reporting

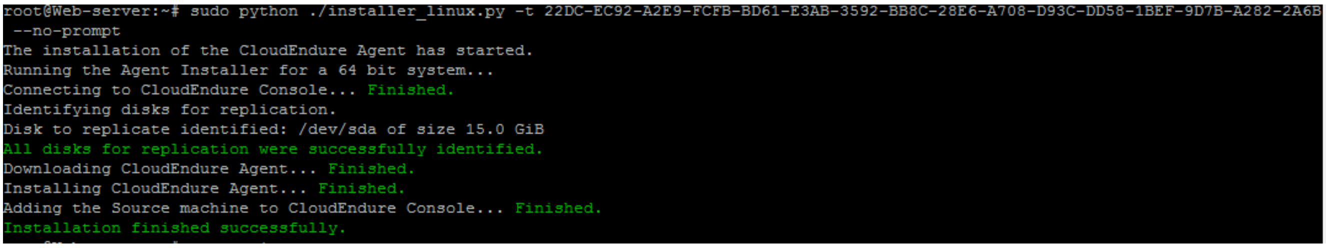
Figure 1: GCP account

2 Move you application to the cloud

To move an application to GCP it is required to perfom the following steps:

- Launch the VM Migration Service from the Google Cloud Platform Console.
- Install the replication agents and copy the virtual machines into Compute Engine.
- Launch the newly created virtual machines.

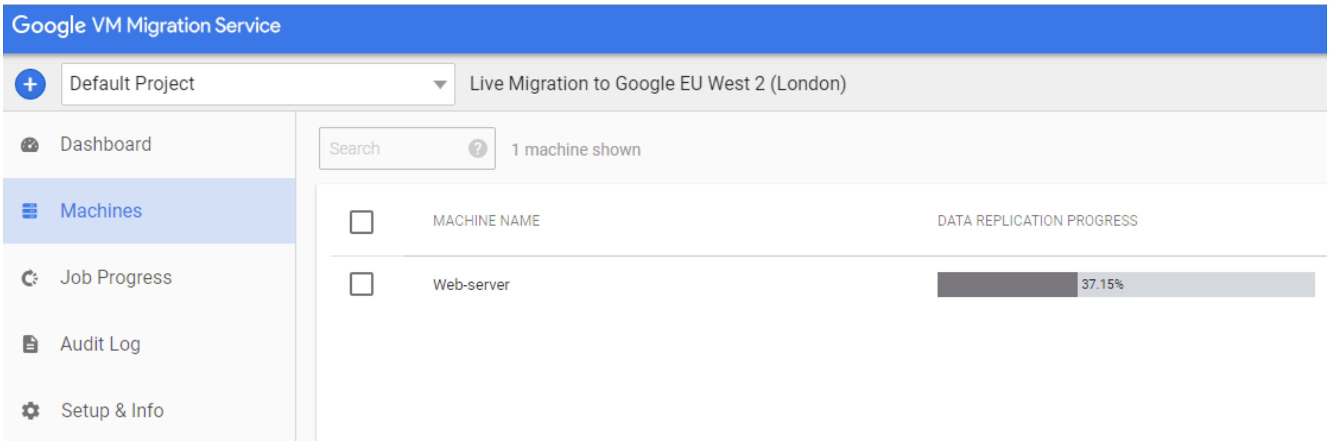
The result of migration is shown in Figure 2



```
root@Web-server:~# sudo python ./installer_linux.py -t 22DC-EC92-A2E9-FCFB-BD61-E3AB-3592-BB8C-28E6-A708-D93C-DD58-1BEF-9D7B-A282-2A6B
 --no-prompt
The installation of the CloudEndure Agent has started.
Running the Agent Installer for a 64 bit system...
Connecting to CloudEndure Console... Finished.
Identifying disks for replication.
Disk to replicate identified: /dev/sda of size 15.0 GiB
All disks for replication were successfully identified.
Downloading CloudEndure Agent... Finished.
Installing CloudEndure Agent... Finished.
Adding the Source machine to CloudEndure Console... Finished.
Installation finished successfully.
```

Figure 2: Migration Virtual Machine

And you can see the process of migration to Google EU West 2 (London) on Figure 3.



Live Migration to Google EU West 2 (London)

MACHINE NAME	DATA REPLICATION PROGRESS
□ Web-server	37.15%

Figure 3: Migration Virtual Machine

3 Datastore

The Cloud SQL was chosen for DB instance. The process of dividing DBMS from WordPress contains the following steps:

1. Creating Buckets for storing files.
2. Exporting DB from WordPress and uploading it to Buckets.
3. Creating DB in MySQL Database.
4. Upload DB to MySQL Database.
5. Editing WordPress config file by changing IP address for DB.

4 Improve application performance

To improve the performance of the web server, we applied load balancing by installing another instance. For us, it's enough to create another additional server. After that, we created a group of instances and load balancing.

5 Using GCP API

in order to be able to create a snapshot, we should have the following permissions.

- compute.disks.createSnapshot
- compute.disks.createSnapshot

And we need to know the following:

- Project name.
- Zone name.
- Disk name to be snapshot.

Also we need the compute engine API to be enabled, this can be done from here: https://console.developers.google.com/apis/api/compute.

And finally we need the google api library for python , which can be installed by this command: pip install –upgrade google-api-python-client.

After all the previous steps, now we can start the following script in order to create a snapshot of all the disks inside the project, first the code will loop thought all the instance name and then get the all the attached disks name and then do the snapshot for all these disks.

```
from pprint import pprint
from googleapiclient import discovery
from oauth2client.client import GoogleCredentials

credentials = GoogleCredentials.get_application_default()

service = discovery.build('compute', 'v1', credentials=credentials)

project = 'lia-hyper-v-friends'

zone = 'europe-west3-b'
request = service.instances().list(project=project, zone=zone)
while request is not None:
    response = request.execute()
    for instance in response['items']:
        for DiskName in instances[disks]:
          disk = instances.DiskName.initializeParams.diskName
          snapshot_body = {
          }

          request = service.disks().createSnapshot(project=project,
          zone=zone, disk=disk, body=snapshot_body)
          response = request.execute()
          pprint(response)

    request = service.instances().list_next(previous_request=request,
    previous_response=response)
```

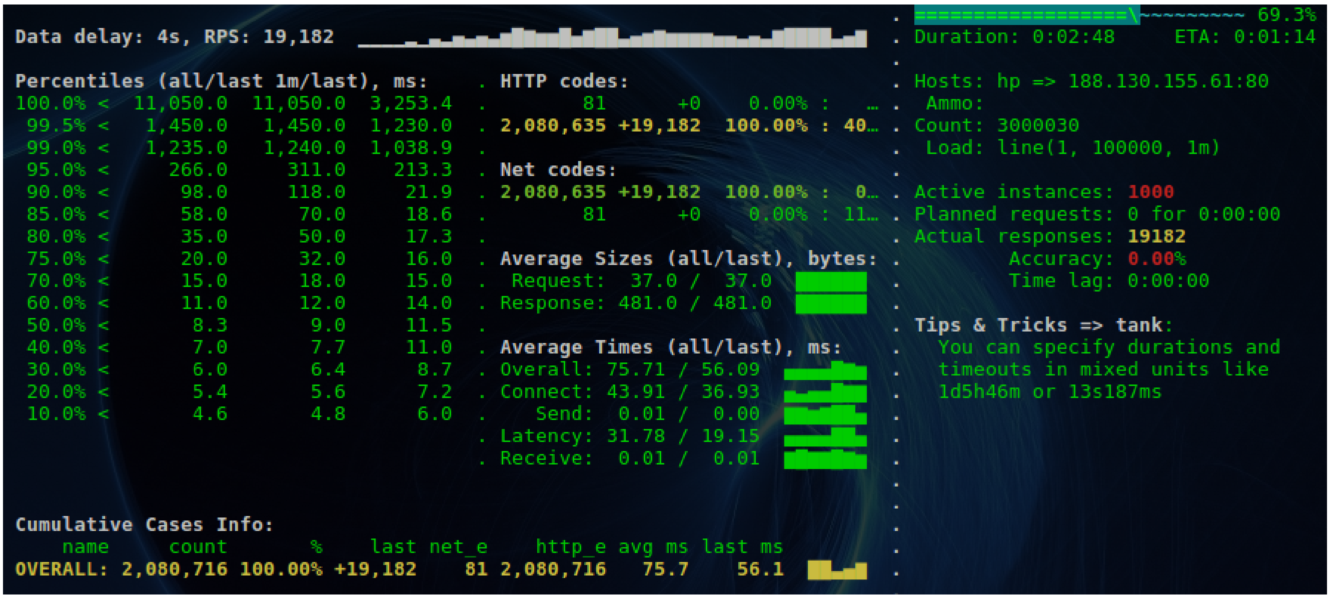
6 Restrict access to your application

Restriction access can be configured only on instances. We used Apache configuration to do it. There is example of configuration below: There is an attempt of getting access from German IP address.

```
<Directory "/var/www/html">
Options Indexes MultiViews FollowSymLinks
AllowOverride None
Order deny,allow
Deny from all
Allow from 188.130.155.0/24 2.16.154.0/24 2.16.159.0/24 2.16.160.0/23 2.16.168.0/
</Directory>
```

7 Try to make DOS attack to your APP

DOS attack was performed with `yandex.tank` utility in the linear mode. The linear load from 1 to 100 000 rps, duration - 10 minutes. (Figure 4)



```
                                                                          . ==================\~~~~~~~~~~ 69.3%
Data delay: 4s, RPS: 19,182                                               . Duration: 0:02:48     ETA: 0:01:14
                                                                          .
Percentiles (all/last 1m/last), ms:    . HTTP codes:                      . Hosts: hp => 188.130.155.61:80
100.0% <  11,050.0  11,050.0  3,253.4  .        81       +0     0.00% :   … .  Ammo:
 99.5% <   1,450.0   1,450.0  1,230.0  . 2,080,635 +19,182  100.00% : 40… . Count: 3000030
 99.0% <   1,235.0   1,240.0  1,038.9  .                                  .  Load: line(1, 100000, 1m)
 95.0% <     266.0     311.0    213.3  . Net codes:                       .
 90.0% <      98.0     118.0     21.9  . 2,080,635 +19,182  100.00% :  0… . Active instances: 1000
 85.0% <      58.0      70.0     18.6  .        81       +0     0.00% : 11… . Planned requests: 0 for 0:00:00
 80.0% <      35.0      50.0     17.3  .                                  . Actual responses: 19182
 75.0% <      20.0      32.0     16.0  . Average Sizes (all/last), bytes: .         Accuracy: 0.00%
 70.0% <      15.0      18.0     15.0  .  Request:  37.0 /  37.0          .         Time lag: 0:00:00
 60.0% <      11.0      12.0     14.0  . Response: 481.0 / 481.0          .
 50.0% <       8.3       9.0     11.5  .                                  . Tips & Tricks => tank:
 40.0% <       7.0       7.7     11.0  . Average Times (all/last), ms:    .   You can specify durations and
 30.0% <       6.0       6.4      8.7  . Overall: 75.71 / 56.09           .   timeouts in mixed units like
 20.0% <       5.4       5.6      7.2  . Connect: 43.91 / 36.93           .   1d5h46m or 13s187ms
 10.0% <       4.6       4.8      6.0  .    Send:  0.01 /  0.00           .
                                       . Latency: 31.78 / 19.15           .
                                       . Receive:  0.01 /  0.01           .
                                                                          .
                                                                          .
Cumulative Cases Info:                                                    .
    name       count       %     last net_e     http_e avg ms last ms     .
OVERALL: 2,080,716 100.00% +19,182     81 2,080,716   75.7     56.1       .
```

Figure 4: DOS attack

8 Questions

Question 1

Describe the differences between Google cloud storage options. Which one is most suitable for your application, why?

Answer

There are following Google cloud storage options:

1. Persistent Disk. Fully-managed, price-performant block storage that is suitable for virtual machines and containers.

2. Google Cloud Storage. A scalable, fully-managed, highly reliable, and cost-efficient object / blob store.

3. Google Cloud Bigtable. A scalable, fully-managed NoSQL wide-column database that is suitable for both real-time access and analytics workloads.

4. Google Cloud Datastore. A scalable, fully-managed NoSQL document database for your web and mobile applications. A fully-managed MySQL and PostgreSQL database service that is built on the strength and reliability of Googles infrastructure.

5. Google Cloud SQL. A fully-managed MySQL and PostgreSQL database service that is built on the strength and reliability of Googles infrastructure.

6. Google Cloud Spanner. Mission-critical, relational database service with transactional consistency, global scale and high availability.

7. Google BigQuery. A scalable, fully-managed Enterprise Data Warehouse (EDW) with SQL and fast response times.

8. Google Drive. A collaborative space for storing, sharing, and editing files, including Google Docs.

Since our application is a web server, for our needs most suited the Google Cloud SQL. Because it is designed for the web frameworks, wtructured data and OLTP workloads. In addition, its workload is designed for websites, blogs, and content management systems (CMS).

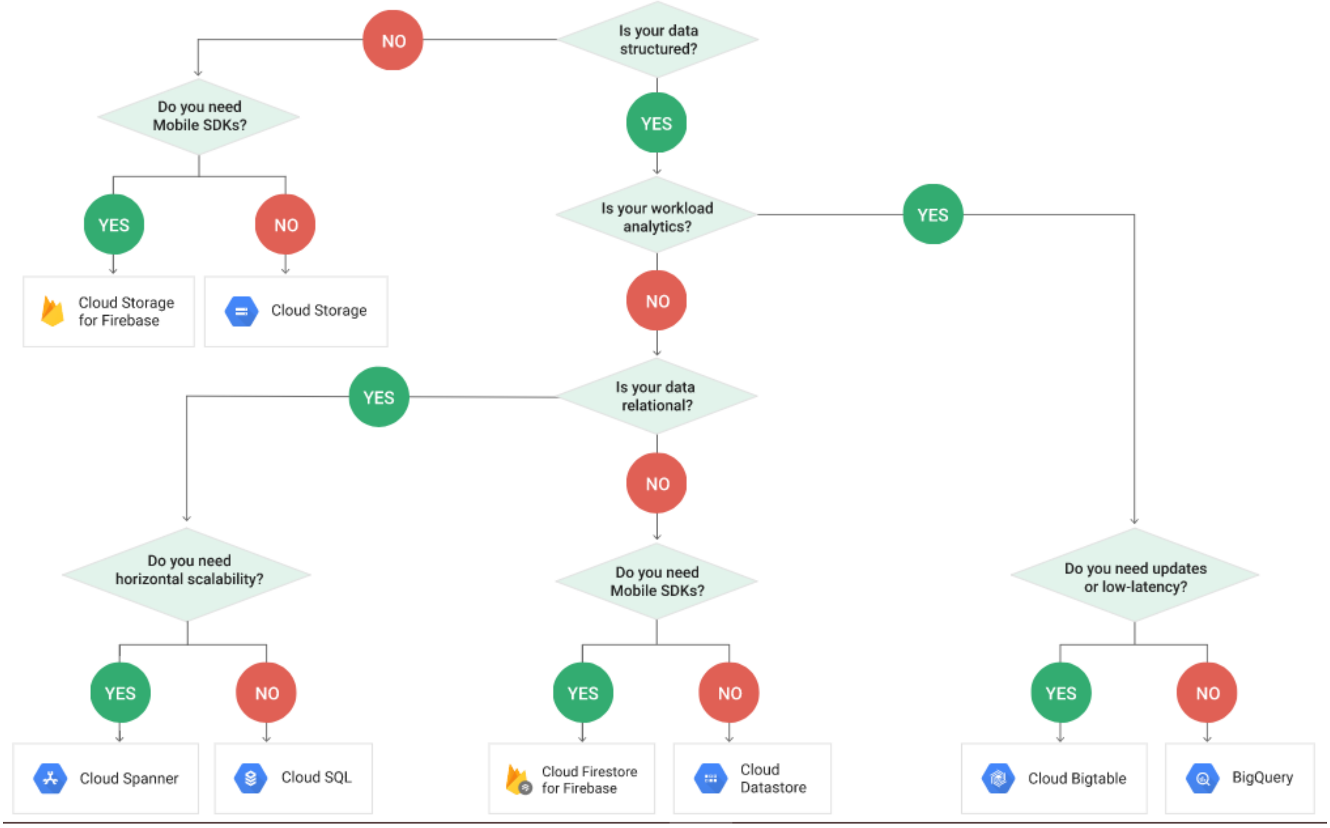


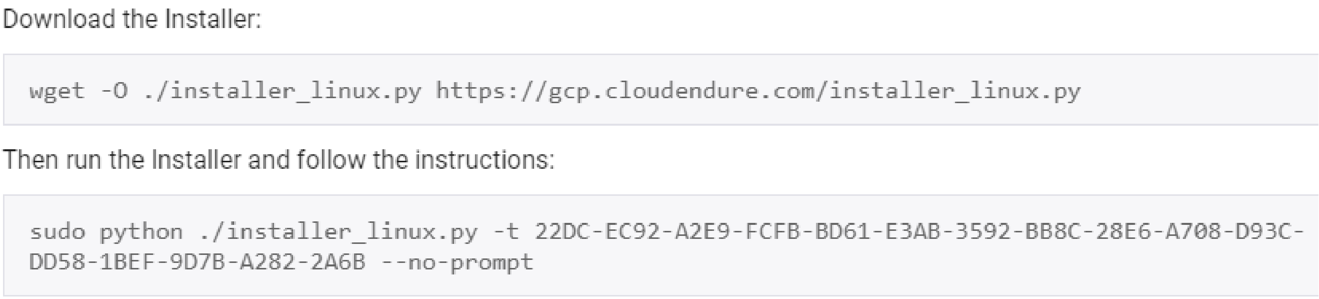
Figure 5: Choosing a storage option

Question 2

How did you move APP to the cloud, explain why?

Answer

We followed by the steps described in instruction (Figure 6).



Download the Installer:

```
wget -O ./installer_linux.py https://gcp.cloudendure.com/installer_linux.py
```

Then run the Installer and follow the instructions:

```
sudo python ./installer_linux.py -t 22DC-EC92-A2E9-FCFB-BD61-E3AB-3592-BB8C-28E6-A708-D93C-
DD58-1BEF-9D7B-A282-2A6B --no-prompt
```

Figure 6: Instruction of migration to cloud

There are many problems that can solve the migration to the cloud:

- your application has increased traffic, and it becomes increasingly difficult to scale resources ”on the fly” to meet growing demand;
- you need to reduce operating costs, while improving the efficiency of IT processes;
- you want to focus more on development, while reducing the overhead of infrastructure;
- cloud computing environments allow you to remotely access applications and work through the Internet;

- cloud-based disaster recovery systems can be implemented much faster and give you much better control over your resources;
- some cloud computing models handle many administrative tasks, such as database backup, software updates, and periodic maintenance.

Question 3

Can you use TCP load balancer in your setup? Describe pros and cons of using TCP load balancer.

Answer

Yes we can, but according to google documentation, using TCP load balancer for HTTP/HTTPS traffic is not recommended and it is better to use the HTTP/SSL Proxy load balancing.

Pros:

- It is a global load balancing service, so it can redirect user to the closest region to that user.
- Does not cause performance degradation on the server
- Recovers and redistributes the workload within seconds.

Cons:

- Single point of failure.
- Since it is a proxy, we can not identify incoming connection by ip address (we can not ip-base blocking) since all the connection we come from the load balancer ip address.
- Does not make smarter load-balancing decisions (only layer 3 and layer 4 routing), and to apply optimizations and changes to the content (such as compression and encryption)

Question 4

How can you protect your APP from the DOS attacks?

Answer

1. Abandon Windows Server. The reason for failure lies in the Windows network stack: when there are a lot of connections, the server certainly begins to respond poorly.

2. The second important condition is the rejection of Apache. If you have Apache, then at least put before it a caching proxy - nginx or lighttpd.

3. Attackers can send several tens of thousands of requests to the search engine. We can temporarily disable search. Let customers not be able to search for the information with built-in tools, but the entire main site will remain operational until you find the root of all problems. Nginx supports non-standard code 444, which allows you to simply close the connection and do nothing in return.

4. Use load balancing and server farms to displace bandwidth.

5. Use intelligent routers that control the activity of incoming requests.

6. You can severely restrict individual countries from which you receive a multitude of unwanted queries.

7. Track the number of requests per second.

8. Limit resources (buffer sizes).

9. Set timeouts.