**What challenges do you think will be faced if you deploy it on premise data center?**

We think that the following challenges will be faced if you deploy applications on premise data centers:

1. Infrastructural Cost will go up: Creating and maintaining a data center is expensive. It involves a huge infrastructural cost, security costs, disaster management recovery costs, upgradation costs, etc. This will be a huge overhead for deploying applications.
2. Scalability is not so easy: Performance is another issue that a data center must address. Scalability is not easy for on premise data centers. Again, more cost is involved here.
3. Expansion of business becomes more expensive: expanding business to new areas will be costlier as new on-premises data centers will be needed to setup there.
4. Training costs: As data centers require trained employees to look after the data. This venture will incur training costs for employees responsible for the data centers. Moreover, this cost will linearly increase with scaling.

**List the advantages or improvements you see if implemented on cloud?**

With cloud, the following improvements come into picture:

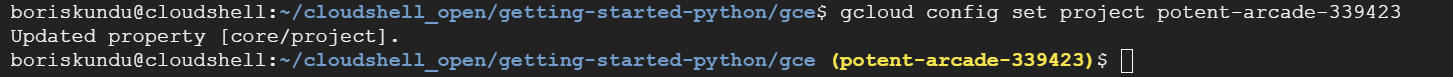
1. Infrastructural cost goes down: it is statistically observed that the infrastructure costs go down when cloud services are used.
2. Maintainability: the cloud services are more maintainable. A lot of advantages are provided free of cost to users including physical security, upgradation of software and hardware, disaster recovery, etc. Other services are provided to the user/company which are only a click away.
3. Scalability: scaling applications is way easier and cheaper with cloud services. Improving application’s performance is as simple as scaling the infrastructure from cloud services.
4. Reduced Downtime: all cloud services provide less downtime than any on-premises data centers. Cloud services have more stable and secure ways to handle downtimes. Moreover, cloud services provide cluster deployments where replicas of applications can run on a cluster. This helps in improving the applications availability and reducing downtime.
5. Expansion: Business expansion is easier with cloud services, as it only requires upscaling the clusters to support more requests.

**A simple ‘Hello World‘ application on using Compute engine service on GCP**

**Note**: VM instance is currently shutdown to avoid incurring charges. Access URL changes as external IP changes every time instance is brought up. Please email me (kundubs@mail.uc.edu) and I will share updated URL. Once you hit the URL, please wait for at least 2 minutes for the page to come up. I am using US central server and there could be latency.

**Steps:**

1. Selected an existing project
2. Ensured billing is enabled for our project
3. Enabled the Compute Engine API
4. Started Google Cloud shell
5. Configure the gcloud CLI to use our new Google Cloud project

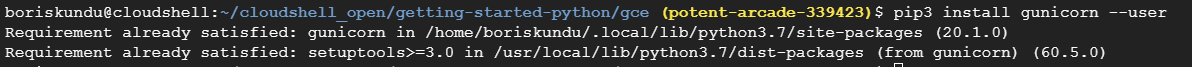


1. Installing dependencies

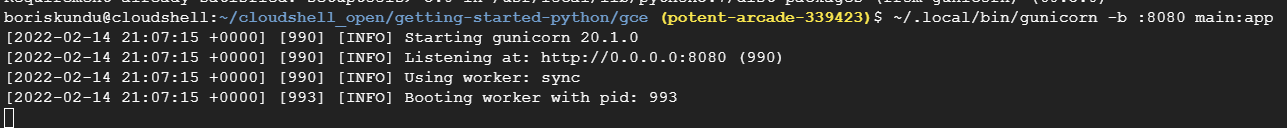
Text

Description automatically generated

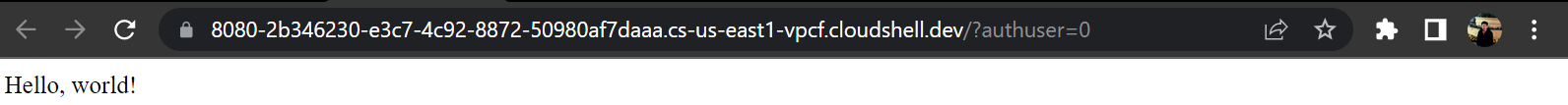
1. Install the Gunicorn HTTP server:



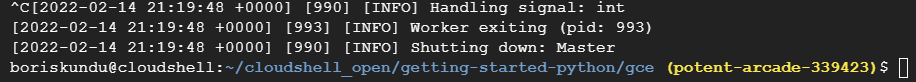
1. Run the Gunicorn HTTP server:



1. Click on Web preview => **Preview on port 8080**



1. To stop the local web server, press Control+C.



1. Startup script to initialize instance

# Install Stackdriver logging agent  
curl -sSO https://dl.google.com/cloudagents/install-logging-agent.sh  
sudo bash install-logging-agent.sh  
  
# Install or update needed software  
apt-get update  
apt-get install -yq git supervisor python python-pip python3-distutils  
pip install --upgrade pip virtualenv  
  
# Account to own server process  
useradd -m -d /home/pythonapp pythonapp  
  
# Fetch source code  
export HOME=/root  
git clone https://github.com/GoogleCloudPlatform/getting-started-python.git /opt/app  
  
# Python environment setup  
virtualenv -p python3 /opt/app/gce/env  
/bin/bash -c "source /opt/app/gce/env/bin/activate"  
/opt/app/gce/env/bin/pip install -r /opt/app/gce/requirements.txt  
  
# Set ownership to newly created account  
chown -R pythonapp:pythonapp /opt/app  
  
# Put supervisor configuration in proper place  
cp /opt/app/gce/python-app.conf /etc/supervisor/conf.d/python-app.conf  
  
# Start service via supervisorctl  
supervisorctl reread  
supervisorctl update

1. Create and configure compute instance.

MY\_INSTANCE\_NAME="my-app-instance"  
ZONE=us-central1-a  
  
gcloud compute instances create $MY\_INSTANCE\_NAME \  
    --image-family=debian-10 \  
    --image-project=debian-cloud \  
    --machine-type=g1-small \  
    --scopes userinfo-email,cloud-platform \  
    --metadata-from-file startup-script=startup-script.sh \  
    --zone $ZONE \  
    --tags http-server

Text

Description automatically generated

1. Check the progress of the instance creation:

gcloud compute instances get-serial-port-output my-app-instance --zone us-central1-a

1. Create a firewall rule to allow traffic to your instance

Text

Description automatically generated

1. Get the external IP address of your instance:

gcloud compute instances list

Graphical user interface, text, application

Description automatically generated

1. Check our app running, enter this URL with the external instance IP in the browser:

http://34.69.241.113:8080



1. Monitor instance

Graphical user interface, text

Description automatically generated with medium confidence

1. Open SSH to connect to VM

Text

Description automatically generated

1. Delete individual resources to avoid cost or delete the project.

gcloud compute instances delete my-app-instance --zone= us-central1-a --delete-disks=all  
gcloud compute firewall-rules delete default-allow-http-8080

**Challenges**:

I personally did not face any challenges while using Compute Engine.

**Code:**

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/', methods=['GET'])

def say\_hello():

return "Hello, world!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='127.0.0.1', port=8080, debug=True)

**References**

* <https://cloud.google.com/python/docs/getting-started/getting-started-on-compute-engine>
* <https://github.com/GoogleCloudPlatform/getting-started-python/tree/main/gce>

**A simple “Hello World” application using App engine service on GCP**

**Note**: App Engine is currently disabled. Please email our group leader (kundubs@mail.uc.edu) and I will bring it up. Once you hit the URL, please wait for at least 2-3 minutes for the page to come up.

**Steps:**

1. Created a project.

Graphical user interface, application

Description automatically generated

1. Enabled the App Engine API.

Graphical user interface, text, application

Description automatically generated

1. Activate an App Engine (Create Application)

Graphical user interface, text, application

Description automatically generated

1. Clone the repository in google cloud shell.

Graphical user interface, text, application

Description automatically generated

1. Navigate to the application.

Used cd command to navigate to the Hello World Application

1. Deploy the application

Graphical user interface, text

Description automatically generated

**Output (https://helloworld-341405.uc.r.appspot.com/):**

Graphical user interface, application, Word

Description automatically generated

**App Engine Deployment**

For deploying the “Hello World” application in App Engine I have chosen Standard Environment version because the source code of the application is written on python 3.7 and Standard Environment is optimal for applications written in this language. In addition to that this application is intended to run for free or at a very low cost. This application can scale to 0 when there is no traffic. Furthermore, App Engine in Standard Environment supports sudden and extreme spikes of traffic which require immediate scaling. I did not choose the flexible environment because for this simple application complex features such as Customizable infrastructure, Performance options, Native feature support, etc. are not required.

**References**:

* <https://cloud.google.com/appengine/docs/standard/python3/quickstart>

**Comparison of Compute Engine vs App Engine**

|  |  |
| --- | --- |
| **App Engine** | **Compute Engine** |
| It is PaaS (Platform as a Service) | It is IaaS (Infrastructure as a Service) |
| It has a lot of things automated for us such as deployment, scaling, resources etc. | It offers a bare virtual machine with an OS that we need to install and maintain ourselves. |
| It is limited in the sense what we can do with it. | It has wider use as we can install and run almost anything we like. |
| It offers users greater flexibility and ease of operation. | It offers administrators more direct control over operating system. |

**Conclusion on Deployment App Engine vs Compute Engine**

It really depends on what kind of application we are trying to build and how comfortable are we in working on a raw environment from scratch versus a standard prepared environment prepared. The same Hello World application was deployed on both the Compute and App Engines. App Engine is more user friendly, has less deployment steps and we can focus more on coding. However, Compute Engine is more admin friendly, gives us more control, and is more versatile.

**Exploring GCP Marketplace**

# We identified Firebase Hosting as a ready to use component that can support our Hello World application which is a static webpage. It gives us the infrastructure, features, and tooling tailored to deploying and managing static website. By using the Firebase CLI, we can deploy files from local directories on our computer to our Hosting server.

# Reference:

<https://console.cloud.google.com/marketplace/product/google-cloud-platform/firebase-hosting?project=potent-arcade-339423>