

Dublin City University School of Computing.  
CA674: Cloud Architecture Project report  
Course – MCM

# Comparison of different Paas OSS

Heroku, Cloud Foundry and Google App engine.

Vishvesh Kadam - 18210026  
Paritosh Gupta - 18210686  
Shashank Ravishankar - 18210902  
12/19/2018

## Contents

|  |    |
|--|----|
| ABSTRACT:.....   | 2  |
| CLOUD FOUNDRY:.....  | 2  |
| Introduction to Cloud Foundry: .....                       | 2  |
| Architecture of Cloud foundry .....                        | 2  |
| Deployment details:.....                                   | 3  |
| Setting the environment .....                              | 3  |
| Navigating to the directory .....                          | 3  |
| Deploying the application to the cloud:.....               | 3  |
| HEROKU:.....   | 5  |
| Introduction to Heroku: .....                              | 5  |
| Architecture of Heroku: .....                              | 5  |
| Deployment details: .....                                  | 6  |
| Setting the environment.....                               | 6  |
| Navigating the directory .....                             | 6  |
| Deploying the application to the cloud.....                | 6  |
| Challenges Faced.....                                      | 7  |
| Google App Engine:.....                                    | 8  |
| Introduction to Google Cloud: .....                        | 8  |
| Google Cloud Architecture:.....                            | 8  |
| Deployment Details.....                                    | 9  |
| Setting the environment: .....                             | 9  |
| Deploying the app to cloud .....                           | 9  |
| Web App Demo:.....   | 10 |
| COMPARISON OF DIFFERENT CLOUD PLATFORMS:.....              | 11 |
| PERFORMANCE EVALUATION OF DIFFERENT CLOUD PLATFORMS: ..... | 13 |
| CONCLUSION:.....   | 14 |
| REFERENCES: .....  | 14 |

## ABSTRACT:

Cloud is the most influencing technology in recent times. There are many opensource cloud providers in the market, where we can explore the methodologies of the deployment and the performance analysis. In general, the cloud operators will provide various services as app deployment, database, load balancing, DNS which provides the support for major challenges such as Compute, Storage, Networking, Bigdata and many other services. In the market there are several providers of cloud for platform as service(paas), to name few, Amazon web services, IBM, Open-shift by red hat, google cloud, Heroku, cloud foundry and many.

In this report, we have deployed the web application in three different open source cloud providers Heroku, Google cloud and cloud foundry. This contains an overview of the all the cloud service provider along with deployment details which contains a detailed analysis of deployment in terms of difficulty and methods. Also, an extensive analysis of the performance in terms of uptime, response time for the different location and different web browsers.

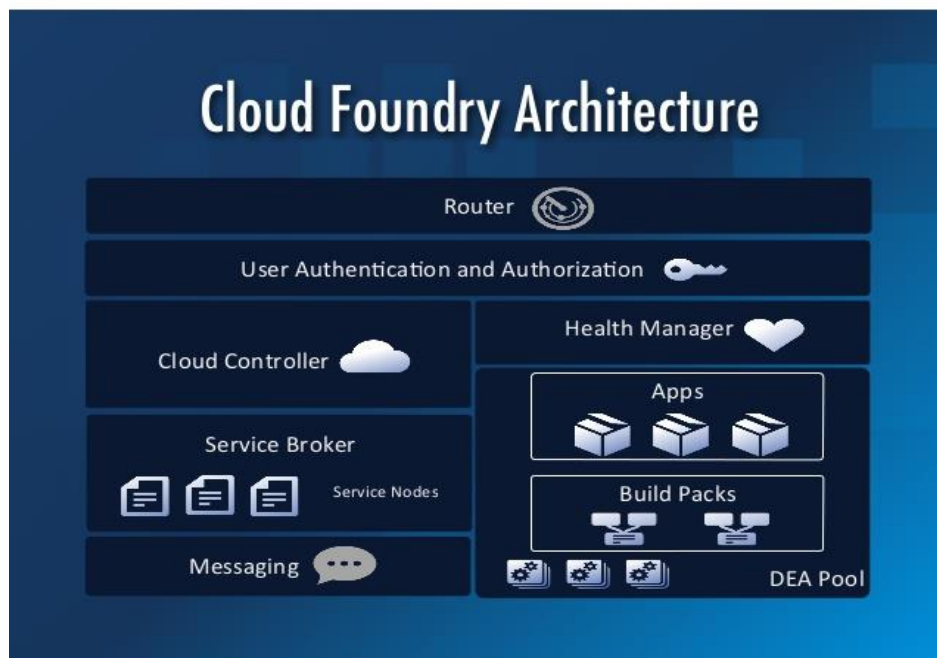
## CLOUD FOUNDRY:

### Introduction to Cloud Foundry:

With the platform as a service, cloud foundry provides the service to build, test, scale and application deployment. With the public cloud distributions and cloud instances, it is accessible to all as this is a opensource project. The Paas is developed on the java and EC2 framework.



Architecture of Cloud foundry: [1]Source: <https://www.slideshare.net/SpringCentral/>



## Deployment details:

**Setting the environment:** Login to pivotal cloud foundry: Initially the pivotal cloud foundry was initialized with user name and password and the CLI – command line interface was downloaded for the same. Using CLI, navigated to the api.run.pivotal.io with the command “cf login -a api.run.pivotal.io”. After the successful login it will be navigated to the Targeted org and Targeted space which we have created in the pivotal cloud foundry dashboard.

```
C:\Users\shash>cf login -a api.run.pivotal.io
API endpoint: api.run.pivotal.io

Email> shashank.ars@gmail.com

Password>
Authenticating...
OK

Targeted org student-org

Targeted space development

API endpoint: https://api.run.pivotal.io (API version: 2.128.0)
User: shashank.ars@gmail.com
Org: student-org
Space: development
```

**Navigating to the directory:** Now with the command prompt instructions we need to navigate to the folder, where all the files related to the project has been placed.

```
C:\Users\shash>cd cloud_project

C:\Users\shash\cloud_project>dir
Volume in drive C has no label.
Volume Serial Number is DABB-CBEE

Directory of C:\Users\shash\cloud_project

12/17/2018  05:29 PM    <DIR>          .
12/17/2018  05:29 PM    <DIR>          ..
12/17/2018  05:28 PM    <DIR>          Project
12/17/2018  05:27 PM                5,485,127 Project_NEW.zip
               1 File(s)                5,485,127 bytes
               3 Dir(s)  64,024,190,976 bytes free
```

**Deploying the application to the cloud:** After the environment setup and navigating to the directory where the project or application is present, to deploy the project in the code “cf push cloud\_pro” command is used. Here cf is the CLI command which stands for cloud foundry, push is the command to push the project to the cloud, “cloud\_pro” is the user given name, which we will use as part of URL creation after the successful deployment.

```
C:\Users\shash\cloud_project\Project>cf push cloud_pro
Pushing app cloud_pro to org student-org / space development as shashank.ars@gmail.com...
Getting app info...
Creating app with these attributes...
+ name:      cloud_pro
  path:      C:\Users\shash\cloud_project\Project
  routes:
+   cloudpro.cfapps.io
```

The project is pushed successfully to cloud foundry: Once the project is successfully deployed, it will display the details of the name, requested state, routes, last uploaded, type, instances and memory usage.

```
Waiting for app to start...




name:      cloud_pro
requested state: started
routes:     cloudpro.cfapps.io
last uploaded: Mon 17 Dec 17:34:09 GMT 2018
stack:      cflinuxfs2
buildpacks: php 4.3.66

type:      web
instances: 1/1
memory usage: 1024M
start command: $HOME/.bp/bin/start

state since      cpu    memory    disk    details
#0  running  2018-12-17T17:34:24Z  0.0%  0 of 1G  0 of 1G
```

**In Pivotal dashboard:** After the deployment, we will be able to see the details, Allocated memory, Disk allocated. Also, the dashboard displays the disk and memory allocated for the instance. The other important aspect of the dashboard is we can navigate to the deployed web application, by clicking on the “VIEW APP”.

Home / student-org / development / cloud\_pro

APP  
cloud\_pro    Running [VIEW APP](#)

Overview Services Route (1) Logs Tasks Settings Buildpack: N/A

Events Last Push: 05:33 PM 12/17/18

- Started app shashank.ars@gmail.com 12/17/2018 at 05:33:46 PM
- Mapped route to app shashank.ars@gmail.com 12/17/2018 at 05:33:35 PM
- Created app shashank.ars@gmail.com 12/17/2018 at 05:33:35 PM

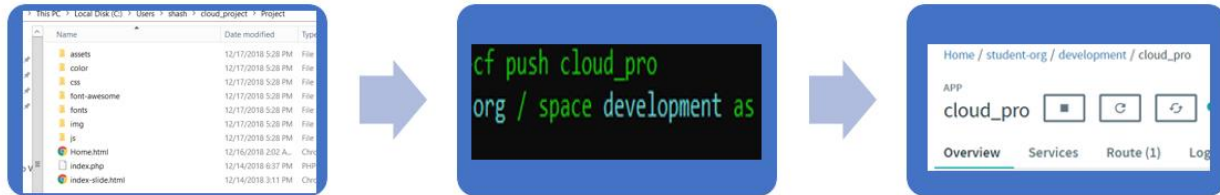
App Summary

|                       |                    |                  |
|-----------------------|--------------------|------------------|
| Instances / Allocated | Memory / Allocated | Disk / Allocated |
| 1 / 1                 | 0.03 / 1.00 GB     | 0.22 / 1.00 GB   |

Processes and Instances [View in PCF Metrics](#)

| web                                  |     |                  |           |   |
|--------------------------------------|-----|------------------|-----------|---|
| Instances                            | 1   | Memory Allocated | 1 GB      | Disk Allocated 1 GB <a href="#">SCALE</a> |
| <input type="checkbox"/> Autoscaling |     |                  |           |   |
| #                                    | CPU | Memory           | Disk      | Uptime                                    |
| 0                                    | 0%  | 34.84 MB         | 226.74 MB | 8 min                                     |

Below is the cloud deployment of the web application in the cloud foundry in a Nut shell. Initially a web project is created, the same has been moved to cloud using cloud foundry CLI- command line interface. After the successful implementation, we can see the details of the hosted application in pivotal cloud foundry dashboard.



## HEROKU:

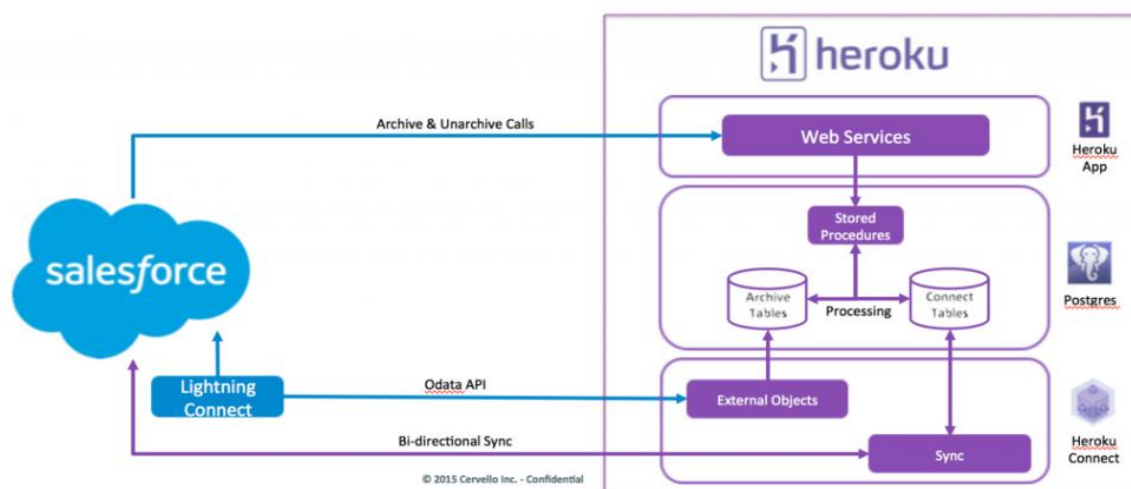
### Introduction to Heroku:

Heroku is one of the Platform as a Service provider owned by Salesforce, one of the leading giant in Software as a Service provider, provides cloud platform to build and develop the app followed by monitoring and scaling of it. It allows the developer to spend most of the time on development of the application code, not on the managing and monitoring of the servers. It provides the application development on programming languages such as Ruby, Node.js, Java, Python, Clojure, Scala, Go and PHP. It has its own Database Postgres SQL which can be accessed from any programming language with PostgreSQL driver. Its Infrastructure platform is managed by Amazon EC2. [2]



### Architecture of Heroku:

[3] Source- <http://mycervello.com/blog/salesforce-data-archiving-leveraging-lightning-connect-heroku-2/>



## Deployment details:

**Setting the environment:** First create the account on Heroku web portal and create the project on dashboard. Once you have the Heroku account, we required to download 'Git Bash' and 'Command Line Interface- CLI' to deploy any app. Using CLI interface command '\$heroku login', login the Heroku dashboard from CLI and by '\$ git init' which will initialize the Git to the repository. Once the repository is initialized, try to connect to project create on CLI by below command. In our case 'herukodcu' was the project created.

```
parit@Paari MINGW64 ~ (master)
$ heroku git:remote -a herukodcu
set git remote heroku to https://git.heroku.com/herukodcu.git
```

**Navigating the directory:** Now navigate to the directory, where the project files are placed. Please the below screenshot for the command.

```
parit@Paari MINGW64 ~ (master)
$ pwd
/c/Users/parit

parit@Paari MINGW64 ~ (master)
$ cd Git\ folder/

parit@Paari MINGW64 ~/Git folder (master)
$ ls -ltr
total 4
drwxr-xr-x 1 parit 197609 0 Dec 14 18:35 Project/
```

**Deploying the application to the cloud:** After the environment is setup and directory is navigated to project, use '\$git add --all', which will add the directory and will be ready for commit.

```
parit@Paari MINGW64 ~/Git folder (master)
$ git add --all

parit@Paari MINGW64 ~/Git folder (master)
$ git commit -am "this is the first commit"
[master 3af3ab3] this is the first commit
8 files changed, 32 insertions(+), 20 deletions(-)
rewrite Project/img/logo-dark.PNG (99%)
copy Project/img/{logo-dark.PNG => logo-dark2.PNG} (100%)
delete mode 100644 Project/img/team/11.jpg
rewrite Project/img/team/2.jpg (93%)
delete mode 100644 Project/img/team/22.jpg
delete mode 100644 Project/img/team/33.jpg
delete mode 100644 Project/img/team/44.jpg
```

We must perform the push mechanism to deploy the code in cloud. From the below command, we have pushed all the project files into the cloud.

```

parit@Paari MINGW64 ~/Git folder (master)
$ git push heroku master
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 423.65 KiB | 11.77 MiB/s, done.
Total 8 (delta 3), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:

```



Once the project is deployed successfully on Heroku, it will display the project folder in which it is deployed and the application link.

```







remote:      https://herukodcu.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/herukodcu.git
   04e2468..3af3ab3  master -> master

```

**In Heroku Dashboard:** After the deployment we will be able to see the details of the memory allocation and the region in which it is deployed.

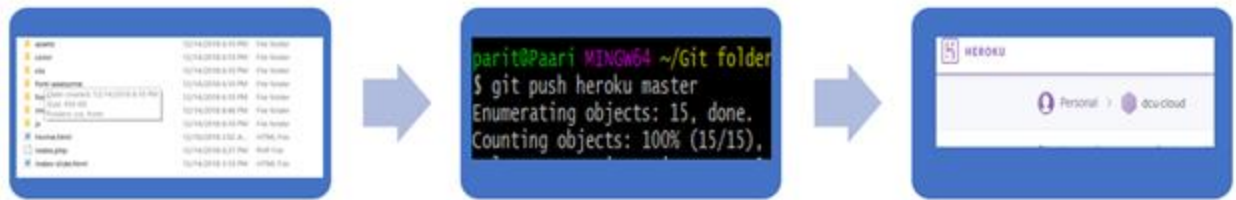
|      |                |  |
|------|----------------|--|
| Info | Region         |  Europe |
|      | Stack          | heroku-18  |
|      | Framework      |  PHP  |
|      | Slug Size      | 20.6 MiB of 500 MiB  |
|      | Heroku Git URL | <a href="https://git.heroku.com/herukodcu.git">https://git.heroku.com/herukodcu.git</a>  |

**Challenges Faced:** Initially we have made the application on HTML but it wasn't supported Heroku and the deployment got failed. We have induced PHP page to call HTML page which is successfully deployed on cloud. Please refer the below screenshot for the deployment logs.

|   |   |  |
|---|---|--|
|  |  | vishveshkadam@gmail.com: Deployed <a href="#">837d7533</a><br>Dec 14 at 6:38 PM · v3           |
|  |  | vishveshkadam@gmail.com: Build succeeded<br>Dec 14 at 6:38 PM · <a href="#">View build log</a> |
|  |  | vishveshkadam@gmail.com: Build failed<br>Dec 14 at 6:23 PM · <a href="#">View build log</a>    |

Below is the workflow diagram for the deployment, from project directory, pushing the code in cloud through CLI and the application on dashboard.





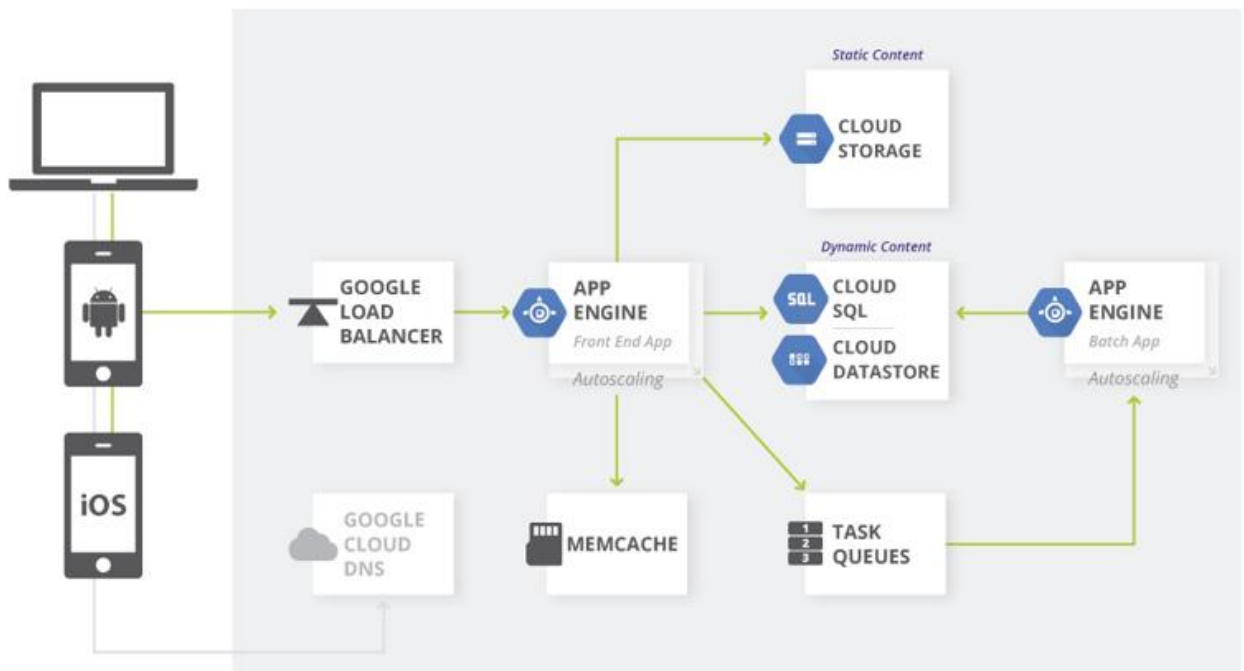
## Google App Engine:

### Introduction to Google Cloud:

Google has a reputation for highly reliable, high performance infrastructure. With App Engine you can take advantage of the 10 years of knowledge Google has in running massively scalable, performance driven systems. [4] An App Engine application enables developers to build platform which are easy to use and scale up as the need grows, using any language without worrying about monitoring and operation. Developers have the freedom to choose from many of the popular languages like Java, PHP, Node.js, Python, C#, .Net, Ruby and Go or bring your own language runtimes and frameworks if you choose. With abilities, for example, programmed scaling-up and downsizing of your application, completely managed fixing and management of your servers, you can offload all your framework worries to Google. [5]

### Google Cloud Architecture:

[6] Source: <https://cloud.google.com/solutions/architecture/webapp>



In an App Engine venture, you can send different microservices as isolated administrations, recently known as modules in App Engine. These administrations have full seclusion of code; the best way to execute code in these administrations is through a HTTP innovation, for example, a client asks for or a RESTful API call. The code in one administration can't straightforwardly call code in another administration. The code can be conveyed to administrations autonomously, and distinctive administrations can be written in various dialects, for example, Python, Java, Go, and PHP. Autoscaling, stack adjusting, and machine example types are altogether overseen autonomously for administrations.

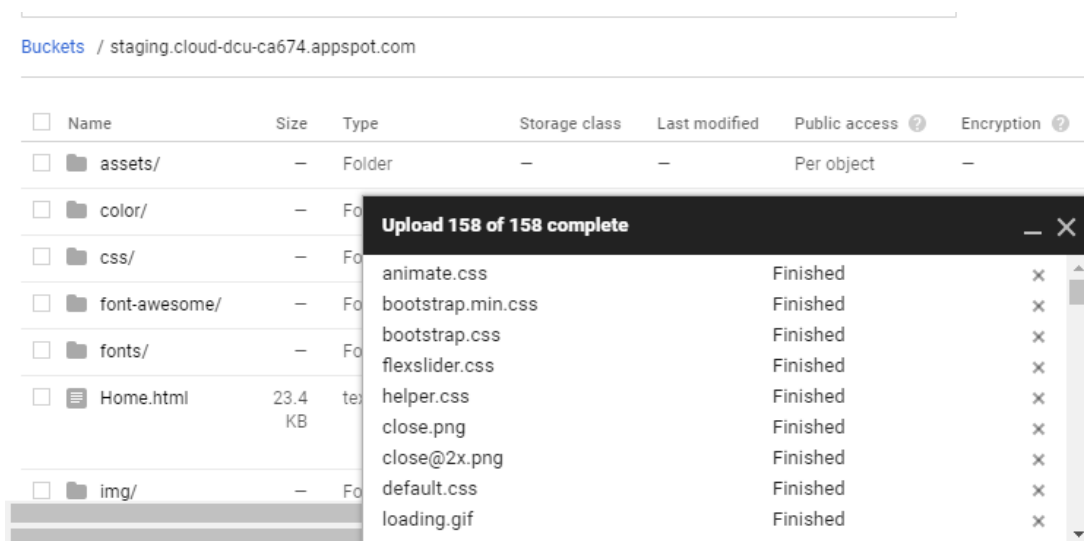
## Deployment Details: Google App Engine

### Setting the environment:

1. Firstly, I created Google App DCU account using links provided by university.
2. Then created Google App Engine location: Application Engine is territorial, which implies the foundation that runs your applications is situated in an explicit area and is overseen by Google to be repetitively accessible over every one of the zones inside that locale.
3. Meeting your inertness, accessibility, or strength prerequisites are essential elements for choosing the locale where your applications are run. You can commonly choose the area closest to your application's clients. For Ireland, I choose Europe-west2.
4. You can view the region by running the 'gcloud app describe' command or opening the App Engine Dashboard in the GCP Console.

### Deploying the app to cloud:

1. Then, on the dashboard, I tried to deploy entire code using drag and drop functionality. But as the web application was heavy, it failed to load our embedded video and other animations.



2. So, to overcome this situation, I had to download Google SDK
3. Then using yaml file to configured the resources and placed it in Project directory.

- Finally, deploy the code using command line scripts. Once the code is deployed, you can directly open your web app using `$ gcloud app browse` command.

```
83df163b37c3bec5c0ddbbb8c76dbb62eb00614a55d9838144e98f97 size: 3504
Finished Step #1
PUSH
DONE

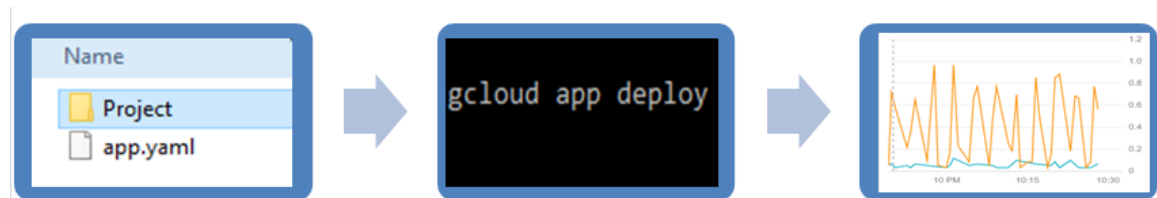
-----

Updating service [default] (this may take several minutes)...done.
Setting traffic split for service [default]...done.
Deployed service [default] to [https://cloud-dcu-ca674.appspot.com]

You can stream logs from the command line by running:
$ gcloud app logs tail -s default

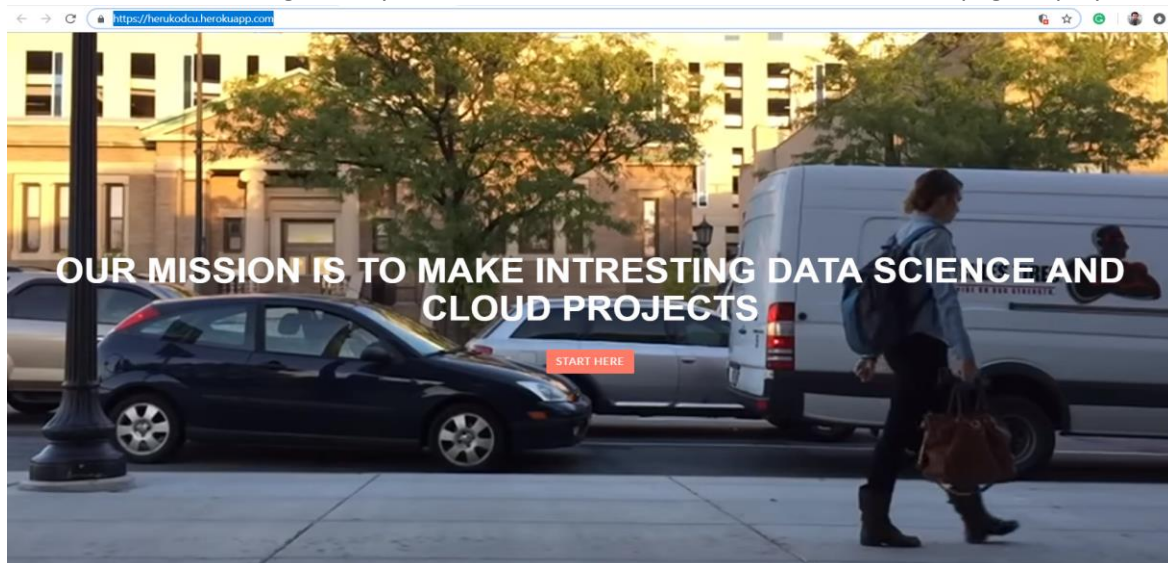
To view your application in the web browser run:
$ gcloud app browse

E:\Vishvesh\Project>
```



## Web App Demo:

We have created the website to represent our area of interest and work, and the modules which we have covered through this syllabus. Below are the screenshots from the webpage deployed on cloud.



## CLOUD PROJECT

Compare Contrast different Paas - Openshift, Cloud Foundry and Heroku.



You will never win, if you never begin

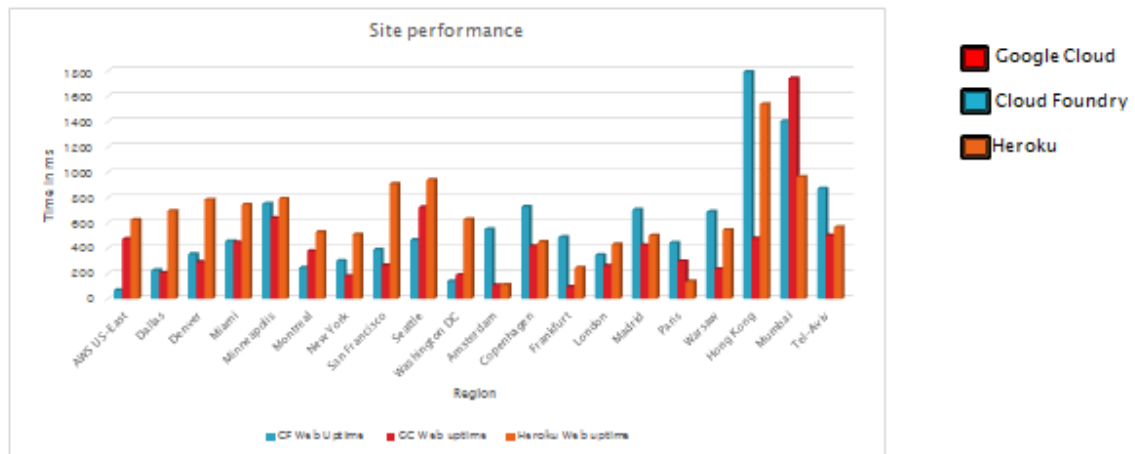
### COMPARISON OF DIFFERENT CLOUD PLATFORMS:

| Parameters        | HEROKU         | Google App Engine | Cloud Foundry   |
|-------------------|----------------|-------------------|---|
| 1. Popularity     | Very Popular   | Popular           | Less Popular  |
| 2. Deployment     | Easy           | Easy              | Easy  |
| 3. Workload       | Strong         | Basic             | Leading   |
| 4. Security       | Strong         | Strong            | Strong  |
| 5. Queueing       | Basic          | Basic             | Strong  |
| 6. Scalability    | Simple scaling | Auto scaling      | Basic   |
| 7. Datastores-SQL | Leading        | Leading           | For Google SQL driver, you have to insert the DB connection into the code manually. |
| 8. Java Support   | Poor           | Basic             | Best  |

|                                  |   |   |  |
|----------------------------------|---|---|--|
| <b>9. Prices</b>                 | Heroku's PaaS unit is called a dyno, and it offers 512MB, unknown CPU power and 100MB swap space for free. From there if you need more dynos for scaling, it costs \$0.05 per hour. | The dyno equivalent on AppEngine is called a FrontEnd instance, and it costs \$0.08 per hour.   | By Quote   |
| <b>10. Supported Integration</b> | Salesforce App Cloud: Heroku Enterprise comes with 150 pre-integrated add-ons.  | Google Cloud Platform supports multiple programming languages, such as Java, Ruby, and Python.  | Strong   |
| <b>PROS</b>                      | Offers standard SQL. Relatively painless deployment. Simpler pricing model.   | Access to the rest of Google services. Running asynchronous tasks much easier than Heroku. Runs on Google's own cloud infrastructure. | Perfectly aligned with springboot<br>Application health management<br>Free service discovery (Eureka)<br>Free distributed tracing (zipkin)   |
| <b>CONS:</b>                     | Not as large as Google. Hosted on AWS which suffers major outages   | Platform tie-in, lack of flexibility. No standard SQL DB.   | Does not support stateful containers<br>Supports showing logs,<br>This makes relying on Cloud Foundry's logs very unreliable.<br>The logs have to be persisted using other third party tools like Elk and Kibana |

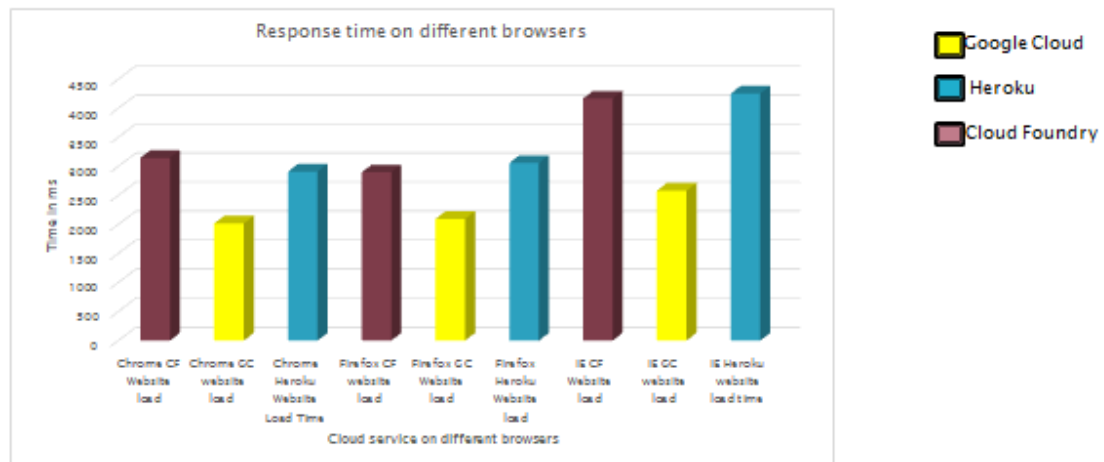
## PERFORMANCE EVALUATION OF DIFFERENT CLOUD PLATFORMS:

### Application HTTP Response Time



Data gathered from- <https://dotcom-monitor.com>

### App Load Time - Browser Compatibility



Data gathered from- <https://dotcom-monitor.com>

## CONCLUSION:

Through this assignment, we got to learn about different features, pros and cons of three leading Cloud Platforms like Heroku, Cloud Foundry and Google App engine. We got to know about the deployment procedures, its ease and challenges. However we found google app engine more user friendly and easy for deployment, it had some difficulty when done by drag and drop mechanism. Other two platform required use of CLI and git, but had definite steps and didn't give any issues while opening heavy web application. Overall the learning acquired was fruitful and we could compare several factors of each platform through this activity.

## REFERENCES:

1. <https://www.slideshare.net/SpringCentral/>
2. <https://www.upguard.com/articles/heroku-appengine>
3. <http://mycervello.com/blog/salesforce-data-archiving-leveraging-lightning-connect-heroku-2/>
4. <https://stackshare.io/stackups/cloud-foundry-vs-google-app-engine-vs-heroku>
5. <https://cloud.google.com/appengine/>
6. <https://cloud.google.com/solutions/architecture/webapp>