Platform as a Service

Tobias Reincke

January 7, 2020

1 Abstract

The objective of the research of this paper is to have an insight on 'Platform as a Service' technology and usage with Google App Engine as example.

Contents

2 Introduction

Platform as a service is a Cloud Computing technology similar to Infrastructure as a Service. it provides an Environment to host web and mobile applications via virtual machines, whereas single processes might be scaled up automatically to provide enough power needed. It supports development, management and run of the applications.

Google App Engine is one of the most used Cloud Computing Providers and one of the biggest 'Platform as a Service' out there. This Paper looks a bit into what makes it attractive and features and offers a look into the development history of the Engine. I will also show an example on how to use the api using java (and Python) as axample for the supported languages. A prominent short for Platform as a Service is 'PaaS'.

3 What exactly is the Google App Engine

The Google App Engine - in the following reffered to as App Engine or (G) AE - is a 'Platform as a Service'-service part of the Google Cloud, hosted by Google Inc. It is specifically used to host web applications uploaded by the user. It provides automatic scaling of resources and splits and app across multiple servers. It was released in 2008 in Beta and fully in 2011. As PaaS in falls under the public option as you only need a free google account to use it.

4 Definition

To Host an application a Platform needs the following:

- Server and Networks in form of a VLAN
- Storage and Storage Systems
- an Operating System
- corresponding firm and middleware
- Database services
- Usage and Scaling of Virtual Machines
- Software
- An Infrastructure and APIs connecting these services

What distinguishes Infracstructure as a Service and Platform as a Service is the Level they work on. Whereas IaaS provides an *infrastructure* to develop and to host your own virtual machines on, PaaS is comparably high level and provides a *platform* to only develop high-level applications, whatever they may

be intended to do. The platform also contains Database services, compared to not only storage systems, to work with.

There are multiple kinds of PaaS infrastructures, most prominent the Public, the Private and Hybrid.

Public PaaS is derived from Publice Software as Service, but hosts the virtual machines instead of software.It is accessed online. The pay model is based on a subscription basis. Overall Public Paas can be classified being placed somewhere between SaaS and IaaS. Google App Engine can be classified as such.

Private PaaS software is to be installed either in a public Cloud or in a company's own Data Center. Software will then rearrange the downloaded application and the databases into one functioning platform.

Hybrid PaaS is mix of both Public and Private deployments.

Google App Engine is public platform: Visual machines are connected in the Google Cloud. Furthermore as a subset of the Google Cloud it also is connected to Database services, storage systems such as Memcache and Datastore. It is operated on via Google Cloud Console or Shell (Google-Cloud-Sdk in your own Shell, or Cloud Shell).

5 What does Google App Engine offer ? (A short Overview)

Languages

AE's second generation runtime API prominently supports for Python 3 and Java 11 (and other Languages based on the Java Virtual Machine ,such as Kotlin). Popular scripting languages, that are supported, are javascript (Node.js), PHP, aswell as Ruby and Go. Those supports are generally up-to-date with the version of the language itself. Furthermore it includes support for Microsoft's DotNet Controller View Model applications and support for the c# language. Supports for relational databanks using SQL and MySQL are embedded since 2011. Limited support for older types within the first generation runtime are Python 2(.7), Java 8, GO 1.9 and PHP 5.5. .

APIs

All API's conform the Rest-standart. Social Graph API, Google Accounts can used.

Else

As mentioned in the section above, Google provides automatic resource scaling. Google App Engine requires a Google account to get started, and an account may allow the developer to register up to 25 free applications and an unlimited number of paid applications. It also includes APIs using Google Accounts for automatically sending emails and authentication. It can be accessed via Shell or via Google's "Cloud Console". (The Cloud Web Shell is basically a bash Shell based run on Google's Server). It's not recommended for permanent use since aside from the root and home directory. Easy to set up server / and applications. (Providing a short tutorial in the later sections.)

Google App Engine defines usage quotas for free applications this is one of the main attractions of the app engine. Extensions to these quotas can be requested, and application authors can pay for additional resources.

Dedicated Memcache As the name suggests, the memchache is basically a memory cache, but with some helpful features, dedicated to the users specific web application. What it does is save the returns for requests that occur especially often. It does this by providing direct access to the memory system via Key-Mapping. Google offers up to 100 GB of dedicated memcache (only on the us-central server). As long as not fully used the memcache will save any requests. The free version does not offer dedicated in this regard, as it excludes control over the size of the cache, which can make the app, depending on what it runs , exponentially slower. (A Chess engine like Stockfish for an example..) You can not hope for anything, which is disappointing.
Other mankos may be the limitation of key length: 250 Byte, though it is rather big and should not cause problems in most cases. Every other given key will be hashed. For efficient use we recommend the keys rather short and in relation to the dedicated cache not that big though. Memcache can be accessed manually via Cloud Console. Querying, adding and removing manual keys with the Gui is possible by hand.

BigQuery "Big Query is a scalable, interactive ad hoc query system for analysis of read-only nested data." -Wikipedia Big Query is the service offered to analalyze used Data. According to Hackernoon.com [1] "BigQuery is append-only and does not have support for primary keys", which leads to duplicate, although this can be addressed with workarounds in the apps.
Google also has a record of bad communication and reporting including the status page on `https://status.cloud.google.com/` not updating and showing wrong information.

Cloud SQL Cloud SQl is google intern database as a service offer to host and make traditional relational database systems in MySQL and PostgreSQL

[1] `https://hackernoon.com/going-gae-our-experience-with-google-app-engine-deaf2b7171c1`

compatible within Google Cloud including App Engine. MySQL 5.6 or 5.7, and provide up to 416 GB of RAM and 30 TB

Graph Query Language GQL is an intelligent modern query language , being able to provide Queries in and via the Json format, that supports a property being a combination of multiple types (and more types overall).

- only select statement

Data Store One of App Engine's ways of saving things and part of Google Cloud's Firestore. It is used to save Objects written in an Object-oriented No-SQL language like java with the command "thisobject.put(); ". The write time is fast, so it's good to use for moments with a lot of traffic or writing intensive tasks in general.

TaskQueue This API was made to support asynchronous work between multiple workers outside of called Https requests in the background. There are two kinds of Taskqueue flavors:

- **Push Queues** are for scheduled HTTPS requests to App Engine's worker engines meaning they come with some of the same restrictions as other requests. Automatic scaled tasks need to finish within 10 minutes if scaled not manually, 24 hours if manually. This Feature allows managing the rate at which new requests are scheduled. (!) Usage scenarios usually include slow, but short processes that can be divided easily, like for a round mail in social media where multiple people need to receive the same notification or message.
- **Pull Queues** are for distribution of collected leased jobs from the worker services and give controll over when jobs are handled by which worker in a certain and with what priority. An usage example might the updating of a live leaderboard which is split into multiple processes fetching high scores from single players which then are pulled in by the queue.

The operations using Task Queues are prized at 40 cent with the first 5 Millions, of which the first Million is free entirely.

Cloud Storage {Will be finished in the final issue}

Social Graph API {Will be finished in the final issue}

6 How to Operate on it

Via Shell Depending on your personal computer's operating system you first need to install the google-cloud-sdk. (or go to `https://ssh.cloud.google.com/cloudshell/editor` and continue pretending this is your local computer.

Files may be up-/downloaded via the options menu on the right upper corner clicking the 3 dots.)
Since the whole Google Cloud sdk is for interacting with the entirety we just need a small subset of commands to interact with App Engine specifically. Those can be found printed by typing 'gcloud app'. To create a project type 'gcloud app create'. On first login a website, requiring you to login via Google Account, will be opened. You will have to enter a project name. To upload an application you need to create a directory and insert multiple files:

- requirements.txt containing <framework name>=<versions number> for every framework used.
- main file for any kind of language mentioned in the subsection, code according to framework
- app.yaml for listing the runtime of the app according to yaml format: runtime: <language>

The deployment follows with 'gcloud app deploy'. The process includes uploading all files to App Engine, so please supply a sufficient internet connection. To not use deploy for every change made, it is sufficient to just use 'gcloud app update' in the future. 'gcloud app deploy' opens the application in the browser. For additional information either 'gcloud app describe' or open the Cloud Console in your browser. (type 'gcloud app open-console').

Via Console The Console on the main page contains key information on the current project, requests and perfomance over time information, resources, a billing estimate for the current period and information on availability of Google Cloud's Services, and a few redirections on tutorials. Using the sidebar you can get to the overview pages of Task Queue and Memcache, and .

7 The Pay Model

The Pay Model is a simple Pay-as-you-go model, which makes it really attractive for dynamic environments. There is also the free of charge limit of 1 Gigabyte memory and Traffic. You can use AE for free until you have reached that limit. Some functions have a different limit though; "to guarantee to stability of the system", according to Google. The model also occupies a choosable resource contingent for day and minute. You can combine this with an optional cost contigent. Whenever one these is reached, the applications will be shut down until the reached contigent is renewed. The guarantees both User safety and a fair system, and saves unnecessary counting. Application instance pricing: For both runtime APIs there are nine instance classes of resources available to you. The charge is accordingly to the instance class chosen. They determine (1st) how much memory and processor speed are available for your programm and (2nd) how to handle the scaling of the programm: either automatically or

Instance class	Cost per hour per instance
B1	$0.05
B2	$0.10
B4	$0.20
B4_1G	$0.30
B8	$0.40
F1	$0.05
F2	$0.10
F4	$0.20
F4_1G	$0.30

Figure 1:

manually. The Handling has no influence on the pricing though. The B8 class is only comes with manual handling. Storage: App Engine offers one Gigabyte for free a month. Writing, deleting and changing entries differ in pricing: Everything extending 1 Gygabyte will be charged at these prices.

8 Restrictions

Developers have read-only access to the filesystem on App Engine. Applications can use only virtual filesystems, like gae-filestore.
App Engine can only execute code called from an HTTP request (scheduled background tasks allow for self calling HTTP requests). "Users may upload arbitrary Python modules, but only if they are pure-Python; C and Pyrex modules are not supported."
Java applications may only use a subset (The JRE Class White List) of the classes from the JRE standard edition. This restriction does not exist with the App Engine Standard Java8 runtime. This issue was one of the safety restrictions included in App Engines early stages, but does not play any important role nowadays.
A process started on the server to answer a request can't last more than 10 minutes or 24 hours via TaskQueues (with the 1.4.0 release, this restriction does not apply to background jobs anymore). Session affinity (also known as sticky sessions) is are not supported, meaning you can't scale up the performance of one job without having to split workload onto multiple workers. Overall the user is limited to writing his apps in high level languages and not being to set up his own virtual machines.

9 In Hindsight/Conclusion

Pros Google makes it very easy to access your own litte Apps run on the Web. For us it took just to log in and access the console. (We also installled the Google Cloud SDK). Then you initalize it and insert your code and upload. It's relatively fast and easy. It is also as an public suited to for either Individuals who do not want to set up their own little server for their applications, or smaller start-ups. Also the console is a big plus for providing such amount of information on performance in this compact format in the cloud console. The support for Python is far-ranged, making development quite easy and fast, and expandable. The free to use options exceed Amazon Webservice's free options in performance and quotas. [2].

Cons There is a fundamental lack of low level cofiguration freedom, as it in all of Platform as a Service. Adding to this are lots of smaller limitations, like those mentioned in the Restrictions section. The pool of supported languages is also quite small and limited JVM languages and script languages with an interpreter, excluding popular basic level languages using a Compiler, due to safety concerns. The API on Google's side may be good, but due to their own specific APIs having raised portability issues and concerns, it is not recommend to start bigger projects using App Engine with the intention of migrating. App Engine is quite expensive too, like all PaaS tend to be. If an app makes money on Google scale, it is guaranteed to make money. Deep Data analysis is rather difficult due to minor limitations.

9.1 Popular examples of applications using App Engine

- Snapchat

10 sources

https://www.icsr.agh.edu.pl/ malawski/google-appengine-ieee-2011.pdf

https://cloud.google.com/appengine/docs/standard/?hl=de http://googlecode.blogspot.com/2011/10/goog cloud-sql-your-database-in-cloud.html https://hackernoon.com/going-gae-our-experience-with-google-app-engine-deaf2b7171c1 16:62 23.10.19 https://cloud.google.com/appengine/docs/standard/pytho

11 unfinished things /notes (will be excluded in the final form, please ignore)

INSERT G(APP) ZU GAP JOKE WEIL BEHIND

pros: + ez to use +"sexy python libraries" - you cannot include your own c modules!! +much options +for free options +good options for you budget +can

[2] https://www.icsr.agh.edu.pl/~malawski/google-appengine-ieee-2011.pdf

be expansive +hard to calculate yourself if you've got the time to calculate it +fast

-complex - maybe a bit too much options portability concerns + fear being locked to google (open source projects to fix that) -limited to google's api no c or c++ support overall: very good good no sticky sessions (modules, that when loaded can't be removed with usual unload / purge)