## Lesson 7: Google App Engine

### 7.1. Introduction

In our previous lesson, we learnt about virtualization. In our lesson today we will look at Google App Engine. Google App Engine (GAE or simply App Engine) is a platform as a service (PaaS) cloud computing platform for developing and hosting web applications in Google servers. Also defined as a cloud computing technology used to virtualize applications across multiple servers and data centers.

### 7.2 Lesson structure

This lesson is structured as follows:

- Introduction
- Lesson structure
- Google App engine overview
- Google App engine architecture
- Google App engine development cycle
- Installation and configuration of Google App Engine
- Advantages of Google app engine
- Disadvantages of Google app engine
- Revision questions
- Summary
- Suggested reading

### 7.3. Lesson objectives

By the end of this lesson, you will be able to:

- Define google app engine
- Describe the architecture of google app engine
- Describe the process of installation and configuration of GAE
- Describe the pros and cons of developing app in google app engine
- Describe the google app engine development life cycle

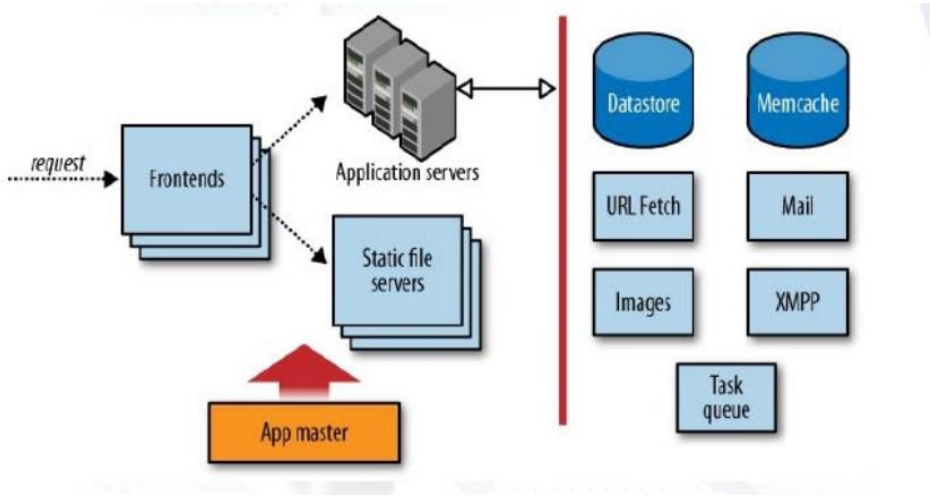
### 7.4. Google App Engine overview

Google App Engine (GAE) is a cloud computing platform for developing web-based applications. The Google App Engine Development Kit offers tools for developing and testing the entire application locally. Once you've developed the application, Google App Engine provides a management tool which gives the control of your application using a simple web based dashboard. It also scales automatically. GAE is free to use until your application reach a certain amount of resources use. After reaching this amount, if you plan to scale your application resources, you must pay for the resources used. When you write a Google Application Engine Application - you are running in the Google Cloud. Google App Engine requires that application developers use only a limited set of supported programming languages together with a small set of APIs and frameworks that are mostly dedicated to Web applications. The applications are executed in a secure hosting environment running on the computing infrastructure that Google

provides and manages. Google App Engine is designed to host applications with many simultaneous users. When an application can serve many simultaneous users without degrading performance, we say it scales. Applications written for google App Engine scale up as more people use the application. App Engine allocates more resources for the application and manages the use of those resources engine scale automatically. In addition, with Google App Engine, you only pay for the resources you use. To help you get started, every developer gets a certain amount of resources for free, enough for small applications with low traffic. Google App Engine primarily supports PHP, Go, Java, Python, Node.js, .NET, and Ruby applications, although it can also support other languages. Fees are charged for additional storage, bandwidth, or instance hours required by the application.

### 7.5. Google App Engine architecture

System architecture is concerned with the components and their functions. The Google App Engine is designed to address your concerns regarding scalability and reliability. To archive this a number of components are involved:




**Google accounts:** This component is included in the application to provide user control. Customer's applications can allow users to sign in their product by using Google accounts instead of implementing their own user account system

**Frontend:** The web server that accepts the requests from the users before passing them to application server. On the basis of data from app master, the frontend can send the request to static file servers or to app servers.

**URLFetch:** This service is fast as other Google services and can retrieve other web sources by using Google infrastructure. It allows Google App Engine applications to exchange data with other hosts using HTTP requests.

**App master:** This component ensures that all the files are in place on a web server before using the new files to handle requests. The app master is responsible for governing the frontends, app servers, and static file servers. It is also responsible for deploying new versions of application

software and configuration, and updating the "default" version served on an app's user-facing domain.

**Static file servers:** A server intended to serve static files such as JavaScript, images, and CSS in addition to handling dynamic requests.

**Application servers:** This is where the frontend routes requests for your application code.

**Data store:** Google owns its proprietary database which is called Bigtable. This is a distributed storage system entirely built by Google, having other Google services such as Scheduler, GFS, Lock Service and MapReduce8 as its base. Its purpose is to manage structured data that is designed to scale to a very large size.

**Memcache**: This is a distributed in-memory data cache in front of or in place of reliable constant storage is often use by high performance scalable web applications for some tasks. Because of this reason Google App Engine supports memory cache service [13]. The Memcache service assures user applications with a high performance in-memory key-value cache that is available by numerous instances of the applications.

**Images**: Google provide as well and Image service which supports manipulating image functions, like resize, crop, rotate and flip images both in JPEG and PNG formats. This is very useful service for customer's applications which are dealing with photographs or just provide users with a service to prepare their avatars.

**Mail:** Every user application can send message using App Engine's mail service, which also uses Google's company infrastructure. This is very useful way for applications to communicate with users and notify other users for any activity or updates.

**XMPP:** Extensible Messaging and Presence Protocol, a set of open technologies for instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data. The XMPP server provides basic messaging, presence, and XML routing features.

**Task queues:** This component allows the applications perform work, called *tasks*, asynchronously outside of a user request. If an app needs to execute work in the background, it adds tasks to task queues. The tasks are then executed later.

### 7.6. Google App Engine development life cycle

Generally, there are five steps involved when developing an apple for google app engine.

1. **Build**: Using a supported programming tools like python, write the code
2. **Test:** Run the application locally to detect any errors.
3. **Deploy:** Push the application to google servers
4. **Manage:** Administer the application via web console.
5. **Upgrade**: The process of replacing the application with a newer version.

### 7.7. Installation and configuration of Google App Engine

The steps followed will vary depending on the language used e.g. python.

1. Ensure you have google account.

2. Download and install the latest release of Python 3.
3. Download, install, and then initialize the Cloud SDK(Software Development Kit). If you already have then update your SDK using the command: `gcloud components update`
4. Create a new project e.g. `gcloud projects create project-1`
   Then verify that Verify the project was created type the command below:
   `gcloud projects describe project-1`
5. Initialize your App Engine app with your project and choose its region:
6. Make sure billing is enabled for your project. A billing account needs to be linked to your project in order for the application to be deployed to App Engine.
7. Install the prerequisites such as Git:
   `gcloud components install app-engine-python`
8. Prepare your environment for Python development

### 7.8. Advantages of Google App Engine

- **Security:** The application data and code are stored in highly secure servers.
- **Scalability:** GAE has automatic scaling feature. Thus, regardless of the amount of data or number of users that your app stores, the app engine can meet your needs by scaling up or down as required.
- **Cost savings:** Scale downs to zero when no traffic/requests. Besides there is no need to hire stuff who will manage the IT infrastructure.
- **Platform Independence:** Developer can move all their data to another environment without any difficulty as there are not many dependencies on the app engine platform.
- **Infrastructure**: GAE uses high-speed Google infrastructure. The developer only needs to focus on writing the code, rest everything is taken care.
- **Ease of use**: GAE incorporates the tools that you need to develop, test, launch, and update the applications.

### 7.9. Disadvantages of Google App Engine

- The cost of switching to a different vendor is so high that you will is essentially get stuck with Google.
- App Engine can only execute code called from an HTTP request.
- Not suitable for CPU intensive calculations. They are slower and expensive.
- Not all programming languages are supported by GAE.

### 7.10. Revision questions

a) Outline any three advantages of Google App Engine
b) Differentiate between google app engine and google compute engine
c) Briefly explain the steps followed in google app engine development.
d) Outline the function of each of the following components in the google app engine architecture:
   i. URLFetch
   ii. Memcache
   iii. Static file servers

### 7.11. Summary

In this lesson, we have learnt that google app engine is a cloud computing technology used to virtualize applications across multiple servers and data centers. We have looked at various components of google app engine:

- Google account- For authentication and user control.
- Frontend: The web server that accepts the requests from the users before passing them to application server.
- App master: Used for governing the frontends, app servers, and static file servers.
- Static file servers: Used to serve static files such as JavaScript, images, and CSS in addition to handling dynamic requests.
- Application servers: This is where the frontend routes requests for your application code.
- XMPP: Technology for instant messaging, presence, multi-party chat, voice and video calls and routing XML data.
- Memcache: Used to provide high performance in-memory key-value cache
- Data store: Used to provide a distributed data storage and operations.
- URLFetch: Used for invoking external URLs.
- Mail- Used for sending mail from the GAE application.
- Task queues- Used for invoking background processes.
- Images- Used for image manipulation.

### 7.12. Reference


http://cse.tkk.fi/en/publications/B/5/papers/1Zahariev_final.pdf

https://www.youtube.com/watch?v=jWRtX8vs_cM

https://codelabs.developers.google.com/codelabs/cloud-app-engine-go/index.html?index=..%2F..index#0

http://www.cse.hut.fi/en/publications/B/5/papers/1Zahariev_final.pdf

https://nuxtjs.org/faq/appengine-deployment/

https://stackabuse.com/deploy-node-js-apps-on-google-app-engine/

https://www.ijeit.com/vol%202/Issue%203/IJEIT1412201209_30.pdf

https://upcommons.upc.edu/bitstream/handle/2099.1/16104/85328.pdf?sequence=1&isAllowed=y

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

https://cloud.google.com/appengine/docs/flexible/python/quickstart