Cloud Service Models - Part 2

This lesson continues covering types of cloud service models.

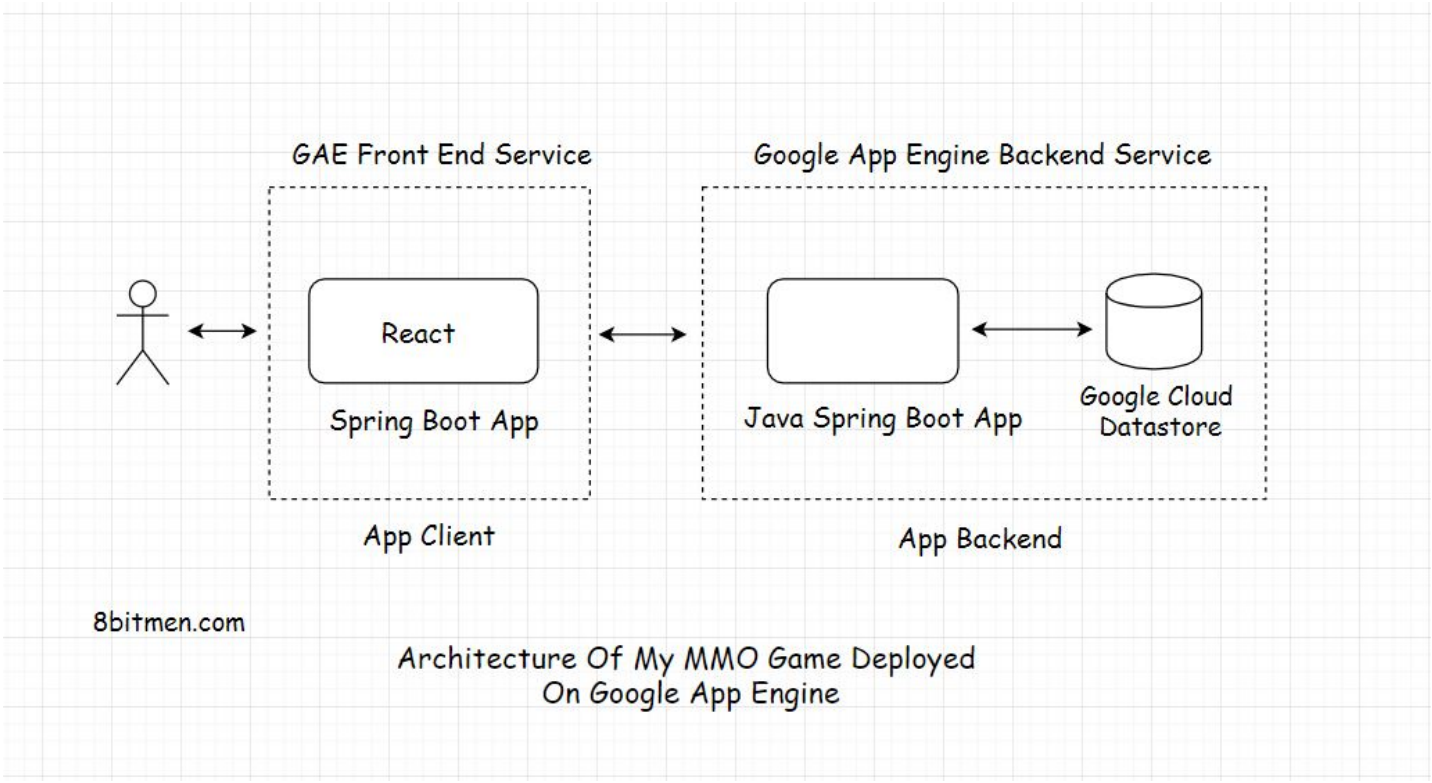
We'll cover the following

- My experience building an app using a PaaS
- Software as a Service (SaaS)

My experience building an app using a PaaS

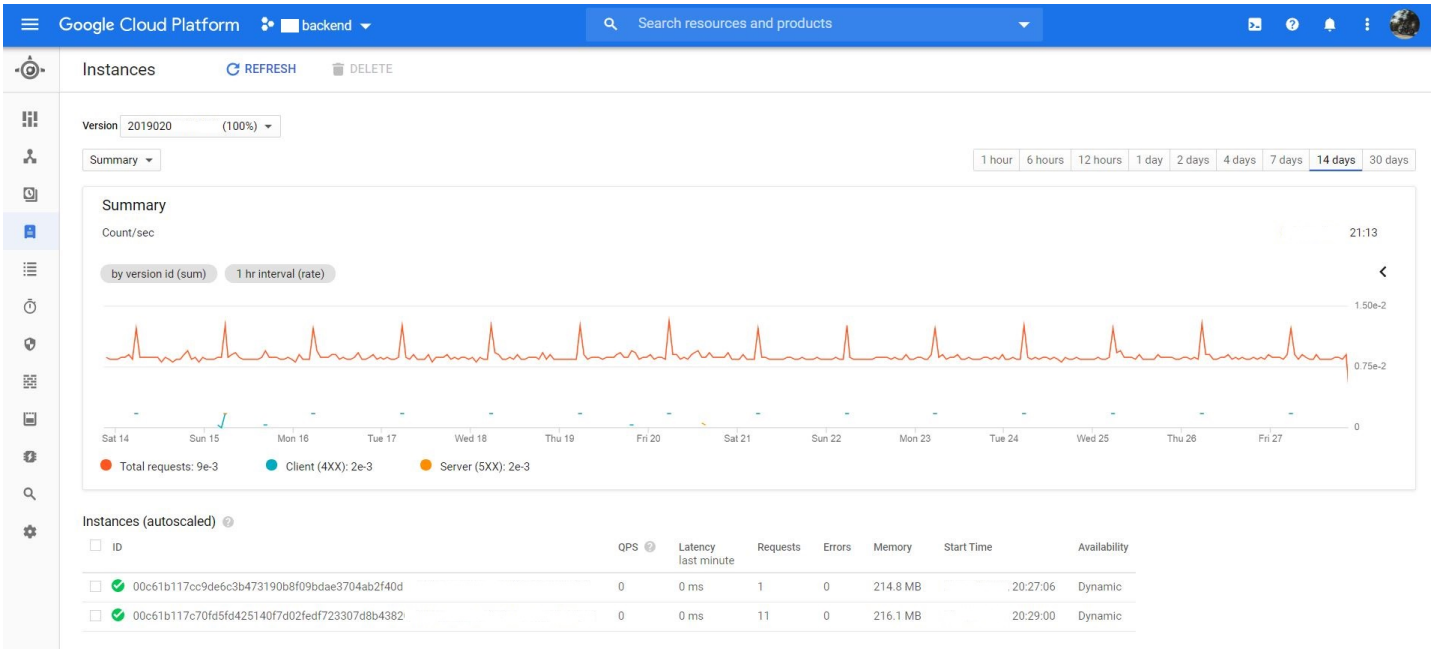
A while back I built an online multiplayer business simulation game using Google's *PaaS*. The backend was written using *Java, Spring Boot*, and the database was *Google Cloud Datastore*. This is a proprietary product of *GCP*. The UI was in *React*.

The in-game notifications, background jobs, and all the asynchronous events were handled by the task queue service that the platform offered via an API. I did not have to worry about using a message queue like *RabbitMQ* or *Kafka*. The *UI* and the *Backend* were decoupled into separate *Spring Boot* apps and were hosted separately as two *Google App Engine* services. This decoupling facilitated the introduction of new clients without the need to update the backend code.



Architecture Of My MMO Game Deployed On Google App Engine

There was a bit of a learning curve with respect to Google Cloud Datastore. However, once I wrote the game and deployed it, I didn't have to do anything regarding managing the servers, technology, log management, process monitoring, and whatnot. Everything was on auto-mode and taken care of by the platform. Here is a screenshot of the app dashboard during the testing phase of the game.



I ran the service for a while. It was free to play; no money involved. Players could sign up easily using their *Facebook OAuth Login*. The game had one particular module that enabled the players to trade virtual goods with each other. As it gained traction, the frequency of read-writes, due to the nature of the module, became way too high.

I eventually ran out of money, and I didn't have the time to add premium features or raise money. The platform's generous free tier offered up to 1 GB of outgoing data bandwidth per day. That didn't cut it. So, I finally had to pull the plug on the service. It was fun running it though :)

Okay!! moving on to the *SaaS* service model.

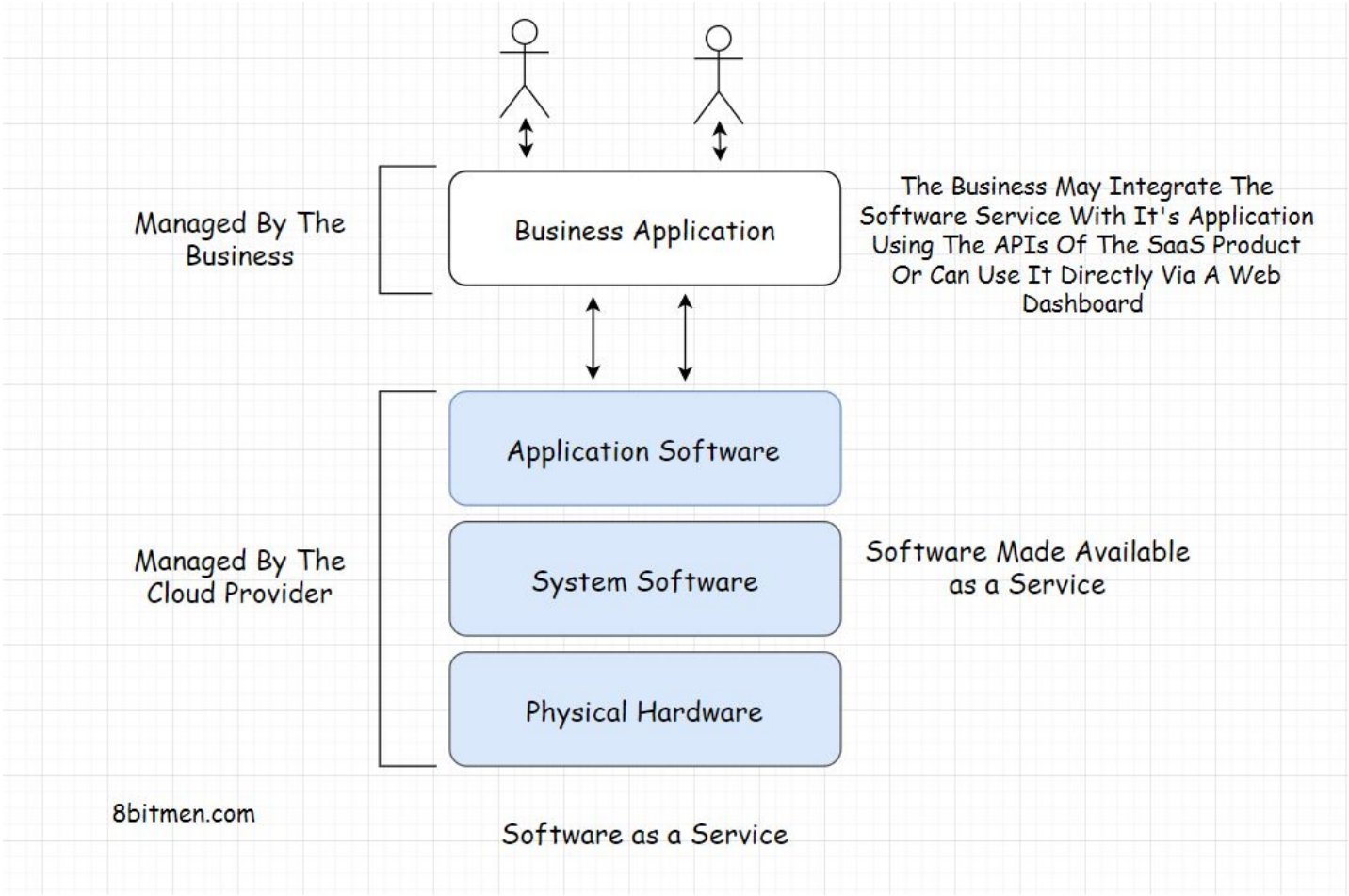
Software as a Service (SaaS)

Think of using a *SaaS* model as accessing a full-fledged web service run by a third-party provider. The service can be a search service, a web-based workplace communication tool, or any service that runs on the web.

Let's understand *SaaS* with the help of an example. Say you need to add a product

search module on your e-commerce platform to enable users to easily search the products. There are two ways to pull this off. Write the search service all by yourself and manage it continually or leverage a ready-made, customizable third-party search service via a *SaaS* service model and integrate it with your application.

Using a third-party search service would make sense if you are looking for a quick, easy-to-use, and fully managed search solution for your platform. *SaaS* is also known as *on-demand software* and is made available via third-party providers on a subscription basis or license. These *SaaS* services are plug and play in nature and can be integrated with our product using their *APIs* or the *Web-hooks*. The integration largely depends upon the nature of the service.



The provider of a *SaaS* manages everything. The users of the service don't have to worry about the technology stack the service is built on, how the service is deployed, the scaling strategies, software upgrades, and so on.

The trade-offs involved when using these services are:

- The business has to share its data with the service provider. Not ideal for a business looking to keep everything on their premise.
- There is a limit to the customization flexibility and the features a business has

access to.

- If the service goes down or has high latency, the business can't do much from an engineering point of view.

This is a good read on *SaaS* – How Slack became the fastest growing B2B SaaS business ever.

Alright, so we've covered the three types of cloud service models. Let's discuss the fourth cloud service model that is *Functions as a Service (FaaS)*. However, before you learn about FaaS, you'll take a quick quiz in the next lesson.