# AWS vs Google Cloud: Flexibility vs operational simplicity

Published September 26, 2015 (updated: August 8, 2019) in Cloud.

I wanted to revisit a theory I first wrote about in my 15 April 2015 e-mail newsletter—how the approach of Amazon Web Services is different from Google Cloud Platform—and add that to my theory on how containers are core to Google's Cloud Platform strategy.

## The core theory

On the surface, AWS and GCP are very similar, but their approach to product design is actually quite different:

- **Amazon Web Services**: *Flexibility*. They provide a huge range of sophisticated functionality that allows you to replicate everything you might have in your own data centre. This ranges from control over low level networking and compute through to providing multiple critical infrastructure products like logging, monitoring, access control and load balancing; and application level services like search, email delivery, file sharing and databases. It's up to the developer to use them correctly.
- **Google Cloud Platform**: *Operational simplicity*. They provide the same kind of products as AWS (although not as many, yet) but have a particular opinion about how they should be used, hiding options they don't think you should need to consider. Google has the experience, let them deal with most things for you in the best way.

## Example: pricing

A great way to illustrate this is through the approaches to pricing.

Amazon gives you a huge range of options: on-demand, reserved (specifying region, zone, OS, utilisation and selling reservations on a market) and spot market bidding (with lowest price and diversified pricing). Then there's the other usage based costs such as i/o charges, volume size charges, provisioned IOPS and choosing reserved capacity.

In contrast, Google gives you 1 price for the instance costs, automatic "sustained usage" discounts which just apply based on your usage and then option to have preemptive instances at significant discount but with limited lifespan. Disks are priced based on volume size and give predictable performance that scales linearly. And datastore products give guaranteed throughput per node, so you scale up performance alongside cost.

Here, Amazon is the most **flexible** and gives you a huge range of variables to optimise and calculate the most cost efficient mix of on-demand and reserved (with some upfront payments), whereas Google pricing is pretty **simple** based mostly on what scale you need now, growing linearly as needed.

## Example: databases

Google is famous for its Bigtable service. The paper describing this technology led to products like Cassandra and HBase. Amazon has a similar technology which it published as Dynamo (which Cassandra also takes some ideas from), resulting in products like Riak and Aerospike.

The cloud platform equivalents are Google's Cloud Bigtable and Amazon's DynamoDB. The former has only recently been launched but is priced based on a simple per node + storage volume model. Each node can process a set number of QPS and the price grows linearly as you add more nodes for additional performance. In contrast, DynamoDB pricing is based on provisioned throughput (read and write separated) in "capacity units", billed per hour + index data storage + optional features like cross region replication and streams. There's also reserved capacity and network charges too.

Again, Amazon gives you a lot of **flexibility** to choose the features and precise capacity you need, so long as you calculate it. Google prefers to ask what scale you need and how much data, and then give a **simple** price that can be predicted linearly.

# Philosophical origins

I think these different approaches can be explained by where the products are coming from internally.

Google has built an internal platform. All these public services are productised versions of what Google engineers use to build Google services internally. The stated goal is that Google engineers do not need to know where data lives, what throughput needs to be provisioned nor how to deal with replication across data centres...they just consume an API and let the Google software deal with everything underneath, supported by their Reliability Engineers. It's a true platform as a service and Google wants you, as a GCP customer, to be in the same position—trust Google to know what is best. It's **operational simplicity**—just get on and build your app.

Amazon is much more about building a product range to replace your data centre. When you run things yourself you have full **flexibility** and control over everything. Amazon seems to believe that you shouldn't have to deal with the really low level things like data centre power, environmentals, networking equipment, etc, but it should still expose as many options as it can through the APIs. Just looking at Amazon's VPC product shows the sophistication of the implementation of Software Defined Networking and the options it gives you. You know your application best, you should have the **flexibility** to optimise it.

# How this applies to containers

Google is pushing the container agenda with Kubernetes and the open formats because containers will be the future of how applications are deployed. If the underlying infrastructure doesn't matter and containers can be ported and run anywhere, the differentiator is the supporting services they consume.

Google is trying to make the most important decisions for you so its services become the best in class. With a full range of IaaS/SaaS products you can consume just based on the **simple** calculations of scale and data size requirements, it eventually makes GCP the best place to run your application. This applies to running the containers themselves —Google deals with deploying them, providing the requested resources and redundancy. Just upload the container and go.

Amazon has less experience running container technologies, which Google pioneered. Amazon may be winning now, but by the time GCP has matured, their bet is that containers will be fully portable and GCP will become a much more compelling proposition as the best place to run container based applications.

# Which approach is better?

Amazon is the biggest provider but Google Cloud has only been around in it's current, truly competitive form for a 18–24 months. The question is: do developers prefer to have full **flexibility** or would they like the hard problems like scaling databases and cross-datacentre networking solved for them as part of an operationally **simple** platform.

PaaS has been very popular but typically gets very expensive at scale. You can optimise better when you have more control. But that was before, both Google and Amazon are fighting hard on price competitiveness, and the services have been unbundled out of a monolithic application platform.

Cloud environments are typically cheaper if you consume all the supporting services, so perhaps customers migrating from their own datacentres will want to maintain the flexibility. But new, greenfield applications don't have all those legacy requirements.

They're also the most likely to be able to use containers rather than trying to port old applications into them.

I think this differentiation is important and we'll continue to see AWS and GCP release features, differing on **flexibility** vs **simplicity**. AWS might be best for migrations to the cloud. GCP might be better for new, container based applications. It's a long game.

## About the author

Co-founder & CEO of Console (the best tools for developers). Researching sustainable computing at Imperial College London. Previously Co-founder & CEO of Server Density (acquired by StackPath).