Why Use Cloud? – Part 1

This lesson discusses the reasons for deploying apps on the cloud.

We'll cover the following

The driving factors behind why more and more businesses are deploying their services on the cloud are both *economic* and *technical* in nature.

We'll begin with the economic reasons and then move on to discuss the technical ones.

Economic reasons

No upfront deployment costs

I talked a bit about this in the introductory cloud lesson, but with so many businesses onboard cloud providers have achieved economies of scale. As a result, the cost of hosting a single service has come down significantly. So much so that these platforms are able to provide a perpetual free tier of their services to the businesses.

With a free tier, these businesses can experiment a bit to get an idea of whether the cloud service offerings go well with their requirements and test the waters simultaneously. If a service has minimal traffic and its resource consumption remains below a certain threshold, there is literally no hosting charge that the business has to pay.

The upfront deployment costs for a workload is zero. When we host our app on the

cloud, there is no need for us to set up our data center before we launch our service. There are no IT costs involved in setting up and managing things. With these cloud platforms, all we have to do is deploy our workload on their infrastructure to go global within minutes without spending a single penny. *I mean, how cool is that?*

Think about this from an indie developer standpoint. Imagine that I build an augmented reality game like Pokémon Go. With these freemium services, I don't have to worry much about the app's hosting costs during the initial days. By leveraging a cloud service, I can bring my idea to reality, show it to the world, have some initial users, get feedback, and pitch it to potential investors without paying a dime for hosting or infrastructure. *Well, what more can I say?*

This is the whole reason the cloud service model blew up. It provided a way for solo, indie developers to bootstrap their business and get a foothold in the market by just focusing on idea implementation and letting the cloud platform take care of the rest.

Pay as you go pricing model

All the cloud platforms offer a pay as you go pricing model. There are no long-term contracts, hidden charges, or complex licensing of any sort. It's just like we pay for our electricity usage. Pay only for what you use. The consumption of computing resources is charged per minute or even seconds. So, if we wish to power down a service, we are charged only for the time the service is up, we don't have to pay for the entire day or even the next hour. Also, there is no service termination fee involved.

The *pay-as-you-go* model enables a business to adapt to the traction its product gets. There is no overcommitting money for the infrastructure.

Some cloud providers like AWS also enable businesses to buy reserved instances in advance at a much less price than that of on-demand compute instances. Reserved instances are the compute power at a certain availability zone that businesses can buy for a longer period of time. Reserving computing power brings down the hourly instance price significantly.

Cloud providers also provide a volume discount if a service consumes a large amount of computing power on their platform. This brings down the hosting costs even further for the business. These providers keep coming up with innovative

and optimized pricing models to keep businesses happy, onboard, and making good profits at the same time.

These are the economic reasons for using cloud hosting. Now, let's talk about the technical reasons behind picking a cloud host for our application.

Technical reasons

Provisioning the right amount of hardware

No service wants to go down ever, no matter how much traffic load it has to deal with. To handle the traffic bursts, we need to provide the right amount of computing power to tackle the traffic influx at peak times.

When setting up their infrastructure, businesses have to make an estimate of the maximum amount of traffic their platforms could receive, and then they have to go shopping for hardware that can handle that much load. This is technically known as *provisioning*.

In this scenario, businesses have to buy all the hardware upfront, investing quite a sum of money to get their platforms ready to handle the traffic surge.

Here are a few things to consider when a business buys all the hardware upfront:

What if the platform doesn't receive the amount of predicted traffic? What if the traffic falls short by quite an extent?

What if there are just a few days in a month when the traffic really spikes, and the rest of the days go pretty normal?

In this scenario, all the extra servers just sit there waiting for the next traffic spike. This is under-utilization and a waste of resources. To handle spikes for a very small amount of time, the business has to spend quite an amount of money upfront.

Let's consider the opposite scenario.

What if the business blows up right after launch and the platform receives three times the anticipated traffic? How will the modest server fleet handle that amount of traffic?

The servers can't do much. Rather, they have to salute each other and go down with dignity.

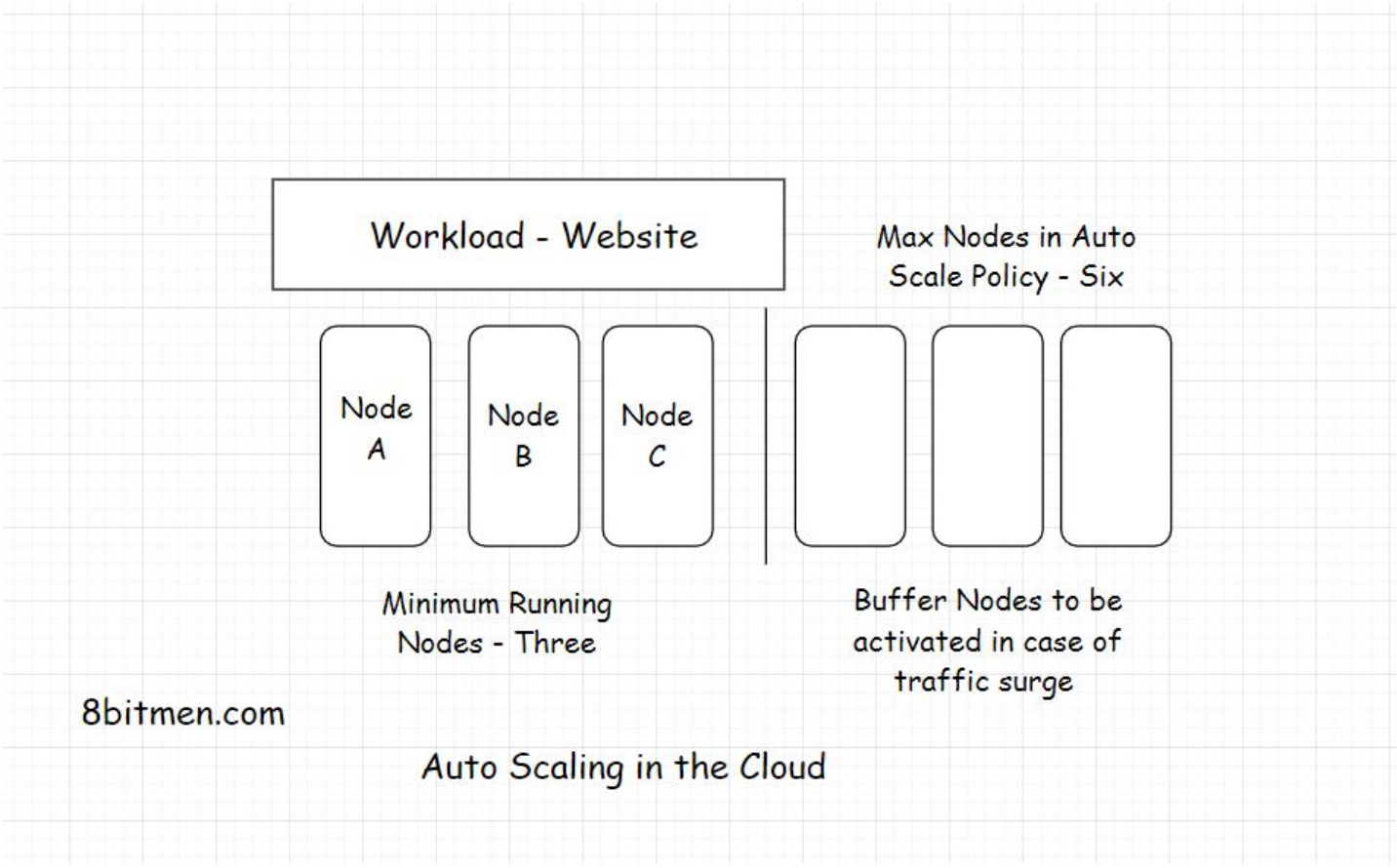
In this situation, the business has to rush to the market to buy new hardware and set everything up quickly, or they will continue to lose customers. On the other hand, they could buy all the hardware upfront like crazy, investing everything they have, anticipating three to five times the estimated traffic.

I ask, what are the odds of a platform getting hit with three to five times the estimated traffic? Unlikely, isn't it?

Therefore, buying so much hardware upfront is not a practical solution for obvious reasons.

Provisioning instances dynamically on cloud

With cloud computing, all these scenarios are easily dealt with without the need to invest any money in hardware upfront. When a workload runs on the cloud, additional server nodes get added and removed on the fly based on the traffic patterns.



In the case of a traffic surge, new servers are deployed dynamically to meet the demand. When the traffic subsides, the additional servers that were added dynamically are powered down and removed from the fleet. All this addition and removal of servers happens automatically on the fly without the need for any kind

of human intervention. Pretty convenient, isn't it?

Why make our lives harder, buying and managing all the hardware upfront, when the cloud makes our lives much easier?

A good real-life example of this is *Pokémon Go*. I brought this up in the first lesson. The Pokémon Go engineering team deployed their augmented reality game on the Google Cloud Platform.

They anticipated a one-time player traffic with a worst-case estimate of five times the anticipated traffic. However, the game blew up right after launch receiving fifty times the anticipated traffic. Google Cloud handled this crazy traffic tsunami, effectively spinning up additional server instances as and when required.

If it wasn't for the cloud, scaling a service with fifty times the expected traffic was just not possible.

Let's continue this discussion in the next lesson where we talk about more technical reasons behind deploying our apps on the cloud.