Background

In the 1964, before the internetwas the internet we could draw all the nodes on the ARPANET on a geographic map

source

by 1982 the network is so crowded the map starts becoming less ledgible so we started clustering things in to sets

source

After the privatization of the network the internet maps start getting lines with curves.

source

Before the privatization there was a single network there may have been multiple links from one node to the other but since they were all operated by the same entity there really wasn't much point in distinguishing them. After the privatization we started to delineate links with curves so that we can overlay different ISPs on the same map without them bleeding in to each other too much.

And that lasted a hot minute before we gave up strated drawing a bunch of eccentric circles to denote the boundaries of the external system. Which look like clouds, so we gave up all together and started drawing clouds to mean, a system that is too big and too dynamic to be meaningfully mapped.

Today when someone refers to the cloud, they maybe referring to the internet, but more often than not a they are probably talking about cloud computing and/or cloud computing platforms like AWS, GCP or Azure.

By Sam Johnston - Created by Sam Johnston CC BY-SA 3.0, Link

Promise of the cloud

The promise of the cloud, or rather cloud computing, is that everything within the boundary of the "cloud" is handled for you.

If a hard drive fails, your cloud provider will replace it. If top of the rack switch fails, your cloud provider will replace it. If the rack is borked, your cloud provider will replace it.

Yes your cloud provider will take care of all these issues for you, with a catch. There will be down-time.

Your cloud provider will migrate you to the working versions of the resources you're paying for, which is great. But they won't be gurantee failover capacity unless you're paying for it.

And that is to be expected, because the cloud is not homogenus.

Your cloud keeps shouting "No Homo"

Microsoft, Google and Amazon are the biggest public cloud providers. In 2016 Gartner estimated Google alone has 2.5 million servers. So it should come as no surprise that a typical Google datacenter employs ~150 people. Those 150~ employees are replacing HDDs, SSDs, Memories, CPUs, switches, racks and stuff as quickly as they are failing.

While non of the big cloud providers speak of these numbers publicly, backblaze does. Accodring to them the drive failure rate is around 0.89%. So let's assume Google, has 2.5 million servers, let's say each server has 24 HDD slots, let's assume all of them are filled with HDDs and not SSDs.

2,500,000 x 24 = 60,000,000 drives in total 60,000,000 x 0.89 = 53,400,000 drive failures per year 53,400,000 ÷ 365 =146,301.369863

If you haven't worked for a Hyperscaler, you might be thinking no way Google destroys 146,301 Hard Disk Drives per day, and you're right, 2,5million servers were from a 2016 estimate, I suspect it'll be more today.

Say Hi to Denise, her full time job is destroying HDDs.

Also here is a bit more information about Google's scale, https://www.youtube.com/watch?v=P_D2LLgI5uI

According to research is only %9 percent of the market. https://www.srgresearch.com/articles/quarterly-cloud-spending-blows-past-30b-incremental-growth-continues-riseso the numbers for AWS and Azure are going to be even higher.

Which is why Google has invested in a robot to destroy hard drives that fail https://www.datacenterknowledge.com/google-alphabet/robots-now-annihilate-hard-drives-google-data-centers

All this to say; at Microsoft, Google and Amazon's scale, procuring the same hardware for millions if not billions of servers at different continents from different vendors is simply impossible.

So the way these cloud platforms standerdise their platforms is by virtualizing your hardware. If software defines the hardware charactheristics it's easier to make every machine have the same ammout of sockets, cores, memory by converging them on the same common denominator.

Multitenancy

Which brings us to multitenancy. Multitenancy is how your cloud provider makes money. You tell them the size of the machine you want, and they allocate it to you.

Almost every talk I gave at Cloudflare with Zach, we used a version of this image, to describe the

overhead of virtual machines.

source: Cloud Computing without Containers

While you can see there is a lot of process overhead with packing virtual machines on the the same machine, since all the VMs all have their own network stack, operating system and so on, that is not your cloud providers problem and all of them do the same exact thing so they are all pretty much on the same footing there.

So while your cloud provider will be able to migrate you on the same machine (if your HDD or the CPU that your process is running on fails), things get iffier when you move to another machine.

A bunch oh years ago I saw Mark Russinovich demo live migrating a VM with a running application to another physical machine and the proof of the demo working was monotonic clock would jump as much as the duration it takes to migrate the VM.

Cloud is not magic Ofcourse your cloud provider can transfer your VM to another DC but it would take a (relatively) long time because they have to copy the entire state of the machine including the VM disk. Given that no cloudprovider is magic and can't beat the speed of light, the further they move you from the original location more you'll notice very subtle ways your application breaks. If you move from one machine to another, the services you used to be 0 hops from will now be 1 hop, if you move to a new rack thats another hop, if you move to a new asile add another hop, if you move to another hall add another hop and so on.

If your cloud provider could throw your VMs around like a small rag dolls, they would, heck they can't even do that for their own services.

You're not Netflix

Unless you work for Netflix you're not doing Active-Active for Multi-Regional Resiliency.

So let's just not pretend that you're using the cloud to it's full potential.

Can you run your own cloud?

YES

Since September I've been running probably one of the buggieststack of software out there with 97.578% uptime.

What I have learned is it's actually easy to approach public cloud level reliability.

But first discuss why this isn't a **99.999%** uptime.

- **Hardware failures** My first and biggest outagewas due to a hardware failuure. As the old addage goes, all software eventually work and all hardware eventually fail. It's a good idea to buy stuff in triplicate or duplicate.
- **Running hot** When the previous error occured I was not able to recover from the incident quickly because a bunch of services assumed 3 node cluster for quorum reasons. In a scenario where a stateful serice can't reach quorum, simply adding a new node doesn't always help you reach quorum again. You need to either revive the machine or bootstrap a new cluster. This is because without a quorum you can't change the cluster state. So it's a good idea to have spares incase hardware fails

and you need to failover.

- **Internet Providers break all the time** as of writing this I live in northwest london if that wasn't obvious. one of the most common reasons of outages were, you guess it, Virgin Media. For hours at a time. So it's a good idea to back that up too.
- **Me** for the most part I didn't touch stuff all that much when they were running but once in a blue moon I'd cycle all the machines at the same time and that cause down time. I am a lot more careful about that.

What went right?

Well Nomad and Proxmox are just great. K8s is just way too complicated and isn't designed as coherently as nomad.

Don't get me wrong, k8s can do many more things than nomad can, but not all of them great. While nomad is a lot better at doing things because it does fewer things.

Commodity hardware is just great. When every I walk by a pawn shop I look for i5-i7 lenovo thikpads or thinkstations to add to my cluster. I don't run huge applications so everything (VM and Containers) have modest size requirements so it's easy to fit them anywhere and move them anywhere.

What went wrong?

I tried running k8s, sank 3 days in to setting it up. What you get out of k8s is the ecosystem, more sophisticated features and networking.

I found that ecosystem (helm charts and other k8s recipes) are very irrelevant to me. The sophisticated features are too. I really wanted to give the networking bit a go but turns out what I wanted to do doesn't work outside the box there either. So K8s' value prop for me isn't there either.

What's next?

Well the thing I'm currently not having fun is assiging IPs to things manually. I am hoping to change this very soon and switch my entire home network to a calico.

We'll see how that goes.