A Case for Technological Diversity

> **Note:** The following is an opinion piece on the perils of relying solely on a single cloud provider, be it AWS or another entity. It was written out of a concern that readers may limit themselves to an AWS-centric attitude and not fully explore what the world of cloud computing has to offer.

The bulk of this book has been focused on explaining fundamental concepts related to cloud computing and the way in which they relate to AWS and AWS-adjacent services, therefore it might not come as a surprise to the reader that I am quite a big fan of AWS.

But I am a fan of AWS in the same way that an alcoholic might be especially fond of vodka. After all society, in its infinite wisdom, would still call that person an alcoholic and not an vodkaholic.

At my core, I find the convenience of spinning up physical infrastructure and using them without having to buy and setup any hardware myself, an ability which cloud computing provides me with to be worth the faustian bargain of having to pay rent and not completely owning or controlling the systems my applications are dependent on. It is therefore, the cloud that I am addicted to and not AWS.

AWS is simply the cloud provider that I (and evidently, 31% of the industry as a whole) am most fond of, it is my poison of choice. And it need not be yours. In fact, the whole concept that a business should attach itself to a single cloud provider, i.e. have a poison of choice, is dumb.

Just as any smart investor diversifies their portfolio in order to hedge against potential catastrophes and not be over-reliant on any single company/industry, good businessmen too should diversify their technology in order to hedge against potential catastrophes and not be beholden to any single entity.

Not to be a snob, but there is a quote from the 1965 novel Dune that I believe most succinctly describes my attitude on the matter:

`"He who can destroy a thing, has real absolute control over it."`

If a person's business can be completely put to halt by a single power outage in an Amazon data center in California, or be made entirely unprofitable by an update in AWS pricing policy, then that person might own the business on paper, but AWS has the real absolute control over it.

There are three major reasons why I believe adopting multiple cloud providers would be prudent. These are enumerated below:-

- Mitigate chances of complete infrastructure breakdown
- Protecting oneself against changes in policy and pricing
- Developing attachment with proprietary services rather than general technologies

Mitigating Infrastructure Breakdown

The most widely used region within AWS is `us-east-1` , located in the area spanning the eastern coast of the United States, concentrated particularly around areas in the north of Virginia. A great deal of very large organizations have a history of being foolishly over-reliant on AWS resources in the region, such that an outage in the `us-east-1` region can halt their entire business operation.

On June 13, 2023 for example, Amazon Web Services (AWS) experienced an incident that impacted a number of services in the US-EAST-1 region. The

incident, which lasted more than 2 hours, was first detected around 18:50 UTC, and affected organizations such as **The New York Metropolitan Transportation Authority** and the **Associated Press**. With the MTA even having to send a tweet apologizing to the residents of New York City because of the incident. Also, curiously, Amazon has still yet to publicly disclose the cause of the outage more than a year after the incident.

Though few and far between, these outages of AWS services are not an uncommon occurrence and especially when running a business which cares about minimizing down-time, it would be wise not to rely on a single cloud provider.

Additionally, though remembering the durability and up-time guarantees given to us by AWS with regards to a particular service is necessary, blindly putting one's trust in them is another matter entirely. For example, the S3 service guarantees us with 99.99% uptime, and for the most part, it does deliver on the promise. But AWS is not expecting it to be available 99.99% of the time, in fact, Amazon expects failures to happen and considers it as part of the process. It just hopes that its customers are smart enough to plan for contingencies, with AWS giving credits that you can use within AWS as compensation for whenever "shit hits the fan" as they say in more cultured parts of the world.

Don't believe me? This was a slide prominently displayed at the AWS re:Invent conference in 2013:

Whenever we create an EC2 instance or spin up a S3 bucket we enter into a Serivce-Level Agreement (SLA) with the Amazon corporation, agreeing to accepting compensation for any downtime incurred due to faults or failures caused by AWS in the form of AWS credits, usually according to the structure laid below:

Monthly Uptime Percentage	Service Credit Percentage
Less than 99.99% but equal to or greater than 99.0%	10% (of the estimated monthly bill)
Less than 99.0% but equal to or greater than 95.0%	30%
Less than 95.0%	100%

The truth of the matter is, AWS is rather component in handling its infrastructure and making sure its operating smoothly but in the rare cases that somehow it is not able to, AWS is also large enough to be able to pay the costs outlined in its SLA.

What AWS is not however, is an entity that will give you 24/7 uptime all day, every day. **Expect things to fail.** And prepare for contingencies. These contingencies usually offer themselves to us in one of two forms, both of which are listed and explored below:

- Multi-AZ and Multi-Region environments
- Multi-cloud environments

Multi-AZ and Multi-Region

It is definitely possible to assure against possible technological disasters whilst limiting oneself to AWS, with the most obvious manner of doing so being the spreading of AWS resources across multiple availability zones and/or multiple regions. This helps ensure high availability and is quite easy to set up. It requires learning no new cloud technologies and if planned properly, very little added cost.

Outages happening at the same time across multiple AWS regions is basically unheard of, especially for regions that are not adjacent to one another geographically. Therefore this is definitely a valid choice for enterprises that wish

to make their existing AWS infrastructure more resilient regardless of their stance on utilizing multiple cloud providers.

Multi-Cloud

If an organization is scared that its infrastructure is over-reliant on a single cloud provider, and wishes to mitigate against infrastructure breakdown, then the logical solution would be to spread the infrastructure and workload across multiple cloud providers. Indeed, according to a survey of over 1500 corporations across the world done by the Oracle corporation, **98 percent** of all business enterprises rely on more than a single cloud provider for the purposes of infrastructure provisioning and management.

This normally means keeping accounts with multiple cloud providers and handling operations across multiple cloud platforms, which does involve a whole lot more work than if an organization were relying solely on a single cloud provider. A popular method of going about this is through the utilization of **Terraform** and **Ansible**, which are both IaC (Infrastructure as Code) tools that allow us to provision and configure infrastructure in the cloud through the use of scripts that are **cloud-agnostic**. Meaning that the same programming language and code structure (sometimes even the same exact lines of code) can be used for AWS, Azure, GCP and 60+ other infrastructure providers. Both tools are liberally used by cloud engineers and DevOps teams alike.

Protection against Policy and Pricing Changes

Another major reason why not to rely on a single cloud provider would be to hedge against potential changes in (A) How a service operates and (B) How a service is priced. For example, if you are running a company that heavily utilizes Amazon's S3, then any change in how S3 is priced or operated could affect you in greatly outsized manners.

This is exactly what Dropbox, one of the largest digital storage companies in the world, realized in 2015 when it decided to move away from AWS (and it never

looked back). This allowed Dropbox to avoid paying the extremely high costs that had recently become customary when dealing with large volumes of data in S3, reportedly saving the company upwards of $75 Million over the next two years.

However, one is much less susceptible to being greatly affected by changes in policy and pricing if their business is technologically diversified and relying on a greater number of cloud providers, each with its own set of policy and pricing standards, greatly reducing the chance of a single change in policy or pricing leading to outsized impact on the business.

Proprietary Services and General Technologies

The final major reason why I believe it to be important not to rely on a single cloud provider is to avoid excessive attachment, comfort and limited expertise with a specific proprietary service rather than a general type of technology leading to a type of cloud administrator who say, realizes how to use S3 but will never bother learning what object storage is and will never explore other cloud providers and object storage services, limiting themselves to an AWS-centric world.

That is not the type of cloud user I wish for the reader to become, after all the shackles of habit are too weak to be felt until they are too strong to be broken. AWS should simply be treated as the on-boarding point for the reader's cloud journey. It provides you with the largest and most popular cloud platform in the world, but it is not however to be conflated with the only cloud platform in the world.

Therefore, in order to aid the reader in broadening their horizon, small tables of the most popular AWS services discussed in the book, the general technology they represent, and their alternatives is provided to the reader, so that they can take the AWS-centric knowledge that they have gained in this book and use it as a jumping off point for more broader cloud endeavors.

Storage Services

AWS Service	General Technology	Azure alternative	GCP alternative	3rd party alternative
Simple Storage Service	Object Storage	Azure Blob Storage	Google Cloud Storage	Storj
Elastic File System	File Storage	Azure Files	Google Filestore	OpenStack Manilla
Elastic Block Store	Block Storage	Azure Disk Storage	Google Persistent Disk	DigitalOcean Volumes
AWS DataSync	Data Transfer	Azure Data Box	Storage Transfer Service	Resilio Connect
Snowball and Snow family	Data Transfer + Edge	Azure Data Box	GCP Transfer Appliance	N/A

Compute Services

AWS Service	General Technology	Azure alternative	GCP alternative	3rd party alternative
EC2	Virtual Machines	Azure Virtual Machines	Google Compute Engine	DigitalOcean Droplets
ECS	Containers	Azure Containers	Google Kubernetes Engine	Docker Swarm
Lambda	FaaS	Azure Functions	Google Cloud Functions	Cloudflare Workers

Database Services

AWS Service	General Technology	Azure alternative	GCP alternative	3rd party alternative
RDS	Managed SQL DB	Azure SQL Databases	Cloud SQL	SQL on Oracle Cloud
DynamoDB	NoSQL DB	CosmosDB	Firestore	MongoDB
Redshift	Data Warehouse	Azure Synapse	BigQuery	Snowflake

Networking Services

AWS Service	General Technology	Azure alternative	GCP alternative	3rd party alternative
CloudFront	Content Delivery Network (CDN)	Azure CDN	Google Cloud CDN	Cloudflare
Elastic Load Balancer	Load Balancers	Azure Load Balancer	Google Cloud Load Balancing	Nginx
API Gateway	API Management	Azure API Management	Google Cloud Endpoints	Kong
Route53	DNS Provider	Azure DNS	Google Cloud DNS	Oracle DNS

Application Integration Services

AWS Service	General Technology	Azure alternative	GCP alternative	3rd party alternative
SQS	Message Queue	Azure Queue Service	Google Pub/Sub	RabbitMQ
SNS	Notification Service	Azure Notification Hub	Google Pub/Sub	Twilio
EventBridge	Event Bus	Azure EventGrid	Google EventArc	Zapier
SES	Email Service	Azure Communication Services	N/A	SendGrid, Mailgun
Kinesis	Real-time Data Streaming	Azure Event Hubs	Google Cloud Dataflow	Apache Kafka
Quicksight	Data Analytics	Power BI	Google Data Studio	Tableau