AWS Solution Architect Training

Module 05
Storage and Databases

Instructor: Tim Platt, Cloud Solution Architect

Simple Storage Service (S3)

Object Storage

Storage re-imagined for the cloud era.

Key Points

- A magical “Bucket” that NEVER fills up. You can upload millions or billions of files – there is no upper limit on the amount of space or number of files (or type of file!). S3 provides Internet scale read/write throughput.
- Each file (aka “object”) must be <= 5 Terabytes (TB!)
- Select a “storage class” to match your data access pattern for cost efficiency – everything from “Frequent Access” to “Glacier Deep Archive”
- Support for:
  - Encryption at rest, Encryption in Transit (HTTPS)
  - Versioning of files (must be enabled)
  - Access: Public (downloadable by anyone - be careful!) or Private (the default)
- SUPERPOWER: Each file gets a unique URL – easy to access
- SUPERPOWER: Your data is redundantly stored in multiple locations within the AZ (One Zone) or REGION (S3 Standard and other classes)

Why is it called Object Storage?

We're not dealing with files, an "object" consists of three things

Key

- Each object is uniquely identified within the bucket by a "Key"
- The S3 service can efficiently and quickly return the "Value" (the file contents) upon request when given the Key
- Nested folders structure by placing one or more "/" in the Key

Value

The file itself. It can be text, json, csv, binary, mp4, iso, jpeg, encrypted data – S3 doesn't care.

Up to 5 Terabytes in size (per file)

Metadata

Metadata (tags) that you can use to annotate and describe the "value"

NOTE: S3 is not a file system (like NTFS, EXT4, or ZFS) nor is it a network file share.

Making an S3 bucket PUBLIC – (Be careful!)

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket-name/*"
        }
    ]
}
```

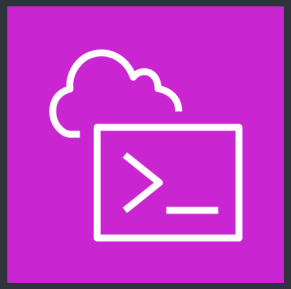
- This policy is a RESOURCE POLICY. It is attached to a resource rather than an IDENTITY. That's why it has a Principal element – that's the "who" that can perform the action(s).
- Make sure Block Public Access is Disabled at the bucket level
- Make sure Block Public Access is Disabled at the account level
- Apply a Bucket Policy to the bucket such as the one shown to the left.

NOTE: Principal of "*" – means ANYONE can download (GetObject)

CloudShell – easiest way to use the AWS CLI

CloudShell

Easiest way to use the AWS CLI (Command Line Interface)



Key Points

- It's a simple linux shell right in your browser – nothing to install and your credentials are ready to go!
- Temporary environment – don't plan to save files here long term (Put them in a bucket!)
- Some examples:

```
aws help
aws sts get-caller-identity
aws s3 ls
aws s3 ls s3://todo-builds-123456789012-dev
aws ec2 describe-instances
```

File Storage Options

Operating Systems like Windows and Linux do not know how to use Object Storage (which is a Web Service). Legacy applications might require ***network file shares*** that can be accessed using traditional protocols

Elastic File System (EFS)

- Network File System (NFS) v4 in the cloud
- Great for Unix and Linux
- NFS protocol

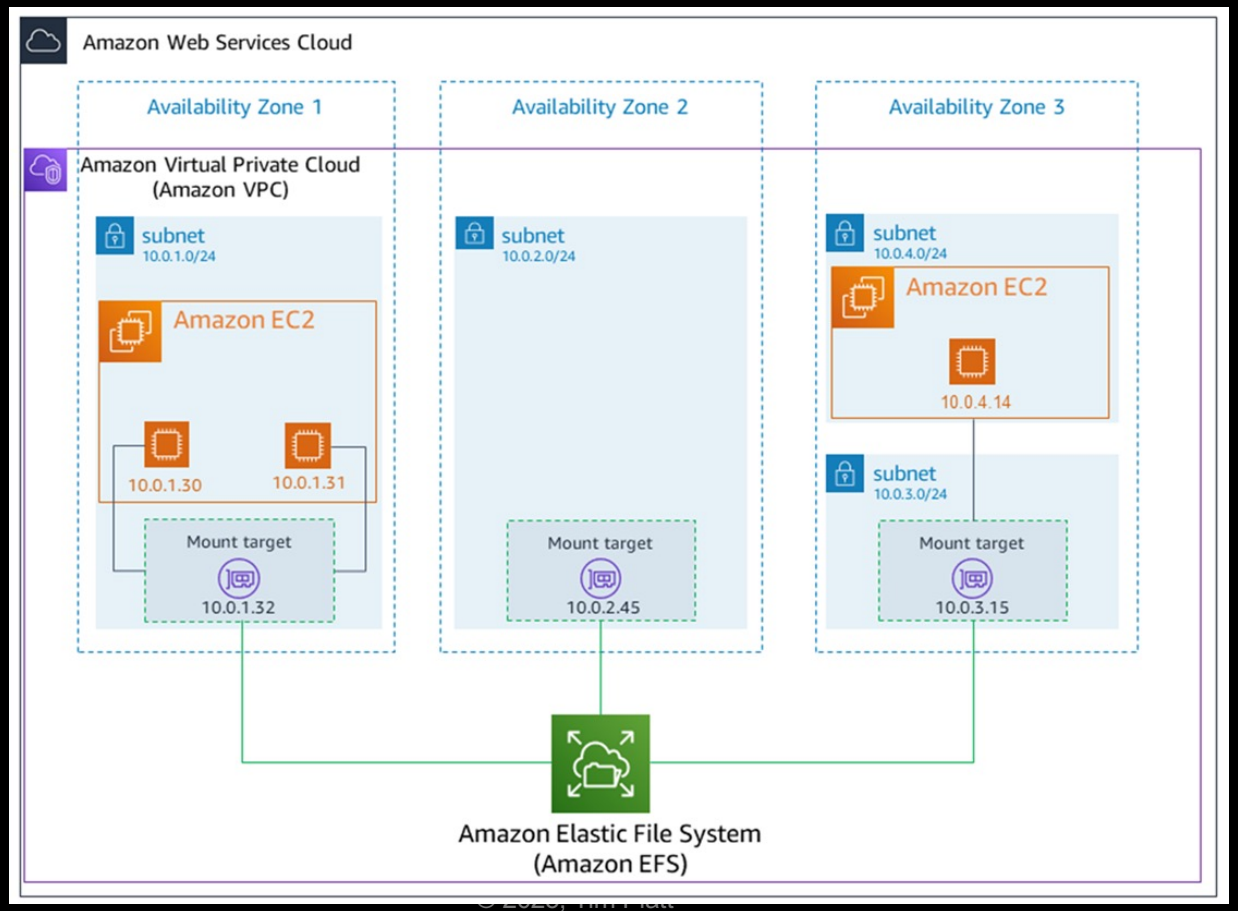
FSx for Windows File Server

- Good ole Windows File Share – in the cloud
- Perfect for Windows servers
- SMB (Server Message Block) protocol

FSx for Lustre

- Linux/Unix alternative for HPC (High Performance Compute)
- Massive read/write throughput

EFS – File System in the Region can be accessed from multiple subnets and SIMULTANEOUSLY by multiple EC2 instances



Relational Database Service (RDS)

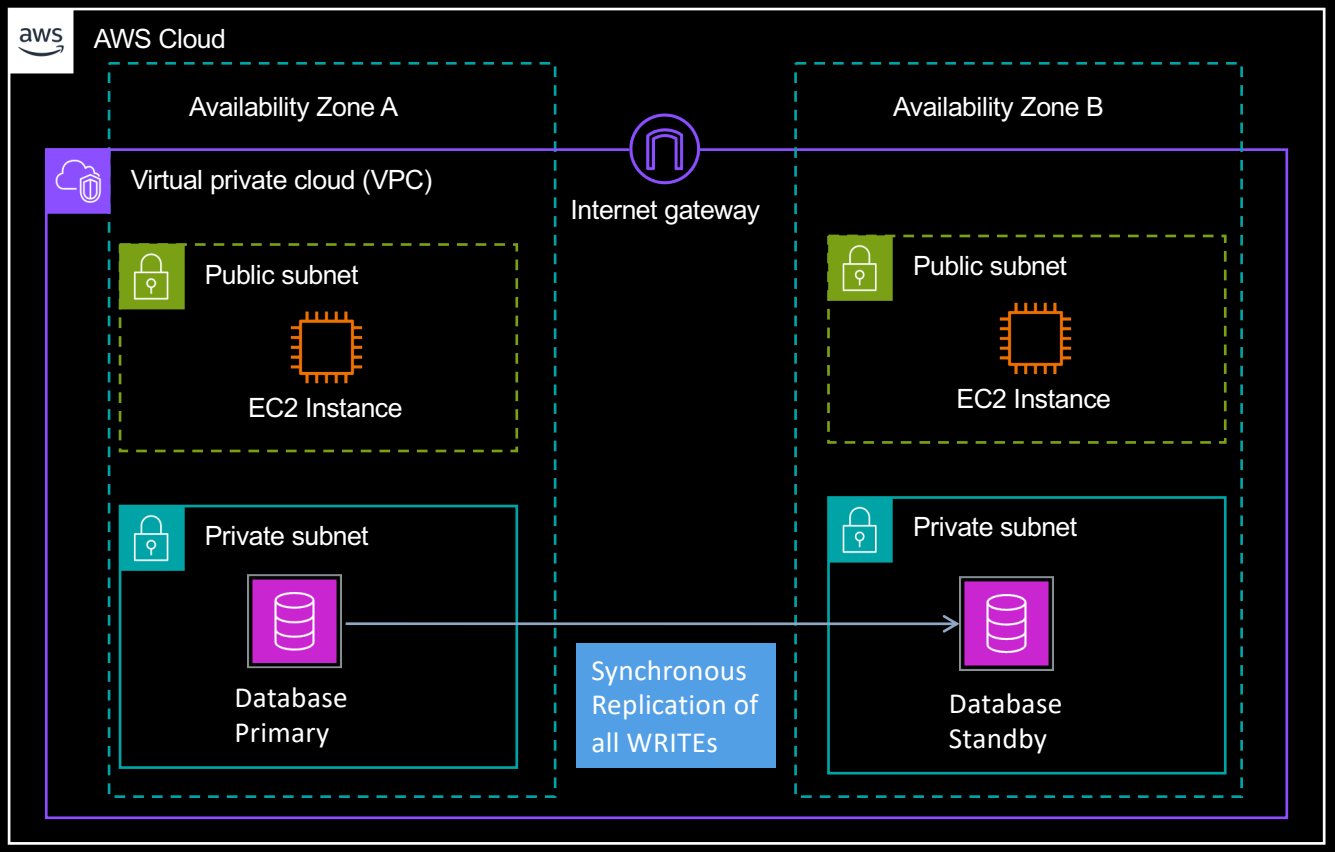
RDS

Relational Databases - MS SQL Server, MySQL, Postgresql, Oracle, Aurora and more.

Key Points

- Database in the cloud – have a multi-node resilient cluster up and running in minutes.
- These are the relational databases that you know and love – tables, indexes, foreign keys, integrity constraints, stored procedures – and access them using SQL queries.
- Support for:
  - Backup (snapshots) and Restore
  - Point in Time Restore
  - Maintenance Windows
  - Version Upgrades
  - Encryption
  - Multi-node clusters with Read Replicas (in other regions if desired)
- SUPERPOWER: RDS Service makes it easy to setup and maintain (upgrades, backups, restores, etc.)
- SUPERPOWER: The Aurora database type is relational database engineered for the cloud era – it's super scalable and high performing at a fraction of the cost of traditional commercial databases.

RDS Database resides in your VPC

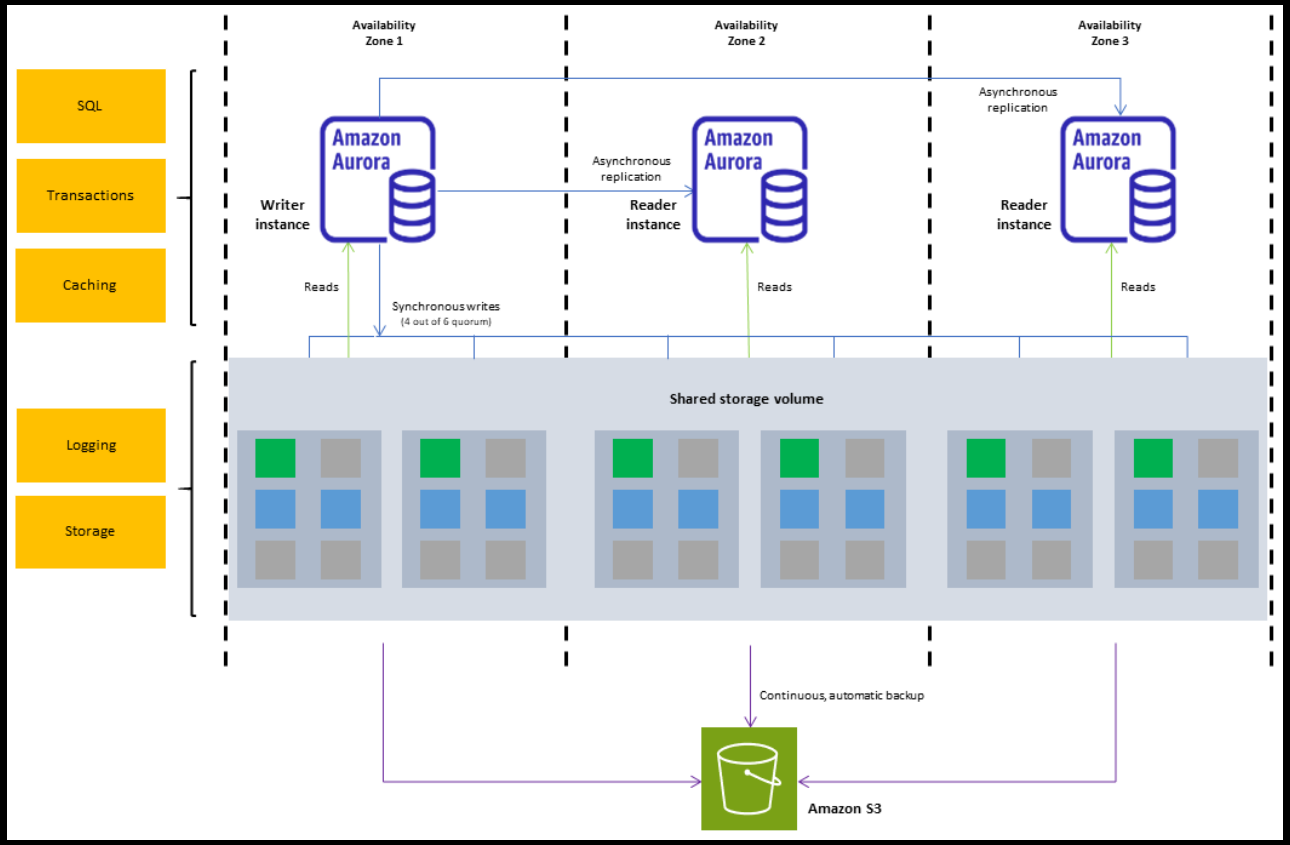


Why?

- Place them in PRIVATE subnets
- Your EC2 instances in other subnets can easily access them (don't forget Network ACLs and SGs!)
- This technology not intended to be accessed directly from the Internet!
- Standby will receive all updates over Synchronous replication (**Multi-AZ DB**)
- Read Replicas will receive updates via Asynchronous replication

NOTE: The Standby is in a DIFFERENT AZ - for availability!

Amazon Aurora – Separation of Compute and Storage for extreme performance and scalability



- Compute is separate from the storage
- Storage spans 3 AZ and 2 copies of the data in each – 6 copies total
- You can have up to 15 Read Replicas for extreme read capacity
- Continuous backups to S3 for PITR – Point In Time Restore capability
- MySQL or Postgresql compatible with your existing application!

DynamoDB

DynamoDB

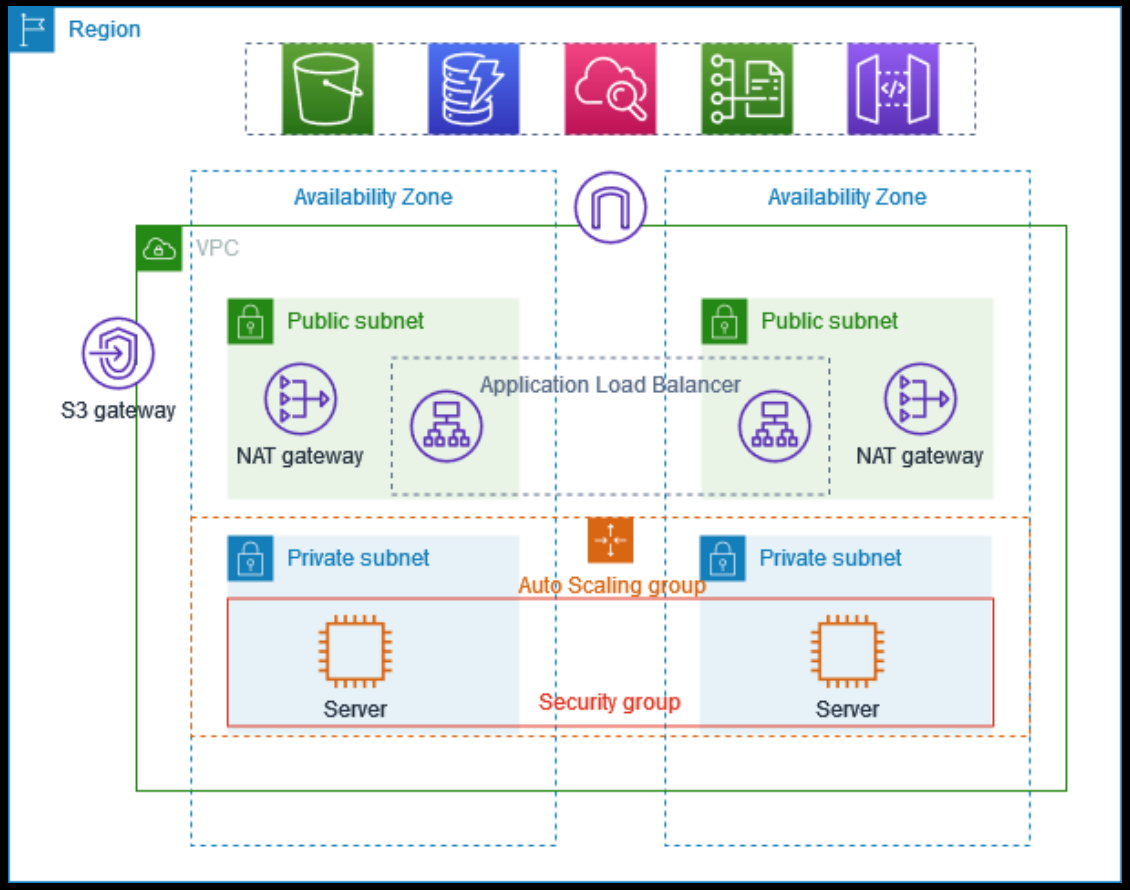
A Non-relational (NoSQL) database – Internet scale and accessed as a WEB SERVICE.

Key Points

- Based on Key/Value access pattern (Does that sound familiar?)
- No upper limit on table size or number of rows (called “Items” in DynamoDB)
- Massive, internet scale read/write throughput (if you need it)
- Significantly different than relational database. You don’t use SQL. DynamoDB has its own Application Programming Interface (API) with GETs, PUTs, QUERIES, SCANS. Accessed over the Internet as a WEB SERVICE.
- Originally invented for the Amazon.com Shopping Cart
- SUPERPOWER: Schema-less database – great for varied, diverse datasets and JSON
- SUPERPOWER: Single digit millisecond latency for GETs and PUTs (single Item access using a key) at any scale
- SUPERPOWER: Managed service using multiple AZs automatically. No upgrades or maintenance to deal with.

NOTE: Due to the extreme differences – migrating from Relational database to DynamoDB is a lot of work

DynamoDB – like S3 and EFS lives in the REGION, not your VPC



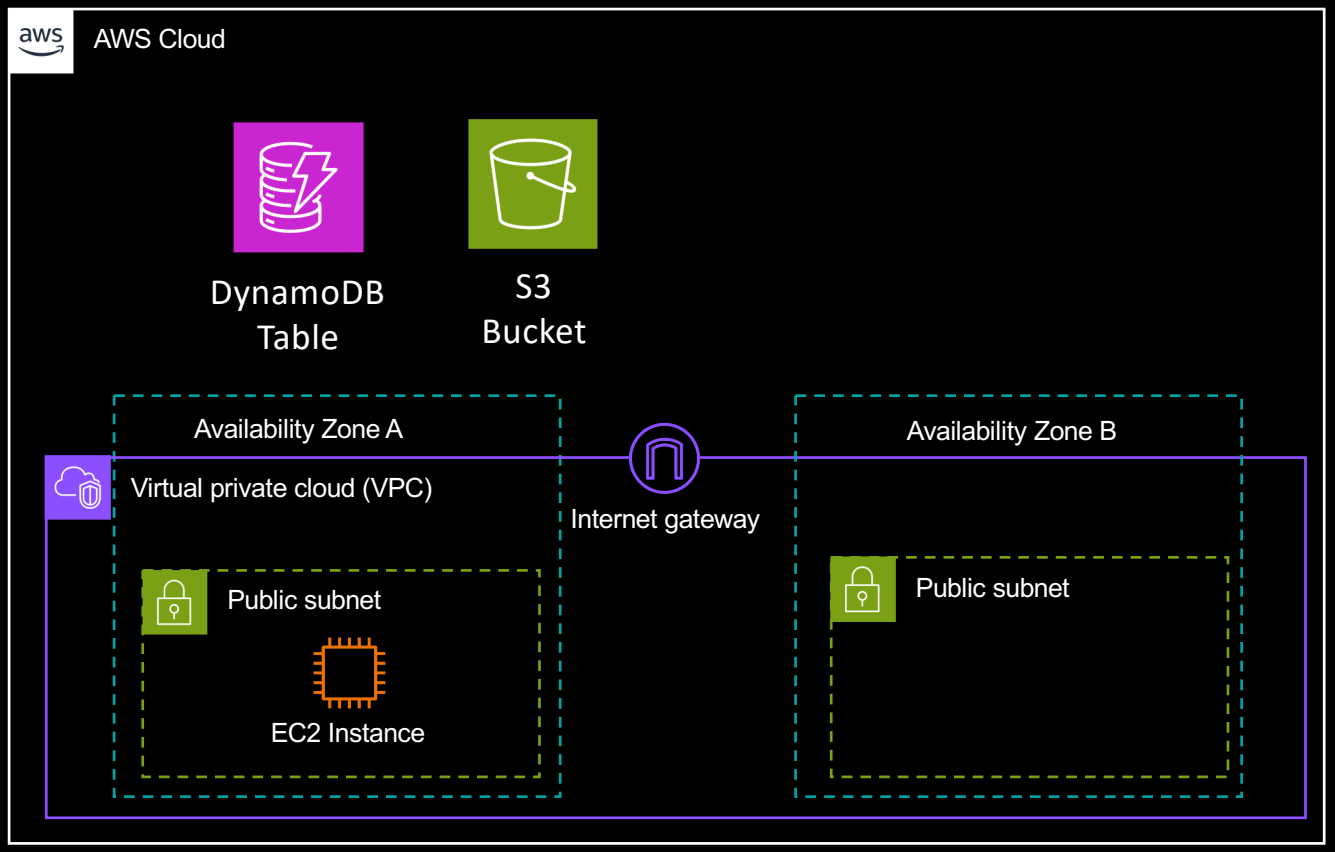
Many services such as DynamoDB, S3, EFS do not reside in your VPC. They reside in the region.

They are web services so you MUST have INTERNET access to connect to them (via Internet Gateway or NAT gateway)

OR

You must use a Gateway or Interface Endpoint to connect to them

Let's use run the To Do App and use DynamoDB for our database and S3 to store CSV exports



Why?

- Simple API – it's a Web Service so it's easy to store our data
- DynamoDB is a Serverless service – no versions, or updates, or patching for us to do!
- DynamoDB tables live in the Region and automatically use multiple three AZs to ensure your data is redundantly stored
- S3 also Serverless – never fills up, easy to use, great place to put files!

- S3 Bucket Policies (Resource Policies attached to an S3 Bucket): https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html
- Relational Database Service (RDS): https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html
- Multi-AZ Database Instance: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZSingleStandby.html
- Amazon Aurora: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Overview.html
- DynamoDB Developer Guide: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html#ddb-characteristics