**AWS CSA-Pro Notes ACloudGuru Section 2 Data Stores**

**Exam Tips:**

**Know when to use various data stores:**

**RDS:**

* Traditional relational data models
* Existing apps requiring RDBMS
* OLTP, ACID compliant
  + RDS is best used when you want to limit your DB admins overhead after migrating to the Cloud.
  + You get various benefits of not having to manage patching, installing the DB, managing the instance and that time you save can go to more important things.

**DynamoDB:**

* High I/O needs
* Scale dynamically
  + Serverless, Key/Value, unpredictable data, NoSQL
  + Offers more High Availability than RDS so from that aspect always think DynamoDB
  + When migrating applications, it can be hard to switch from Relational to Non-Relational because of how the apps are usually structured
  + New apps should always look to be serverless first for the ease of management, cost-effective perspective, and the better availability and fault tolerance

**S3:**

* BLOBs
  + Static websites, Data lakes, Logs
  + Object files

**EC2:**

* Database not supported under RDS
* Need complete control over data
  + If you have questions on database types not supported in AWS EC2 is your fall back for any database types
  + Also if you ever complete control

**Redshift:**

* OLAP, Datawarehouse
  + High read databases that may analyze data from data lakes

**Concepts:**

**Data Persistence:**

**Persistent Data Store:**

* Data is durable and sticks around after reboots, restarts, or power cycles.
* Glacier, RDS, S3, DynamoDB, EBS

**Transient Data Store:**

* Data is just temporarily stored and passed along to another process or persistent store
* SQS, SNS, Kinesis

**Ephemeral Data Store:**

* Data is lost when stopped
* EC2 Instance Store, Memcached

**IOPS VS. Throughput:**

**IOPS:**

* Input/Output Operations Per Second
* Measures of how fast we can read and write to a device
  + Sport car that moves fast (or slow)

**Throughput:**

* Measure of how much data can be moved at a time
  + Truck that carries a lot (or a little) per trip

**Consistency Models – ACID & BASE:**

**A**tomic – Transactions are “all or nothing”

**C**onsistent – Transactions must be valid

**I**solated – Transactions can’t mess with one another

**D**urable – Completed transactions must stick around

**Ba**sic availability – Values available even if stale

**S**oft-state – Might not be instantly consistent across stores

**E**ventual Consistency – Will achieve consistency at some point

ACID models do not scale very well (Relational databases)

BASE models is not inconsistent, but rather accommodates for changes in a more parallel manner (DynamoDB, S3)

**S3:**

* S3 is an Object Store
* Used in other AWS services – directly and behind-the-scenes
* Max object size is 5TB; Largest object in a single PUT operation is 5 GB
* Recommended to use multi-part uploads if larger than 100MB for a PUT

The S3 pathing is not a file path, but rather a key. It looks like a file path.



**S3 Consistency:**

* S3 provides read-after-write consistency for PUTs of new objects
* HEAD or GET requests of the key before an object exists will result in eventual consistency
* S3 offers eventual consistency for overwrite PUTs and DELETEs
* Updates to a single key are atomic

Atomic means only one person can update an object at a time. If 2 requests to updates an object are made, they are processed in order of timestamp and you will see the update as soon as it is replicated elsewhere.

**GET-** Getting an object in S3

**PUT-** Placing an object in S3

**DELETE-** Deleting an object out of S3

**HEAD-** Returns the objects metadata without returning the actual object

**S3 Security:**

* Resource-based (Object ACL, Bucket Policy)
* User-based (IAM Policies)
* MFA Delete

**S3 Data Protection:**

**Versioning:**

* You get a new version of your object with each write
* This enables “roll-back” and “un-delete” capabilities
* Old versions count as billable size until they are permanently deleted
* Integrated with lifecycle management to keep costs low and clean up old versions

**MFA:**

* Safeguard against accidental deletion of an object
* Change the versioning state of your bucket

**Cross-Region Replication:**

* Security
* Compliance
* Latency

**S3 Lifecycle Management:**

* Optimize storage costs
* Adhere to data retention policies
* Keep S3 volumes well-maintained

**S3 Analytics:**

**Data Lake Concept:**

* Athena, Redshift Spectrum, QuickSight

**IoT Streaming Data Repository:**

* Kinesis Firehose can spool directly to S3 Buckets

**Machine Learning and AI Storage:**

* Rekognition, Lex, MXNet, can use S3 as training sets or ingest for processing

**Storage Class Analysis:**

* S3 Management Analytics

**S3 Encryption at REST:**

* SSE-S3 – Use S3’s existing encryption key for AES-256
* SSE-KMS – Use a key generated and managed by AWS KMS
* SSE-C – Upload your own AES-256 encryption key which S3 will use when it writes the objects
* Client-Side Encryption – Encrypt objects using your own local encryption process before uploading to S3

**Other S3 Features:**

**Transfer Acceleration-** Speed up data uploads using CloudFront in reverse

**Requester Pays-** The requester rather than the bucket owner pays for requests and data transfers.

Requester needs their own AWS Account.

**Tags-** Assign tags to objects for use in costing, billing, security, etc.

**Events-** Trigger notification to SNS, SQS, or Lambda when certain events happen in your bucket

**Static Website Hosting-** Simple and massively scalable static website hosting

**BitTorrent-** Use the BitTorrent protocol to retrieve any publicly available object by automatically generating a .torrent file

**Glacier:**

* Cheap, slow to respond, seldom accessed
* Cold Storage
* Used by AWS Storage Gateway VTL
* Integrated with AWS S3 via Lifecycle Management
* Faster retrieval speed options if you pay more

**Glacier Concepts:**

**Glacier Vault-** Like an S3 Bucket

**Glacier Archive-** S3 Object

**Policies:**

**Glacier Vault Lock-** Defines what rules the vault most obey by.

It may enforce a rule like no deletes or MFA

Immutable

**IAM Policies-** To provide access to Glacier. Admin the vault.

**Archive Concepts:**

* Immutable
* File, Zip, Tar, etc.
* Max size 40TB

**How Vault Locks Work:**

1. You first create your Vault Lock and initiate it onto your Glacier Vault of choice.
2. You then have 24 hours to decide if it is working correctly.
3. If 24 hours runs out then it goes away
4. If you do lock it you can initiate the complete Vault Lock command.
5. You can bypass the 24 hour period by doing the Abort Vault Lock command.

**Elastic Block Storage:**

* Think “virtual hard drives”
* Can only be used with EC2
* Tied to a single AZ
* Variety of Optimized choices for IOPS, Throughput, and cost
* Snapshots are great
* Pay for all blocks allocated whether in use or not

**EBS VS. Instance Stores:**

**Instance Stores:**

* Temporary
* Ideal for caches, buffers, work areas
* Data goes away when EC2 is stopped or terminated
* Locked to a specific EC2 instances
* Better performance; Direct attached storage

**EBS:**

* Not locked to Instances. Can attach and detach them
* Create snapshots
* Rides over the network of AWS

**EBS Snapshots:**

* Cost-effective and easy backup strategy
* Share data sets with other users or accounts; Copy snapshots to other accounts or regions
* Migrate a system to a new AZ or Region
* Convert unencrypted volume to an encrypted volume

Snapshots are incremental.

First snapshots contain the whole of your data. When you add data and take another snapshot the next snapshot will only record the newly put data.

**Elastic File System:**

* Implementation of the NFS file share
* Elastic storage capacity, and pay for only what you use (unlike EBS)
* EFS is more expensive
* Multi-AZ metadata and data storage
* Configure mount-points in one, or many, AZs
* Can be mounted from on-premises systems only by using Direct Connect

**Storage Gateway:**

* Virtual machines that you run on-premises with VMWare or HyperV
* Provides local storage resources backed by AWS S3 and Glacier
* Often used in DR preparedness to sync to AWS
* Useful in Cloud Migrations

**Different Versions:**

**File Gateway-** Exposes files as NFS or SMB shares. Allows on-prem or EC2 instances to store object in S3 via NFS or SMB mount points.

**Gateway-Stored Volumes-** Uses iSCSI. Async replication of on-prem data to S3.

**Gateway-Stored Cached-** Uses iSCSI. Primary data stored in S3 with FA data cached locally on-prem

**Tape Gateway-** Uses iSCSI. Virtual media changer and tape library for use with existing backup software.

**Amazon Workdocs:**

* Secure, fully managed file collaboration service
* Can integrate with AD for SSO
* Web, mobile and native clients (no linux clients)
* HIPPA, PCI DSS, and ISO compliance requirements
* Available SDK for creating complementary apps

**Database on EC2:**

* Run any database with full control and ultimate flexibility
* Must manage everything like backups, redundancy, patching, scale
* Good option if you require a database no yet supported by RDS such as IBM DB2 or SAP HANA
* Good option if it is not feasible for you to migrate to an AWS database at the time

**RDS:**

* Managed database option for MySQL, Maria, PostgreSQL, MSS, Oracle, and MySQL-compatible Aurora
* Best for structured, relational data store needs
* Aims to be drop-in replacement for existing on-prem instances of same databases
* Automated backups and patching in customer-defined maintenance windows
* Push-button scaling, replication, and redundancy

**Amazon RDS Anti-Patterns:**

* Lots of large binary objects (BLOBS) = S3
* Automated scalability = DynamoDB
* Named/Value data structures = DynamoDB
* Data is not well structured/unpredictable = DynamoDB
* Other DB platforms not supported= EC2
* Complete control over the DB = EC2

InnoDB is the only way to support replication.

Multi-AZ replication is synchronous

Read Replicas replication is asynchronous

If the master AZ fails and you have Multi-AZ then the stand-by assumes the role of the master.

If a whole region dies and you have a Read Replica in a separate region you need to first promote it to it’s own stand alone database then make it a multi-az deployment.

**DynamoDB:**

* Managed, multi-AZ NoSQL data store with Cross-Region Replication option
* Defaults to eventual consistency reads, but can use strongly consistent reads
* Priced on throughput, rather than compute
* Provision read and write capacity in anticipation of need
* Autoscale capacity adjusts per configured min/max levels
* On-Demand capacity for flexible capacity at a small premium cost
* Achieve ACID compliance with DynamoDB Transactions

**Relational vs NoSQL:**

**Relational:**

* Best when the data is structured well
* Tables and fields

**Non-Relational:**

* Excel with managing name/value pairs
* Name/value pairs are known as an attribute
* Item is the whole collection of name/value pairs
* Table is all the Items put together

**Key Types:**

**Primary Key:**

* A key that uniquely defines an Item
* Must be unique
* Hash attribute

**Partition Key and Sort Key:**

* Have 2 keys
* Partition Key does not have to be unique
* Sort key needs to be unique
* Sort key can also be called a Range key

**Secondary Indexes:**

**Global Secondary Index:**

* Partition key and sort key can be different from those on the table
* It is not restricted to just the partitioning set forth by the partition key

**Local Secondary Index:**

* Same partition key as the table, but a different sort key
* It has to stay local to the table

**Limits:**

* Max 5 of each global and local secondary indexes
* Max 20 attributes across all indexes
* Indexes take up storage space

**Redshift:**

* Fully managed, clustered peta-byte scale data warehouse
* Extremely cost-effective as compared to some other on-prem data warehouse platforms
* PostgreSQL compatible with JDBC and ODBC drivers available; compatible with most BI tools out of the box
* Features parallel processing and columnar data stores which are optimized for complex queries
* Option to query directly from data files on S3 called Redshift Spectrum

**Data Lake:**

* Query raw data without extensive pre-processing
* Lessen time from data collection to data value
* Identify correlations between disparate data sets

Use an S3 bucket to dump a lot of data into and use it as your data lake:

* Transaction Logs
* Sensor Readings
* Social Media Stream
* Weather Data

Then we can use Amazon Redshift Spectrum to query the S3 bucket.

Lastly, we can send the results to an Analytics Tool like Amazon Quicksights

**Neptune:**

* Fully-managed graph databases
* Supports open graph APIs for both Gremlin and SPARQL
* Optimized for relationships
  + Social networks
  + Product recommendation engines

Store interrelationships and query them in a very effective manner.

**Amazon Elasticache:**

* Fully managed implementations of 2 popular in-memory data stores – Redis and Memcached
* Push-button scalability for memory, writes, and reads
* In Memory key/value store – not persistent in the traditional sense
* Really fast, faster than DynamoDB
* Billed by node size and hours of usage

**Use Cases for Elasticache:**

**Web Session Store:**

* In cases with load-balanced web servers, store web session info in Redis so if a server is lost, the session info is not lost and another web server can pick-up

**Database Caching:**

* Use Memcached in front of AWS RDS to cache popular queries to offload work from RDS and return results faster to users

**Leaderboards:**

* Use Redis to provide a live leaderboard for millions of users of your mobile app

**Streaming Data Dashboards:**

* Provide a landing spot for streaming sensor data on the factory floor, providing live real-time dashboard display

**Memcached vs Redis:**

**Memcached:**

* Simple, no-frills, straight forward
* You need to scale out and in as demand changes
* You need to run multiple CPU cores and threads
* You need to cache objects like DB queries

**Redis:**

* You need encryption or HIPPA compliance
* Support for clustering
* Complex data types
* HA and Backup and Restore
* Pub/Sub capability
* Geospacial indexing

**Amazon Database Options:**

**Database on EC2:**

* Preferred database is not available under RDS
* Ultimate control over database

**Amazon RDS:**

* Need traditional relational database for OLTP
* Your data is well-formed and structured

**DynamoDB:**

* Name/Value pair data or unpredictable data structure
* In-memory performance with persistence

**Redshift:**

* Massive amounts of data
* Primarily OLAP workloads

**Neptune:**

Relationships between objects is a major portion of data value

**Elasticache:**

* Fast temporary storage for small amounts of data
* Highly volatile data

**Extra Notes:**

In storage which consistency model values availability over consistency. What about consistency over availability? Which AWS services align with each model?

BASE values availability over consistency

* S3
* Dynamo DB

ACID values consistency over availability

* RDS
* Redshift

How do you setup EFS to multi-az?

1. Create EFS mount targets in each AZ.
2. Configure each EC2 instance to mount to the common mount targets FQDN

What is the main benefit of using an Instance Store?

It gives you faster overall disk I/O because of the locality of where it is.

EBS needs to go over the network of AWS while Instance stores are attached locally to your EC2 instances.

This is also why Instance Stores must stay attached to their original instances and EBS can be transferred between instances.

What is DAX? What are the 3 main scenarios you would use it in?

1. Use it as an In-Memory cache to reduce the response times of eventually-consistent read workloads from single digit millisecond to microseconds.
2. Reduces operational and application complexity by providing a managed service that is API compatible with DynamoDB.
3. Read heavy or bursty workloads, DAX provides increased throughput and potential operational cost savings by reducing the need to over-provision read capacity units.

What is the main disadvantage of using requester pays?

Each user needs their own AWS account to access your S3 objects when you turn on Requester pays.

What is AWS Glue? When should you use it?

AWS glue is a fully managed ETL (Extract, Transform, and Load) service that makes it simple and cost effective to categorize your data, clean it, enrich it, and move it reliably between various data stores.

You should use it to:

* Build a Datawarehouse to organize, cleanse, validate, and format data
* Run serverless queries against your S3 data lake using Athena and Redshift Spectrum
* Event-Driven ETL Pipelines
* Understand data assets better

What is the AWS Glue Data Catalog?

It is your persistent metadata store.

It is a managed service that lets you store, annotate, and share metadata in the AWS Cloud.

What is an AWS Glue Crawler?

It allows you to scan data in all kinds of repositories, classify it, extract schema info from it, and store the metadata automatically into the AWS Glue Catalog.

What is Amazon Quicksights?

A business analytics service you can use to build visualizations, perform ad hoc analysis, and get business insights from your data.