

Domain 1: Resilient architectures

Multi-tier solutions

Exam item will require you to understand and implement several aspects across topics.

Multi-tier solutions refers a general framework that divides up independently scalable applications components, that can be independently developed, managed, and maintained from each other.

Access pattern

Refers to what kind of pattern does the user do to access data? Does it have a peak at certain days, and it everyday.

High availability vs fault tolerance

They do share overlaps in concepts but they vary slightly.

High availability: Design for a minimum downtime, reducing the negative impact on the end user (the clients who are using the applications), by focusing on restoring the important services when a component or application fails.

Fault tolerance: Design an architecture that have zero downtime and service interruption, but much higher cost of operation due to replication and redundancy.

So basically fault tolerance means that if one part of the architecture failed, there is immediately another part that kicks in to replace that failed part. While, high availability just means you are trying to reduce downtime, minimizing the impact on the user when you are trying to restore the failed application.

Single point of failure

To assess single point of failure, it is helpful to think backwards from the point of failure. Work backward, see what happens if one of the web server failed, the database failed, and so on until you see the single point of failure

Disaster recovery objective

Recovery time object: Measures how quickly after an outage an application must be available again

Recovery point objective: Refers to how much data loss your application can tolerate. How old can the data be when this application is recovered? Is it 5 seconds before the disaster? 3 seconds? The lower the time, the better.

The lower the better for both of these objective

Data loss is measured from most recent backup (your recovery point) to the point of disaster. Down

Disaster recovery strategy

Active/passive: Creates an environment that is not intended to be live for IT production until a disaster occurs. This have large initial cost savings on the architecture.

It is further divided into different kind of active/passive strategy:

1. Backup & restore: RPO/RTO in hours. Lower priority use cases, supplying the AWS resources after disaster, and restore a backup prior to the event. Cost = \$
2. Pilot light: RPO/RTO in 10s of minutes. Provision some AWS resources after the event then scale accordingly. Cost = \$\$
3. Warm standby: RPO/RTO in minutes. A **scaled** down version of the business critical application and is always running can be run into a different region. Then scale up the AWS resources after the event have occurred. Cost = \$\$\$

Active/active: Deployment of a second identical live architecture that continually replicates with the first site. However, it has high cost and the requirement for high bandwidth with low latency.

Both have ideal use cases, and don't have to be used independently

Decoupling mechanisms

Refers to components remaining autonomous and unaware of each other as they complete their work as part of a larger system. Essentially, break down your architecture into components that can function completely independently of each other.

Synchronous decoupling

Involves at least two components, both must always be available in order for things to function properly

Asynchronous decoupling

Decoupling involves communication between components through more durable components.

Ex: Multiple instances that are processing messages, the messages are stored into a queue, and if the instances are down the messages are persisted inside the queue until the instances are restored.

Queues

AWS queue services are usually used as a decoupling mechanism to serve as the durable component for communication between two independent components.

AWS standard queue is best-effort ordering that ensures messages are generally delivered in the same order as they are sent, but occasionally say more than one copy of a message might be delivered out of order.

AWS FIFO queue offer first-in-first-out delivery and exactly once processing. Meaning it is a true queue.

Resilient storage

To understand how to pick a resilient storage you need to go through the following process

1. Define the strategy to ensure durability of your data

Know how different storage handle durability and what scenario each storage is suitable for, when to use EC2 store? Amazon S3 glacier?

Knowing this will allow you to pick the correct solution.

2. Define how data service consistency will affect the operation

Know how the data consistency are done, since it impacts how you retrieve the data and the type of data that you get back.

Also you also need to know the functionality of each storage service. If you have a read-heavy load use this, if you deal with global vs single IP range use that. Knowing access pattern will let you pick the best services

3. Know how to implement across a variety of architectures, including hybrid or non-cloud native applications

Know that there are other services outside of AWS being incorporated into the architecture.

Revision #1

Created 4 January 2023 21:06:12 by Tamarine

Updated 6 February 2023 18:24:02 by Tamarine