Module 2: Introducing Cloud Architectting

Tuesday, September 23, 2025 8:28 PM

A. Learning Outcomes
- Define cloud architecture
- Describe how to design and evaluate architectures using the AWS Well-Architected Framwork
- Explain the best pratice for building solutions on AWS
- Describe how to make informed decisions about where to place AWS resourse

B Cloud Architecture
- Cloud architecture is the ***practice of applying cloud characteristics to a solution*** that uses ***cloud services and features to meet an organization's technical needs and business use cases***

- Role of a cloud architect:
  + Plan:
    * Set technical business strategy with business leads
    * Analyze solution for business needs and requirements
  + Research:
    * Investiagte cloud services sepcifications and workload requirements
    * Review existing workload architectures
    * Design prototype solutions
  + Build:
    * Design the transformation roadmap with milestones, work streams, and owners
    * Manage the adoption and migration

- The role of a cloud architect is as follows:
  + Engage with decision makers to ***identify the business goals and the capabilities that need improvement.***
  + Ensure alignment between the technology deliverables of a solution and the business goals.
  + Work with the delivery teams that are implementing the solution to ensure that the technology features are appropriate.

- Cloud architects are responsible for managing an organization's cloud computing architecture. They have in-depth knowledge of the architectural principles and services used to do the following:
  + Develop the technical cloud strategy based on business needs.
  + Assist with cloud migration efforts.
  + Review workload requirements.
  + Provide guidance about how to address high-risk issues

=> User can use AWS services to create highly available, scalable, and reliable architectures.
=> Cloud architects are responsible for managing an organization's cloud computing architecture.

C AWS Well-Architecture Framework
- Six pillars of Well-Architecture Framework including Operational Exellence, Security, Reliability, Performance Efficiency, Cost Optimization, and Sustainability.

1. Operational Excellence Pillar:
- **Support development** and **run workloads effectively**
- Gain **insight into workload operations**
- Continuously **improve processes and procedures** to **deliver business value**
- Besy practices:
  + Perform opeartions as code
  + Make frequent, small, reversiable changes
  + Refine operations procedures frequently
  + Anticipate failure
  + Learn from all operational failures

2. Security Pillar:
- **Protect data, systems, and assets** to take advantage of cloud technologies to improve user's security
- Best practices:
  + Implement a strong identity foundation
  + Enable traceability
  + Apply security at all layers
  + Automate security best practices
  + Protect data in transit and at rest
  + Keep people away from data
  + Prepare ior security events

3. Reliability Pillar:
- **Ensuring a workload can perform its intended function** correctly and consistently when it's expected to
- This includes the ability to operate and test the workload through its total lifecycle
- Best practices:
  + Automatically revocer from failure

+ Test recovery procedures
+ Scale horizontally to increase aggregate workload availability
+ Stop guessinf capacity
+ Manage change in automation

4. Performance Efficiency Pillar:

- The ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve.
- Best practices;
  + Go global in minutes
  + Democratize advanced technologies
  + Use serverless architectures
  + Experiment more often
  + Consider mechanical sympathy

5. Cost Optimization Pillar:

- The ability to run systems to deliver business value at the lowest price point
- Best practices:
  + Implement Cloud Financial Management
  + Adpot a consumption model
  + Measure overall efficiency
  + Stop spending money on undifferentiated heavy lifting
  + Analyze and attribute expenditure

D. Best practices for building solutions on AWS

1.. Desing trade-offs

- Evaluate trade-offs as user can select an optimal approach
- Characteristics:
  + Trade consistency, durability, and space for time and latency => High performance
  + Prioirize speed over cost
- Base design decisions on emprical data

2.. Implementing Scalability

- **Proactive and Quick Scaling:** AWS allows you to quickly and proactively scale your infrastructure to meet demand.
- **Scalability at Every Layer:** It's important to design your entire infrastructure to be scalable.
- **Anticipating Need:** Scalability helps user's anticipate the need for more capacity and deliver it before your system is overwhelmed.
- **Automated Scaling with CloudWatch and EC2 Auto Scaling:**
  + **Amazon CloudWatch** is a monitoring solution used to detect when resource usage, like **CPU utilization**, reaches a certain threshold (e.g., staying above 60% for more than 5 minutes).
  + When a CloudWatch alarm is triggered, **Amazon EC2 Auto Scaling** automatically launches new instances.
  + This ensures new capacity is available before it's needed, providing a seamless user experience.
- **Elasticity:** The system should also be **elastic**, meaning it can scale down when demand decreases. This helps you avoid paying for unused resources.
- **Reactive vs. Proactive Scaling:** The text contrasts automated, proactive scaling with manual, reactive scaling.
  + **Manual scaling** is inefficient because users are blocked from accessing the application when servers reach full capacity.
  + Administrators then have to manually launch new instances, and the startup time for these instances means users can't access the application for a period of time. This increases the downtime.

3.. Automating user environment

- **AWS provides built-in tools:** AWS offers monitoring and automation tools at nearly every level of your infrastructure.
- **Automation is crucial for quick response:** These tools allow your infrastructure to respond quickly to changes and failures without manual intervention.
- **Manual response is inefficient:** Without these tools, you have to manually detect and respond to failures, which involves an administrator manually launching and configuring a new server.
- **CloudWatch and EC2 Auto Scaling for failure recovery:**
  + **CloudWatch** can detect unhealthy resources, such as a crashed application server.
  + **EC2 Auto Scaling** can automate the launch of a replacement server.
- **Automated process:** The tools can automatically detect a failure, launch and configure a new server, and notify an administrator.
- **Logging and tracking:** Changes are logged to a change management solution for tracking purposes.

4.. Using IaC (Infrastructure as Code)

- **Rapid Deployment of Duplicate Environments**: IaC allows user to use a single template to quickly and consistently deploy identical environments, eliminating repetitive manual tasks.
- **Reduced Configuration Errors**: By replacing manual processes with code, IaC significantly reduces human error. If an error occurs due to a code update, you can easily roll back to a previous stable configuration.
- **Consistent Change Propagation**: IaC enables you to make a change in a single template and apply it consistently across all your infrastructure stacks. This ensures uniformity and reliability in deployments.

5. Treating resources as disposable

- **Hardware Mindset:** In the traditional hardware model, you over-provision resources to prepare for future demand. This is expensive and inflexible because upgrades are difficult due to the sunk cost of the hardware.
- **Disposable Mindset:** In the cloud, resources are treated as temporary and interchangeable. This makes it easy to migrate, respond to capacity changes, and upgrade applications and underlying software. You don't get locked in by physical assets.

- **Flexibility and Agility:** This practice allows for quick and efficient management of your infrastructure, promoting flexibility and agility.

6. Using loosely compled components

- **Tightly Coupled Infrastructure:** In a tightly coupled system:
  + Servers are integrated in a chain, each with a specific purpose.
  + The **failure of one component can lead to a complete system failure.**
  + Scaling is difficult and complex, as adding or removing servers at one layer requires changes to every server in connected layers.

- **Loosely Coupled Infrastructure**
  + Intermediaries: This design uses managed solutions, such as load balancers and message queues, to act as intermediaries between layers.
  + Failure Handling: These intermediaries automatically handle failures. For example, a load balancer (like Elastic Load Balancing or ELB) will automatically stop sending traffic to an unhealthy server and redirect it to the healthy ones.
  + Simplified Scaling: The intermediaries also manage the scaling of components, making it easier to add or remove servers without affecting other layers.
  + Example: The provided example of an ELB shows how it routes requests between web and application servers. If one application server fails, the ELB simply routes traffic to the remaining healthy servers, preventing system disruption.

7. Designing services, not servers

- **Consider all options**: While EC2 provides great flexibility, other options like **containers** or **serverless solutions** might be more suitable for your specific needs. It's important to choose the right tool for the job.
- **Focus on the service**: With AWS serverless and managed services, you can focus on the application logic and service itself, rather than the underlying infrastructure. You don't need to provision, configure, or manage entire EC2 instances.
- **Benefits of managed services**: Managed services are often more performant, cost-effective, and have a lower operational profile than server-based solutions. Examples include AWS Lambda, Amazon SQS, Amazon DynamoDB, Elastic Load Balancing (ELB), Amazon Simple Email Service (SES), and Amazon Cognito.

8. Choose the right database solution

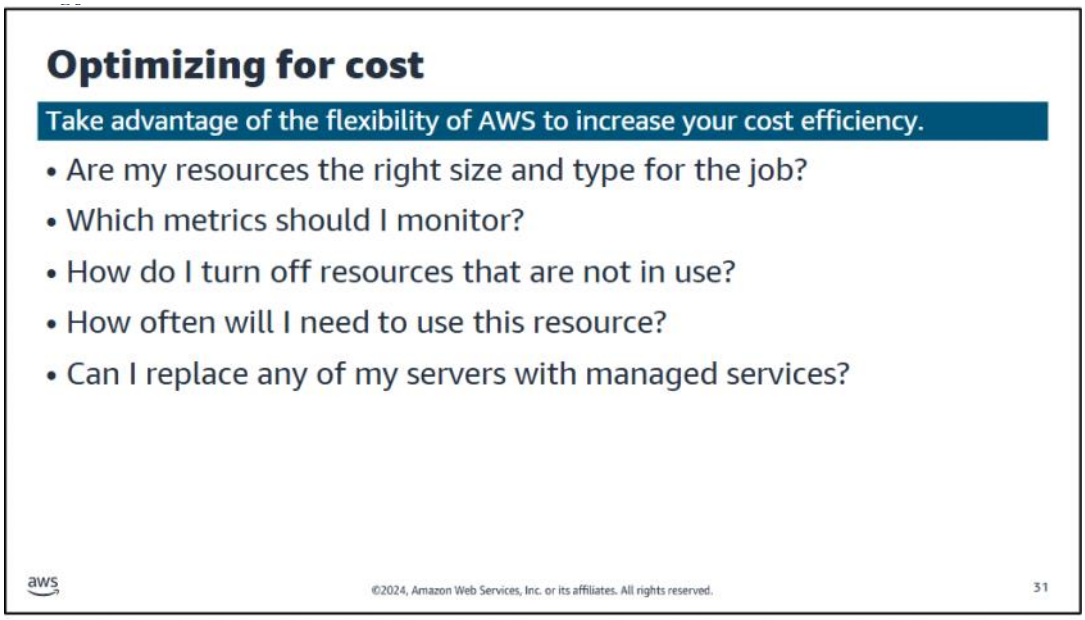
- Read and write needs
- Total storage requirements
- Typical object size and nature of access to these objects
- Durability requirement
- Latency requirements
- Maximum concurrent users to support
- Nature of queries
- Required strength of integrity control

9. Avoiding single points of failure

- A single point of failure (SPOF) is any part of your system that, if it fails, causes the entire system to stop working. The best practice is to design your architecture to avoid them.

- **How to Avoid SPOFs**
- **Don't Duplicate Everything:** You don't always need to have redundant components. Instead, you can use automated solutions that launch new components only when necessary, or rely on AWS managed services that automatically replace malfunctioning hardware for you.
- **Database Example:** A common SPOF is a single database server. If it fails, all connected application servers will also go down.
  - To prevent this, you can set up a **secondary (standby) database server**.
  - You then **replicate data** from the primary to the secondary.
  - If the main database goes offline, the application servers can automatically failover and send their requests to the standby database, ensuring continuous operation.
- **Design for Failure:** The practice of eliminating SPOFs also reinforces the idea of designing applications to treat resources as **disposable**, meaning they can handle the failure, removal, or replacement of the underlying hardware without disruption.

10. Optimizing for cost



11. Using caching

- Caching is a technique for improving performance by temporarily storing copies of data in a location closer to the user. This reduces the need to repeatedly access the primary, often slower, storage layer. The primary goal is to lower latency and network costs.

- How Caching Works (Using the AWS Example)

+ Initial Request: When a user first requests a file, the request goes to **Amazon CloudFront**, which acts as a cache. If the file is not already cached, CloudFront retrieves it from the primary storage location, **Amazon S3**.

+ Caching at the Edge: CloudFront stores a copy of the file at an **edge location**—a data center geographically closer to the user. It then delivers the file to the user.

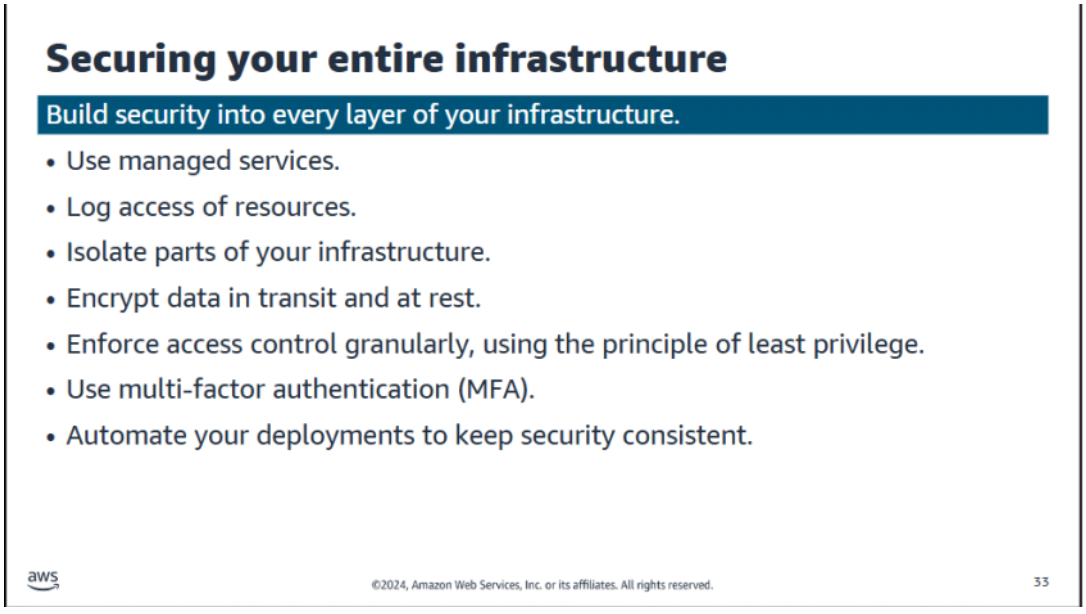
+ Subsequent Requests: Any subsequent requests for the same file from users in that region are served directly from the CloudFront edge location. This is much faster and more cost-effective.

+ Benefits:

*** Reduced Latency**: Users get the data much faster because it is served from a nearby edge location.

*** Lower Cost**: You only pay for the initial data transfer from S3 to CloudFront, not for subsequent requests served from the cache.

12.Securing user entire infrastructure



D AWS Global Infrastructure

1. AWS Regions

- A **Region** is a **georaphical area**
- Each region **consist two or more AZ**
- Communication between Regions uses AWS backbone network infrastructure
- Users enable and control data replication across Regions.

2. Availability Zones

- Select the Azs:

+ Made up of one or more data centers
+ Designed for fault tolerance
+ Interconnect with other AZs in a Region using high-speed private links

- AWS recommends replicating across AZs for resiliency

3. AWS Local Zones

- Local Zones make it possible for user to run latency-sensitive portions of applications closer to end users and resources in a specific georaphy
- They are extension of a Region
- With Local Zones, users can place AWS compute, storage, database, and other select services closer to large population, industry, and IT centers where no Regions exist today.
- Local Zones are managed and supported by AWS

4. AWS data centers

- Data centers are where the data resides and data processing occurs
- A data center typically has tens of thousands of servers
- All data centers are online and serving customers
- AWS custom network equipment includes:
  + Sourced from multiple ODMs
  + Has a customized network protocol stack

5. AWS points of presence (PoPs)

- ***Edge location:*** AWS data centers and servers located close to cutomers and designed to delver services with the lowest latency possible
- ***Regional edge cache:*** AWS data centers between the origin server and the edge location that have a longer cache