Architecting for the Cloud

Architecting for the Cloud is one of the key subjects tested on the Cloud Practitioner exam. This can be dry subject, especially if you're from a non-technical background, but please ensure you're familiar with the concepts at a high-level as questions do come up on the exam.

The information on this page has been extracted from the AWS whitepaper "Architecting for The Cloud: Best Practices" which can be downloaded from this link.

Also, please read the following AWS Blog article: https://aws.amazon.com/blogs/apn/the-5-pillars-of-the-aws-well-architected-framework/

Cloud computing differs from a traditional environment in the following ways:

IT assets become programmable resources

On AWS, servers, databases, storage, and higher-level application components can be instantiated within seconds.

You can treat these as temporary and disposable resources, free from the inflexibility and constraints of a fixed and finite IT infrastructure.

This resets the way you approach change management, testing, reliability, and capacity planning.

Global, available and unlimited capacity

Using the global infrastructure of AWS, you can deploy your application to the AWS Region that best meets your requirements.

For global applications, you can reduce latency to end users around the world by using the Amazon CloudFront content delivery network.

It is also much easier to operate production applications and databases across multiple data centers to achieve high availability and fault tolerance.

Higher level managed services

AWS customers also have access to a broad set of compute, storage, database, analytics, application, and deployment services.

These services are instantly available to developers and can reduce dependency on in-house specialized skills and allow organizations to deliver new solutions faster.

These services are managed by AWS, which can lower operational complexity and cost.

Security built-in

The AWS cloud provides governance capabilities that enable continuous monitoring of configuration changes to your IT resources.

Since AWS assets are programmable resources, your security policy can be formalized and embedded with the design of your infrastructure.

Design Principles

Scalability

Systems that are expected to grow over time need to be built on top of a scalable architecture.

Scaling Vertically

Scaling vertically takes place through an increase in the specifications of an individual resource (e.g., upgrading a server with a larger hard drive or a faster CPU).

On Amazon EC2, this can easily be achieved by stopping an instance and resizing it to an instance type that has more RAM, CPU, IO, or networking capabilities.

Scaling Horizontally

Scaling horizontally takes place through an increase in the number of resources (e.g., adding more hard drives to a storage array or adding more servers to support an application).

This is a great way to build Internet-scale applications that leverage the elasticity of cloud computing.

The table below provides more information on the differences between horizontal and vertical scaling:

Horizontal Scaling	Vertical Scaling
Add more instances as demand increases	Add more CPU and/or RAM to existing instances as demand increases
No downtime required to scale up or down	Requires a restart to scale up or down
Automatic using services such as AWS Auto-Scaling	Would require scripting or automation tools to automate
Unlimited scalability	Scalability limited by maximum instance size

Stateless applications:

- A stateless application is an application that needs no knowledge of previous interactions and stores no session information.
- A stateless application can scale horizontally since any request can be serviced by any of the available compute resources (e.g., EC2 instances, AWS Lambda functions).

Stateless components:

- Most applications need to maintain some kind of state information.
- For example, web applications need to track whether a user is signed in, or else they might present personalized content based on previous actions.
- Web applications can use HTTP cookies to store information about a session at the client's browser (e.g., items in the shopping cart).
- Consider only storing a unique session identifier in a HTTP cookie and storing more detailed user session information server-side.
- DynamoDB is often used for storing session state to maintain a stateless architecture.
- For larger files a shared storage system can be used such as S3 or EFS.
- SWF can be used for a multi step workflow.

Stateful components:

- Databases are stateful.
- Many legacy applications are stateful.
- Load balancing with session affinity can be used for horizontal scaling of stateful components.
- Session affinity is however not guaranteed and existing sessions do not benefit from newly launched nodes.

Distributed processing:

- Use cases that involve processing of very large amounts of data (e.g., anything that can't be handled by a single compute resource in a timely manner) require a distributed processing approach.
- By dividing a task and its data into many small fragments of work, you can execute each of them in any of a larger set of available compute resources.

Disposable Resources Instead of Fixed Servers

Think of servers and other components as temporary resources.

Launch as many as you need, and use them only for as long as you need them.

An issue with fixed, long-running servers is that of configuration drift (where change and software patches are applied over time).

This problem can be solved with the "immutable infrastructure" pattern where a server is never updated but instead is replaced with a new one as required.

Instantiating compute resources

You don't want to manually set up new resources with their configuration and code.

Use automated, repeatable processes that avoid long lead times and are not prone to human error.

Bootstrapping:

- Execute automated bootstrapping actions to modify default configurations.
- This includes scripts that install software or copy data to bring that resource to a particular state.
- You can parameterize configuration details that vary between different environments.

Golden Images:

- Some resource types can be launched from a golden image.
- Examples are EC2 instances, RDS instances and EBS volumes.
- A golden image is a snapshot of a particular state for that resource.
- Compared to bootstrapping golden images provide faster start times and remove dependencies to configuration services or third-party repositories.

Infrastructure are Code:

- AWS assets are programmable, so you can apply techniques, practices, and tools from software development to make your whole infrastructure reusable, maintainable, extensible, and testable.

Automation

In a traditional IT infrastructure, you often have to manually react to a variety of events.

When deploying on AWS there is a lot of opportunity for automation.

This improves both your system's stability and the efficiency of your organization.

Examples of automations using AWS services include:

- AWS Elastic Beanstalk – the fastest and simplest way to get an application up and running on AWS.
- Amazon EC2 Auto Recovery – You can create an Amazon CloudWatch alarm that monitors an Amazon EC2 instance and automatically recovers it if it becomes impaired.
- Auto Scaling – With Auto Scaling, you can maintain application availability and scale your Amazon EC2 capacity up or down automatically according to conditions you define.
- Amazon CloudWatch Alarms – You can create a CloudWatch alarm that sends an Amazon Simple Notification Service (Amazon SNS) message when a particular metric goes beyond a specified threshold for a specified number of periods.
- Amazon CloudWatch Events – The CloudWatch service delivers a near real-time stream of system events that describe changes in AWS resources.
- AWS OpsWorks Lifecycle events – AWS OpsWorks supports continuous configuration through lifecycle events that automatically update your instances' configuration to adapt to environment changes.
- AWS Lambda Scheduled events – These events allow you to create a Lambda function and direct AWS Lambda to execute it on a regular schedule.

Loose Coupling

As application complexity increases, a desirable attribute of an IT system is that it can be broken into smaller, loosely coupled components.

This means that IT systems should be designed in a way that reduces interdependencies—a change or a failure in one component should not cascade to other components.

Design principles include:

- **Well-defined interfaces** – reduce interdependencies in a system by enabling interaction only through specific, technology-agnostic interfaces (e.g. RESTful APIs).
- **Service discovery** – disparate resources must have a way of discovering each other without prior knowledge of the network topology.
- **Asynchronous integration** – this is another form of loose coupling where an interaction does not need an immediate response (think SQS queue or Kinesis).
- **Graceful failure** – build applications such that they handle failure in a graceful manner (reduce the impact of failure and implement retries).

Services, Not Servers

With traditional IT infrastructure, organizations have to build and operate a wide variety of technology components.

AWS offers a broad set of compute, storage, database, analytics, application, and deployment services that help organizations move faster and lower IT costs.

Managed services:

- On AWS, there is a set of services that provide building blocks that developers can consume to power their applications.
- These managed services include databases, machine learning, analytics, queuing, search, email, notifications, and more.

Serverless architectures:

- Another approach that can reduce the operational complexity of running applications is that of the serverless architectures.
- It is possible to build both event-driven and synchronous services for mobile, web, analytics, and the Internet of Things (IoT) without managing any server infrastructure.

Databases

With traditional IT infrastructure, organizations were often limited to the database and storage technologies they could use.

With AWS, these constraints are removed by managed database services that offer enterprise performance at open source cost.

Relational Databases

Relational databases (often called RDBS or SQL databases) normalize data into well-defined tabular structures known as tables, which consist of rows and columns.

They provide a powerful query language, flexible indexing capabilities, strong integrity controls, and the ability to combine data from multiple tables in a fast and efficient manner.

Amazon RDS is a relational database service.

Scalability:

- Relational databases can scale vertically (e.g. upgrading to a larger RDS DB instance).
- For read-heavy use cases, you can scale horizontally using read replicas.
- For scaling write capacity beyond a single instance data partitioning or sharding is required.

High Availability:

- For production DBs, Amazon recommend the use of RDS Multi-AZ which creates a synchronously replicated standby in another AZ.
- With Multi-AZ RDS can failover to the standby node without administrative intervention.

Anti-Patterns:

- If your application primarily indexes and queries data with no need for joins or complex transactions consider a NoSQL database instead.
- If you have large binary files (audio, video, and image), it will be more efficient to store the actual files in S3 and only hold the metadata for the files in your database.

NoSQL Databases

NoSQL is a term used to describe databases that trade some of the query and transaction capabilities of relational databases for a more flexible data model that seamlessly scales horizontally.

NoSQL databases utilize a variety of data models, including graphs, key-value pairs, and JSON documents.

DynamoDB is Amazon's NoSQL database service.

Scalability:

- NoSQL database engines will typically perform data partitioning and replication to scale both the reads and the writes in a horizontal fashion.

High Availability:

- DynamoDB synchronously replicates data across three facilities in an AWS region for fault tolerance.

Anti-Patterns:

- If your schema cannot be denormalized and your application requires joins or complex transactions, consider a relational database instead.
- If you have large binary files (audio, video, and image), consider storing the files in Amazon S3 and storing the metadata for the files in your database.

Data Warehouse

A data warehouse is a specialized type of relational database, optimized for analysis and reporting of large amounts of data.

It can be used to combine transactional data from disparate sources making them available for analysis and decision-making.

Amazon Redshift is a managed data warehouse service that is designed to operate at less than a tenth the cost of traditional solutions.

Scalability:

- Amazon Redshift achieves efficient storage and optimum query performance through a combination of massively parallel processing (MPP), columnar data storage, and targeted data compression encoding schemes.
- RedShift is particularly suited to analytic and reporting workloads against very large data sets.

High Availability:

- Redshift has multiple features that enhance the reliability of your data warehouse cluster.
- Multi-node clusters replicate data to other nodes within the cluster.
- Data is continuously backed up to S3.
- RedShift continuously monitors the health of the cluster and re-replicates data from failed drives and replaces nodes as necessary.

Anti-Patterns:

- Because Amazon Redshift is a SQL-based relational database management system (RDBMS), it is compatible with other RDBMS applications and business intelligence tools.
- Although Amazon Redshift provides the functionality of a typical RDBMS, including online transaction processing (OLTP) functions, it is not designed for these workloads.

Search

Applications that require sophisticated search functionality will typically outgrow the capabilities of relational or NoSQL databases.

A search service can be used to index and search both structured and free text format and can support functionality that is not available in other databases, such as customizable result ranking, faceting for filtering, synonyms, stemming, etc.

Scalability:

- Both Amazon CloudSearch and Amazon ES use data partitioning and replication to scale horizontally.

High Availability:

- Both services provide features that store data redundantly across Availability Zones.

Removing Single Points of Failure

A system is highly available when it can withstand the failure of an individual or multiple components.

Automate recovery and reduce disruption at every layer of your architecture.

Introducing Redundancy

Single points of failure can be removed by introducing redundancy.

In standby redundancy when a resource fails, functionality is recovered on a secondary resource using a process called failover, which typically take some time to complete.

In active redundancy, requests are distributed to multiple redundant compute resources, and when one of them fails, the rest can simply absorb a larger share of the workload.

Detect Failure

Build as much automation as possible in both detecting and reacting to failure.

Services like ELB and Route53 mask failure by routing traffic to healthy endpoint.

Auto Scaling can be configured to automatically replace unhealthy nodes.

You can also replace unhealthy nodes using the EC2 auto- recovery, OpsWorks and Elastic Beanstalk.

Durable Data Storage

Design your architecture to protect both data availability and integrity.

Data replication is the technique that introduces redundant copies of data.

It can help horizontally scale read capacity, but it also increase data durability and availability.

Replication can take place in a few different modes:

- Synchronous replication – transactions are acknowledged only after data has been durably stored in both the primary and replica instance. Can be used to protect data integrity (low RPO) and scaling read capacity (with strong consistency).
- Asynchronous replication – changes on the primary node are not immediately reflected on its replicas. Can be used to horizontally scale the system's read capacity (with replication lag), and data durability (with some data loss).
- Quorum-based replication – combines synchronous and asynchronous replication and is good for large-scale distributed database systems.

Automated Multi-Data Center Resilience

With traditional infrastructure, failing over between data centers is performed using a disaster recovery plan.

Long distances between data centers mean that latency makes synchronous replication impractical.

Failovers often lead to data loss and costly data recovery processes.

On AWS it is possible to adopt a simpler, more efficient protection from this type of failure.

Each AWS region contains multiple distinct locations called Availability Zones (AZs).

Each AZ is engineered to be isolated from failures in other AZs.

An AZ is a data center, and in some cases, an AZ consists of multiple data centers.

AZs within a region provide inexpensive, low-latency network connectivity to other zones in the same region.

This allows you to replicate your data across data centers in a synchronous manner so that failover can be automated and be transparent for your users.

Fault Isolation and Traditional Horizontal Scaling

Though the active redundancy pattern is great for balancing traffic and handling instance or Availability Zone disruptions, it is not sufficient if there is something harmful about the requests themselves.

If a particular request happens to trigger a bug that causes the system to fail over, then the caller may trigger a cascading failure by repeatedly trying the same request against all instances.

One fault-isolating improvement you can make to traditional horizontal scaling is called sharding.

Similar to the technique traditionally used with data storage systems, instead of spreading traffic from all customers across every node, you can group the instances into shards.

In this way, you are able to reduce the impact on customers in direct proportion to the number of shards you have.

Optimize for Cost

Just by moving existing architectures into the cloud, organizations can reduce capital expenses and drive savings as a result of the AWS economies of scale.

By iterating and making use of more AWS capabilities there is further opportunity to create cost-optimized cloud architectures.

Right Sizing:

- In some cases, you should select the cheapest type that suits your workload's requirements.
- In other cases, using fewer instances of a larger instance type might result in lower total cost or better performance.
- Benchmark and select the right instance type depending on how your workload utilizes CPU, RAM, network, storage size, and I/O.
- Reduce cost by selecting the right storage solution for your needs.
- E.g. S3 offers a variety of storage classes, including Standard, Reduced Redundancy, and Standard-Infrequent Access.
- EC2, RDS, and ES support different EBS volume types (magnetic, general purpose SSD, provisioned IOPS SSD) that you should evaluate.

Elasticity:

- Plan to implement Auto Scaling for as many EC2 workloads as possible, so that you horizontally scale up when needed and scale down automatically to reduce cost.
- Automate turning off non-production workloads when not in use.
- Where possible, replace EC2 workloads with AWS managed services that don't require you to take any capacity decisions. For example:
    - ELB.
    - CloudFront.
    - SQS.
    - Kinesis Firehose.
    - Lambda.
    - SES.
    - CloudSearch.
- Or use services for which you can modify capacity as and when need. For example:
    - DynamoDB.
    - RDS.
    - Elasticsearch Service.

Take Advantage of the Variety of Purchasing Options:

- EC2 On-Demand instance pricing gives you maximum flexibility with no long term commitments.
- There are two more ways to pay for Amazon EC2 instances that can help you reduce spend: Reserved Instances and Spot Instances.

Reserved Capacity

EC2 Reserved Instances allow you to reserve Amazon EC2 computing capacity in exchange for a significantly discounted hourly rate compared to On- Demand instance pricing.

This is ideal for applications with predictable minimum capacity requirements.

Spot Instances

For less steady workloads, you can consider the use of Spot Instances.

EC2 Spot Instances allow you to bid on spareEC2 computing capacity.

Since Spot Instances are often available at a discount compared to On-Demand pricing, you can significantly reduce the cost of running your applications.

Spot Instances are ideal for workloads that have flexible start and end times.

If the Spot market price increases above your bid price, your instance will be terminated automatically and you will not be charged for the partial hour that your instance has run.

As a result, Spot Instances are great for workloads that have tolerance to interruption.

Caching

Caching is a technique that stores previously calculated data for future use.

This technique is used to improve application performance and increase the cost efficiency of an implementation.

It can be applied at multiple layers of an IT architecture.

Application Data Caching

Applications can be designed so that they store and retrieve information from fast, managed, in-memory caches.

Cached information may include the results of I/O-intensive database queries or the outcome of computationally intensive processing.

Edge Caching

Copies of static content and dynamic content can be cached at Amazon CloudFront, which is a content delivery network (CDN) consisting of multiple edge locations around the world.

Edge caching allows content to be served by infrastructure that is closer to viewers, lowering latency and giving you the high, sustained data transfer rates needed to deliver large popular objects to end users at scale.

Security

Most of the security tools and techniques that you might already be familiar with in a traditional IT infrastructure can be used in the cloud.

At the same time, AWS allows you to improve your security in a variety of ways.

AWS is a platform that allows you to formalize the design of security controls in the platform itself.

Utilize AWS Features for Defence in Depth

Network level security includes building a VPC topology that isolates parts of the infrastructure through the use of subnets, security groups, and routing controls.

Services like AWS WAF, a web application firewall, can help protect web applications from SQL injection and other vulnerabilities in application code.

For access control, you can use IAM to define a granular set of policies and assign them to users, groups, and AWS resources.

Finally, the AWS platform offers a breadth of options for protecting data, whether it is in transit or at rest with encryption.

Offload Security Responsibility to AWS

AWS operates under a shared security responsibility model, where AWS is responsible for the security of the underlying cloud infrastructure and you are responsible for securing the workloads you deploy in AWS.

Reduce Privileged Access

When you treat servers as programmable resources, you can capitalize on that for benefits in the security space as well.

Eliminate the need for guest operating system access to production environments.

If an instance experiences an issue you can automatically or manually terminate and replace it.

In a traditional environment, service accounts would often be assigned long-term credentials stored in a configuration file.

On AWS, you can instead use IAM roles to grant permissions to applications running on Amazon EC2 instances through the use of short-term credentials.

Security as Code

Traditional security frameworks, regulations, and organizational policies define security requirements related to things such as firewall rules, network access controls, internal/external subnets, and operating system hardening.

You can implement these in an AWS environment as well, but you now have the opportunity to capture them all in a script that defines a “Golden Environment.”

This means you can create an AWS CloudFormation script that captures your security policy and reliably deploys it.

Security best practices can now be reused among multiple projects and become part of your continuous integration pipeline.

You can perform security testing as part of your release cycle, and automatically discover application gaps and drift from your security policy.

Real-Time Auditing

Testing and auditing your environment is key to moving fast while staying safe.

Traditional approaches that involve periodic checks are not sufficient, especially in agile environments where change is constant.

On AWS, it is possible to implement continuous monitoring and automation of controls to minimize exposure to security risks.

Services like AWS Config, Amazon Inspector, and AWS Trusted Advisor continually monitor for compliance or vulnerabilities.

With AWS Config rules you will also know if some component was out of compliance even for a brief period of time.

You can implement extensive logging for your applications (using Amazon CloudWatch Logs) and for the actual AWS API calls by enabling AWS CloudTrail.

Logs can then be stored in an immutable manner and automatically processed to either notify or even take action on your behalf, protecting your organization from non-compliance.

You can use AWS Lambda, Amazon EMR, the Amazon Elasticsearch Service, or third- party tools from the AWS Marketplace to scan logs to detect things like unused permissions, overuse of privileged accounts, usage of keys, anomalous logins, policy violations, and system abuse.