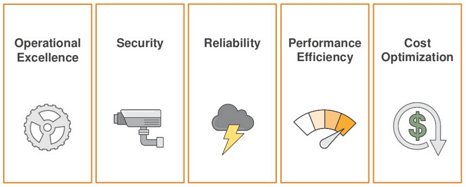
In this, we will learn to list the different cloud architecture design principles.

Amazon’s Well-Architected Framework aids users in the development of secure, high-performing, resilient, and efficient infrastructure for their applications. Cloud architecture design ideas are based on five pillars—

* operational excellence
* security
* reliability
* performance efficiency
* cost optimization

**Cloud Architecture Design Principles**

* Operational Excellence is concerned with the operation and monitoring of systems in order to deliver corporate value and enhance processes and procedures on a continuous basis. It consists of
  + managing and automating changes
  + responding to events
  + defining standards to successfully manage daily operations.
* Security – Focuses on protecting information and systems. It includes
  + confidentiality and integrity of data
  + identifying and managing who can do what with privilege management
  + protecting systems
  + controls to detect security events
* Reliability – It focuses on the ability to prevent, and quickly recover from failures to meet business and customer demand. It includes
  + foundational elements around setup
  + cross project requirements
  + recovery planning
  + how we handle change.
* Performance Efficiency – It focuses on using IT and computing resources efficiently. It includes
  + selecting the right resource types and sizes based on workload requirements
  + monitoring performance
  + making informed decisions to maintain efficiency as business needs evolve.
* Cost Optimization – It focuses on avoiding un-needed costs. It includes
  + understanding and controlling where money is being spent,
  + choosing the proper number of resource kinds and the most relevant resource types
  + Scaling to fulfill corporate needs without overspending by assessing spend over time.



**Core Principles**

The AWS Architectural is guided by two key cloud architecture principles: elasticity and scalability.

**Elasticity**

* Ability to employ resources in a flexible and effective manner
* Avoids the conventional anti-pattern of over-provisioning infrastructure resources in order to meet capacity demands.
* Elasticity saves money by avoiding the costs of over-provisioned resources like electricity, space, and maintenance.
* AWS operates on a pay-as-you-go/pay-for-what-you-use model.

**Scalability**

* Scalability without having to change the design.
* This is achieved in AWS by scaling out.
* Technology components are commodities that can be thrown away if they fail or added to if they succeed.
* Maintain a unified approach to architecture and development.

To successfully leverage the AWS cloud

* Autoscaling allows you to automatically horizontally scale to accommodate load.
* Bootstrapping allows you automatically setup your servers after they boot. (Using components such as Amazon Machine Images (AMI’s) and CloudFormation to automate)
* Loosely Coupled
* Stateless
* Horizontal
* Treat servers as disposable resources
* Automate Automate Automate

**AWS Resources for automation**

* **AWS Elastic Beanstalk:** It’s the quickest and most straightforward approach to get an AWS application up and running. When they upload their application code, the service takes care of everything else, including resource provisioning, load balancing, auto scaling, and monitoring.
* **Amazon EC2 Auto recovery:** You can set up an Amazon CloudWatch alarm to monitor an Amazon EC2 instance and recover it automatically if it becomes unusable. The instance transfers through an instance reboot during instance recovery. And, any data that is the in-memory is lost.
* **Auto Scaling:** Maintain application availability and scale your Amazon EC2 capacity up or down automatically according to conditions you define.
* **Amazon CloudWatch Alarms:** When a given metric exceeds a specified threshold for a specified number of periods, a CloudWatch alert sends an Amazon Simple Notification Service (Amazon SNS) message.
* **Amazon CloudWatch Events:** The CloudWatch service provides a near-real-time stream of system events that characterize AWS resource changes. Each sort of event can route to one or more targets using rules that can be put up in a matter of minutes:
  + AWS Lambda functions, Kinesis streams, SNS, and other AWS services
* **AWS OpsWorks Lifecycle events:** It provides continuous configuration via lifecycle events, which automatically alter the configuration of your instances to respond to changes in the environment. Chef recipes can be triggered by these events to conduct specific setup chores on each instance.
* **AWS Lambda Scheduled events:** These events allow you to create a Lambda function and direct AWS Lambda to execute it on a regular schedule.

**Principle Implementation –   Focus on services, not servers**

For implementation leverage a set of

* Compute services
* Storage services
* Database services
* Analytics services
* Application services
* deployment services

On AWS, there are two ways to do that.

* Use managed services that include databases, machine learning, analytics, queuing, search, email, notifications, and more.
* Running applications through server-less architectures. Build event-driven and synchronous services for mobile, web, analytics, and the Internet of Things (IoT) without managing any server infrastructure.

**Principle Implementation – Database is the core**

Managed database services

* removes constraints that come with licensing costs
* supports diverse database engines
* Access to the information stored on these databases is the main purpose of cloud computing.

**There are three different categories of databases to keep in mind while architecting:**

* Relational databases – Data is normalized into tables and also provided with
  + powerful query language
  + flexible indexing capabilities
  + strong integrity controls
  + ability to combine data from multiple tables in a fast and efficient manner.
  + Can be scaled vertically and are highly available during failovers.
* NoSQL databases– They have a flexible data model that scales horizontally with ease. Graphs, key-value pairs, and JSON documents are among the data models used by NoSQL databases. They provide
  + ease of development
  + scalable performance
  + high availability
  + resilience
* Data warehouse – A specialised sort of relational database designed for large-scale data analysis and reporting. It bring together transactional data from several sources, allowing for analysis and decision-making.
  + Horizontal-scaling is usually based on the partitioning of the data i.e. each node contains only part of the data
  + Vertical-scaling the data resides on a single node and scaling executes through multi-core i.e. spreading the load between the CPU and RAM resources of that machine.

**Remove SPOF or single points of failure**

When a system can endure the failure of a single or numerous components, it is said to be highly available (e.g., hard disks, servers, network links etc.). By automating recovery and decreasing disruption at every layer of the AWS cloud architecture, you may save time and money.

* Introduce redundancy, by having multiple resources for the same task. Redundancy can implement in standby mode (functionality recovers through failover while the resource remains unavailable) or active mode (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).
* Detection and reaction to failure should both automate as much as possible.
* Have a durable data storage to protect data availability and integrity. Redundant copies of data can implement by synchronous, asynchronous or Quorum based replication.
* Automated Multi –Data Center resilience is done through Availability Zones across data centers that reduce the impact of failures.
* Introduce sharding  or a method of grouping instances into groups called shards, instead of sending the traffic from all users to every node.

**Implement Cost  Optimization**

At the end of the day, it often boils down to cost. Your AWS cloud architecture should be designed for cost optimization by keeping in mind the following principles:

* You can reduce cost by selecting the right types, configurations and storage solutions to suit your needs.
* Implementing Auto Scaling so that you can scale horizontally when required or scale down when necessary can be done without any extra cost.
* Taking advantage of the variety of Instance Purchasing options (Reserved and spot instances) while buying EC2 instances will help reduce cost on computing capacity.
* Caching
* Security