INDEX

Fact Finding Exercises

I Python

1) Define a list and tuple in Python. Provide some examples.

A list is a collection of items, which are ordered and *mutable*. Lists are written with square brackets, and items are separated by commas. For example:

```
my_list = [1, 2, 3, 4, 5]
```

You can also use the list() constructor to create a list, for example:

```
my_list = list((1, 2, 3, 4, 5))
```

You can access items in a list by their index, starting from 0. For example, to access the first item in the list, you would use my_list[0], to access the second item, you would use my_list[1], and so on.

A tuple is a collection of ordered and immutable elements, enclosed in parentheses and separated by commas. For example: (1, "apple", 3.14, [1, 2, 3]). Tuples can contain elements of different types and can be used to store multiple values in a single variable. They are similar to lists, but unlike lists, tuples cannot be modified once they are created. They are also useful for creating data structures that should not be modified, such as coordinates or RGB color values.

2) What is a namespace in Python?

A namespace is a collection of names (also known as identifiers) that are used to refer to objects, such as variables, functions, and classes. Each namespace is unique and contains a set of names that are specific to it. Namespaces in Python are used to prevent naming conflicts and to organize code.

Types of namespaces:

Global namespace: This namespace contains names that are defined at the top level of a script or module.

Local namespace: This namespace contains names that are defined inside a function or class.

Built-in namespace: This namespace contains names that are predefined in Python, such as built-in functions and data types.

When a name is referenced in Python, the interpreter first looks in the local namespace, then in the global namespace, and finally in the built-in namespace. If the name is not found in any of these namespaces, a NameError is raised.

It's also important to note that each module in python creates a unique namespace, this means that the variables, functions, and classes defined in one module are not accessible in other modules, unless they are imported.

3) What is the difference between a local variable and a global variable?

A local variable is a variable that is defined inside a function or class and can only be accessed within that function or class. A global variable, on the other hand, is a variable that is defined outside of any function or class and can be accessed from anywhere in the script or module.

The main differences between local and global variables are:

	Local Variable	*Global Variable*
Scope	Local variables have a limited scope and can only be accessed within the function or class in which they are defined.	Global variables have a global scope and can be accessed from anywhere in the script or module.
Modification	Local variables can be modified within the function or class, but their changes do not affect the global scope.	Global variables, on the other hand, can be modified from anywhere in the script or module and their changes are reflected in the global scope.
Naming conflicts	Local variables can have the same name as global variables without causing any conflicts, because they exist in different scopes.	However, if a local variable and a global variable have the same name and are both modified within the same function or class, the local variable will take precedence.

It's important to use the right variable in the right place, using global variable when they are really needed, and using local variable when they are only needed within a function or class, this is because using global variables when they are not needed can make the code hard to read, understand and debug.

4) What is an IDE? Mention some common IDEs that could be used with Python.

An IDE (Integrated Development Environment) is a software application that provides a comprehensive set of tools for software development, including code editing, debugging, and testing. IDEs are designed to make the development process more efficient and user-friendly by providing a single interface for all the necessary tools and resources.

Some common IDEs that can be used with Python include:

- PyCharm: PyCharm is a popular Python IDE that is developed by JetBrains. It provides a wide range of features, including code completion, code highlighting, and debugging, making it a great choice for professional Python developers.
- IDLE: IDLE is the built-in Python IDE that comes with Python. It is a simple and lightweight IDE that is suitable for beginners who are just getting started with Python.

- Eclipse: Eclipse is a general-purpose IDE that can be used for Python development. It is widely used for developing Java applications, but it also has support for Python through a plugin called PyDev.
- Visual Studio Code: Visual Studio Code is a lightweight and powerful code editor that can be used for Python development. It is developed by Microsoft and has a wide range of features, including code highlighting and debugging.
- Jupyter Notebook: Jupyter Notebook is a web-based IDE that is primarily used for data science and machine learning. It allows the user to write and execute code, as well as create and share documents that contain live code, equations, visualizations, and narrative text.

5) What are modules in Python? Provide some examples.

A module is a file containing Python definitions and statements. A module can define functions, classes, and variables, and can also include other modules. The main advantage of using modules is that they allow you to organize your code and reuse it in different parts of your program.

There are two types of modules in Python:

- Built-in modules: These are modules that are included with Python and can be used without having to install them. Examples of built-in modules include the math module, which provides mathematical functions, and the random module, which provides functions for generating random numbers.

  Importing a module example

  ```
  import math

  print(math.sqrt(24))


  from math import sqrt

  print(sqrt(64))
  ```

- External modules: These are modules that are not included with Python and must be installed separately. Examples of external modules include the NumPy module, which provides support for large, multi-dimensional arrays, and the Pandas module, which provides data structures and data analysis tools.

  ```
  !pip install numpy

  import numpy as np

  a = np.array([1, 2, 3])

  print(a)  # Output: [1 2 3]
  ```

- User-defined modules: These are the modules that the developer creates and can be used within the application.

6) What is the difference between an array and a list?

An array and a list are both data structures that are used to store collections of items, but they have some key differences:

	Array	*List*
Data Type	An array is a collection of items of the same data type	a list can contain items of different data types.
Size	array has a fixed size, which means that its size cannot be changed once it is created.	A list, on the other hand, can grow or shrink dynamically as items are added or removed.
Memory	An array occupies a contiguous block of memory, which makes it more efficient for memory usage and accessing elements.	Lists, on the other hand, are implemented as a linked list, which requires more memory overhead.
Accessing elements	Arrays are indexed, which means that elements can be accessed directly by their position.	Lists, on the other hand, require linear search for accessing elements.
Speed	Arrays are generally faster than lists when it comes to accessing elements, or performing mathematical operations, because of their contiguous memory allocation, and the fact that all elements are of the same data type.	Slow as compared to arrays.

In Python, arrays are implemented as the "array" module, while lists are a built-in data type. When working with large amounts of data, or when performance is critical, it is recommended to use arrays instead of lists, but when you need more flexibility, lists are better.

7) What are operators? Provide some examples.

Operators are special symbols in Python that perform specific operations on one or more operands (values). They help to manipulate the values and variables.

Types of operators in Python:

- Arithmetic operators: These operators perform mathematical operations, such as addition (+), subtraction (-), multiplication (*), division (/), and modulus (%).

  For example: `+,−, *, /, %`

- Comparison operators: These operators compare two values and return a Boolean value (True or False) based on the comparison.

  For example: `> , <, ==, !=`

- Logical operators: These operators perform logical operations, such as and (and), or (or), and not (not).

For example: `and, or, not`

- Assignment operators: These operators are used to assign a value to a variable.

  For example: `+=, -=`

II Database

1) What is the difference between a relational and a non-relational database?

Relational Database	*Non-Relational Database*
A relational database organizes data into tables with defined relationships between them, based on keys.	On the other hand, a non-relational database, also known as NoSQL, stores data in a more flexible, unstructured format, such as JSON, XML or key-value pairs, without the need for a fixed schema.
It uses a structured approach to store and retrieve data, with data being easily searchable and retrievable through the use of SQL (Structured Query Language).	Non-relational databases are typically better suited for handling large amounts of unstructured data and for scalability, as they can handle high volumes of writes and are more flexible in terms of data structure changes. Often use alternative query languages other than SQL.
They are optimized for complex, structured data and relationships between data, and support transactions, referential integrity, and the ability to join data from multiple tables.	Not suited for huge load and complex transactional type applications.
Vertically scalable	Horizontally scalable
SQL databases maintains ACID properties (Atomicity, Consistency, Isolation and Durability)	Follows the Brewers theorem or BASE properties
Synchronous Inserts and Updates	Asynchronous Inserts and Updates
Relational databases are a good fit for applications that require complex relationships between data, transactions, and consistency	Non-relational databases are a better choice for large, unstructured data, and high performance and scalability requirements

2) What are indexes?

Indexes are data structures that are used to improve the speed of data retrieval operations on a database table. They allow for faster search for specific values by reducing the number of rows that need to be scanned. An index contains a sorted list of the values in a specific column of a table along with a pointer to the location of each value in the table. When a query is executed that searches for specific values in a column, the database can use the index to quickly locate the relevant rows, rather than having to scan the entire table.

Indexes can significantly improve query performance, especially for large tables with many rows. However, indexes can also slow down updates and inserts, as the database has to update the index whenever the data in the indexed column is changed. It's important to carefully consider the trade-off between the benefits of improved query performance and the costs of increased update time when deciding whether to create an index and which columns to index.

3) What are primary keys and secondary keys?

Primary keys and secondary keys are types of keys used in relational databases to enforce relationships between tables and ensure data integrity.

A primary key is a unique identifier for each record in a table, used to enforce the integrity of the data and to ensure that each record in a table is distinct. A table can only have one primary key, and primary keys are used as the main reference for foreign keys in other tables to enforce relationships between tables.

A secondary key, also known as an index, is used to improve the performance of searching and sorting operations in a table. Secondary keys do not enforce uniqueness or relationships between tables, but they can be used to quickly find specific records based on the values in the indexed column.

Primary keys are used to enforce data integrity and relationships between tables, while secondary keys are used to improve query performance.

4) What are inner joins and outer joins?

Inner joins and outer joins are types of SQL joins used to combine data from two or more tables. Inner joins and outer joins are used to combine data from two or more tables based on a specified join condition, and they play an important role in relational database design and data analysis.

An inner join returns only the rows from both tables where the join condition is met. It returns only the matching rows from both tables and discards the rest. An inner join returns only the common values between the two tables.

An outer join returns all the rows from one table and the matching rows from the other table. There are three types of outer joins: left join, right join, and full outer join.

- A left join returns all the rows from the left table and the matching rows from the right table, and returns NULL values for non-matching rows from the right table.
- A right join returns all the rows from the right table and the matching rows from the left table, and returns NULL values for non-matching rows from the left table.
- A full outer join returns all the rows from both tables, and returns NULL values for non-matching rows from either the left or the right table.

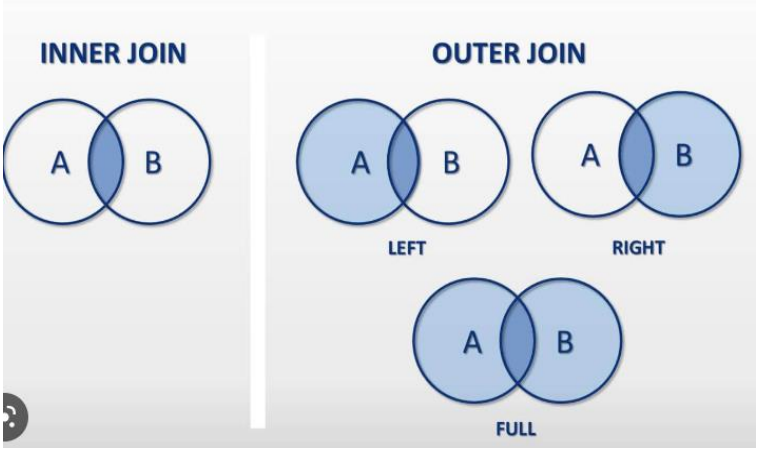


Figure 1 Inner joins and Outer joins

5) What is the difference between DROPTABLE and TRUNCATETABLE?

DROPTABLE is a statement that permanently deletes a table and all the data in it. It removes the table structure and all its associated objects, including triggers, indexes, and constraints. Once a table is dropped, it cannot be recovered, so it should be used with caution. The syntax for DROPTABLE is:

```
DROP TABLE table_name;
```

TRUNCATETABLE, on the other hand, is a statement that is used to remove all the data in a table, but not the table structure itself. The table structure, including its columns, data types, and constraints, remains intact after a TRUNCATE operation. The syntax for TRUNCATETABLE is:

```
TRUNCATE TABLE table_name;
```

6) What are the different datatypes in SQL?

SQL supports various data types that determine the type of data that can be stored in a column of a table. The different data types in SQL include:

1. Numeric: used for numbers such as integers, decimals, and floats. For example: INT, BIGINT, NUMERIC, DECIMAL, FLOAT
2. Character: used for strings of characters. For example: `CHAR, VARCHAR, TEXT`
3. Date and Time: used for date and time values. For example: `DATE, TIME, DATETIME, TIMESTAMP`
4. Binary: used for binary data such as images, videos, or audio. For example: `BINARY, VARBINARY, BLOB`
5. Boolean: used for values that can only be either true or false. For example: `BOOLEAN or BIT`
6. Geometric: used for geometric data such as points, lines, and polygon. For example: `POINT, LINESTRING, POLYGON`
7. Network: used for network address data such as IP addresses. For example: `CIDR, INET`

Different SQL implementations may have different sets of data types, but these are some of the commonly used ones. When choosing data types for a column, it is important to consider the type of data that will be stored, the size of the data, and the range of values that need to be supported.

7) Explain the WHERE and HAVING clauses.

A HAVING clause is like a WHERE clause, but applies only to groups as a whole (that is, to the rows in the result set representing groups), whereas the WHERE clause applies to individual rows.

The WHERE clause is used to filter rows from a table based on specified conditions. It is used in SELECT, UPDATE, and DELETE statements to restrict the rows that are returned or affected by the statement. The WHERE clause is used in the following syntax:

```
SELECT columns FROM table WHERE condition;
```

The condition in the WHERE clause can be a comparison between two values, a range of values, or a combination of multiple conditions. For example, you can use the WHERE clause to find all the employees who earn a salary greater than $50,000:

The HAVING clause is used to filter groups of rows based on aggregate values. It is used in SELECT statements with the GROUP BY clause to restrict the groups that are returned. The HAVING clause is used in the following syntax:

```
SELECT columns, aggregate_function(column) FROM table GROUP BY columns
HAVING condition;
```

The condition in the HAVING clause can be a comparison between the aggregate value and a constant, a range of values, or a combination of multiple conditions. For example, you can use the HAVING clause to find the departments that have an average salary greater than $60,000:

```
SELECT department, AVG(salary) FROM employees GROUP BY department
HAVING AVG(salary) > 60000;
```

III AWS Well-Architected Framework

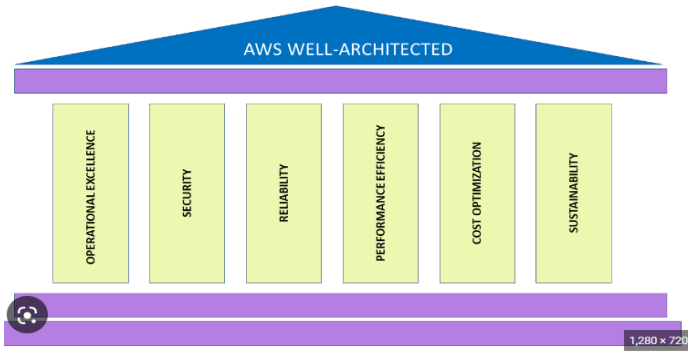


Figure 2 AWS Well-Architected Framework

1) What are the six pillars of the Well Architected Framework (WAF)?

Creating a software system is a lot like constructing a building. If the foundation is not solid, structural problems can undermine the integrity and function of the building. The AWS Well-Architected Framework helps cloud architects build the most secure, high-performing, resilient, and efficient infrastructure possible for their applications. This framework provides a consistent approach to evaluate architectures, and provides guidance to implement designs that scale with your application needs over time. The six pillars of the WAF are:

1. Operational Excellence

The Operational Excellence pillar includes the ability to support development and run workloads effectively, gain insight into their operation, and continuously improve supporting processes and procedures to delivery business value. You can find prescriptive guidance on implementation in the Operational Excellence Pillar whitepaper.

There are five design principles for operational excellence in the cloud:

- Perform operations as code
- Make frequent, small, reversible changes
- Refine operations procedures frequently
- Anticipate failure
- Learn from all operational failures

Best Practices

Operations teams need to understand their business and customer needs so they can support business outcomes. Ops creates and uses procedures to respond to operational events, and validates their effectiveness to support business needs. Ops also collects metrics that are used to measure the achievement of desired business outcomes.

Everything continues to change—your business context, business priorities, and customer needs. It's important to design operations to support evolution over time in response to change, and to incorporate lessons learned through their performance.

2. Security

The Security pillar includes the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security. You can find prescriptive guidance on implementation in the Security Pillar whitepaper.

There are seven design principles for security in the cloud:

- 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 for security events

Best Practices

Before you architect any workload, you need to put in place practices that influence security. You'll want to control who can do what. In addition, you want to be able to identify security incidents, protect your systems and services, and maintain the confidentiality and integrity of data through data protection.

You should have a well-defined and practiced process for responding to security incidents. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

The AWS Shared Responsibility Model enables organizations that adopt the cloud to achieve their security and compliance goals. Because AWS physically secures the infrastructure that supports our cloud services, as an AWS customer you can focus on using services to accomplish your goals. The AWS Cloud also provides greater access to security data and an automated approach to responding to security events.

3. Reliability

The Reliability pillar encompasses the ability of a workload to 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. You can find prescriptive guidance on implementation in the Reliability Pillar whitepaper.

There are five design principles for reliability in the cloud:

- Automatically recover from failure
- Test recovery procedures
- Scale horizontally to increase aggregate workload availability
- Stop guessing capacity
- Manage change in automation

Best Practices

Before building any system, foundational requirements that influence reliability should be in place. For example, you must have sufficient network bandwidth to your data center. These requirements are sometimes neglected (because they are beyond a single project's scope). With AWS, however, most of the foundational requirements are already incorporated or can be addressed as needed.

The cloud is designed to be nearly limitless, so it's the responsibility of AWS to satisfy the requirement for sufficient networking and compute capacity, leaving you free to change resource size and allocations on demand.

A reliable workload starts with upfront design decisions for both software and infrastructure. Your architecture choices will impact your workload behavior across all six AWS Well-Architected pillars. For reliability, there are specific patterns you must follow, such as loosely coupled dependencies, graceful degradation, and limiting retries.

Changes to your workload or its environment must be anticipated and accommodated to achieve reliable operation of the workload. Changes include those imposed on your workload, like a spikes in demand, as well as those from within such as feature deployments and security patches.

Low-level hardware component failures are something to be dealt with every day in an on-premises data center. In the cloud, however, these are often abstracted away. Regardless of your cloud provider, there is the potential for failures to impact your workload. You must therefore take steps to implement resiliency in your workload, such as fault isolation, automated failover to healthy resources, and a disaster recovery strategy.

4. Performance Efficiency

The Performance Efficiency pillar includes the ability to use computing resources efficiently to meet system requirements, and to maintain that efficiency as demand changes and technologies evolve. You can find prescriptive guidance on implementation in the Performance Efficiency Pillar whitepaper.

There are five design principles for performance efficiency in the cloud:

- Democratize advanced technologies
- Go global in minutes
- Use serverless architectures
- Experiment more often
- Consider mechanical sympathy

Best Practices

Take a data-driven approach to building a high-performance architecture. Gather data on all aspects of the architecture, from the high-level design to the selection and configuration of resource types.

Reviewing your choices on a regular basis ensures you are taking advantage of the continually evolving AWS Cloud. Monitoring ensures you are aware of any deviance from expected performance. Make trade-offs in your architecture to improve performance, such as using compression or caching, or relaxing consistency requirements

The optimal solution for a particular workload varies, and solutions often combine multiple approaches. AWS Well-Architected workloads use multiple solutions and enable different features to improve performance

5. Cost Optimization

The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point. You can find prescriptive guidance on implementation in the Cost Optimization Pillar whitepaper.

There are five design principles for cost optimization in the cloud:

1. Implement cloud financial management
2. Adopt a consumption model
3. Measure overall efficiency
4. Stop spending money on undifferentiated heavy lifting
5. Analyze and attribute expenditure

Best Practices

As with the other pillars, there are trade-offs to consider. For example, do you want to optimize for speed to market or for cost? In some cases, it's best to optimize for speed—going to market quickly, shipping new features, or simply meeting a deadline—rather than investing in up-front cost optimization.

Design decisions are sometimes directed by haste rather than data, and as the temptation always exists to overcompensate rather than spend time benchmarking for the most cost-optimal deployment. This might lead to over-provisioned and under-optimized deployments.

Using the appropriate services, resources, and configurations for your workloads is key to cost savings.

6. Sustainability

The discipline of sustainability addresses the long-term environmental, economic, and societal impact of your business activities. You can find prescriptive guidance on implementation in the Sustainability Pillar whitepaper.

There are six design principles for sustainability in the cloud:

1. Understand your impact
2. Establish sustainability goals
3. Maximize utilization
4. Anticipate and adopt new, more efficient hardware and software offerings
5. Use managed services
6. Reduce the downstream impact of your cloud workloads

Best Practices

Choose AWS Regions where you will implement workloads based on your business requirements and sustainability goals.

User behavior patterns can help you identify improvements to meet sustainability goals. For example, scale infrastructure down when not needed, position resources to limit the network required for users to consume them, and remove unused assets.

Implement software and architecture patterns to perform load smoothing and maintain consistent high utilization of deployed resources. Understand the performance of your workload components, and optimize the components that consume the most resources.

Analyze data patterns to implement data management practices that reduce the provisioned storage required to support your workload. Use lifecycle capabilities to move data to more efficient, less performant storage when requirements decrease, and delete data that's no longer required.

Analyze hardware patterns to identify opportunities that reduce workload sustainability impacts by minimizing the amount of hardware needed to provision and deploy. Select the most efficient hardware for your individual workload.

In your development and deployment process, identify opportunities to reduce your sustainability impact by making changes, such as updating systems to gain performance efficiencies and manage sustainability impacts. Use automation to manage the lifecycle of your development and test environments, and use managed device farms for testing.

2) What are the 3 areas of operational excellence in the cloud?

The Operational Excellence pillar includes the ability to support development and run workloads effectively, gain insight into their operations, and to continuously improve supporting processes and procedures to deliver business value.

Amazon describes operational excellence in the cloud through four areas:

1. Organization
2. Prepare
3. Operate
4. Evolve

Organization

Your teams need to have a shared understanding of your entire workload, their role in it, and shared business goals to set the priorities that will enable business success. Well-defined priorities will maximize the benefits of your efforts. Evaluate internal and external customer needs involving key stakeholders, including business, development, and operations teams, to determine where to focus efforts. Evaluating customer needs will ensure that you have a thorough understanding of the support that is required to achieve business outcomes. Ensure that you are aware of guidelines or obligations defined by your organizational governance and external factors, such as regulatory compliance requirements and industry standards, that may mandate or emphasize specific focus. Validate that you have mechanisms to identify changes to internal governance and external compliance requirements. If no requirements are identified, ensure that you have applied due diligence to this determination. Review your priorities regularly so that they can be updated as needs change.

Evaluate threats to the business (for example, business risk and liabilities, and information security threats) and maintain this information in a risk registry. Evaluate the impact of risks, and tradeoffs between competing interests or alternative approaches. For example, accelerating speed to market for new features may be emphasized over cost optimization, or you may choose a relational database for non-relational data to simplify the effort to migrate a system without refactoring. Manage benefits and risks to make informed decisions when determining where to focus efforts. Some risks or choices may be acceptable for a time, it may be possible to mitigate associated risks, or it may become unacceptable to allow a risk to remain, in which case you will take action to address the risk.

Prepare

Operational excellence cannot be achieved in a vacuum. It demands a detailed understanding of each workload, what it is trying to achieve, and how it will achieve those goals. Without these insights, it's impossible to design a system that will surface its status, or to create a procedure that will effectively support that system.

Preparation: Operational Priorities

Successful operations teams are enlightened operations teams. They have a complete understanding of:

- Workloads they're responsible for
- Shared business goals
- Their role in achieving the goals
- Regulatory or compliance requirements

Only with these insights can teams prioritize their efforts. If a particular workload has stringent compliance requirements, those should be prioritized ahead of, say, monitoring enhancements, for example. AWS provides a number of resources to help teams set their operational priorities:

- ✓ AWS Support, including the AWS Knowledge Center, AWS Discussion Forums, and AWS Support Center
- ✓ AWS Documentation, which is now available on GitHub as an open source project
- ✓ AWS Cloud Compliance
- ✓ AWS Trusted Advisor, which includes core checks for environmental improvements

In addition, AWS Certified Partners like Mission can act as an extension of your operations team, providing consulting, professional services, and managed services to ensure that you are setting your operational priorities in such a way that you'll be supporting shared business goals.

Preparation: Design for Operations

Well-architected workloads intentionally consider deployment, updates, and operations in their design. They're observable by design, with logging, instrumentation, and metrics baked in from the start.

AWS enables you to model your entire workload in code, including applications, infrastructure, policy, governance, and operations. Applying rigorous engineering discipline to not only your application code, but to your entire stack, ensure that you're designed from the start for operations. AWS provides a huge number of tools and services that enable you to design for operations, including CloudFormation and the AWS Developer Tools.

Operate

Based upon shared business goals, operations teams should create, publish, and agree upon key metrics and outcomes to define operational success for their business and workload. Clear definitions help your teams to respond to events quickly, and in ways that directly impact your business goals.

To operate successfully, your team must first understand, and then respond.

By sending log data to CloudWatch Logs, baselines can be established that define "normal." CloudWatch Dashboards can then be used to create system level and business level views of those key metrics.

Amazon ElasticSearch with Kibana can help create even more detailed visualizations for your operational health metrics.

For monitoring portions of your workload that are delegated to AWS through the shared responsibility model, you can leverage the AWS Service Health Dashboard and the Personal Health

Dashboard. If you have an AWS support subscription, you can integrate with the Personal Health Dashboard API.

In addition to the tools and services provided by AWS, consider integrating other best of breed tools and services like Logstash and Grafana.

Operation: Responding to Events

Being operationally excellent doesn't guarantee that you won't have to deal with operational events. That said, operational excellence requires you to properly anticipate that failures will happen, and be ready to respond quickly and effectively by leveraging your operational health metrics, processes, and procedures.

Evolve

The most important predictor of success is a passion for learning. Teams that want to achieve operational excellence need to cultivate a culture of curiosity and continuous improvement, where every experience is an opportunity first to learn, and then to share those lessons far and wide.

Evolution: Learning from Experience

While no operations team looks forward to production issues, I've found that the best teams enjoy digging in to learn from failure. As a leader, encouraging operations teams to analyze, experiment, and improve will pay great dividends over time.

AWS provides an extensive platform to enable analysis and experimentation:

Amazon CloudWatch and CloudTrail can be combined with Amazon ElasticSearch with Kibana.

Exporting large amounts of data to Amazon S3 enables analysis with Amazon Athena and Amazon QuickSight, including rich visualizations to help your teams gain insights.

When experimenting and evolving, ensure to pull in other parts of the business to add their points of view. Frequently, new opportunities for improvement will surface when additional perspectives are solicited.

Evolution: Share Learnings

Because of Mission's status as an AWS Certified MSP, we have an opportunity to learn from hundreds of workloads, and dozens of use cases. Mission's engineers enjoy little more than evolving our platform based upon those lessons. Every improvement we make rolls out to all of our customers over time, giving maximum benefit to our entire customer base. Similarly, many organizations have multiple product and operations teams. By sharing lessons broadly, you give the entire company the benefit of your own evolution.

AWS enables sharing of best practices. Your teams can define shared libraries for implementing best practices, including CloudFormation templates, Chef Cookbooks or Ansible Playbooks, Lambda functions for common operational tasks, and more. When sharing resources, leverage AWS IAM to define permissions enabling controlled access.

3) What are the design principles that strengthen system security?

The Security pillar includes the ability to protect data, systems, and assets to take advantage of cloud technologies to improve your security.

There are seven design principles for security in the cloud:

1. Implement a strong identity foundation
2. Enable traceability
3. Apply security at all layers
4. Automate security best practices
5. Protect data in transit and at rest
6. Keep people away from data
7. Prepare for security events

Before you architect any workload, you need to put in place practices that influence security. You'll want to control who can do what. In addition, you want to be able to identify security incidents, protect your systems and services, and maintain the confidentiality and integrity of data through data protection.

You should have a well-defined and practiced process for responding to security incidents. These tools and techniques are important because they support objectives such as preventing financial loss or complying with regulatory obligations.

The AWS Shared Responsibility Model enables organizations that adopt the cloud to achieve their security and compliance goals. Because AWS physically secures the infrastructure that supports our cloud services, as an AWS customer you can focus on using services to accomplish your goals. The AWS Cloud also provides greater access to security data and an automated approach to responding to security events.

4) What are the design principles that increase reliability?

In the cloud, there are a few principles that can help you increase reliability.

1. Automatically recover from failure: By monitoring a workload for key performance indicators (KPIs), you can trigger automation when a threshold is breached. These KPIs should be a measure of business value, not of the technical aspects of the operation of the service. This allows for automatic notification and tracking of failures, and for automated recovery processes that work around or repair the failure. With more sophisticated automation, it's possible to anticipate and remediate failures before they occur.
2. Test recovery procedures: In an on-premises environment, testing is often conducted to prove that the workload works in a particular scenario. Testing is not typically used to validate recovery strategies. In the cloud, you can test how your workload fails, and you can validate your recovery procedures. You can use automation to simulate different failures or to recreate scenarios that led to failures before. This approach exposes failure pathways that you can test and fix *before* a real failure scenario occurs, thus reducing risk.
3. Scale horizontally to increase aggregate workload availability: Replace one large resource with multiple small resources to reduce the impact of a single failure on the overall workload. Distribute requests across multiple, smaller resources to ensure that they don't share a common point of failure.
4. Stop guessing capacity: A common cause of failure in on-premises workloads is resource saturation, when the demands placed on a workload exceed the capacity of that workload

(this is often the objective of denial of service attacks). In the cloud, you can monitor demand and workload utilization, and automate the addition or removal of resources to maintain the optimal level to satisfy demand without over- or under-provisioning. There are still limits, but some quotas can be controlled and others can be managed.

5. Manage change through automation: Changes to your infrastructure should be made using automation. The changes that need to be managed include changes to the automation, which then can be tracked and reviewed.

5) What are the areas to focus on to achieve performance efficiency in the cloud?

Focus on the following areas to achieve performance efficiency in the cloud:

1. Selection

The optimal solution for a particular workload varies, and solutions often combine multiple approaches. Well-architected workloads use multiple solutions and enable different features to improve performance.

AWS resources are available in many types and configurations, which makes it easier to find an approach that closely matches your needs. You can also find options that are not easily achievable with on-premises infrastructure. For example, a managed service such as Amazon DynamoDB provides a fully managed NoSQL database with single-digit millisecond latency at any scale.

- Performance architecture selection
- Compute architecture selection
- Storage Architecture Selection
- Database architecture selection
- Network architecture selection

2. Review

When architecting workloads, there are finite options that you can choose from. However, over time, new technologies and approaches become available that could improve the performance of your workload. In the cloud, it's much easier to experiment with new features and services because your infrastructure is code.

3. Monitoring

After you implement your architecture you must monitor its performance so that you can remediate any issues before they impact your customers. Monitoring metrics should be used to raise alarms when thresholds are breached.

CloudWatch is a monitoring service for AWS Cloud resources and the workloads that run on AWS.

4. Trade-offs

When you architect solutions, think about trade-offs to ensure an optimal approach. Depending on your situation, you could trade consistency, durability, and space for time or latency, to deliver higher performance.

Using AWS, you can go global in minutes and deploy resources in multiple locations across the globe to be closer to your end users. You can also dynamically add read-only replicas to information stores (such as database systems) to reduce the load on the primary database.

AWS offers caching solutions such as Amazon ElastiCache, which provides an in-memory data store or cache, and Amazon CloudFront, which caches copies of your static content closer to end users. Amazon DynamoDB Accelerator (DAX) provides a read-through/write-through distributed caching tier in front of Amazon DynamoDB, supporting the same API, but providing sub-millisecond latency for entities that are in the cache.

Take a data-driven approach to building a high-performance architecture. Gather data on all aspects of the architecture, from the high-level design to the selection and configuration of resource types.

Reviewing your choices on a regular basis, ensures that you are taking advantage of the continually evolving AWS Cloud. Monitoring ensures that you are aware of any deviance from expected performance. Make trade-offs in your architecture to improve performance, such as using compression or caching, or relaxing consistency requirements.

6) What are the different approaches to using AWS resources in a cost-effective manner?

There are five design principles for cost optimization in the cloud:

1. Implement Cloud Financial Management: To achieve financial success and accelerate business value realization in the cloud, you need to invest in Cloud Financial Management /Cost Optimization. Your organization needs to dedicate time and resources to build capability in this new domain of technology and usage management. Similar to your Security or Operational Excellence capability, you need to build capability through knowledge building, programs, resources, and processes to become a cost-efficient organization.
2. Adopt a consumption model: Pay only for the computing resources that you require and increase or decrease usage depending on business requirements, not by using elaborate forecasting. For example, development and test environments are typically only used for eight hours a day during the work week. You can stop these resources when they are not in use for a potential cost savings of 75% (40 hours versus 168 hours).
3. Measure overall efficiency: Measure the business output of the workload and the costs associated with delivering it. Use this measure to know the gains you make from increasing output and reducing costs.
4. Stop spending money on undifferentiated heavy lifting: AWS does the heavy lifting of data center operations like racking, stacking, and powering servers. It also removes the operational burden of managing operating systems and applications with managed services. This allows you to focus on your customers and business projects rather than on IT infrastructure.
5. Analyze and attribute expenditure: The cloud makes it easier to accurately identify the usage and cost of systems, which then allows transparent attribution of IT costs to individual workload owners. This helps measure return on investment (ROI) and gives workload owners an opportunity to optimize their resources and reduce costs.

IV Cloud Foundations 1

1) Define what IaaS, PaaS and SaaS is.

IaaS, PaaS and SaaS are three main categories of cloud computing services.

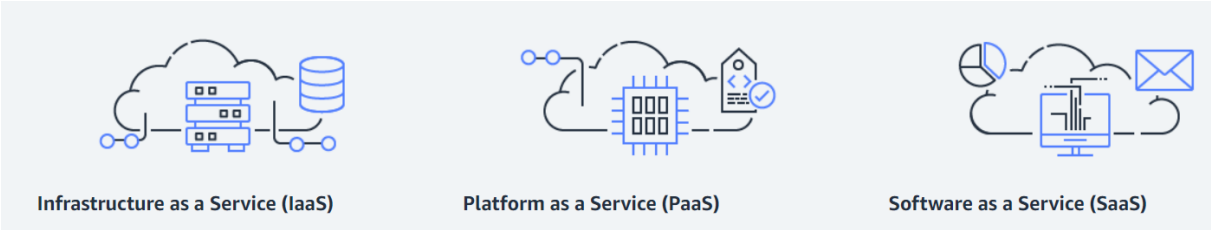


Figure 3 Iaas, Paas and SaaS

1. Infrastructure as a Service (IaaS): IaaS refers to the delivery of computing infrastructure—typically servers, storage, and networking—as a fully outsourced service over the internet. With IaaS, customers rent IT infrastructure on a pay-per-use basis from a cloud service provider, rather than building and maintaining their own in-house infrastructure.
   Example : amazon EC2
2. Platform as a Service (PaaS): PaaS provides a platform for customers to develop, run, and manage their applications without having to worry about the underlying infrastructure. This includes web servers, databases, and development tools, among other things. PaaS is usually delivered over the internet and is designed to support a specific programming language or development framework.
   Example: Amazon Elastic Beanstalk
3. Software as a Service (SaaS): SaaS is the delivery of applications over the internet, on a subscription basis. With SaaS, customers can use software applications without having to install or maintain them on their own computers. SaaS applications are hosted and maintained by the provider, who also handles security, availability, and scalability. Common examples of SaaS include customer relationship management (CRM) systems, email services, and productivity tools.
   Example: Gmail, amazon Rekognition for machine learning

2) Provide 6 advantages of cloud computing.

1. **Trade fixed expense for variable expense** – Instead of having to invest heavily in data centers and servers before you know how you're going to use them, you can pay only when you consume computing resources, and pay only for how much you consume.
2. **Benefit from massive economies of scale** – By using cloud computing, you can achieve a lower variable cost than you can get on your own. Because usage from hundreds of thousands of customers is aggregated in the cloud, providers such as AWS can achieve higher economies of scale, which translates into lower pay as-you-go prices.
3. **Stop guessing capacity** – Eliminate guessing on your infrastructure capacity needs. When you make a capacity decision prior to deploying an application, you often end up either sitting on expensive idle resources or dealing with limited capacity. With cloud computing, these

problems go away. You can access as much or as little capacity as you need, and scale up and down as required with only a few minutes' notice.

4. **Increase speed and agility** – In a cloud computing environment, new IT resources are only a click away, which means that you reduce the time to make those resources available to your developers from weeks to just minutes. This results in a dramatic increase in agility for the organization, since the cost and time it takes to experiment and develop is significantly lower.
5. **Stop spending money running and maintaining data centers** – Focus on projects that differentiate your business, not the infrastructure. Cloud computing lets you focus on your own customers, rather than on the heavy lifting of racking, stacking, and powering servers.
6. **Go global in minutes** – Easily deploy your application in multiple regions around the world with just a few clicks. This means you can provide lower latency and a better experience for your customers at minimal cost.

3) Define what an AWS region and an Availability Zone is.

AWS Cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data center facilities are located in different physical locations. These locations are categorized by regions and Availability Zones.

AWS Regions are large and widely dispersed into separate geographic locations. Availability Zones are distinct locations within an AWS Region that are engineered to be isolated from failures in other Availability Zones. They provide inexpensive, low-latency network connectivity to other Availability Zones in the same AWS Region.

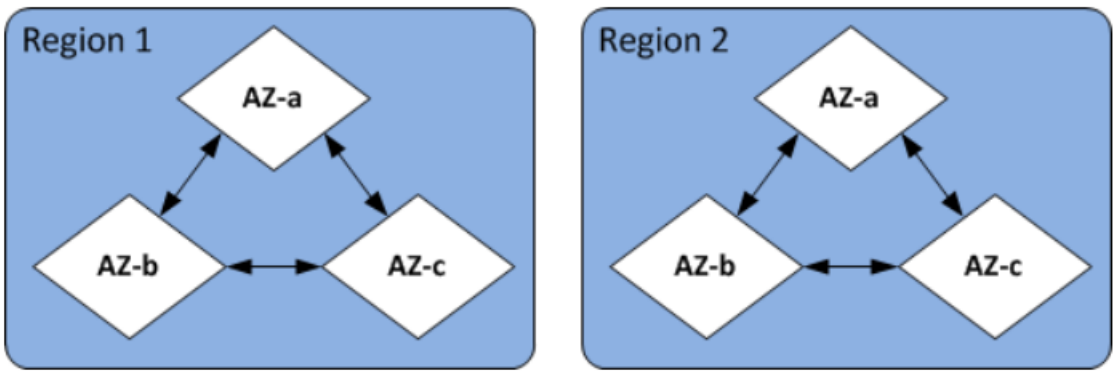


Figure 4- Regions and Availability zones

Note: Each region is completely independent. Any ElastiCache activity you initiate (for example, creating clusters) runs only in your current default region.

4) List all the AWS regions.

Amazon cloud computing resources are hosted in multiple locations world-wide. These locations are composed of AWS Regions, Availability Zones, and Local Zones. Each *AWS Region* is a separate geographic area. Each AWS Region has multiple, isolated locations known as *Availability Zones*.

The list can be viewed from this code from aws cli:

```
for region in `aws ec2 describe-regions --region us-east-1 --output text | cut -f4`
do
     echo -e "\nListing Instances in region:'$region'..."
     aws ec2 describe-instances --region $region
done
```

From Management console→VPC Dashboard→ running Instances→See all Regions

Figure 5 List of AWS Regions

5) What are the categories in which the AWS services are grouped?

1. Compute: This category includes services for running computing workloads, such as Amazon Elastic Compute Cloud (EC2) for scalable virtual machines and AWS Lambda for event-driven, serverless computing.
2. Storage: This category includes services for storing and retrieving data, such as Amazon Simple Storage Service (S3) for object storage and Amazon Elastic Block Store (EBS) for persistent block-level storage.
3. Database: This category includes services for managing relational and NoSQL databases, such as Amazon Relational Database Service (RDS) for SQL databases and Amazon DynamoDB for NoSQL databases.
4. Networking and Content Delivery: This category includes services for connecting and securing networks, such as Amazon Virtual Private Cloud (VPC) for secure and isolated cloud resources and Amazon CloudFront for content delivery.

5. Management Tools: This category includes services for managing and monitoring cloud resources, such as AWS CloudFormation for provisioning and managing infrastructure as code and Amazon CloudWatch for monitoring and logging.
6. Security, Identity & Compliance: This category includes services for securing data and applications, such as AWS Key Management Service (KMS) for encryption and Amazon GuardDuty for threat detection.
7. Analytics: This category includes services for processing and analyzing big data, such as Amazon Redshift for data warehousing and Amazon QuickSight for business intelligence.
8. Machine Learning: This category includes services for building and training machine learning models, such as Amazon SageMaker for building and deploying machine learning models.
9. Mobile Services: This category includes services for building mobile applications, such as Amazon Pinpoint for push notifications and Amazon Mobile Analytics for mobile app analytics.
10. Application Services: This category includes services for building and deploying applications, such as Amazon Simple Queue Service (SQS) for queuing and Amazon API Gateway for building and deploying APIs.

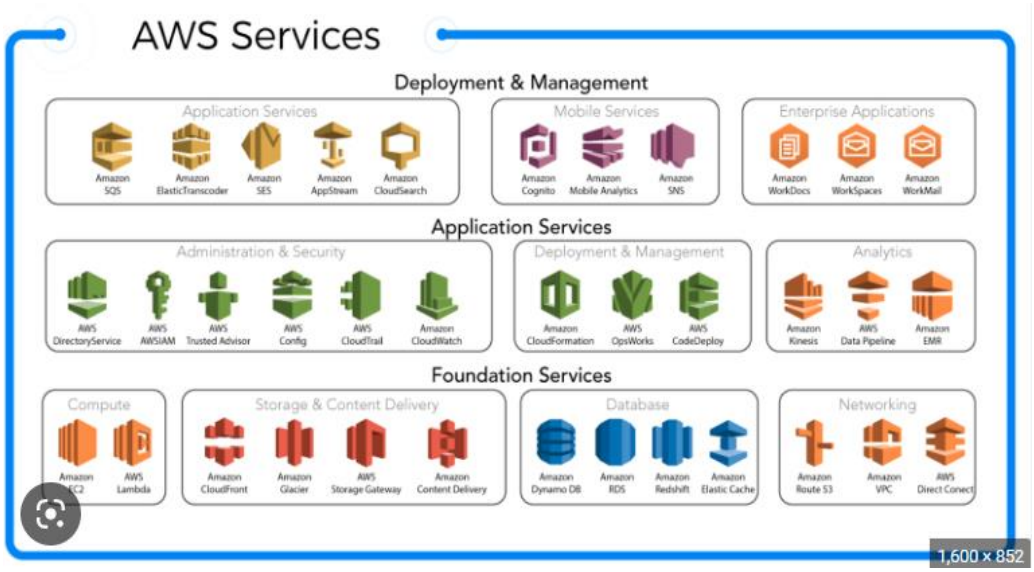


Figure 6 AWS Service Categories

6) What is the difference between object storage and block storage?

Cloud storage is a cloud computing model that enables storing data and files on the internet through a cloud computing provider that you access either through the public internet or a dedicated private network connection. The provider securely stores, manages, and maintains the storage servers, infrastructure, and network to ensure you have access to the data when you need it at virtually unlimited scale, and with elastic capacity. Cloud storage removes the need to buy and manage your own data storage infrastructure, giving you agility, scalability, and durability, with any time, anywhere data access.

There are three types of cloud storage: object, file, and block. Each is ideal for specific use cases and storage requirements.

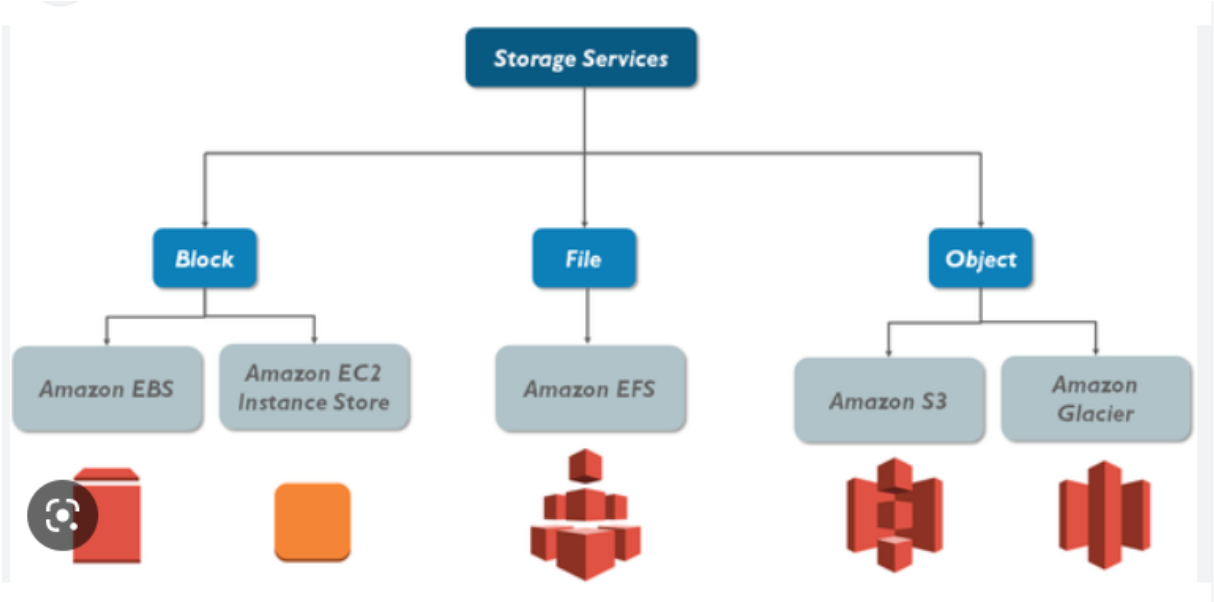


Figure 7 AWS Storage Types

Object storage

Organizations have to store a massive and growing amount of unstructured data, such as photos, videos, machine learning (ML), sensor data, audio files, and other types of web content, and finding scalable, efficient, and affordable ways to store them can be a challenge. Object storage is a data storage architecture for large stores of unstructured data. Objects store data in the format it arrives in and makes it possible to customize metadata in ways that make the data easier to access and analyze. Instead of being organized in files or folder hierarchies, objects are kept in secure buckets that deliver virtually unlimited scalability. It is also less costly to store large data volumes.

Applications developed in the cloud often take advantage of the vast scalability and metadata characteristics of object storage. Object storage solutions are ideal for building modern applications from scratch that require scale and flexibility, and can also be used to import existing data stores for analytics, backup, or archive.

Object storage, as provided by Amazon Simple Storage Service (S3), is a highly scalable and durable storage solution that is optimized for unstructured data, such as text files, images, videos, and backups. S3 is designed to store large amounts of data and provides unlimited capacity, so you can store as much data as you need, for as long as you need. Data in S3 is organized into buckets and can be accessed over HTTP or HTTPS.

Block storage

Enterprise applications like databases or enterprise resource planning (ERP) systems often require dedicated, low-latency storage for each host. This is analogous to direct-attached storage (DAS) or a storage area network (SAN). In this case, you can use a cloud storage service that stores data in the form of blocks. Each block has its own unique identifier for quick storage and retrieval.

Block-based cloud storage solutions are provisioned with each virtual server and offer the ultra-low latency required for high-performance workloads. Block storage provides low latency and high-performance values in various use cases. Its features are primarily useful for structured database storage, VM file system volumes, and high volumes of read and write loads.

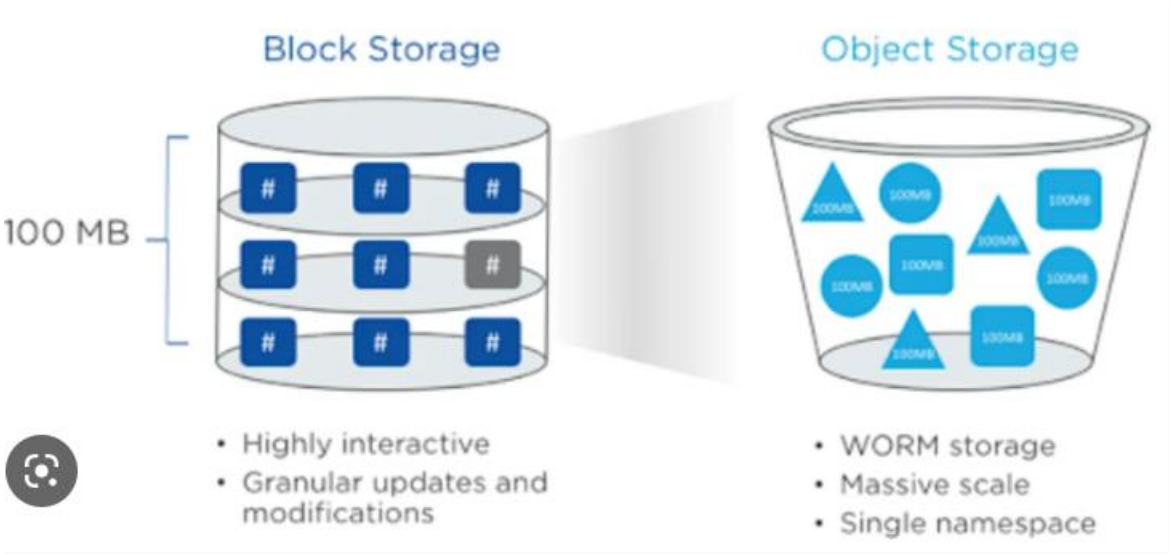


Figure 8 Object and Block Storage

7) List two AWS compute services and explain them.

AWS compute services

- Amazon EC2
- Amazon EC2 Auto Scaling
- Amazon EC2 Image Builder
- Amazon Lightsail
- AWS App Runner
- AWS Batch
- AWS Elastic Beanstalk
- AWS Fargate
- AWS Lambda
- AWS Serverless Application Repository
- AWS Outposts
- AWS Wavelength
- VMware Cloud on AWS

Category	*AWS Services*

Instances (virtual machines)	• **Amazon Elastic Compute Cloud** (Amazon EC2) — Secure and resizeable compute capacity (virtual servers) in the cloud • **Amazon EC2 Spot Instances**— Run fault-tolerant workloads for up to 90% off • **Amazon EC2 Auto Scaling** — Automatically add or remove compute capacity to meet changes in demand • **Amazon Lightsail** — Easy-to-use cloud platform that offers you everything you need to build an application or website • **AWS Batch** — Fully managed batch processing at any scale
Containers	• **Amazon Elastic Container Service (Amazon ECS)** — Highly secure, reliable, and scalable way to run containers • **Amazon ECS Anywhere** — Run containers on customer-managed infrastructure • **Amazon Elastic Container Registry** (Amazon ECR) — Easily store, manage, and deploy container images • **Amazon Elastic Kubernetes Service** (Amazon EKS) — Fully managed Kubernetes service • **Amazon EKS Anywhere** — Create and operate Kubernetes clusters on your own infrastructure • **AWS Fargate** — Serverless compute for containers • **AWS App Runner** — Build and run containerized applications on a fully managed service
Serverless	• **AWS Lambda** — Run code without thinking about servers. Pay only for the compute time you consume.
Edge and hybrid	• **AWS Outposts** — Run AWS infrastructure and services on premises for a truly consistent hybrid experience • **AWS Snow Family** — Collect and process data in rugged or disconnected edge environments • **AWS Wavelength** — Deliver ultra-low latency application for 5G devices • **VMware Cloud on AWS** — Preferred service for all vSphere workloads to rapidly extend and migrate to the cloud • **AWS Local Zones** — Run latency sensitive applications closer to end-users

Cost and capacity management	• **AWS Savings Plan** — Flexible pricing model that provides savings of up to 72% on AWS compute usage • **AWS Compute Optimizer** — Recommends optimal AWS compute resources for your workloads to reduce costs and improve performance • **AWS Elastic Beanstalk** — Easy-to-use service for deploying and scaling web applications and services • **EC2 Image Builder** — Build and maintain secure Linux or Windows Server images • **Elastic Load Balancing** (ELB) — Automatically distribute incoming application traffic across multiple targets

1. Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers.

The simple web interface of Amazon EC2 allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances (called Amazon EC2 instances) to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers and system administrators the tools to build failure resilient applications and isolate themselves from common failure scenarios.

Instance types

Amazon EC2 passes on to you the financial benefits of Amazon scale. You pay a very low rate for the compute capacity you actually consume. Refer to Amazon EC2 Instance Purchasing Options for a more detailed description.

- On-Demand Instances — With On-Demand Instances, you pay for compute capacity by the hour or the second depending on which instances you run. No longer-term commitments or upfront payments are needed. You can increase or decrease your compute capacity depending on the demands of your application and only pay the specified per hourly rates for the instance you use. On-Demand Instances are recommended for:
    - Users that prefer the low cost and flexibility of Amazon EC2 without any up-front payment or long-term commitment

  - Applications with short-term, spiky, or unpredictable workloads that cannot be interrupted
  - Applications being developed or tested on Amazon EC2 for the first time
- Spot Instances —Spot Instances are available at up to a 90% discount compared to On-Demand prices and let you take advantage of unused Amazon EC2 capacity in the AWS Cloud. You can significantly reduce the cost of running your applications, grow your application's compute capacity and throughput for the same budget, and enable new types of cloud computing applications. Spot Instances are recommended for:
  - Applications that have flexible start and end times
  - Applications that are only feasible at very low compute prices
  - Users with urgent computing needs for large amounts of additional capacity
- Reserved Instances—Reserved Instances provide you with a significant discount (up to 72%) compared to On-Demand Instance pricing. You have the flexibility to change families, operating system types, and tenancies while benefitting from Reserved Instance pricing when you use Convertible Reserved Instances.
- Savings Plans—Savings Plans are a flexible pricing model that offer low prices on EC2 and Fargate usage, in exchange for a commitment to a consistent amount of usage (measured in $/hour) for a one or three year term.
- Dedicated Hosts—A Dedicated Host is a physical EC2 server dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server (subject to your license terms), and can also help you meet compliance requirements.

2. AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and Internet Information Services (IIS).

You can simply upload your code, and AWS Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, and auto scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.

8) List two AWS storage services and explain them.

1. Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management

features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

Figure 9 S3- How it works

Use cases

- Build a data lake - Run big data analytics, artificial intelligence (AI), machine learning (ML), and high performance computing (HPC) applications to unlock data insights.
- Archive data at the lowest cost - Move data archives to the Amazon S3 Glacier storage classes to lower costs, eliminate operational complexities, and gain new insights.

2. Amazon File Cache provides a high-speed cache on AWS that makes it easier to process file data, regardless of where it's stored. Amazon File Cache serves as temporary, high-performance storage for data on premises or on AWS. The service allows you to make dispersed datasets available to file-based applications on AWS with a unified view and high speeds.

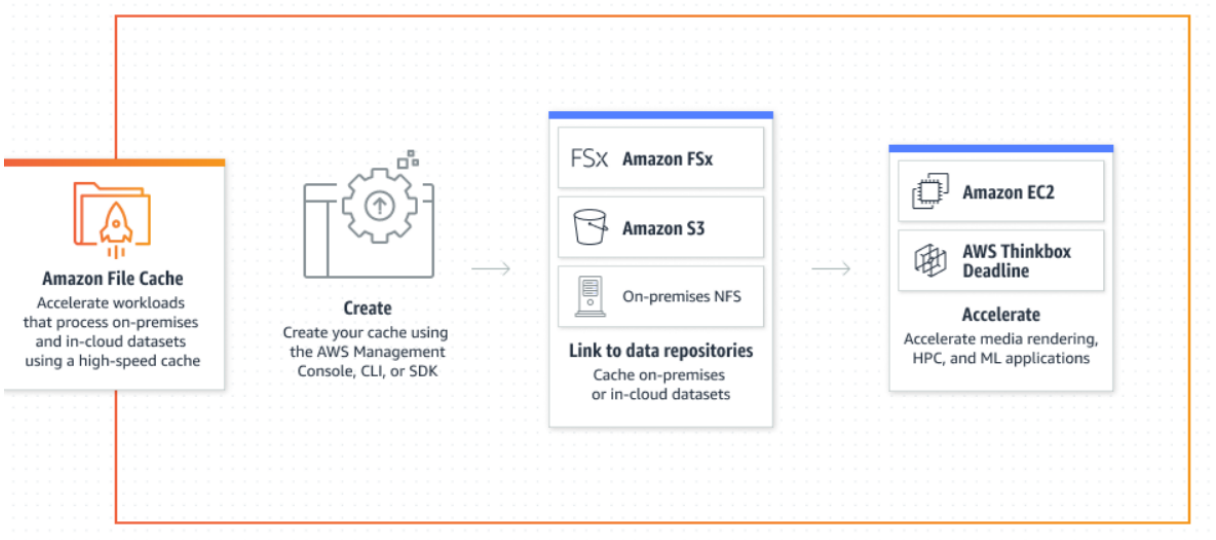


Figure 10 File Cache - How it works

Use cases

- Increase media workload agility - Burst visual effects (VFX) rendering and transcoding workloads to AWS to meet peak compute needs during media production.

- Reduce machine learning (ML) training time - Provide instant access to your on-premises and in-cloud datasets and maximize throughput to your training instances.

V Cloud Foundations 2

1) Explain the AWS Shared responsibility model.

Security and Compliance is a shared responsibility between AWS and the customer. This shared model can help relieve the customer's operational burden as AWS operates, manages and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the service operates.

The AWS Shared Responsibility model defines the security responsibilities between AWS and its customers. Under this model, AWS is responsible for the security of the cloud infrastructure, including the physical security of data centers, networking, and hardware, while the customers are responsible for securing the data and applications they run on AWS.

The Shared Responsibility model consists of two parts:

1. Security of the Cloud: AWS is responsible for securing the cloud infrastructure that includes the compute, storage, database, and networking services offered by AWS. AWS is responsible for the physical security of data centers, hardware, and software infrastructure.

2. Security in the Cloud: Customers are responsible for securing their data and applications in the cloud. This includes the management of access credentials, data encryption, security configurations, patch management, and network security. Customers are also responsible for their applications, including their operating systems, applications, and data.

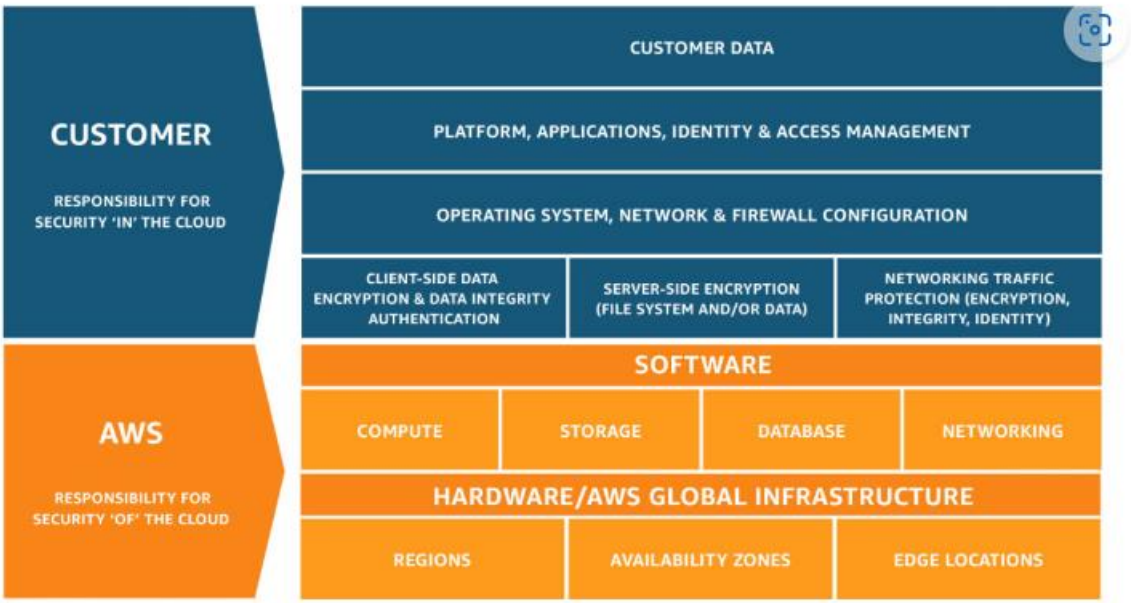


Figure 11 AWS Shared Responsibility model

2) Explain an Identity and Access Management (IAM) Role.

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. For example, you might want to grant users in your AWS account access to resources they don't usually have, or grant users in one AWS account access to resources in another account. Or you might want to allow a mobile app to use AWS resources, but not want to embed AWS keys within the app (where they can be difficult to rotate and where users can potentially extract them). Sometimes you want to give AWS access to users who already have identities defined outside of AWS, such as in your corporate directory. Or, you might want to grant access to your account to third parties so that they can perform an audit on your resources.

3) Explain an Identity and Access Management (IAM) Policy.

An Identity and Access Management (IAM) policy in AWS is a document that defines permissions and access control rules for AWS resources. IAM policies are used to grant or deny access to AWS resources based on specific conditions, such as the user's identity, the resource type, or the action being performed.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, if a policy allows the GetUser action, then a user with that policy can get user information from the AWS Management Console, the AWS CLI, or the AWS API.

An IAM policy is made up of one or more statements, where each statement defines a specific permission or access control rule. Each statement includes the following components:

1. Effect: This defines whether the policy statement is granting or denying access to a resource.
2. Action: This specifies the actions that are allowed or denied for the resource. For example, "s3:GetObject" to allow read access to an S3 object.
3. Resource: This specifies the AWS resource that the policy statement applies to. For example, "arn:aws:s3:::example-bucket/*" to apply the policy to all objects in a specific S3 bucket.
4. Condition: This provides additional conditions that must be met for the policy statement to be applied. For example, restricting access to a resource based on a specific IP address range.

IAM policies can be attached to users, groups, or roles in order to grant or deny access to AWS resources.

In summary, IAM policies are an important tool for controlling access to AWS resources. They allow you to define fine-grained permissions and access control rules based on specific conditions, such as the user's identity or the resource type. IAM policies are a critical part of securing your AWS environment and ensuring that only authorized users have access to your resources.

4) Describe an Amazon Machine Image (AMI).

An Amazon Machine Image (AMI) is a supported and maintained image provided by AWS that provides the information required to launch an instance. You must specify an AMI when you launch an instance. You can launch multiple instances from a single AMI when you require multiple instances with the same configuration. You can use different AMIs to launch instances when you require instances with different configurations.

An AMI includes the following:

- One or more Amazon Elastic Block Store (Amazon EBS) snapshots, or, for instance-store-backed AMIs, a template for the root volume of the instance (for example, an operating system, an application server, and applications).
- Launch permissions that control which AWS accounts can use the AMI to launch instances.
- A block device mapping that specifies the volumes to attach to the instance when it's launched.

5) List the different EC2 instance types with use cases for each type.

1. General Purpose

General purpose instances provide a balance of compute, memory and networking resources, and can be used for a variety of diverse workloads. These instances are ideal for applications that use these resources in equal proportions such as web servers and code repositories.

Use cases : Applications built on open-source software such as application servers, microservices, gaming servers, midsize data stores, and caching fleets.

2. Compute Optimized

Compute Optimized instances are ideal for compute bound applications that benefit from high performance processors. Instances belonging to this family are well suited for batch processing workloads, media transcoding, high performance web servers, high performance computing (HPC), scientific modeling, dedicated gaming servers and ad server engines, machine learning inference and other compute intensive applications.

Use Cases: High performance computing (HPC), batch processing, ad serving, video encoding, gaming, scientific modelling, distributed analytics, and CPU-based machine learning inference.

3. Memory Optimized

Memory optimized instances are designed to deliver fast performance for workloads that process large data sets in memory.

Use cases: Memory-intensive workloads such as open-source databases, in-memory caches, and real-time big data analytics.

4. Accelerated Computing

Accelerated computing instances use hardware accelerators, or co-processors, to perform functions, such as floating point number calculations, graphics processing, or data pattern matching, more efficiently than is possible in software running on CPUs.

Use Cases: Machine learning, high performance computing, computational fluid dynamics, computational finance, seismic analysis, speech recognition, autonomous vehicles, and drug discovery.

5. Storage Optimized

Storage optimized instances are designed for workloads that require high, sequential read and write access to very large data sets on local storage. They are optimized to deliver tens of thousands of low-latency, random I/O operations per second (IOPS) to applications.

Use Cases: These instances maximize the number of transactions processed per second (TPS) for I/O intensive and business-critical workloads which have medium size data sets and can benefit from high compute performance and high network throughput such as relational databases (MySQL, MariaDB, and PostgreSQL), and NoSQL databases (KeyDB, ScyllaDB, and Cassandra). They are also an ideal fit for workloads that require very fast access to medium size data sets on local storage such as search engines and data analytics workloads

6. HPC Optimized

High performance computing (HPC) instances are purpose built to offer the best price performance for running HPC workloads at scale on AWS. HPC instances are ideal for applications that benefit from high-performance processors such as large, complex simulations and deep learning workloads

6) Explain Virtual Private Cloud (VPC).

Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This virtual network closely resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

The following features help you configure a VPC to provide the connectivity that your applications need:

Virtual private clouds (VPC)

A VPC is a virtual network that closely resembles a traditional network that you'd operate in your own data center. After you create a VPC, you can add subnets.

- Subnets

A subnet is a range of IP addresses in your VPC. A subnet must reside in a single Availability Zone. After you add subnets, you can deploy AWS resources in your VPC.

- IP addressing

You can assign IPv4 addresses and IPv6 addresses to your VPCs and subnets. You can also bring your public IPv4 and IPv6 GUA addresses to AWS and allocate them to resources in your VPC, such as EC2 instances, NAT gateways, and Network Load Balancers.

- Routing

Use route tables to determine where network traffic from your subnet or gateway is directed.

- Gateways and endpoints

A gateway connects your VPC to another network. For example, use an internet gateway to connect your VPC to the internet. Use a VPC endpoint to connect to AWS services privately, without the use of an internet gateway or NAT device.

- Peering connections

Use a VPC peering connection to route traffic between the resources in two VPCs.

- Traffic Mirroring

Copy network traffic from network interfaces and send it to security and monitoring appliances for deep packet inspection.

- Transit gateways

Use a transit gateway, which acts as a central hub, to route traffic between your VPCs, VPN connections, and AWS Direct Connect connections.

- VPC Flow Logs

A flow log captures information about the IP traffic going to and from network interfaces in your VPC.

- VPN connections

Connect your VPCs to your on-premises networks using AWS Virtual Private Network (AWS VPN).

7) Differentiate between a Public and a Private subnet

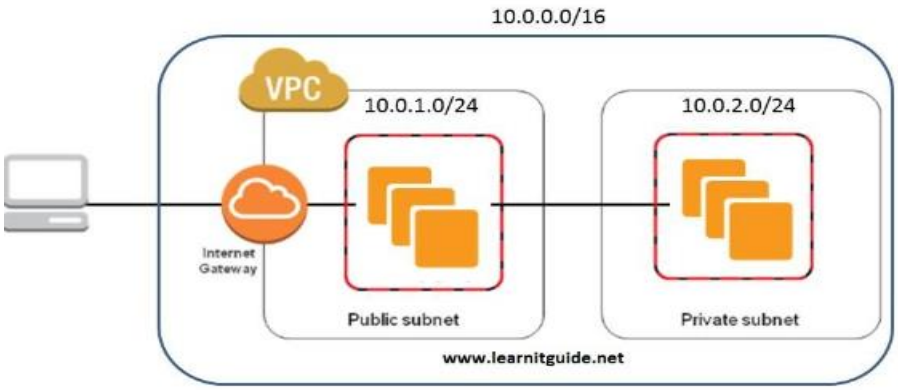
In AWS, subnets are logical partitions of an Amazon Virtual Private Cloud (VPC) that are used to isolate resources and control network traffic. Subnets can be either public or private, depending on their configuration and accessibility.

The main differences between a public subnet and a private subnet are:

1. Public Subnet: A public subnet is a subnet that is directly accessible from the internet. It is associated with a route table that has a default route to the internet gateway, which enables resources in the subnet to communicate with the internet. Resources in a public subnet typically have public IP addresses or Elastic IP addresses (EIPs) assigned to them.

2. Private Subnet: A private subnet is a subnet that is not directly accessible from the internet. It is associated with a route table that does not have a default route to the internet gateway, which prevents resources in the subnet from communicating with the internet. Resources in a private subnet typically have only private IP addresses assigned to them.

The key difference between a public and private subnet is their accessibility to the internet. Public subnets are directly accessible from the internet, while private subnets are not. Public subnets are typically used for resources that require direct internet access, such as web servers or load balancers, while private subnets are used for resources that do not require direct internet access, such as databases or backend application servers.



VI AWS CloudFormation

1) What is Configuration Orchestration?

Configuration Orchestration in AWS is the process of managing and automating the deployment and configuration of resources and applications in an AWS environment. It involves defining and managing the configuration of resources such as EC2 instances, databases, load balancers, and security groups, as well as deploying and managing applications on those resources.

AWS provides several tools for configuration orchestration, including AWS CloudFormation, AWS Elastic Beanstalk, and AWS OpsWorks. These tools allow users to define the desired state of their infrastructure and applications in code and automate the deployment and configuration process.

For example, with AWS CloudFormation, users can define their infrastructure as code using a JSON or YAML template, which specifies the resources needed to run their application. CloudFormation then automatically provisions and configures those resources, ensuring that they are always in the desired state.

Overall, configuration orchestration in AWS helps to streamline the deployment and management of infrastructure and applications, while also improving consistency and reducing errors.

2) What is Configuration Management? List some commonly used tools for Configuration Management.

Configuration Management is the process of managing the configuration of software and hardware components of an IT system. It involves keeping track of the state of these components and ensuring that they are configured correctly and consistently across different environments.

In AWS context, Configuration Management involves managing the configuration of AWS resources such as EC2 instances, databases, and network settings. This includes defining the desired state of these resources, deploying configurations, and monitoring changes to ensure they remain in the desired state.

Some commonly used tools for Configuration Management in AWS are:

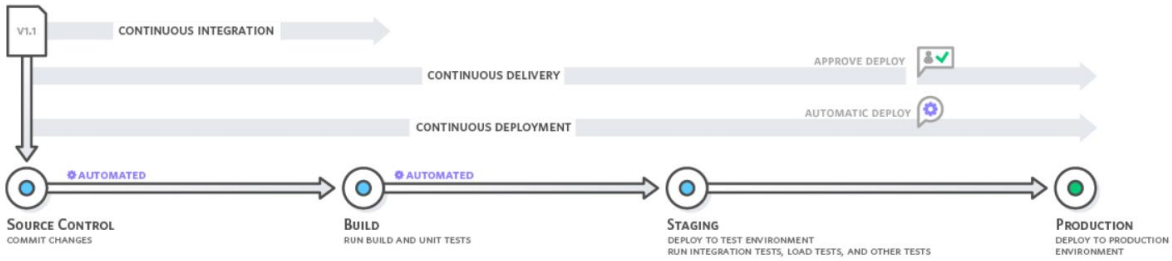
1. AWS Systems Manager: A service that helps you automatically configure and manage EC2 instances, on-premises instances, and other resources.
2. AWS Config: A service that enables you to assess, audit, and evaluate the configurations of your AWS resources.
3. AWS OpsWorks: A configuration management service that helps you deploy and operate applications of all types.
4. AWS CloudFormation: A service that lets you define and deploy infrastructure as code.
5. Ansible: A configuration management tool that helps automate the provisioning, configuration, and deployment of applications.

6. Chef: A configuration management tool that helps automate the configuration and deployment of applications.
7. Puppet: A configuration management tool that helps automate the configuration and deployment of applications.

These tools help simplify the process of managing configurations in an AWS environment, improving consistency and reducing errors.

3) What is Continuous Integration?

Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. Continuous integration most often refers to the build or integration stage of the software release process and entails both an automation component (e.g. a CI or build service) and a cultural component (e.g. learning to integrate frequently). The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.



Continuous integration refers to the build and unit testing stages of the software release process. Every revision that is committed triggers an automated build and test.

Continuous Integration helps to reduce the risk of errors and conflicts in software development, improve code quality, and speed up the overall development process. It also promotes collaboration among team members and helps to ensure that everyone is working towards a common goal.

4) What is Continuous delivery?

Continuous delivery is a software development practice where code changes are automatically prepared for a release to production. A pillar of modern application development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When properly implemented, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

Continuous delivery lets developers automate testing beyond just unit tests so they can verify application updates across multiple dimensions before deploying to customers. These tests may include UI testing, load testing, integration testing, API reliability testing, etc. This helps developers more thoroughly validate updates and pre-emptively discover issues. With the cloud, it is easy and

cost-effective to automate the creation and replication of multiple environments for testing, which was previously difficult to do on-premises.

5) What is AWS CloudFormation? List 3 advantages of Cloud Formation.

AWS CloudFormation enables you to create and provision AWS infrastructure deployments predictably and repeatedly. You can use AWS CloudFormation on services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), Amazon Simple Notification Service (Amazon SNS), Elastic Load Balancing, and Auto Scaling. AWS CloudFormation enables you to use a template file to create and delete a collection of resources, which are managed together as a single unit (a stack). Advantages of using CloudFormation are:

1. Simplify infrastructure management
2. Quickly replicate your infrastructure
3. Easily control and track changes to your infrastructure

AWS CloudFormation simplifies the process of managing and provisioning AWS infrastructure and resources, making it easier to deploy applications and manage changes to your infrastructure over time.

6) What is JSON and YAML? List out 3 differences between them.

JSON and YAML are two popular formats for storing and transmitting data. Both formats are human-readable, text-based, and easy to parse, making them widely used in a variety of contexts, including web development, configuration management, and data serialization.

Here are three differences between JSON and YAML:

	JSON	*YAML*
Syntax	JSON uses a syntax that is based on a subset of the JavaScript language	YAML uses a more compact and intuitive syntax that is designed to be easy to read and write
Data Types	JSON supports a limited set of data types, including strings, numbers, booleans, arrays, and objects	YAML supports a broader range of data types, including strings, numbers, booleans, arrays, objects, and even more complex data structures such as dates, times, and regular expressions.
Comments	JSON does not support comments	Comments are a way to annotate code with human-readable notes and are useful for documenting code and explaining its behaviour. YAML supports both single-line and multi-line comments, making it easier to document your code and communicate with other developers.

Overall, JSON and YAML have different syntax and data types, and each format has its strengths and weaknesses depending on the context in which it is used. However, both formats are widely used and have a large community of developers and tools that support them.

7) What is a stack in AWS CloudFormation?

In AWS CloudFormation, a stack is a collection of AWS resources that are created and managed together as a single unit. The resources in a stack are defined by a CloudFormation template, which specifies the properties and configurations of the resources, as well as any dependencies between them. When you create a stack, AWS CloudFormation provisions the resources in the template, and then manages them as a single unit. This means that you can create, update, or delete a stack with a single operation, and AWS CloudFormation will automatically manage the underlying resources accordingly.

Stacks can include a wide range of AWS resources, such as EC2 instances, S3 buckets, RDS databases, security groups, and more. They can also be used to create complex architectures, such as multi-tier applications or entire environments.

Stacks can be managed using the AWS CloudFormation console, the AWS CLI, or programmatically using the AWS CloudFormation API. By using stacks, you can simplify the management and deployment of AWS resources, as well as automate common tasks such as rolling out updates or creating new environments.

VII AWS Billing

1) List the different types of AWS support plans.

AWS Support offers five support plans:

- Basic
- Developer
- Business
- Enterprise On-Ramp
- Enterprise

Basic Support offers support for account and billing questions and service quota increases. The other plans offer a number of technical support cases with pay-by-the-month pricing and no long-term contracts.

All AWS customers automatically have 24x7 access to these features of Basic Support:

- One-on-one responses to account and billing questions
- Support forums
- Service health checks
- Documentation, technical papers, and best practice guides

Customers with a Developer Support plan have access to these additional features:

- Best practice guidance
- Client-side diagnostic tools
- Building-block architecture support: guidance on how to use AWS products, features, and services together
- Supports an unlimited number of support cases that can be opened by one primary contact, which is the AWS account *root user*.

In addition, customers with a Business, Enterprise On-Ramp, or Enterprise Support plan have access to these features:

- Use-case guidance – What AWS products, features, and services to use to best support your specific needs.
- AWS Trusted Advisor – A feature of AWS Support, which inspects customer environments and identifies opportunities to save money, close security gaps, and improve system reliability and performance. You can access all Trusted Advisor checks.
- The AWS Support API to interact with Support Center and Trusted Advisor. You can use the AWS Support API to automate support case management and Trusted Advisor operations.
- Third-party software support – Help with Amazon Elastic Compute Cloud (Amazon EC2) instance operating systems and configuration. Also, help with the performance of the most

popular third-party software components on AWS. Third-party software support isn't available for customers on Basic or Developer Support plans.

- Supports an unlimited number of AWS Identity and Access Management (IAM) users who can open technical support cases.

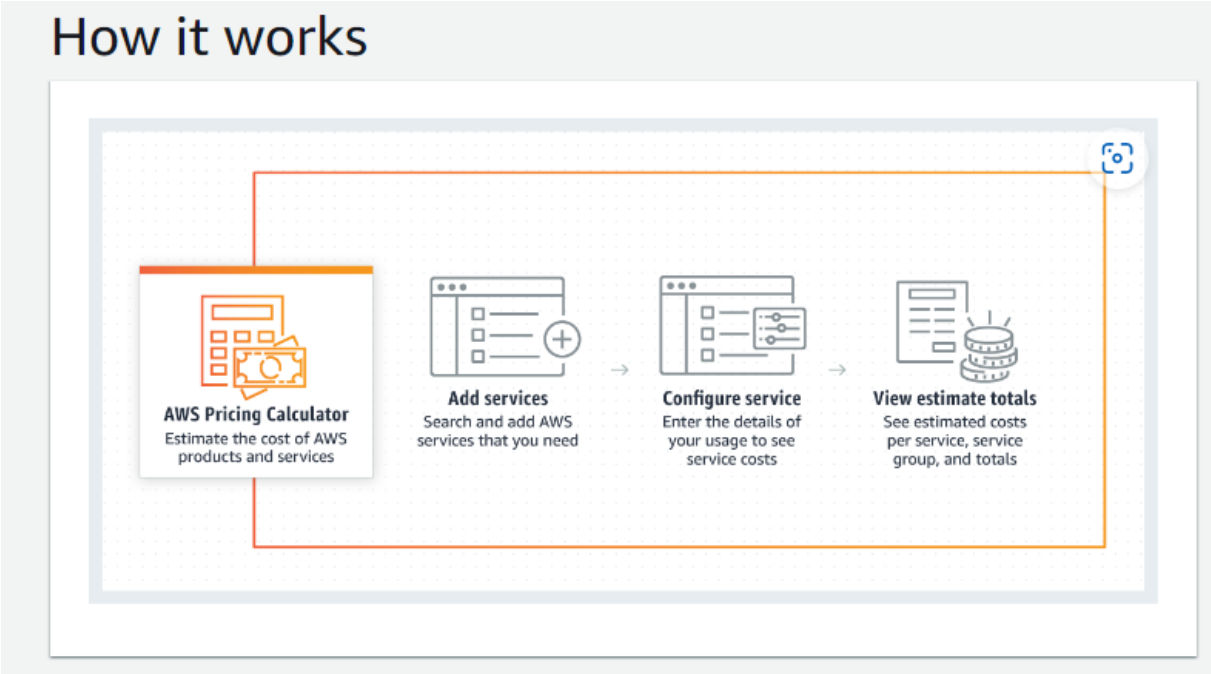
In addition, customers with an Enterprise On-Ramp or Enterprise Support plan have access to these features:

- Application architecture guidance – Consultative guidance on how services fit together to meet your specific use case, workload, or application.
- Infrastructure event management – Short-term engagement with AWS Support to get a deep understanding of your use case. After analysis, provide architectural and scaling guidance for an event.
- Technical account manager – Work with a technical account manager (TAM) for your specific use cases and applications.
- White-glove case routing.
- Management business reviews.

2) Explain the AWS Simple Monthly Calculator.

This tool to estimate your monthly bill, to determine your best and worst case scenarios (if you get Slashdotted, Dugg etc.), and identify areas of development to reduce your monthly costs and even compare it with other service providers who do not offer utility-style of billing (pay-as-you-go). This is now replaced by AWS pricing calculator.

Using AWS pricing calculator you can estimate the cost for your architecture solution.



3) List and explain the different EC2 pricing models

Amazon Elastic Compute Cloud (EC2) offers several pricing models to suit different workloads and use cases. The main EC2 pricing models are:

1. On-Demand Instances: This pricing model allows you to pay for compute capacity by the hour, with no long-term commitments or upfront costs. You can launch and terminate instances at any time, and only pay for the hours that you use.
2. Reserved Instances: This pricing model allows you to make a one-time upfront payment for a discounted hourly rate for EC2 instances, in exchange for a commitment to use the instances for a specific term (1 or 3 years). Reserved Instances can be a good choice for predictable workloads that require long-term compute capacity.
3. Spot Instances: This pricing model allows you to bid on unused EC2 capacity, which can be much cheaper than On-Demand or Reserved Instances. However, Spot Instances can be terminated by AWS with very little notice if the capacity is needed for other customers. Spot Instances can be a good choice for workloads that are flexible and can be interrupted without significant impact.
4. Dedicated Hosts: This pricing model allows you to run EC2 instances on a physical server that is dedicated to your use, rather than sharing the server with other customers. Dedicated Hosts can provide additional control and visibility over your infrastructure, as well as help address compliance requirements.

In addition to these main pricing models, EC2 also offers several other pricing options, such as Savings Plans and EC2 Instance Savings Plans, which provide additional discounts for Reserved Instances, as well as Instance Types, which offer a range of different hardware configurations for EC2 instances.

Overall, the choice of EC2 pricing model depends on the specific needs of your workload, as well as your budget and cost optimization goals. By understanding the different pricing models available and analyzing your usage patterns, you can choose the best option to minimize your EC2 costs while still meeting your performance and capacity requirements.