**AWS**

**Advantages of Cloud Computing:**

1. **Pay as you go**: Instead of investing on initial hardware, we pay only we use computing resources
2. **Benefit from massive economies of scale**: By using cloud computing, you can achieve a lower cost than you can get on your own. Because usage from hundreds of thousands of customers is aggregated in the cloud, 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:** IT resources are only a click away, which means that you reduce the time to make resources available to your developers from weeks to 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. **Realize cost saving:** Companies can focus on projects that differentiate their business instead of maintaining data centers. Cloud computing lets you focus on your customers, rather than on the heavy lifting of racking, stacking, and powering physical infrastructure. This is often referred to as undifferentiated heavy lifting.
6. **Go global in minutes:** Applications can be deployed in multiple Regions around the world with a few clicks. This means that you can provide lower latency and a better experience for your customers at a minimal cost.

**Regions:** Regions are geographic locations worldwide where AWS hosts its data centers. AWS Regions are named after the location where they reside. AWS Regions are independent from one another. Data is not replicated from one Region to another, without explicit customer consent and authorization.

**Choose the Right AWS Region**

When you decide which AWS Region to host your applications and workloads, consider four main aspects – latency, price, service availability, and compliance.

Inside every Region is a cluster of Availability Zones (AZs). An AZ consists of one or more data centers with redundant power, networking, and connectivity. These data centers operate in discrete facilities in undisclosed locations. They are connected using redundant high-speed and low-latency links.

**Maintain resiliency**  
To keep your application available, you must maintain high availability and resiliency. A well-known best practice for cloud architecture is to use Region-scoped, managed services. These services come with availability and resiliency built in. When that is not possible, make sure your workload is replicated across multiple AZs. At a minimum, you should use two AZs. That way, if an AZ fails, your application will have infrastructure up and running in a second AZ to take over the traffic.

Text

Description automatically generated

Ways to interact with AWS API:

1. Management console
2. CLI
3. SDK

Shared Responsibility model:

Customer and AWS both are responsible for the security.

Timeline

Description automatically generated

**AWS responsibility:**

AWS is responsible for security of the cloud. This means AWS protects and secures the infrastructure that runs the services offered in the AWS Cloud. AWS is responsible for:

* Protecting and securing AWS Regions, Availability Zones, and data centers, down to the physical security of the buildings
* Managing the hardware, software, and networking components that run AWS services, such as the physical servers, host operating systems, virtualization layers, and AWS networking components

| **Category** | **Examples of AWS Services in the Category** | **AWS Responsibility** |
| --- | --- | --- |
| Infrastructure services | Compute services, such as Amazon Elastic Compute Cloud (Amazon EC2) | AWS manages the underlying infrastructure and foundation services. |
| Container services | Services that require less management from the customer, such as Amazon Relational Database Service (Amazon RDS) | AWS manages the underlying infrastructure and foundation services, operating system, and application platform. |
| Abstracted services | Services that require very little management from the customer, such as Amazon Simple Storage Service (Amazon S3) | AWS operates the infrastructure layer, operating system, and platforms, in addition to server-side encryption and data protection. |

**Customer responsibility:**

| Category | AWS Responsibility | Customer Responsibility |
| --- | --- | --- |
| Infrastructure services | AWS manages the infrastructure and foundation services. | You control the operating system and application platform, in addition to encrypting, protecting, and managing customer data. |
| Container services | AWS manages the infrastructure and foundation services, operating system, and application platform. | You are responsible for customer data, encrypting the data, and protecting it through network firewalls and backups. |
| Abstracted services | AWS operates the infrastructure layer, operating system, and platforms, in addition to server-side encryption and data protection. | You are responsible for managing customer data and protecting it through client-side encryption. |

Customers are responsible for security in the cloud. When using any AWS service, you’re responsible for properly configuring the service and your applications, in addition to ensuring that your data is secure.  
  
Your level of responsibility depends on the AWS service. Some services require you to perform all the necessary security configuration and management tasks, while other more abstracted services require you to only manage the data and control access to your resources. Using the three categories of AWS services, you can determine your level of responsibility for each AWS service you use.

Due to the varying levels of effort, customers must consider which AWS services they use and review the level of responsibility required to secure each service. They must also review how the shared security model aligns with the security standards in their IT environment, in addition to any applicable laws and regulations.  
  
A key concept is that customers maintain complete control of their data and are responsible for managing the security related to their content. For example, you are responsible for the following:

* Choosing a Region for AWS resources in accordance with data sovereignty regulations
* Implementing data-protection mechanisms, such as encryption and scheduled backups
* Using access control to limit who can access to your data and AWS resources

Root User:

Never user root user for day to day activities

To ensure the safety of the root user, follow these best practices:

* Choose a strong password for the root user
* Never share your root user password or access keys with anyone
* Disable or delete the access keys associated with the root user
* Do not use the root user for administrative tasks or everyday tasks

**IAM**

AWS Identity and Access Management (IAM) is an AWS service that helps you manage access to your AWS account and resources. It also provides a centralized view of who and what are allowed inside your AWS account (authentication), and who and what have permissions to use and work with your AWS resources (authorization).

With IAM, you can share access to an AWS account and resources without sharing your set of access keys or password. You can also provide granular access to those working in your account, so people and services only have permissions to the resources they need.

**IAM user**

An IAM user represents a person or service that interacts with AWS. You define the user in your AWS account. Any activity done by that user is billed to your account. Once you create a user, that user can sign in to gain access to the AWS resources inside your account.

**IAM groups**

An IAM group is a collection of users. All users in the group inherit the permissions assigned to the group. This makes it possible to give permissions to multiple users at once. It’s a more convenient and scalable way of managing permissions for users in your AWS account. This is why using IAM groups is a best practice.

Keep in mind the following features of groups:

* Groups can have many users.
* Users can belong to many groups.
* Groups cannot belong to groups.

The root user can perform all actions on all resources inside an AWS account by default. This is in contrast to creating new IAM users, new groups, or new roles. New IAM identities can perform no actions inside your AWS account by default until you explicitly grant them permission.  
  
The way you grant permissions in IAM is by using IAM policies.

**IAM policies**

To manage access and provide permissions to AWS services and resources, you create IAM policies and attach them to IAM users, groups, and roles. Whenever a user or role makes a request, AWS evaluates the policies associated with them. For example, if you have a developer inside the developers group who makes a request to an AWS service, AWS evaluates any policies attached to the developers group and any policies attached to the developer user to determine if the request should be allowed or denied.

**IAM policy examples**

Most policies are stored in AWS as JSON documents with several policy elements. The following example provides admin access through an IAM identity-based policy.

|  |
| --- |
| ***{ "Version": "2012-10-17", "Statement": [{ "Effect": "Allow", "Action": [ "iam: ChangePassword", "iam: GetUser" ] "Resource": "arn:aws:iam::123456789012:user/${aws:username}" }] }*** |

This policy has four major JSON elements: ***Version***, ***Effect***, ***Action***, and ***Resource***.

* The ***Version***element defines the version of the policy language. It specifies the language syntax rules that are needed by AWS to process a policy. To use all the available policy features, include **"Version": "2012-10-17"** before the **"Statement"**element in your policies.
* The ***Effect***element specifies whether the statement will allow or deny access. In this policy, the Effect is **"Allow"**, which means you’re providing access to a particular resource.
* The ***Action***element describes the type of action that should be allowed or denied. In the example policy, the action is **"\*"**. This is called a wildcard, and it is used to symbolize every action inside your AWS account.
* The ***Resource***element specifies the object or objects that the policy statement covers. In the policy example, the resource is the wildcard **"\*"**. This represents all resources inside your AWS console.

The Seven R’s

1. Rehost
2. Replatform
3. Relocate
4. Refactor
5. Retire
6. Retain
7. Repurchase

Rehost: Lift and shift ( on prem 🡪 AWS Cloud )

1. Recreating the on-prem network, only hosted on AWS
2. Automating with tools such as AWS Application Migration Service
3. Easier to optimize and re-architect applications after migration

Relocate: Hypervisor level lift and shift ( VMWare infrastructure 🡪 VMWare Cloud on AWS )

Shared Responsible Model:

it depends on

A vunrability is a weakness

A Threat is a possibility for an event or act to exploit a vunrability

A risk is the potential for loss, damage or destruction of resources due to a threat

CAF – Cloud