Identity & Access Management (IAM)

Security

Security and Clouds

- Security is a core requirement for any application whether it is hosted on an on premise data center or a cloud such as AWS.

- It is a fundamental service that protects your applications and data from a variety of cyber-attacks, security breaches, accidental or deliberate data deletions, theft, and much more.

Is AWS really secure?

Different layers of security that AWS uses

- 1. **Physical data center security:**
  - The AWS infrastructure is designed and managed according to security best practices and compliance guides.
  - The data centers themselves are housed at non-disclosed locations and entry to them is strictly controlled, managed, logged, and audited on a regular basis.

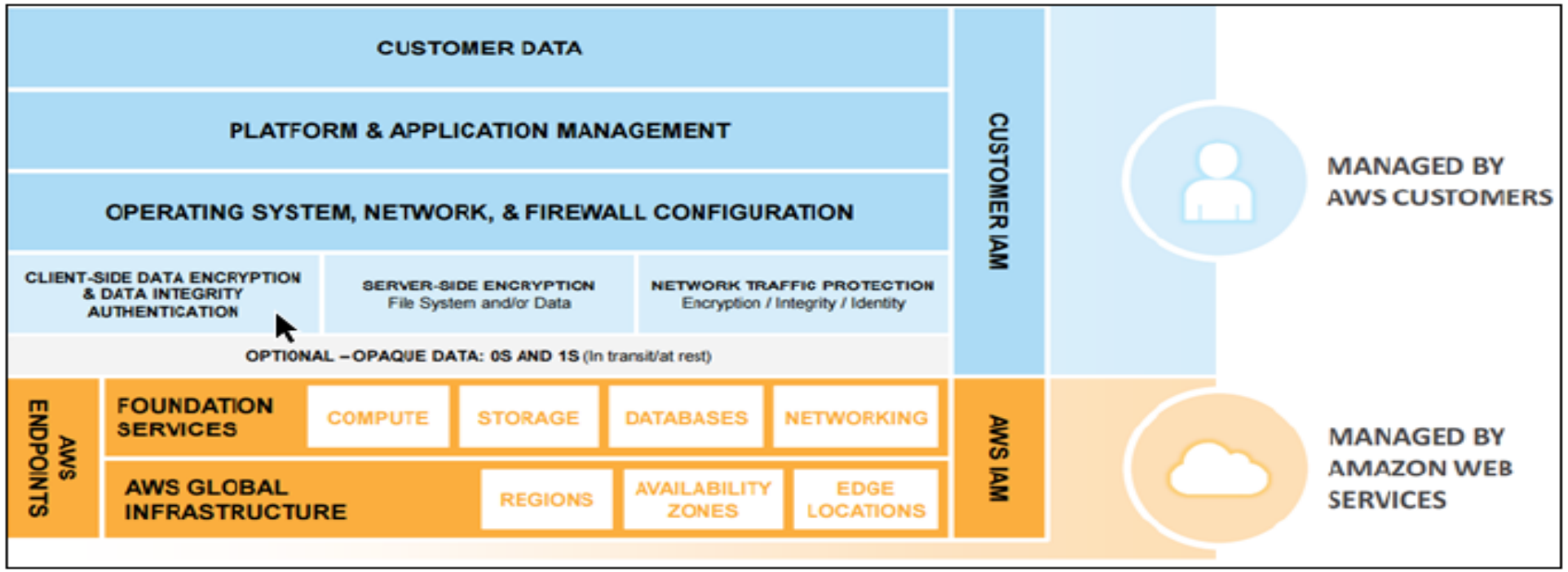
- 2**. Virtualization and OS security:**
  - AWS regularly patches and updates virtualization and operating systems against a variety of attacks such as DDoS, and so on.

Is AWS really secure?

- 3. **Regulatory compliances:**
  - The AWS infrastructure is certified against security and data protection in accordance with various industry and government requirements. Here are a few compliances that AWS is certified against:
  - SOC 1 (formerly SAS 70 Type II),
  - SOC 2, and SOC 3
  - FISMA, DIACAP, and FedRAMP
  - ISO 27001
  - HIPAA

- To read the complete list, visit the AWSrisk and compliance whitepaper at http://aws.amazon.com/security/.

Shared Model for AWS Services



Security AWS provides a few services and products that are specifically designed to help you
secure your infrastructure on the cloud, such as IAM, AWS Multi-Factor Authentication (AWS MFA), AWS Cloud Trail, and much more.

Identity & Access Mana -

AWS IAM

Identity and Access Management

- You can use IAM to create users and groups, assigning users specific permissions and policies, and a lot more.

- The best part of all this is that IAM is completely FREE. Yup! Not a penny is required to use it.

User and Credentials

You can access AWS in different ways depending on the user credentials:

- Console password: A password that the user can type to sign in to interactive sessions such as the AWS Management Console.
- Access keys: A combination of an access key ID and a secret access key. You can assign two to a user at a time. These can be used to make programmatic calls to AWS. For example, you might use access keys when using the API for code or at a command prompt when using the AWS CLI or the AWS PowerShell tools.
- By default, a brand new IAM user has no permissions to do anything
- Each IAM user is associated with one and only one AWS account
- There's a limit to the number of IAM users you can have in an AWS account.

Key Terminology for IAM

- **Users**:

  End Users such as people, employees of an organization

- **Groups**:

  A collections of users. Each user in the group inherit the permission of the group

- **Policies**:

  Policies are made up of documents called policy documents. These documents are in the format of JSON and they give permissions as to what a User/Group/Role is able to do.

- **Roles**:

  You create a role and assign them to AWS resource.

IAM Features

- Centralized control of your AWS resources
- Shared access to your AWS account
- Granular permissions
- Identity Federation (Including Active Directory, FB, LinkedIn)
- Multi Factor Authentication
- Provide temporary access for users/devices and services where necessary
- Allows you setup your own password rotation policy
- Integrated with many different AWS Services
- Supports PCI DSS compliance.

IAM Features

- **Shared access to a single account:**
  - you can create and provide users with shared access to your single account with real ease.

- **Multi-factor authentication:**
  - along with your password, you will also have to provide a secret key/pin from a special hardware device, or even from software apps such as Google Authenticator.

IAM Features

- **Integration with other AWS products:**
  - can be used to provide granular access rights and permissions to each service as required.
- **Identity federation:**
  - IAM can be integrated with an on-premise AD to provide access to your AWS account
- **Global reach:**
  - IAM is the Global, not specific to the Region.
- **Access mechanisms:**
  - IAM can be accessed using a variety of different tools, AWS Management Console, AWS CLI, via SDKs that support different platforms and programming languages such as Java, .NET, Python

Delete your root keys

- Delete your root access keys. Now why would you want to do? What are root access keys?
  - Root keys simply consist of an access ID and a secret key
  - Can be used to programmatically access any AWS service.
  - Each user that you create gets its own set of keys.
  - The secret key has to be protected and kept under lock and key at all costs.

AWS CLI VS Console

- **Access key ID and Secret key ID are used with AWS CLI or API’s**
- Email ID or username and password is used to login to AWS console.
- You cannot login to AWS Console using access and secret keys.
- You cannot login to CLI with Console username and password.

Groups and Policies

- **Group** is a collection of IAM users that has a particular set of permissions assigned to it.
  - For example, a set of users who perform admin tasks can be clubbed under a common group called as administrators.
- **A policy** is a document that lists one or more permissions. You can attach policies to virtually anything in AWS, from users and groups to individual AWS resources as well.

Permissions

- Permissions provide you with access to and control of various AWS resources.
- They are also responsible for controlling actions that you can perform on the resources.
- Permissions can be classified into two main classes
  - **User-based permissions**
  - **Resource-based permissions**

User Based Permissions

- These permissions are attached to IAM users and allow them to perform some action over an AWS resource.
- User-based permissions can be applied to groups as well.
- Two Categories
  - **Inline policies**
    - created and managed completely by you
  - **Managed Policies**
    - created and managed more by AWS itself

Resource Based Permissions

- User has specific level of access to a particular AWS resource along with what actions they can perform on it.

- These categories of permissions are only inline-based this means that they are completely managed and created by you.

Permissions

User Based Permissions

Admin
All Actions on All Resources

DevTL
List, Read, Write on EC2

TestTL
Read on EC2

Resource Based Permissio

S3 Bucket

Admin: List, Read, Write
DevTL: Read, Write
TestTL: Read, Write

Policy – JSON Format

Let's look at a simple policy for our reference:

```
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"ec2:DescribeInstances",
"ec2:DescribeImages"
],
"Resource": "arn:aws:iam::012345678910:user/admin"
}
]
```

Roles

- IAM roles are designed so that our applications can securely make API requests from our instances, without requiring us to manage the security credentials that the applications use. Instead of creating and distributing our AWS credentials, wean delegate permission to make API requests using IAM roles

Roles

- Roles are a group of permissions that grant users access to some particular AWS resources and services.
- Difference:
  - Policies are applied to users and groups that belong to a particular AWS account
  - Roles are applied to users who are generally not a part of your AWS account
  - Use roles to delegate access to users, applications, and services that do not have access to your AWS resources.
  - Use roles to create federated identities where a user from your organization's corporate directory gets access to your AWS resources on a temporary basis.

IAM Best Practices

- Get rid of the Root Account, use IAM wherever necessary. Hide away the Root key and avoid using it unless it's the end of the world!

- Create a separate IAM users for your organization, each with their own sets of access and Secret Keys. *DO NOT SHARE YOUR KEYS OR PASSWORDS! Sharing such things is never a good idea and can cause serious implications and problems*.

- Create separate administrators for each of the AWS services that you use.

- Use roles and groups to assign individual IAM users permissions. Provide only the required level of access and permissions that the task demands.

IAM Best Practices

- Leverage multi-factor authentication (MFA) wherever possible.
- Rotate your passwords and keys on a periodic basis. Create keys only if there is a requirement for it.
- Maintain a logs and history of your AWS account and its services. Use AWSCloudTrail for security and compliance auditing.
- Use temporary credentials (IAM Roles) rather than sharing your account details with other users and applications.
- Leverage AWS Key Management Service to encrypt data and your keys wherever necessary.

Exam Tips

- **IAM is Universal.** It does not apply to regions at this time
- The **root account** is simply the account created when first setup your AWS account. It has complete Admin access.
- New users have NO PERMISSIONS when first created
- New users are assigned Access key ID & Secret Access keys when first created.
- These are not same a passwords. You cannot use access and secret keys to login to the console. You can use this to access AWS using CLI and API’s.
- You only get to view this once. If you loose them , you have to regenerate them. So, save it in secure location.
- Always setup MFA on your root account
- You can create customized login link and password rotation policies.