Securing, Protecting and Managing Data

https://docs.aws.amazon.com/whitepapers/latest/building-data-lakes/securing-protecting-managing-data.html

Securing, Protecting, and Managing Data

PDF | RSS

Building a data lake and making it the centralized repository for assets that were previously duplicated and placed across many siloes of smaller platforms and groups of users requires implementing stringent and fine-grained security and access controls along with methods to protect and manage the data assets. A data lake solution on AWS—with Amazon S3 as its core—provides a robust set of features and services to secure and protect your data against both internal and external threats, even in large, multi-tenant environments. Additionally, innovative Amazon S3 data management features enable automation and scaling of data lake storage management, even when it contains billions of objects and petabytes of data assets.

Securing your data lake begins with implementing very fine-grained controls that allow authorized users to see, access, process, and modify particular assets and ensure that unauthorized users are blocked from taking any actions that would compromise data confidentiality and security. A complicating factor is that access roles may evolve over various stages of a data asset's processing and lifecycle. Fortunately, Amazon has a comprehensive and well-integrated set of security features to secure an Amazon S3-based data lake.

Access Policy Options and AWS IAM

You can manage access to your Amazon S3 resources using access policy options. By default, all Amazon S3 resources—buckets, objects, and related subresources—are private: only the resource owner, an AWS account that created them, can access the resources. The resource owner can then grant access permissions to others by

writing an access policy. Amazon S3 access policy options are broadly categorized as resource-based policies and user policies. Access policies that are attached to resources are referred to as *resource-based policies*. Example resource-based policies include bucket policies and access control lists (ACLs). Access policies that are

attached to users in an account are called *user policies*. Typically, a combination of resource-based and user policies are used to manage permissions to S3 buckets, objects, and other resources.

For most data lake environments, we recommend using user policies, so that permissions to access data assets can also be tied to user roles and permissions for the data processing and analytics services and tools that your data lake users will use. User policies are associated with AWS Identity and Access Management (IAM) service,

which allows you to securely control access to AWS services and resources. With IAM, you can create IAM users, groups, and roles in accounts and then attach access policies to them that grant access to AWS resources, including Amazon S3. The model for user policies is shown in the following figure. For more details and information on securing Amazon S3 with user policies and AWS IAM, please reference: Amazon Simple Storage Service Developers Guide and AWS Identity and Access Management User Guide.

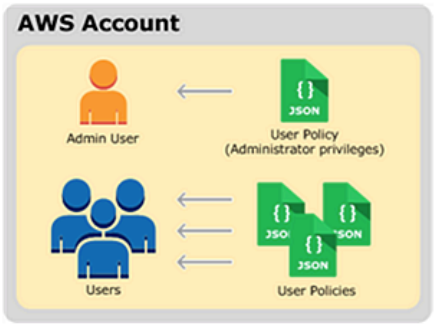


Figure: Model for user policies

---

https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

What is IAM?

PDF | Kindle | RSS

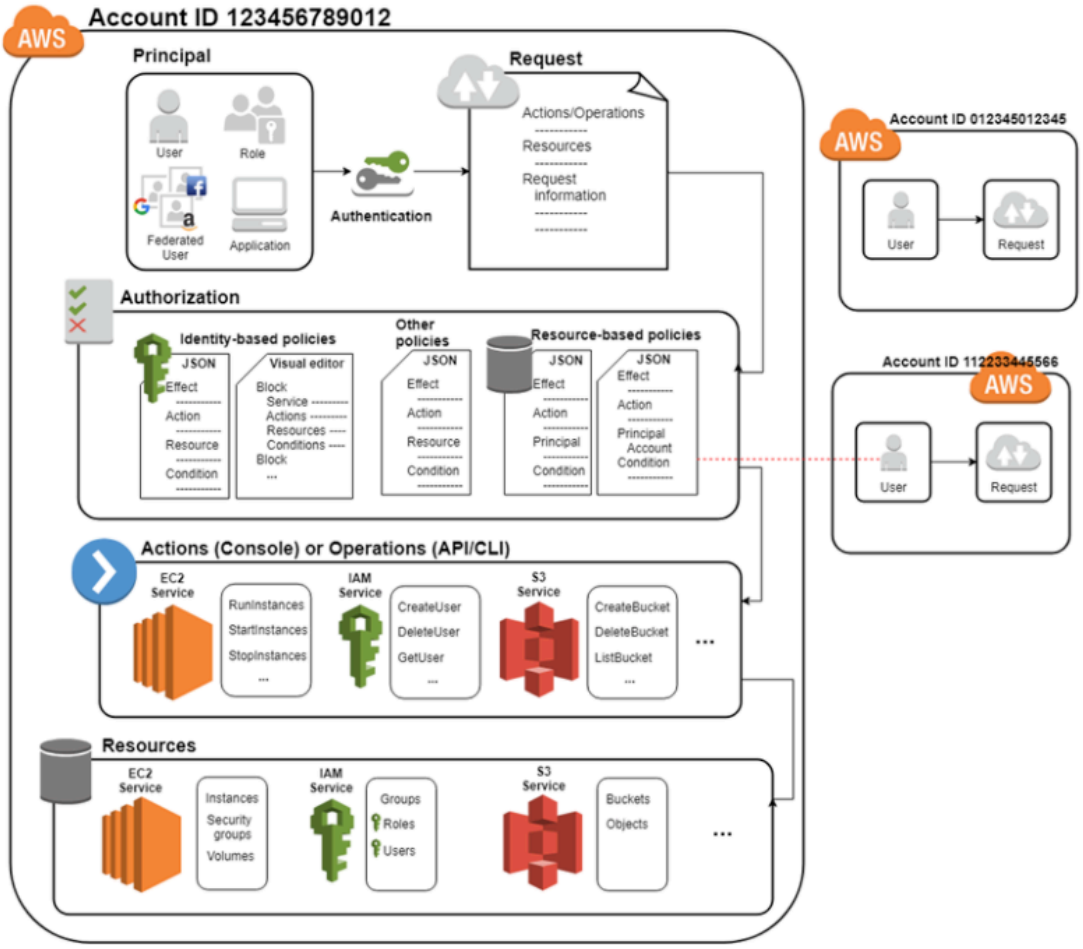
AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

Understanding how IAM works

PDF | Kindle | RSS

Before you create users, you should understand how IAM works. IAM provides the infrastructure necessary to control authentication and authorization for your account. The IAM infrastructure includes the following elements:



Terms

Learn more about IAM terms.

Resources

The user, group, role, policy, and identity provider objects that are stored in IAM. As with other AWS services, you can add, edit, and remove resources from IAM.

Identities

The IAM resource objects that are used to identify and group. You can attach a policy to an IAM identity. These include users, groups, and roles.

Entities

The IAM resource objects that AWS uses for authentication. These include IAM users, federated users, and assumed IAM roles.

Principals

A person or application that uses the AWS account root user, an IAM user, or an IAM role to sign in and make requests to AWS.

Request

When a principal tries to use the AWS Management Console, the AWS API, or the AWS CLI, that principal sends a *request* to AWS. The request includes the following information:

- **Actions or operations** – The actions or operations that the principal wants to perform. This can be an action in the AWS Management Console, or an operation in the AWS CLI or AWS API.
- **Resources** – The AWS resource object upon which the actions or operations are performed.
- **Principal** – The person or application that used an entity (user or role) to send the request. Information about the principal includes the policies that are associated with the entity that the principal used to sign in.
- **Environment data** – Information about the IP address, user agent, SSL enabled status, or the time of day.
- **Resource data** – Data related to the resource that is being requested. This can include information such as a DynamoDB table name or a tag on an Amazon EC2 instance.

AWS gathers the request information into a *request context*, which is used to evaluate and authorize the request.

Authentication

A principal must be authenticated (signed in to AWS) using their credentials to send a request to AWS. Some services, such as Amazon S3 and AWS STS, allow a few requests from anonymous users. However, they are the exception to the rule.

To authenticate from the console as a root user, you must sign in with your email address and password. As an IAM user, provide your account ID or alias, and then your user name and password. To authenticate from the API or AWS CLI, you must provide your access key and secret key. You might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more about the IAM entities that AWS can authenticate, see IAM users and IAM roles.

Authorization

You must also be authorized (allowed) to complete your request. During authorization, AWS uses values from the request context to check for policies that apply to the request. It then uses the policies to determine whether to allow or deny the request. Most policies are stored in AWS as JSON documents and specify the permissions for principal entities. There are several types of policies that can affect whether a request is authorized. To provide your users with permissions to access the AWS resources in their own account, you need only identity-based policies. Resource-based policies are popular for granting cross-account access. The other policy types are advanced features and should be used carefully.

AWS checks each policy that applies to the context of your request. If a single permissions policy includes a denied action, AWS denies the entire request and stops evaluating. This is called an *explicit deny*. Because requests are *denied by default*, AWS authorizes your request only if every part of your request is allowed by the applicable permissions policies. The evaluation logic for a request within a single account follows these general rules:

- By default, all requests are denied. (In general, requests made using the AWS account root user credentials for resources in the account are always allowed.)
- An explicit allow in any permissions policy (identity-based or resource-based) overrides this default.
- The existence of an Organizations SCP, IAM permissions boundary, or a session policy overrides the allow. If one or more of these policy types exists, they must all allow the request. Otherwise, it is implicitly denied.
- An explicit deny in any policy overrides any allows.

To learn more about how all types of policies are evaluated, see Policy evaluation logic. If you need to make a request in a different account, a policy in the other account must allow you to access the resource *and* the IAM entity that you use to make the request must have an identity-based policy that allows the request.

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html

Users and permissions

By default, a brand new IAM user has no permissions to do anything. The user is not authorized to perform any AWS operations or to access any AWS resources. An advantage of having individual IAM users is that you can assign permissions individually to each user. You might assign administrative permissions to a few users, who then can administer your AWS resources and can even create and manage other IAM users. In most cases, however, you want to limit a user's permissions to just the tasks (AWS actions or operations) and resources that are needed for the job.

Imagine a user named Diego. When you create the IAM user `Diego`, you can create a password for that user. You also attach permissions to the IAM user that let him launch a specific Amazon EC2 instance and read (GET) information from a table in an Amazon RDS database. For procedures on how to create users and grant them initial credentials and permissions, see Creating an IAM user in your AWS account. For procedures on how to change the permissions for existing users, see Changing permissions for an IAM user. For procedures on how to change the user's password or access keys, see Managing user passwords in AWS and Managing access keys for IAM users.

Actions or operations

After your request has been authenticated and authorized, AWS approves the actions or operations in your request. Operations are defined by a service, and include things that you can do to a resource, such as viewing, creating, editing, and deleting that resource. For example, IAM supports approximately 40 actions for a user resource, including the following actions:

- `CreateUser`
- `DeleteUser`
- `GetUser`
- `UpdateUser`

To allow a principal to perform an operation, you must include the necessary actions in a policy that applies to the principal or the affected resource. To see a list of actions, resource types, and condition keys supported by each service, see Actions, Resources, and Condition Keys for AWS Services.

Data Encryption with Amazon S3 and AWS KMS

Although user policies and IAM control who can see and access data in your Amazon S3-based data lake, it's also important to ensure that users who might inadvertently or maliciously manage to gain access to those data assets can't see and use them. This is accomplished by using encryption keys to encrypt and de-encrypt

data assets. Amazon S3 supports multiple encryption options. Additionally, AWS KMS helps scale and simplify management of encryption keys. AWS KMS gives you centralized control over the encryption keys used to protect your data assets. You can create, import, rotate, disable, delete, define usage policies for, and audit the use of encryption keys used to encrypt your data. AWS KMS is integrated with several other AWS services, making it easy to encrypt the data stored in these services with encryption keys. AWS KMS is integrated with AWS CloudTrail, which provides you with the ability to audit who used which keys, on which resources, and when.

> Data lakes built on AWS primarily use two types of encryption: Server-side encryption (SSE) and client-side encryption. SSE provides data-at-rest encryption for data written to Amazon S3. With SSE, Amazon S3 encrypts user data assets at the object level, stores the encrypted objects, and then decrypts them as they are accessed and retrieved. With client-side encryption, data objects are encrypted before they written into Amazon S3. For example, a data lake user could specify client-side encryption before transferring data assets into Amazon S3 from the Internet, or could specify that services like Amazon EMR, Amazon Athena, or Amazon Redshift use client-side encryption with Amazon S3. SSE and client-side encryption can be combined for the highest levels of protection. Given the intricacies of coordinating encryption key management in a complex environment like a data lake, we strongly recommend using AWS KMS to coordinate keys across client- and server-side encryption and across multiple data processing and analytics services.
>
> For even greater levels of data lake data protection, other services like Amazon API Gateway, Amazon Cognito, and IAM can be combined to create a "shopping cart" model for users to check in and check out data lake data assets. This architecture has been created for the Amazon S3-based data lake solution reference architecture, which can be found, downloaded, and deployed at https://aws.amazon.com/answers/big-data/data-lake-solution/.

https://aws.amazon.com/solutions/implementations/data-lake-solution/