ConductorOne

4 Ways to Configure AWS Access

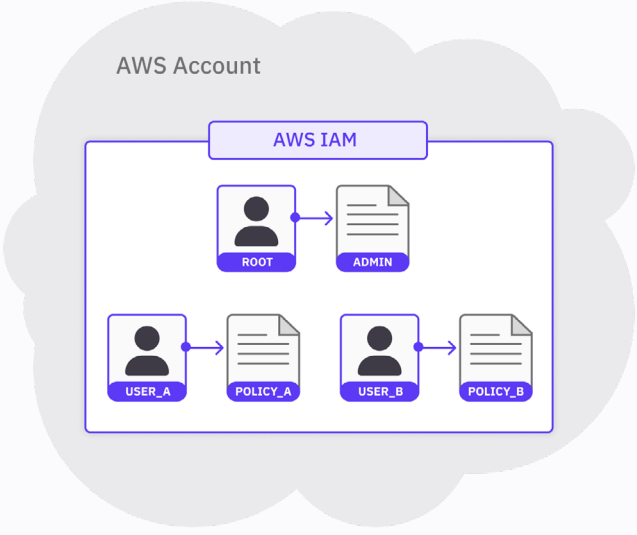
As cloud computing continues to revolutionize the way businesses operate, Amazon Web Services (AWS) has maintained its reputation as one of the field's leading providers. With its vast array of services and features, AWS offers organizations the ability to scale, deploy, and manage cloud resources with ease.

Ensuring that the right people get the right access to these resources is paramount. Unauthorized access can lead to data breaches and a multitude of other security concerns. This is where AWS access control and management models come into play.

Configuring access control in AWS is inherently complex, but we'll make sense of it all in this article. Specifically, we'll explore **four different approaches to configuring AWS access**, each tailored to specific use cases and requirements. By understanding the advantages, disadvantages, and tradeoffs of each method, you'll gain insights into which approach best aligns with your organization's needs.

AWS Access Management Method 1: Local IAM Users

In AWS, Identity and Access Management (IAM) is the service that helps you define access permissions to AWS services and resources. One method of access management is through the use of local IAM users. Consider the following diagram:



The diagram above depicts 3 IAM users in a single AWS account. Each user consists of a name (i.e., `root` , `user_a` , or `user_b`) and a set of credentials. The credentials are represented by an IAM policy that's attached to each user. The policy (i.e., `admin` , `policy_a` , or `policy_b`) defines the actions that the user is allowed to perform within the account. Whenever you log into an AWS account, you log in as a specific IAM user.

The `root` user is special. It comes bundled with each new AWS account and has admin privileges to every AWS service and resource. Because of this, AWS advises against using the root user to perform everyday tasks. To limit permissions for certain users and minimize the impact of potential mistakes, you can create additional users with fewer permissions to perform day-to-day operations.

To summarize this first method: Using the `root` user, we can create other users that have fewer permissions. Then, to follow security best practices, you can log in as one of the other users to perform daily operations instead of as the `root` user.

Advantages of This Method

- **Granular controls:** For each local IAM user, you can customize exactly what services, actions, and resources that user is allowed access to. This high level of granularity can help organizations tailor access rights to the specific needs of individual users or groups.
- **Flexibility:** IAM allows you to set access controls to specific actions and resources, making it very flexible and customizable. In addition, IAM is completely managed by AWS, so organizations can manage user accounts entirely within the AWS environment.
- **Local credential management:** Each local IAM user manages their own set of access keys (access key ID and secret access key), which are used for programmatic access to AWS services.
- **Audit trail:** AWS provides detailed logging and monitoring for all IAM user activities. This audit trail helps organizations track user actions and changes to access permissions in case something goes awry.

Disadvantages of This Method

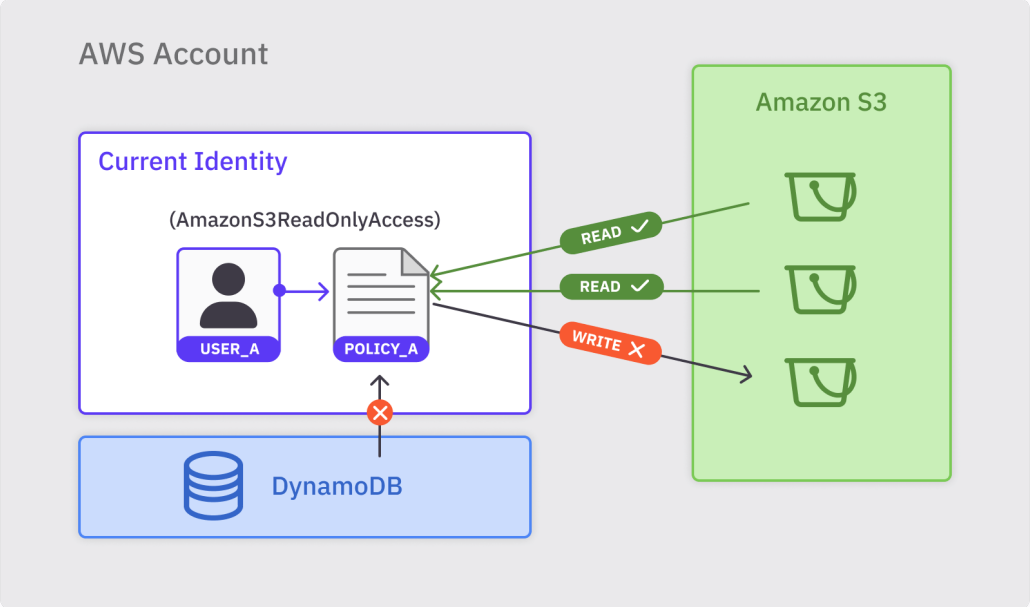
- **Admin overhead:** As the number of users and permissions grows, managing local IAM users becomes a huge operational burden. The process of creating, modifying, and revoking user access may require significant effort, particularly in larger organizations with complex access requirements.
- **Management complexity:** Keeping track of permissions and ensuring they are up-to-date can be challenging. Without proper organization and documentation, managing access control configurations can lead to errors and inconsistencies.
- **Lack of centralized control and visibility:** If your organization manages multiple accounts, it can be hard to enforce consistent security policies across all of them since local IAM users are managed at the individual account level.
- **Limited integration with external Identity Providers (IdPs):** IAM may not seamlessly integrate with external IdPs. If your organization already has an established IdP, this can result in redundant access management processes.

Setting It Up

Suppose we have a user, `user_a` , that only has to be concerned with reading from a couple of specific S3 buckets. To set up `user_a` , we might first create an IAM policy, `policy_a` , as follows:

```
{
    “Version”: “2012-10-17”,
    “Statement”: [
        {
            “Effect”: “Allow”,
            “Action”: [
                “s3:Get*”,
                “s3:List*”,
                “s3-object-lambda:Get*”,
                “s3-object-lambda:List*”
            ],
            “Resource”: “*”
        }
    ]
}
```

Note that this is the same as the AWS Managed Policy AmazonS3ReadOnlyAccess. If we attach this policy to `user_a` and log into the AWS account as `user_a` , here's a visual depiction of the actions we're allowed to perform:



For the above diagram:

- `user_a` is allowed to perform `READ` actions on S3 buckets.
- `user_a` is `not` allowed to perform `WRITE` actions on S3 buckets.
- `user_a` is `not` allowed to perform any actions on any other services (i.e., DynamoDB).

Doing this protects us from unexpected operations, such as an accidental `s3:putObject` that overrides an object, or even worse, a `dynamodb:deleteTable` that removes an entire database table. If all we care about is reading from S3 buckets, we should be logged in as a user that contains S3 read permissions only.

Security Best Practices for this Method

- **Use Multi-Factor Authentication (MFA):** Unsurprisingly, enforcing MFA for local IAM users is a fundamental security best practice and greatly reduces the risk of unauthorized account access.
- **Implement the Principle of Least Privilege (PoLP):** Assign the minimum necessary privileges to each user whenever possible. Avoid granting broad or excessive permissions, as this could inadvertently expose sensitive resources to security risks.
- **Regularly rotate access keys:** Set up a mechanism to automatically rotate access keys at regular intervals. This reduces the window of opportunity for attackers to exploit compromised or leaked keys.

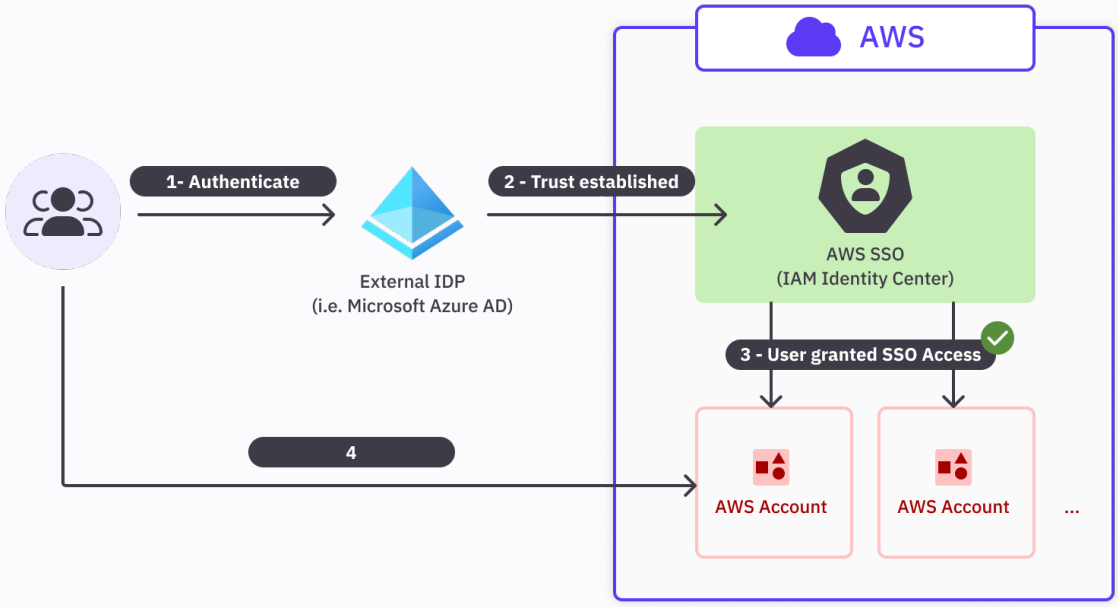
AWS Access Management Method 2: Account Level SSO

Local IAM users tend to work well in the beginning. However, as your organization starts to scale, you'll probably need a more centralized access control method. This is where Account Level Single Sign-On (SSO) comes in.

As its name implies, in an SSO model, once the user is authenticated by an Identity Provider (IdP), they gain access to all the AWS accounts they are authorized to access without having to sign in multiple times. This eliminates the need to manage separate IAM credentials for each individual account, and greatly simplifies the authentication experience.

For account administrators, account-level SSO also helps streamline the process of access management by providing a centralized place to manage user identities. That centralized place is the IdP itself, which is useful for organizations that already use their own IdP but still want to use AWS.

In AWS, IAM Identity Center is the service that enables organizations to set up SSO to their AWS accounts. Note that IAM Identity Center used to be called AWS Single Sign-On up until July 2022. Here's a high-level diagram of how account-level SSO works.



In the above diagram:

1. Users first authenticate through an external IdP. This example uses Microsoft Azure Active Directory (or Microsoft Entra ID), which is fully compatible with IAM Identity Center.
2. If user authentication succeeds, the user is directed to a portal in IAM Identity Center.
3. The user is granted SSO access to all AWS accounts, cloud applications, and other SAML-enabled applications that they're assigned to, as defined in IAM Identity Center.
4. The user can then access any of these AWS accounts and applications without having to log in separately or manage different AWS credentials for each account.

Advantages of This Method

- **Centralized management:** Account administrators can handle access management to multiple AWS accounts through a central IdP. This helps enforce consistent access control policies with minimal surprises.
- **Streamlined user experience:** A single sign-on experience is generally better for users, who no longer have to remember or manage separate IAM credentials for each account.
- **Enhanced security (compared to local users):** SSO can leverage the advanced authentication capabilities of external IdPs, including built-in features like MFA. Centralized control also enables quicker response to security threats and policy violations.
- **Federated identity:** Account-level SSO enables organizations to establish trust between their AWS accounts and external IdPs. This facilitates federated identity management, allowing users to log in using their existing corporate credentials without having to create new passwords.

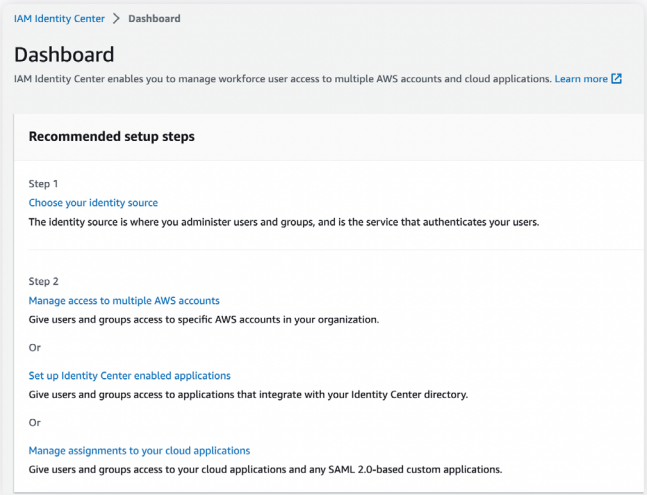
Disadvantages of This Method

- **Initial Setup Complexity:** Implementing account-level SSO requires initial configuration and integration with an external IdP. Depending on the complexity of the organization's existing infrastructure, setting up SSO can take time and effort.
- **Dependency on external IdPs:** Any downtime in the IdP can severely affect the ability of your users to access AWS accounts and resources, so be sure to choose a reliable IdP.
- **Limited Flexibility:** Certain accounts may have unique access control requirements. These might be better handled outside of a centralized identity management system.
- **Compliance Considerations:** Organizations must consider compliance requirements related to handling user identities and sensitive data within the IdP. Ensure the chosen IdP aligns with current regulatory standards and security best practices.
- **Delegated Administration of Access to the IdP Owner:** In this model, the IdP owner has significant power and responsibility in managing user identities across AWS accounts. Organizations must ensure their IdP owners can be trusted and/or at least consider this implication in understanding your threat model.

Setting It Up

You can set up account-level SSO entirely from the AWS console. If you haven't already enabled the Identity Center service, you can do so from the Identity Center console. The account needs to be managed by AWS Organizations, but you can also choose to create an organization when you enable Identity Center.

Once you've enabled Identity Center, the service dashboard contains links and guides to help you through the setup process.



Security Best Practices for This Method

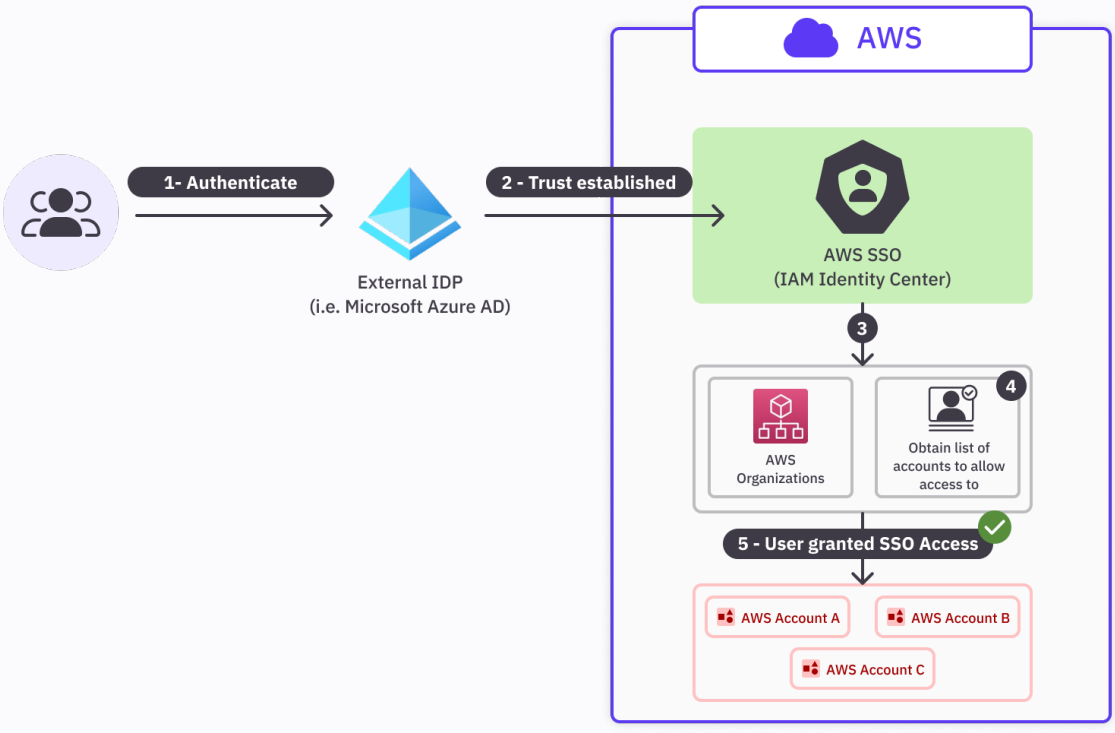
- **Use a secure and reliable IdP:** Choose a trusted and reputable Identity Provider for your Account Level Single Sign-On (SSO) implementation. AWS supports all of the major cloud identity providers, so this shouldn't be a big issue.
- **Implement MFA:** Enable Multi-Factor Authentication (MFA) for all users accessing AWS accounts through Account Level SSO.
- **Enforce strong password policies:** Your IdP should enforce strong password policies for all accounts. Strong passwords should be long, complex, and include a combination of uppercase and lowercase letters, numbers, and special characters.
- **Monitor external IdP security:** Because so much of SSO relies on your IdP, regularly keep track of security updates, patches, and changes in your IdP's configuration. Stay informed about any security incidents or vulnerabilities that may impact your IdP, and take prompt action to mitigate risks.

AWS Access Management Method 3: Organization-level SSO Groups and Users

In account-level SSO, the focus is on providing single sign-on capabilities for multiple AWS accounts *within an organization*. Organization-level SSO is very similar, but it allows entire organizations to create groups and users centrally within AWS Organizations. These groups and users can then be granted access to specific AWS accounts and resources via IAM roles and policies.

Setting up organization-level SSO generally still requires the use of an external IdP, but user management is centralized through AWS Organizations. Administrators create and manage groups at the organization level and assign them to specific AWS accounts with the necessary IAM roles and policies.

The following is a high-level diagram of what organization-level SSO might look like. Note that the only difference is the extra focus on AWS Organizations, which Identity Center uses to determine which accounts the user is allowed to access. Technically, AWS Organizations was involved in the account level SSO method as well, but since we didn't need it to handle accounts at scale, we left it out of the previous diagram for simplicity.



In the above diagram:

1. Users first authenticate through an external IdP. This example also uses Microsoft Azure AD as the IdP.
2. If the user authentication succeeds, the user is directed to a portal in IAM Identity Center.
3. In the backend, IAM Identity Center consults AWS Organizations.
4. AWS Organizations defines the list of accounts that the authenticated user is allowed access to.
5. The user is granted SSO access to all AWS accounts as defined by AWS Organizations.

Advantages of This Method

- **Centralized management:** Within AWS Organizations, administrators can control user access policies from a single, centralized location.
- **Streamlined user experience:** As with account-level SSO, users can enjoy a seamless single sign-on experience.
- **Consistent security policies across the organization:** Organization-level SSO enables organizations to enforce consistent security policies across all owned AWS accounts. With centralized IAM roles and policies, security measures are standardized and easier to maintain.
- **Easy user provisioning and de-provisioning:** When a user joins or leaves the organization, it's easy for the administrator to grant or revoke access, and/or for SCIM to automatically do so, to the relevant AWS accounts. This greatly reduces the risk of orphaned accounts or unauthorized access.
- **Fine-grained access control:** IAM roles and policies form the backbone of this access method. This allows for fine-grained access control, ensuring users only have access to the specific AWS resources required for their roles.

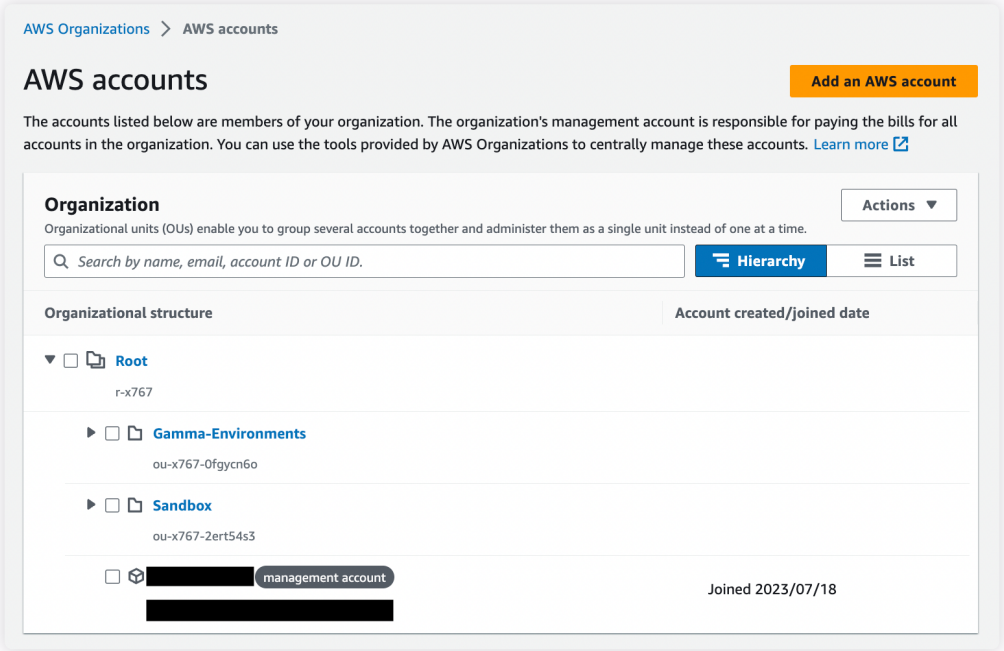
Disadvantages of This Method

- **Initial setup complexity:** Integrating with an external IdP may involve complex setup procedures. In addition, administrators must be careful when setting up trust relationships and configuring IAM roles and policies.
- **Dependency on External IdPs:** SSO depends on your IdP; if the IdP goes down, this may impact your organization's access to AWS resources.
- **Limited Flexibility:** If your accounts have special access control requirements, a centralized access management system might not be the best choice for your organization.
- **Vendor lock-in:** Choosing an IdP must be done with care, since changing it down the road might involve significant costs and effort.
- **Delegated Administration of Access to the IdP Owner:** In this model, the IdP owner has significant power and responsibility in managing user identities across AWS accounts. Organizations must ensure their IdP owners can be trusted and/or at least consider this implication in understanding your threat model.

Setting It Up

To set up organization-level SSO, you'll have to create an AWS Organizations organization and enable Identity Center, just like we did for account-level SSO.

First, configure your organization. In AWS Organizations, an organizational unit (OU) is a group of accounts that share the same access control policies. In the following screenshot, there are two OUs: Sandbox and Gamma-Environments.



For each OU, you can then configure any number of users to grant access to. For instance, if you wanted to grant `user_a` access to all Gamma-Environments accounts, you can do so from the AWS Organizations console.

Once you set this up, when `user_a` authenticates via SSO from their Identity Center user portal, they'll be able to access the accounts in Gamma-Environments.

Security Best Practices for This Method

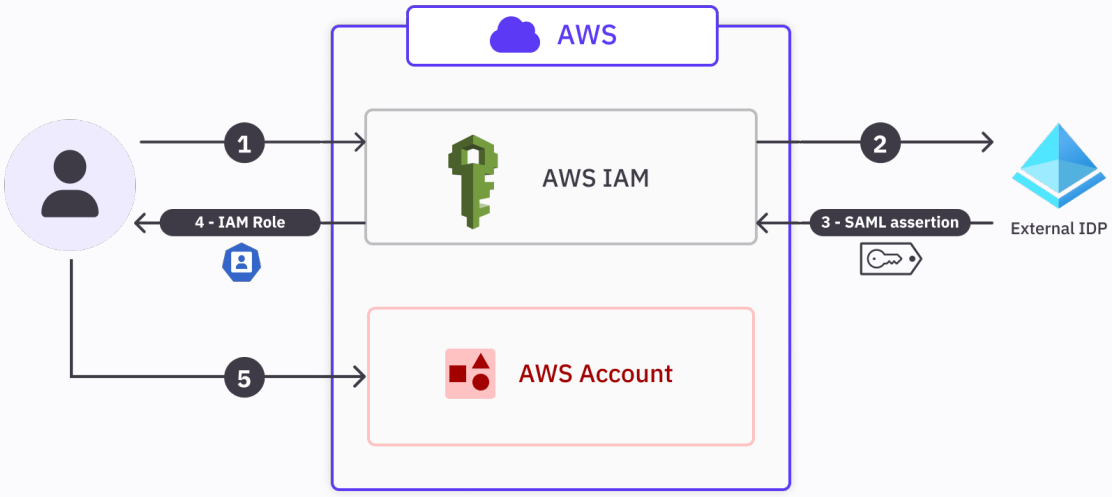
- **Utilize Role-Based Access Control (RBAC):** RBAC assigns users specific roles with pre-defined permissions based on their job responsibilities within the organization. This ensures that all users only have access to the resources necessary for their roles.
- **Regularly monitor and audit security:** Use tools like AWS CloudTrail to track user activity and changes to permissions. Regular security audits help detect suspicious activities and potential security vulnerabilities.
- **Implement PoLP:** The Principle of Least Privilege (PoLP) is a central theme and best practice across all of AWS. Avoid overly permissive roles, and periodically review and adjust user permissions based on changes in roles or responsibilities.

AWS Access Management Method 4: JIT Authorization from Identity Provider

The final AWS access management method is Just-In-Time (JIT) authorization from an IdP. This method allows for on-demand, automatic provisioning of user access to AWS resources when they attempt to sign in.

Central to JIT authorization is the Security Assertion Markup Language (SAML). SAML is an open standard for transmitting identity data between an IdP and a cloud service provider like AWS.

In this access model, when a user attempts to access AWS resources, they're first directed to the IdP for authentication. After the user provides their credentials, the IdP generates a SAML assertion. This contains **SAML groups**, which are elements of information about users such as their identity, group memberships, and any additional attributes relevant to their AWS access credentials. The IdP sends this SAML assertion back to AWS, which then maps these SAML groups over to IAM roles. Here's a diagram illustrating what this workflow typically looks like.



In the above diagram:

1. The user wants to access AWS resources and provides authentication credentials.
2. AWS checks the credentials with a trusted external IdP.
3. The IdP returns a SAML assertion, informing AWS IAM of the various group memberships the user has access to.
4. Based on the SAML groups, IAM provisions and assigns a temporary IAM role to the user.
5. The user can then use this role to access the desired AWS resources.

Advantages of This Method

- **Reduced attack surface:** JIT authorization doesn't use long-term IAM credentials, which greatly reduces the potential for attacks. Users receive temporary credentials for a limited duration, minimizing the window of opportunity for attackers to steal credentials.
- **Minimal credential lifespan:** JIT authorization ensures that access rights are only valid for a short, necessary period of time, reducing the risk of unauthorized access if credentials are compromised.
- **Improved security:** Overall, JIT provides strengthened security since there is little to no risk of inactive or lingering credentials.
- **Simplified user management:** Administrators won't have to manually manage IAM roles for each user. Instead, IAM can automatically provision roles based on the SAML assertions returned by the IdP.

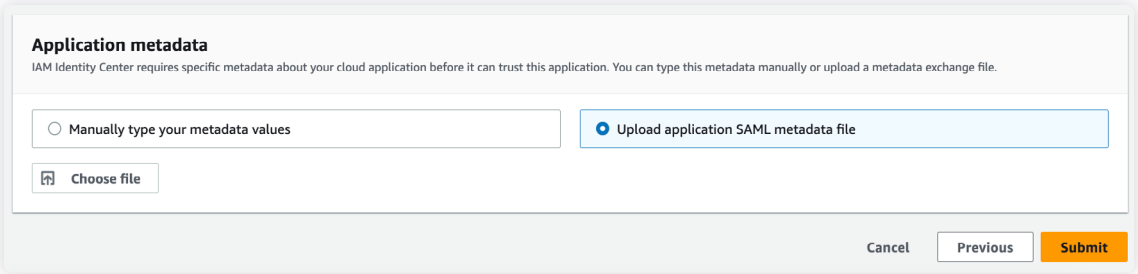
Disadvantages of This Method

- **IdP becomes a single point of failure:** Like the other IdP-centric methods, JIT authorization relies heavily on the availability and reliability of the IdP. Downtime in the IdP often means users won't be able to access AWS resources.
- **Increased complexity of configuration:** Configuring JIT authorization and integrating it with an IdP can be more complex than traditional IAM user management.
- **Deep coupling to IdP and configuration:** With JIT authorization, your IdP and AWS auth become deeply coupled. This can make it challenging to switch to another IdP or modify configurations without impacting access to resources.
- **Difficulty in visibility of access:** Since credentials used in JIT authorization are temporary, it can be challenging to gain real-time visibility into user access rights. Administrators may need to rely on detailed logging to track access patterns effectively.

Setting It Up

In our high-level diagram above, we depicted AWS IAM as the service entity that checks with the IdP and vends credentials back to the user based on the SAML assertion. In practice, AWS Identity Center acts as the entity that does all this and provides SSO capabilities.

As with the other SSO methods, you'll first have to choose an external IdP and enable the Identity Center service. Then, in the Identity Center console, create a custom SAML integration application for your chosen IdP. You'll need to provide the metadata or SAML endpoint URLs provided by the IdP during this setup.



Next, you can use attribute mappings to map group memberships listed by your IdP to AWS IAM roles in the console. Verify that your setup is working as expected by having users sign in through the IdP and access AWS resources.

Security Best Practices for This Method

- **Choose a reputed IdP:** Ensure that your IdP adheres to industry security standards and best practices. A reliable IdP with a proven track record will bolster the security of your SSO environment.
- **Implement MFA and PoLP:** Tried-and-true security measures such as MFA and PoLP are important in making any access management method secure. Both of these mitigate the risk of unauthorized access, even if primary passwords are compromised.

General Best Practices for AWS Access

Regardless of which access model you end up choosing, there are a few general best practices that apply in all cases.

Monitor and Audit User Access Regularly

Have robust monitoring and auditing mechanisms in place to track user access and activities within your AWS environment. Regularly reviewing access logs and audit trails can help you detect suspicious activity early.

You may consider setting up automated alarms that fire when you detect repeated unauthorized access attempts.

Consider Implementing Time-based Access Control

Implement time-based access control policies to restrict user access to AWS resources. For instance, you might have a policy set up to restrict access to sensitive resources during non-business hours, except for select folks who are on-call and need access to them.

Enable User Activity Notifications (or Audit Trail)

Enable AWS CloudTrail to capture user activity logs and API calls. CloudTrail provides a detailed audit trail of events, allowing you to monitor and review user actions and changes to your AWS infrastructure in real-time.

Regularly Review Active Accounts to Ensure Security and Reduce Costs

Conduct periodic reviews of active AWS accounts to ensure that only authorized users have access. Your organization should define clear processes for deactivating or removing accounts for employees who no longer require AWS access to minimize security risks and control costs.

Conclusion

In this article, we covered four methods to control and secure user access to AWS resources. We started by understanding the fundamental concepts of AWS access control by exploring the most straightforward model provided by AWS IAM. From there, we explored the advantages and disadvantages of using different methods, including account-level SSO, organization-level SSO, and JIT authorization with an IdP.

Overall, local IAM users offer granular controls and more flexibility but come with more administrative overhead. On the other hand, the SSO-based methods provide centralized management and streamlined user experiences at the cost of additional setup complexity and a hard dependency on external IdPs. To effectively manage access to AWS resources, your organization must carefully select the appropriate access management method that best aligns with your security needs and user workflows.

If you need a reliable identity security solution, consider ConductorOne to help simplify your access management workflows. With ConductorOne, you can effortlessly implement JIT access control, centralize access management, and all the security best practices we discussed in this article, all while providing a seamless user experience.