AWS Directory Service

What is AWS Directory Service?

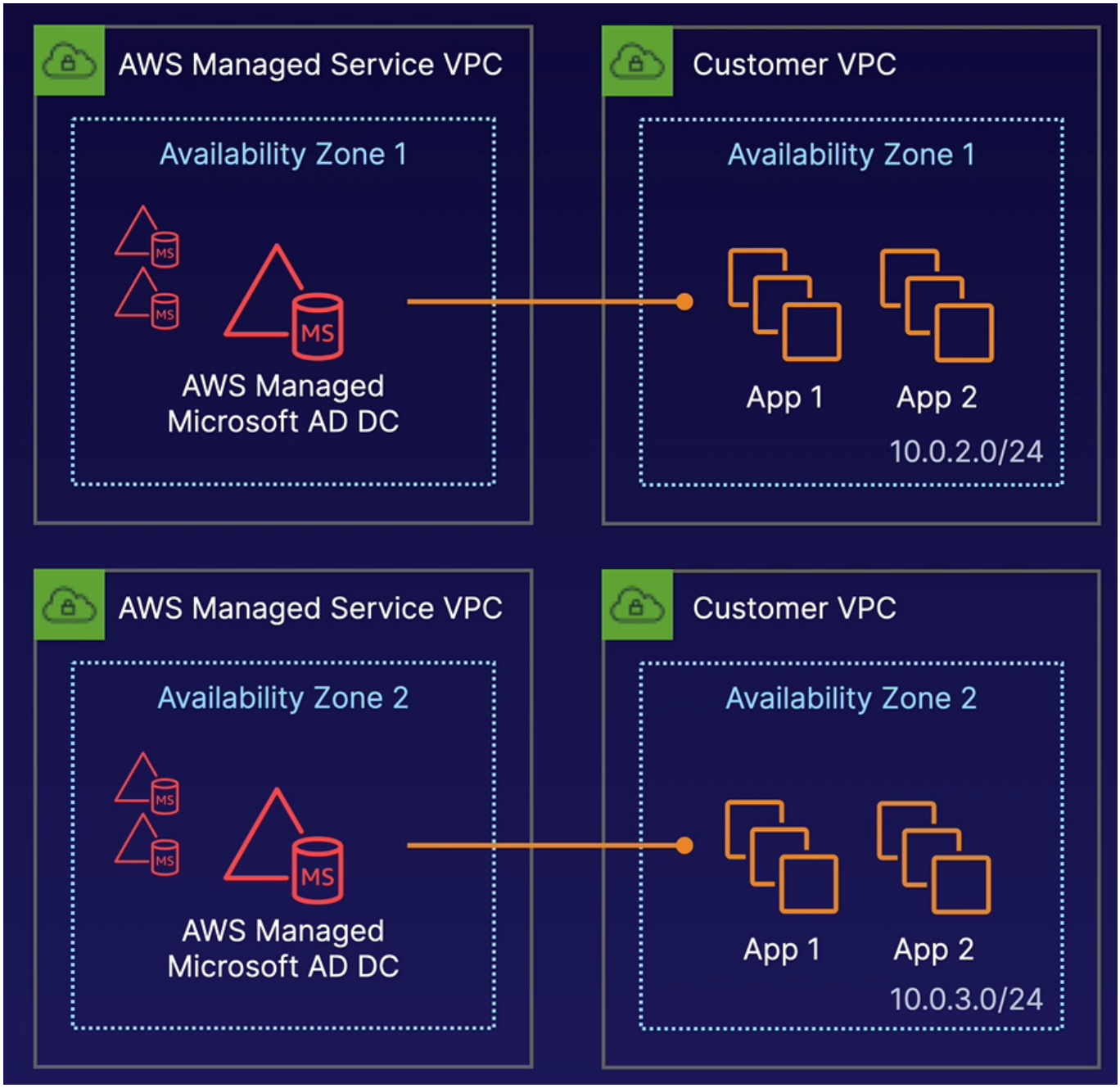
- Family of managed services
- Connect AWS resources with on-premises AD
- Standalone directory in the cloud
- Use existing corporate credentials
- Can also log into AWS management console using the same existing corporate credentials
- SSO (Single-sing-on) to any domain-joined EC2 instance

What is Active Directory

- On-premises directory service
- Hierarchical database of users, groups, computers -- trees and forests
- Group policies
- Supports LDAP and DNS
- Kerberos, LDAP, and NTLM authentication
- Highly available configuration

AWS Managed Microsoft AD

- AD domain controllers (DCs) running Windows Server
- Reachable by applications in your VPC
- Add DC's for HA and performance
- Exclusive access to DCs
- Extend existing AD to on-premises using AD Trust



AWS Managed Microsoft AD

AWS

- Multi-AZ deployment
- Patch, monitor, recover
- Instance rotation
- Snapshot and restore

Customer

- Users, groups, GPOs
- Standard AD tools

- Scale our DCs
- Trusts (resource forest)
- Certificate authorities (LDAPS)
- Federation

Simple AD

- Standalone managed directory
- Basic AD features
- Small: <= 500; Large <= 5,000 users
- Easier to manage EC2
- Linux workloads that need LDAP
- Does not support trusts (can't join on-premises AD)

AD Connector

- Directory gateway (proxy) for on-premises AD
- Avoid caching information in the cloud
- Allow on-premises users to log in to AWS using AD
- Join EC2 instances to your existing AD domain
- Scale across multiple AD Connectors

Cloud Directory

- Directory-based store for developers
- Multiple hierarchies with hundreds of millions of objects
- Use cases: org charts, course catalogs, device registries
- Fully managed service

Amazon Cognito User Pools

- Managed user directory for SaaS applications
- Sign-up and sign-in for web or mobile
- Works with social media identities

AD vs Non-AD Compatible Services

AD

- Managed Microsoft AD (a.k.a., Directory Service for Microsoft Active Directory)
- AD Connector
- Simple AD

Not AD Compatible

- Cloud Directory
- Cognito user pools

IAM Policies

Amazon Resource Name (ARN)

Uniquely identifies any resource within AWS.

ARNs begin with: `arn:partition:service:region:account_id`

- `partition` → [aws | aws-cn] , aws-cn is for china region
- `service` → [s3 | ec2 | rds]
- `region` → [us-east-1 for example]
- `account_id` → 123456789012

ARNs end with:

```
resource
resource_type/resource
resource_type/resource/qualifier
resource_type/resource:qualifier
resource_type:resource
resource_type:resource:qualifier
```

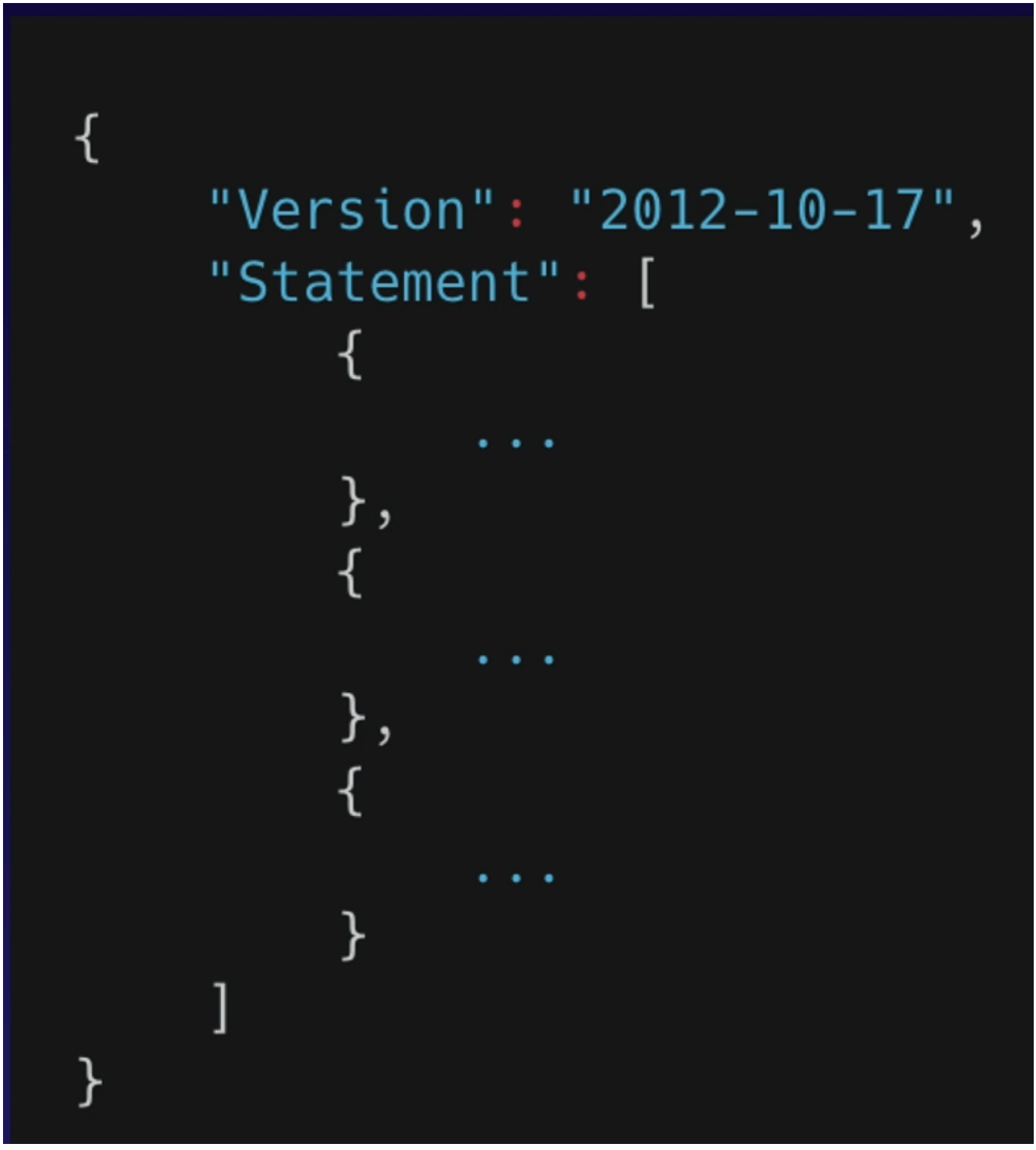
Examples:

```
arn:aws:iam::123456789012:user/mark
arn:aws:s3:::my_awesome_bucket/image.png
arn:aws:dynamodb:us-east-1:123123123123:table/orders
arn:aws:ec2:us-east-1:123456789012:instance/*
```

- in the first one the double `::` after `iam` is there to omit the region, this is because `iam` is a global region; meaning it doesn't belong to a particular region
- in the second example `:::` the region and account id are omited, this is because there is no specific region or account id need to identiyfy an object within S3. All bucket names in S3 are already globally unique.
- in the last example, we are reffering to all EC2 instance using the `*` wildcard

IAM Policies

- JSON document that defines permissions
- Identity policy
- Resource policy
  - attached to resources, S3 buckets for example
- No effect until attached
- List of statements



- A policy document is a list of statements.
- Each statement matches AWS API request.
- Effect is either `Allow` or `Deny`
- Matched based on their `Action`
- And the `Resource` the action is against

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGet*",
                "dynamodb:DescribeStream",
                "dynamodb:DescribeTable",
                "dynamodb:Get*",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWrite*",
                "dynamodb:CreateTable",
                "dynamodb:Delete*",
                "dynamodb:Update*",
                "dynamodb:PutItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/MyTable"
        }
    ]
}
```

Creating a Policy

IAM Console → Policies →

- There are two types of polices:
  - AWS managed policies, preconfigured policies created by AWS for your convience: NOT EDITABLE
  - Customer managed polices, polices created by you, and these are editable
  Create Polic → Can use visual editor or JSON, choose JSON →

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::test"]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": ["arn:aws:s3:::test/*"]
        }
    ]
}
```

Example:
We want this policy to:

- Allow a number of actions enabled on a bucket in s3
  - In the first statment, we allow the listbucket action on the resource `test` in s3
- In the second statement, we allow 3 different api calls to take effect on the object, and because the api calls oeprate on the object level, we use the wildcard `*` to apply it to all objects within the bucket
  → Review → provide it a name, description is optional → Create policy

The policy has no effect until it's attached to something.

Lets attach the policy to a role
IAM Console → Roles → Create Role → EC2 → filter the name given above → Review → provide it a name → Create role

Note:
Inline policies are limited to just the role it is attached to, they can not be attached to other roles. It lives only within the scope of the role it is attached to.

Exam Tips

- Not explicitly allowed == implicitly denied
- Explicit deny > everything else
- Only attached policies have effect
- AWS joins all applicable policices
- AWS-managed vs customer-managed policies

Permission Boundaries

- Used to delegate administration to other users
- Prevent privilege escalation or unnecessarily broad permissions
- Control maximum permissions an IAM policy can grant
- Use cases:
    - Develoeprs creating roles for Lambdo functions
    - Application owners creating roles for EC2 instances
    - Admins creating ad hoc users

AWS Resource Access Manager (RAM)

RAM

AWS Resource Acess Manager (RAM) allows resouce sharing between accounts.

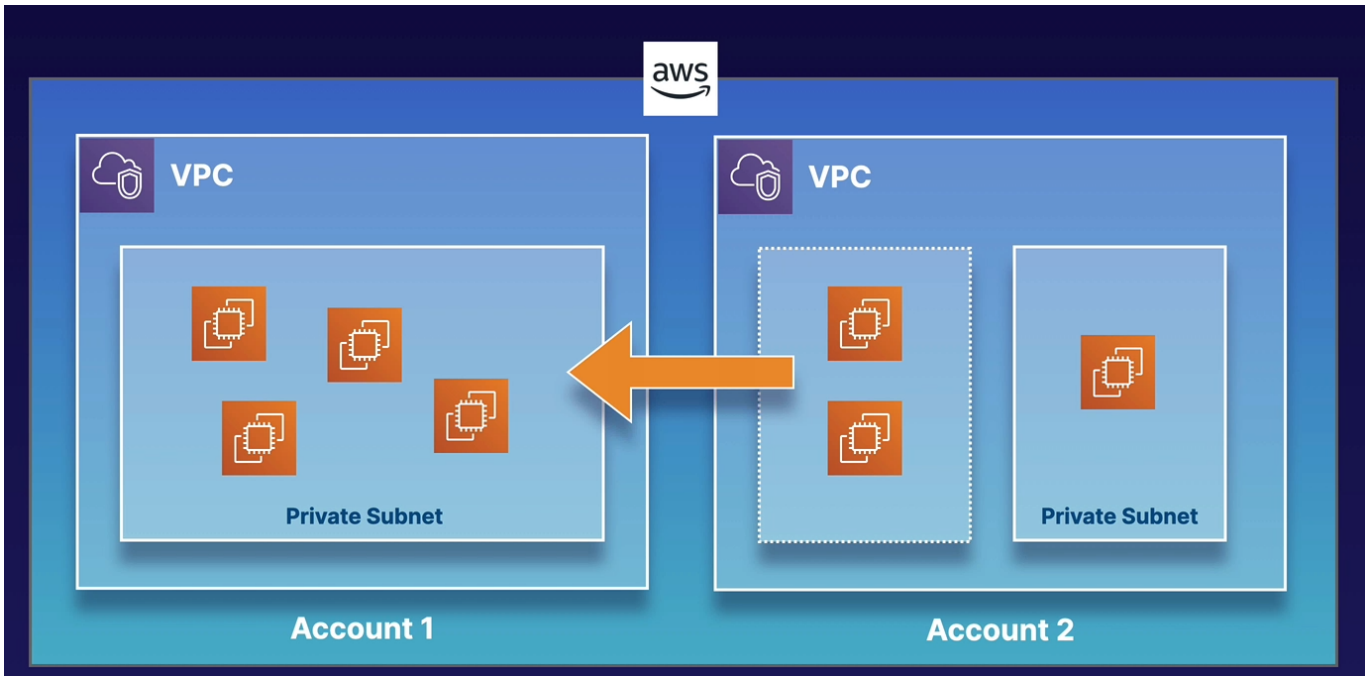
- Individual Accounts
- AWS Organizations

Which AWS resources can you share using RAM?

- APP Mesh
- Aurora
- CodeBuild
- EC2
- EC2 Image Builder
- License Manager
- Resource Groups
- Route53

Example

Launch EC2 instances in a shared subnet



Example 2

What if you want to share a database between two accounts, ofr example an aurora database

Account One
The account with the database access initially
RAM Console → Create a resouce share → provide it a name → and under resource type select "Aurora DB Clusters" → Select the datanase to share with account two, and under "Principals"; need to allow external accounts, enter the account two accound id → Create Resource Share

The resource isn't shared until account two accepts the resource share

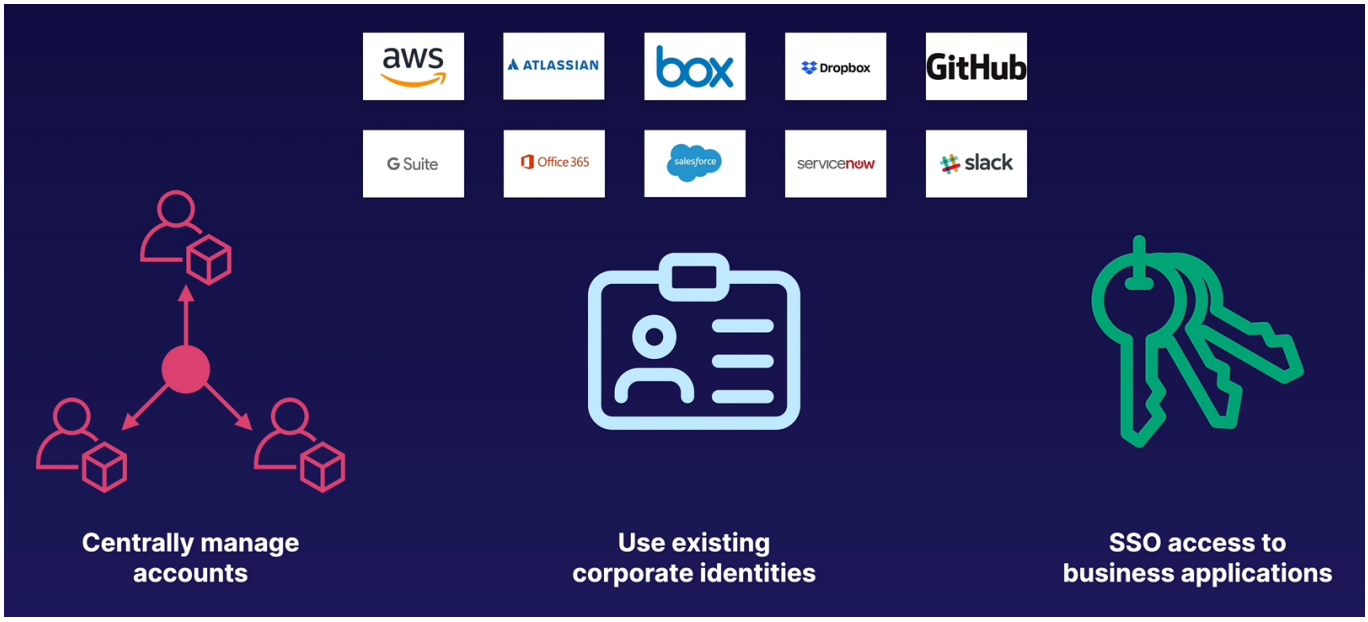
Account Two
RAM Console → Shared with me → Resource Shares → Select the database shared by acocunt one → Accept Resource Share → OK

Now the database is shared with the acoounts, and account two can clone the database

AWS Single Sign-On

What is AWS Single Sign-On

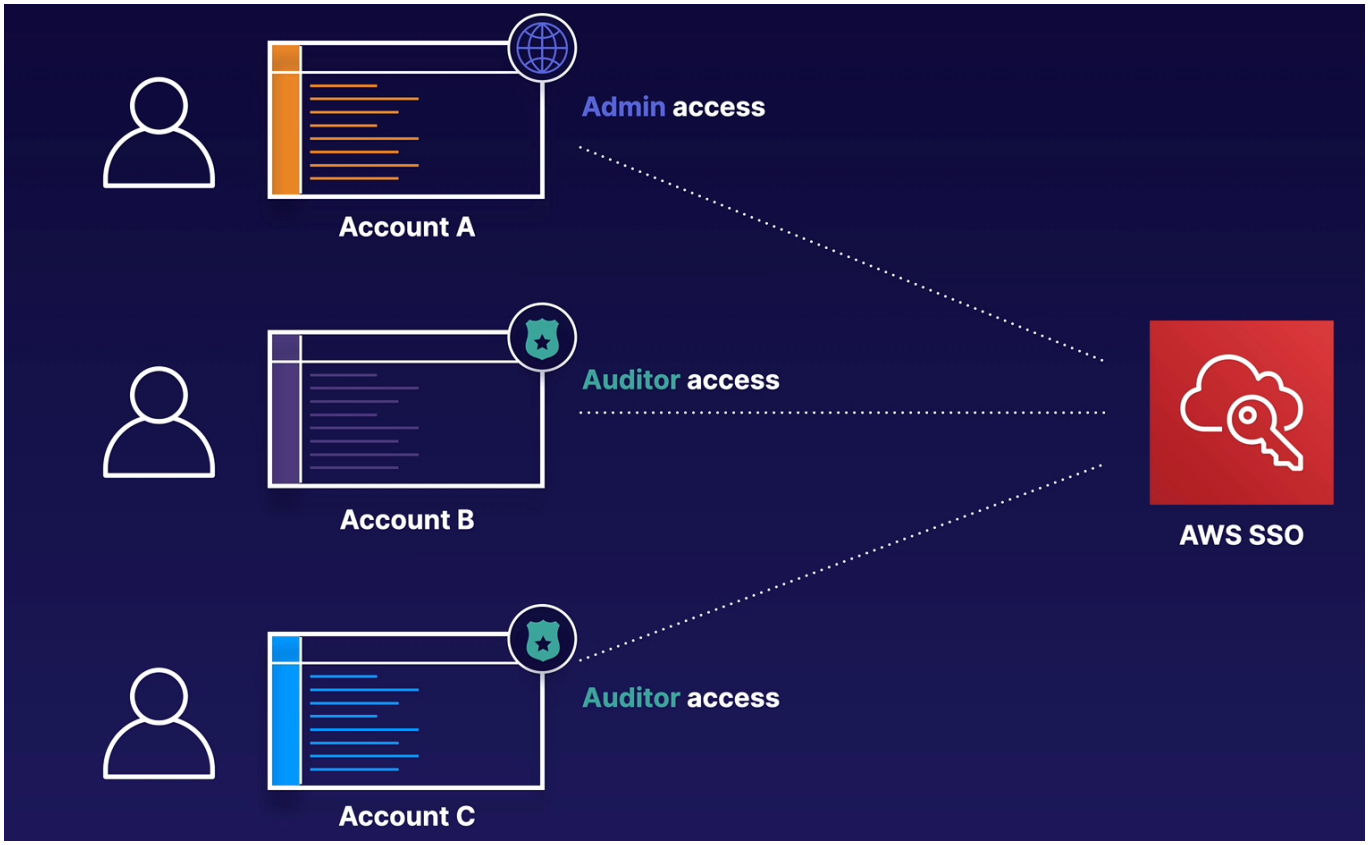
- Single sign-on (SSO) service helps centrally manage access to AWS accounts and business applictions



- Centrally manage accounts
- Use existing corporate identities
- SSO access to business applications

Granular Account-Level Permissions

If you want to grant admin access to security team that are only running security tools, but also want to grant auditor permission to other AWS accounts

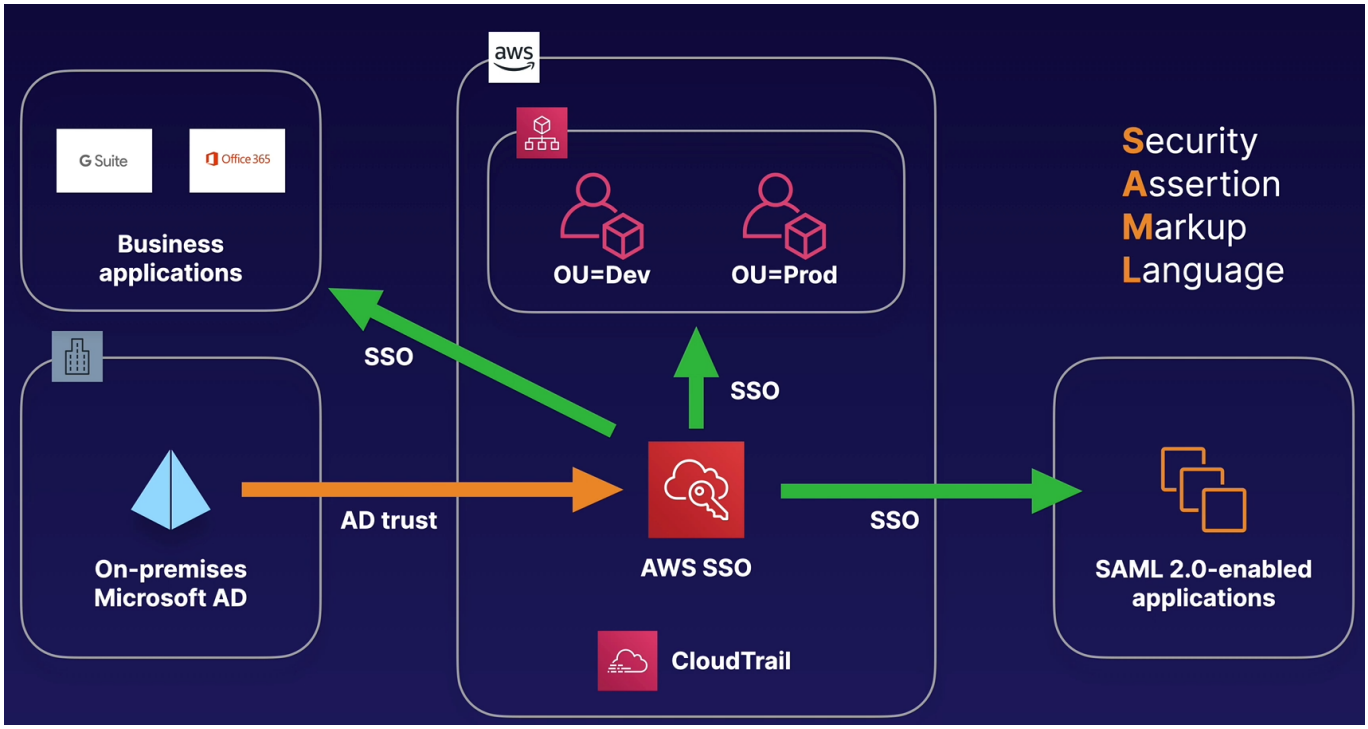


Active Directory and SAML Integration

- SSO allows you to login to business applications
- It also integrates with Active Directory, or any SAML 2.0 identity provider (e.g. Azure AD)
    - This allows them to log into AWS SSO portable with their active directory credentials
    - This can also be used to grant access to AWS organizations
- Also enables access to any SAML 2.0-enabled applications

SAML - Security Assertion Markup Language

All sign on activity is logged with CloudTrail



Advanced IAM Summary

- Active Directory
- Connect AWS resources with on-premises AD
- SSO to any domain-joined EC2 instance
- AWS Managed Microsoft AD
- AD trust
- AWS vs Customer responsibility

- Simple AD
- Does not support trusts
- AD Connector
    - directory gateway or proxy for your on-premise AD
- Cloud Directory -- nothing to do with Microsoft AD
    - a service looking to work with hierarchical data
- Cognito user pools -- nothing to do with AD
    - managed user pools, used with social media identities
- AD vs Non-AD compatible services

IAM Policies

- ARN
- IAM policy structure
- Effect/Affect/Resource
- Identity vs resource policies
- Policy evaluation logic -- allows vs denys
- AWS managed vs customer managed
- Permission boundaries

Resource Access Manager

- Resource sharing between accounts
- Individual accounts and AWS Organizations
- Types of resources you can share

Single Sign On

- Centrally manage access
- Examples: G Suite, Office 365, Salesforce
- Using existing Identities
- Account-level permissions
- SAML