**Advanced IAM**

**Sections**

1. AWS Directory Service
2. IAM Policies
3. AWS Resource Access Manager (RAM)
4. AWS Single Sign-on Service

**AWS Directory Service**

1. What is AWS Directory Service
   1. Family of managed services
   2. Connect to AWS resources with on-premises AD (Active Directory)
   3. Standalone directory in the cloud
   4. Use existing corporate credentials
   5. SSO (Single Sign-on) to any domain-joined EC2 instances (Joined AD domain). No need to manage credentials on EC2 instances
2. What is Active Directory
   1. On-premises directory service
   2. Hierarchical databases of users, groups, computers – trees and forests
   3. Group policies
   4. LDAP (Lightweight Directory Access Protocol) and DNS (Domain Name Service) protocols
   5. Authentication – Kerberos, LDAP, and NTLM
   6. Highly available, also difficult to manage
3. AWS Managed Microsoft AD
   1. AD domain controllers (DCs) running Windows Server
   2. Reachable by applications in your VPC
   3. Add DCs for HA and performance
   4. Exclusive access to DCs
   5. Extend existing AD to on-premises using **AD Trust**
4. Responsibilities
   1. AWS
      1. Multi-AZ
      2. Patch, monitoring, recovery
      3. Instance rotation
      4. Snapshot and restore
   2. Your responsibility
      1. Users, groups, GPOs
      2. Standard AD tools
      3. Scale out DCs
      4. Trust (Resource forest)
      5. Certification authorities (LDAPS)
      6. Federation
5. Simple AD
   1. Standalone managed directory
   2. Basic AD features
   3. Small <= 500 users; large <= 5000 users
   4. Easier to manage EC2
   5. Linux workloads that need LDAP
   6. Does not support TRUST (CANNOT join on-premises AD)
6. AD Connector
   1. Directory gateway (Proxy) for on-premises AD
   2. Avoid caching information in the cloud
   3. Allow on-premises users to log into AWS using AD
   4. Join EC2 instances to your existing AD domain
   5. Scale across multiple AD Connectors
7. Three Microsoft Compatible Services
   1. **Microsoft Managed AD**
   2. **Simple AD**
   3. **AD Connector**
8. Cloud Directory
   1. Directory-based store for developers
   2. Multiple hierarchies with hundreds of millions of objects
   3. Use cases: Org charts, Course categories, Device registries
   4. Fully managed AWS service
9. Amazon Cognito User Pools
   1. Managed user directory for SaaS (Software as a Service) applications
   2. Sign-up and sign-in for web or mobile apps
   3. Works with Social Media identities
10. AD Compatible / Not AD Compatible
    1. Compatible
       1. Managed Microsoft AD (a.k.a, Directory Service for Microsoft Active Directory)
       2. AD Connector
       3. Simple AD
    2. Not Compatible
       1. Cloud Directory – Directory base store – Similar to DynamoDB?
       2. Cognito User Pools
       3. **Cognito is region specific**

**IAM Policies**

1. Amazon Resource Name (**ARN**)
   1. Uniquely identify any AWS resource
   2. Begin with “**arn:partition:service:region:account\_id**”
      1. Partition: aws | aws-cn (China), Service: s3 | ec2 | rds
   3. End with “**resource, resource\_type/resource. etc**”
2. IAM Policies (JSON **document**)
   1. **Identity Policy** – Attach to User/Group/Role – What an Identity can do
   2. **Resource Policy** – Attach to Resources (for example, s3 bucket)
   3. **No effect until attached**
   4. List of **statements** – Each statement matches one or more AWS API requests (**Actions**) (console or cli or sdk (boto3), etc)
   5. **Principle** section and **Resource** section
3. Inline Policies
   1. Ad hoc to a particular role
   2. Can not be used by any other roles
4. Final Effect
   1. Not explicitly allowed == implicitly denied
   2. Explicitly deny > Everything else
   3. AWS joins all applicable policies
5. Permission Boundaries
   1. Used to delegate administration to other users
   2. Prevent privilege escalation or unnecessary broad permissions
   3. Controls maximum permissions on IAM policy can grant (Permission boundary like Policy document, but it does not grant permissions by itself. It set the max possible permissions of the role/user. Groups have no permission boundary)
   4. Use cases
      1. Developers creating roles for Lambda functions
      2. Application owners creating roles for EC2 instances

**AWS Resource Access Manager (RAM)**

1. Why Resource Access Manager
   1. **Multiple AWS accounts** or an AWS Organization
   2. RAM is used to share resources with other accounts
2. Resources can be shared (8)
   1. App mesh
   2. **Aurora**
   3. **CodeBuild**
   4. **EC2**
   5. EC2 Image Builder
   6. License Manager
   7. Resource Groups
   8. **Route 53**
3. Go to Resource Access Manager, choose the resource, and put in the account id to share with. The other account then accept the sharing. It will be shared and visible

**AWS Single Sign-on Service**

1. SSO service helps centrally manage access to **WAS accounts** and **business applications**
2. SSO integrates with **Microsoft AD** and any (**Security Assertion Markup Language**) **SAML 2.0** Identity Provider
3. If **SAML 2.0** => **Look for SSO**

**Summary**

1. Directory Service
2. IMA Policies
3. Resource Access Manager (RAM)
4. SSO service
   1. Using Existing Identity
   2. SAML 2.0