1. You are a cloud architect for a medium-sized company that has just started migrating its on-premises infrastructure to AWS. The company has various departments including **Finance, Development, and Marketing**. Each department has a distinct set of users with different access needs. For example:
   * **Finance** users need access to sensitive financial data stored in **Amazon S3** and the ability to run reports using **Amazon Athena**.
   * **Development** users need access to launch and manage **EC2 instances, RDS databases**, and access source code stored in **S3**.
   * **Marketing** users need access to certain **S3** buckets that store customer-facing content and analytics data from **Amazon QuickSight**.
   * Access to AWS resources must follow the principle of least privilege.
   * Developers should be allowed only to access EC2 instance with specific tags.

Ans:

Create User Groups with Permission Policies :

**Finance-Policy**

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"athena:TagResource",

"athena:TerminateSession",

"athena:UntagResource",

"athena:StartQueryExecution",

"athena:GetQueryResultsStream",

"athena:GetQueryResults",

"athena:GetQueryExecutions",

"athena:ListQueryExecutions",

"athena:ListNamedQueries",

"athena:GetSessionStatus",

"athena:StopQueryExecution",

"athena:GetQueryExecution",

"athena:StartSession",

"athena:ListExecutors",

"athena:ListPreparedStatements"

],

"Resource": "\*"

},

{

"Sid": "VisualEditor1",

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:GetObject",

"s3:UntagResource",

"s3:TagResource",

"s3:ListBucket"

],

"Resource": [

"arn:aws:s3:::finance-data-s3/\*",

"arn:aws:s3:::finance-data-s3"

]

}

]

}

**Development-Policy**

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"ec2:DescribeInstances",

"ec2:DeleteVpcEndpoints",

"ec2:CreateKeyPair",

"ec2:StartInstances",

"ec2:DisassociateSecurityGroupVpc",

"ec2:CreateInternetGateway",

"ec2:DescribeVolumes",

"ec2:AssociateRouteServer",

"ec2:DetachVolume",

"ec2:ImportKeyPair",

"ec2:CreateTags",

"ec2:CreateInstanceConnectEndpoint",

"ec2:ModifyNetworkInterfaceAttribute",

"ec2:DeleteNetworkInterface",

"ec2:DetachInternetGateway",

"ec2:StopInstances",

"ec2:AssignPrivateIpAddresses",

"ec2:CreateVolume",

"ec2:EnableVolumeIO",

"ec2:CreateNetworkInterface",

"ec2:AssociateSecurityGroupVpc",

"ec2:CreateDefaultVpc",

"ec2:DescribeInstanceConnectEndpoints",

"ec2:AssociateSubnetCidrBlock",

"ec2:DeleteNatGateway",

"ec2:DeleteVpc",

"ec2:CreateSubnet",

"ec2:DescribeSubnets",

"ec2:DeleteKeyPair",

"ec2:ModifyVpcEndpoint",

"ec2:AttachVolume",

"ec2:DeleteInstanceConnectEndpoint",

"ec2:CreateNatGateway",

"ec2:DescribeInstanceAttribute",

"ec2:CreateVpc",

"ec2:CreateSubnetCidrReservation",

"ec2:CreateDefaultSubnet",

"ec2:DescribeSecurityGroupVpcAssociations",

"ec2:ModifyVpcAttribute",

"ec2:DescribeInstanceStatus",

"ec2:RebootInstances",

"ec2:AssociateNatGatewayAddress",

"ec2:TerminateInstances",

"ec2:DetachNetworkInterface",

"ec2:DescribeTags",

"ec2:DescribeNatGateways",

"ec2:DisassociateSubnetCidrBlock",

"ec2:DescribeSecurityGroups",

"ec2:DisassociateNatGatewayAddress",

"ec2:CreateInstanceEventWindow",

"ec2:DeleteInstanceEventWindow"

],

"Resource": "\*",

"Condition": {

"StringEquals": {

"aws:ResourceTag/Department": "Development"

}

}

},

{

"Sid": "VisualEditor1",

"Effect": "Allow",

"Action": [

"rds:AddTagsToResource",

"rds:RestoreDBInstanceFromS3",

"rds:ModifyTenantDatabase",

"rds:DeleteTenantDatabase",

"rds:StopDBInstance",

"rds:StartDBInstance",

"rds:ListTagsForResource",

"rds:CreateDBSnapshot",

"rds:RestoreDBInstanceFromDBSnapshot",

"rds:RebootDBInstance",

"rds:CreateDBInstance",

"rds:DescribeDBInstances",

"rds:ModifyDBInstance",

"rds:RemoveTagsFromResource",

"rds:DescribeDBClusters",

"rds:CreateTenantDatabase",

"rds:DeleteDBInstance"

],

"Resource": "\*"

},

{

"Sid": "VisualEditor2",

"Effect": "Allow",

"Action": [

"s3:PutObject",

"s3:GetObject"

],

"Resource": "arn:aws:s3:\*:853973692277:accesspoint/\*/object/\*"

},

{

"Sid": "VisualEditor3",

"Effect": "Allow",

"Action": [

"s3:ListTagsForResource",

"s3:PutObject",

"s3:GetObject",

"s3:UntagResource",

"s3:TagResource",

"s3:ListBucket"

],

"Resource": "arn:aws:s3:::development-source-code-s3"

}

]

}

**Marketing-Policy**

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"quicksight:DescribeDashboard",

"quicksight:ListAnalyses",

"quicksight:ListDashboards",

"quicksight:TagResource",

"quicksight:GetDashboardEmbedUrl",

"quicksight:UntagResource",

"quicksight:SearchDashboards",

"quicksight:ListTagsForResource"

],

"Resource": "\*"

},

{

"Sid": "VisualEditor1",

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:ListBucket"

],

"Resource": [

"arn:aws:s3:::customer-facing-content-s3/\*",

"arn:aws:s3:::customer-facing-content-s3"

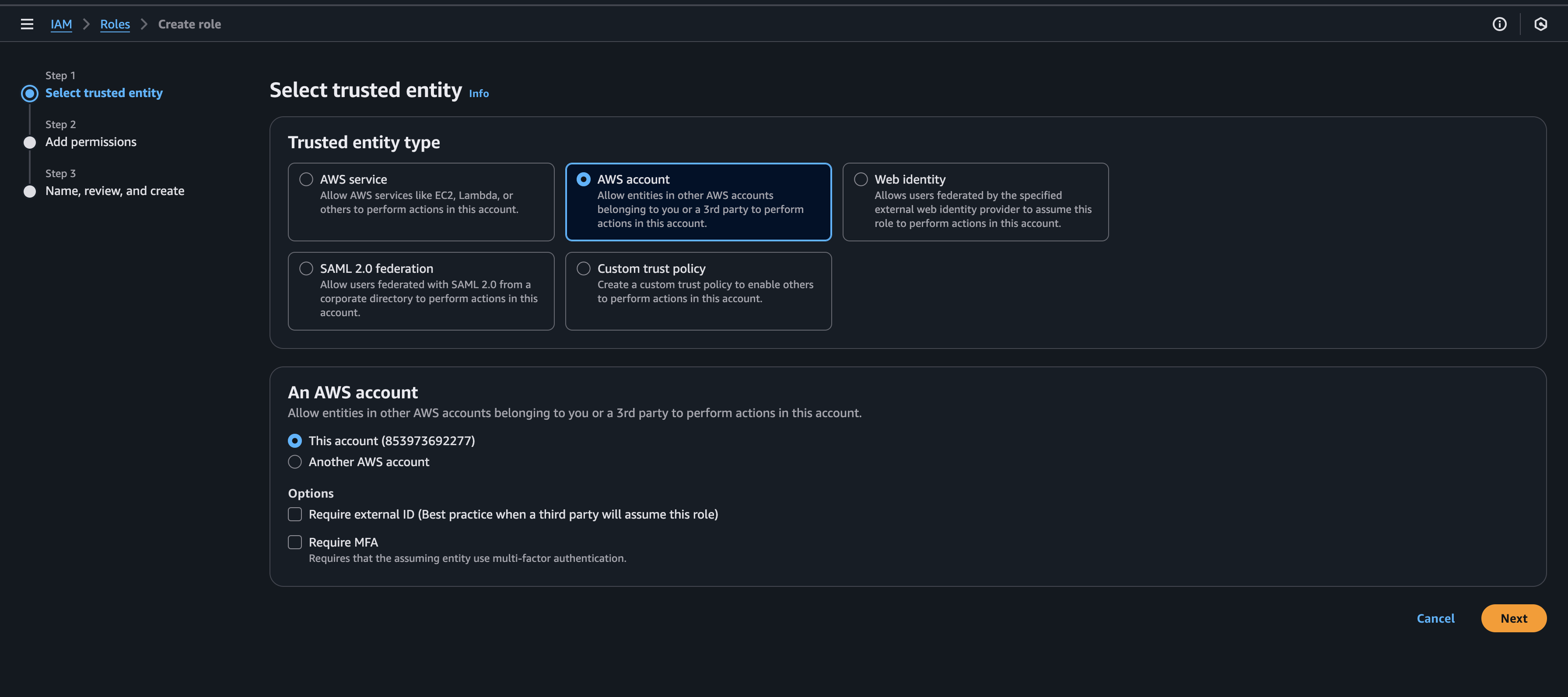
]

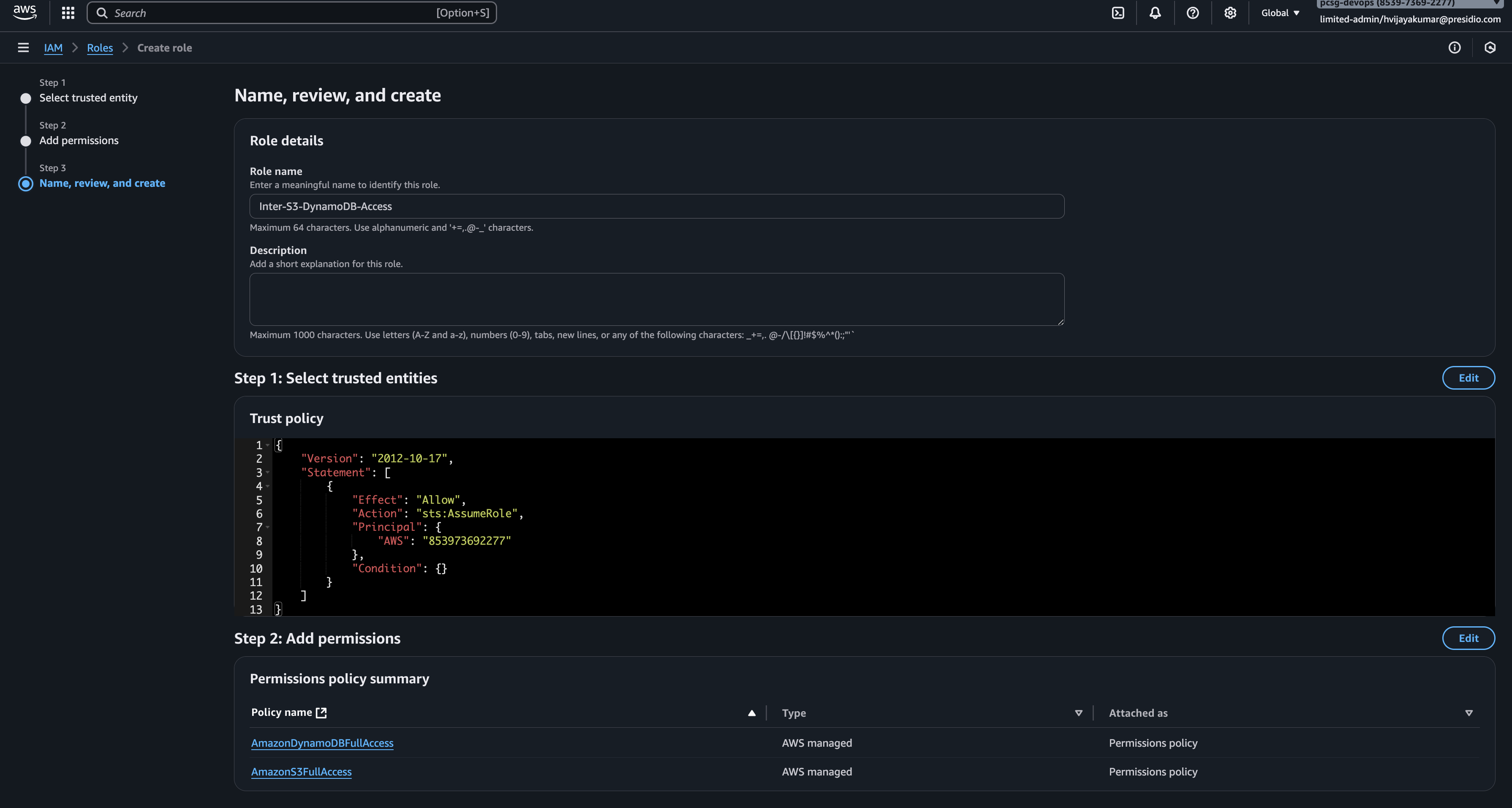
}

]

}

1. You are working on an application hosted in **Account A**, but some required resources, like an S3 bucket and a DynamoDB table, are in **Account B**. To securely access these resources, you will use **IAM Assume Role**.  
     
   **Task:** Create IAM roles to achieve the above requirement.  The role creation in the same account would suffice, if you don’t have access to two accounts.





1. You are the cloud architect for a large university that uses AWS for various academic  
   and research projects. The university has multiple departments (Computer Science,  
   Biology, Physics, etc.), and each department runs its projects on AWS. Technical staLs, Non-technical staLs and students in each department already have accounts in the university’s Central Database system, which is managed by the IT department.  
   The university wants to integrate the existing **Central Database** with AWS so that users from different departments can access the AWS Management Console and other  
   services. However, there are some strict guidelines:  
     
     
   **Tasks (Describe how will you implement the scenario):**
   * **How would you integrate the university's Central Database with AWS so that users can log in with their login credentials?**

Ans :

If university already had an Identity Provider Organisation Account, then it can be connected with AWS IAM Identity Center and create an AWS Organization.

If university doesn’t have any Identity Provider, Then University can integrate their central database with AWS Active Directory and deploy any DB based Identity Provider.

* + **How would you assign different permissions to faculty and students, ensuring they only access resources relevant to their department?**

Ans :

Creating user groups with LPP : Student, Faculty

Creating separate AWS accounts for each department and assign users to the AWS accounts based on their department.

* + **How would you enforce MFA for users before they can access AWS?**

Ans :

We can enforce MFA by two different ways :

1. Through AWS IAM Identity Center
2. Through Identity Provider

To enforce MFA for users before they can access AWS, Identity Provider MFA is utilized.

* + **How would you configure the system so that users can only access the AWS account assigned to their department, even though all accounts are part of a single AWS Organization?**

Ans : IAM Identity Center uses Assignment to assign the permission to the Account. ABAC (Attribute Based Access Control) can also configured. We can also enforce Service Control Policy for denying access of AWS Account from other department users.

Every user has attribute department and role like aws:PrincipalTag/department and aws:PrincipalTag/role. We can assign these with help of SCIM (System of Cross-Domain Identity Management).