A) Understand the basics of IAM, its purpose, and key concepts.

- Topics:

  - Overview of AWS IAM and its importance in managing access to AWS services.

AWS IAM Overview

AWS Identity and Access Management (IAM) enables you to manage access securely to AWS services and resources. It is critical for defining who can access what within AWS.

- Key concepts: Users, Groups, Roles, Policies.
  - Users :- authentication, authorisation
  - authentication methods:-
  - password, key based authentication (secret key/access key)

Key Concepts

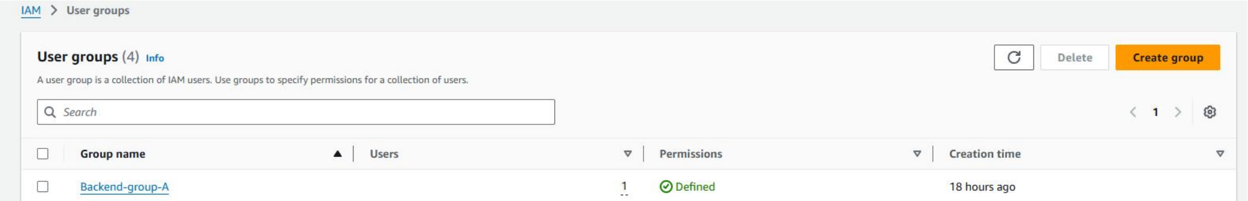
1. **Users**
   - **Purpose**: Represents individuals or applications that need access to AWS services.
   - **Authentication**: The process of verifying the identity of a user.
     - **Methods**:
       - **Password Authentication**: Users sign in with a username and password.
       - **Key-Based Authentication**: Users authenticate using an **access key ID** and **secret access key** for programmatic access via the AWS CLI or API.
   - **Authorization**: Specifies what actions a user can perform.

2. **Groups**
   - **Purpose**: Organizes multiple users and assigns permissions to the group, simplifying the management of permissions.
   - **Example Setup**:
     - **Backend Developers Group**: Access to backend servers for instance management.
     - **Database Administrator Group**: Access limited to managing RDS (Relational Database Service).
3. **Roles**
   - **Purpose**: Enables AWS services or applications to assume a set of permissions temporarily, usually for cross-account access or application services.
   - Roles have no long-term credentials, increasing security.
4. **Policies**
   - **Purpose**: Defines permissions as JSON documents specifying what actions are allowed or denied.
   - **Types**:
     - **AWS Managed Policies**: Predefined by AWS, suitable for common use cases.
     - **User Managed Policies**: Custom policies created and managed by users.
   - **Identity-Based Policies**: Attach to users, groups, or roles to control what actions they can perform.
   - **Resource-Based Policies**: Attach directly to AWS resources (e.g., S3 bucket policies) to control access.

➢ Groups

setting up 2 different groups with different resource access

1. backend developers should have access to backend servers so they can manage those instances

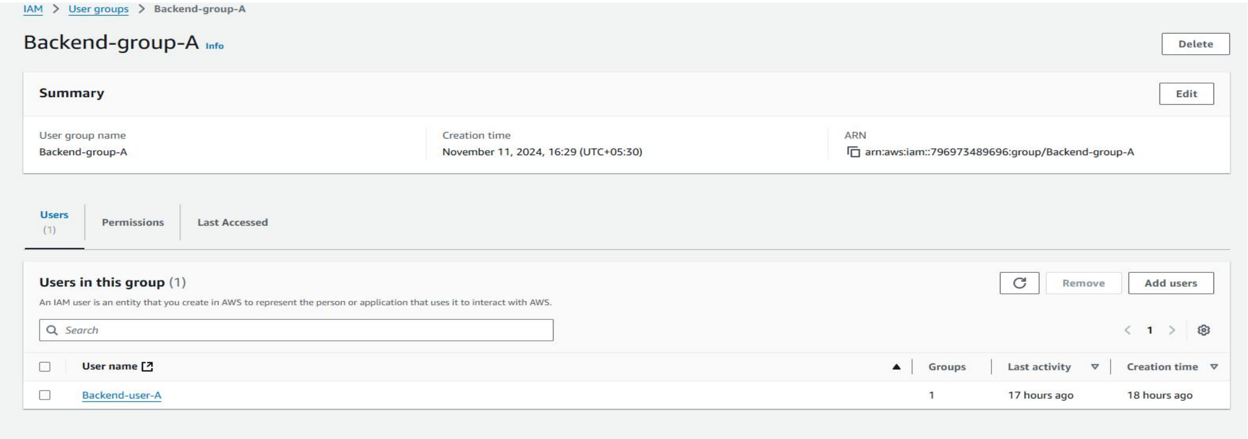


IAM > User groups

User groups (4) Info

A user group is a collection of IAM users. Use groups to specify permissions for a collection of users.

□	Group name	Users	Permissions	Creation time
□	Backend-group-A	1	✓ Defined	18 hours ago



IAM > User groups > Backend-group-A

Backend-group-A Info

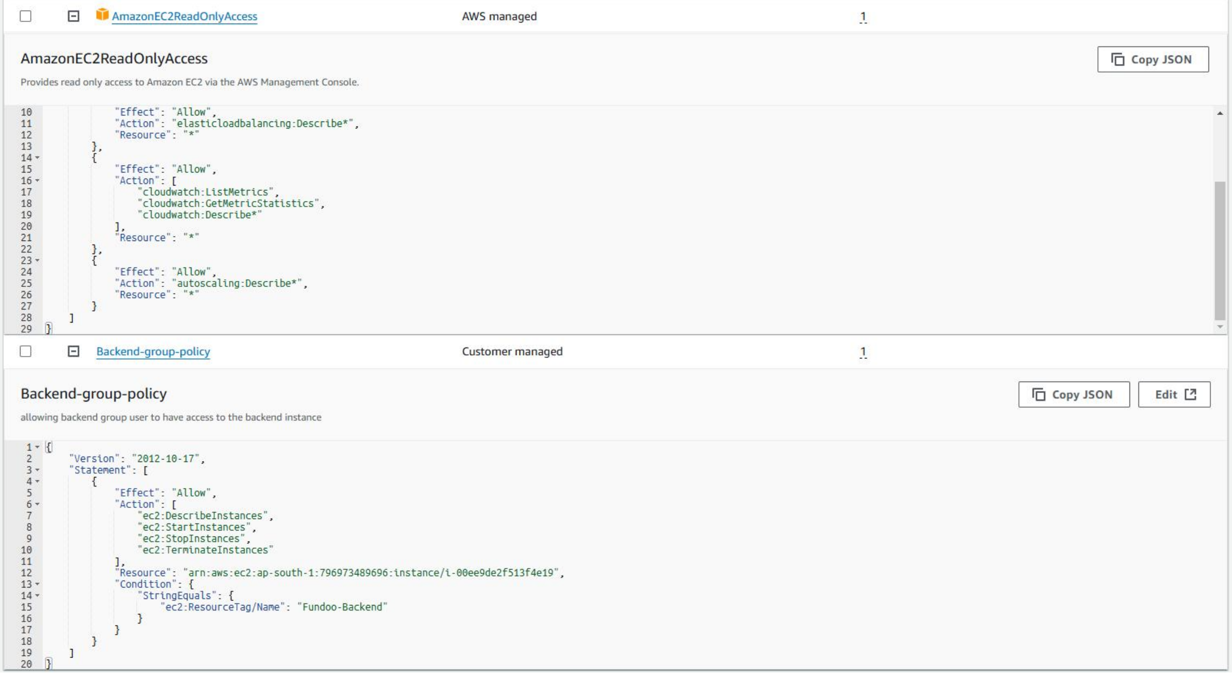
Summary

User group name	Creation time	ARN
Backend-group-A	November 11, 2024, 16:29 (UTC+05:30)	arn:aws:iam::796973489696:group/Backend-group-A

Users in this group (1)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

□	User name	Groups	Last activity	Creation time
□	Backend-user-A	1	17 hours ago	18 hours ago



□	AmazonEC2ReadOnlyAccess	AWS managed	1

AmazonEC2ReadOnlyAccess

Provides read only access to Amazon EC2 via the AWS Management Console.

```
                "Effect": "Allow",
                "Action": "elasticloadbalancing:Describe*",
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "cloudwatch:ListMetrics",
                    "cloudwatch:GetMetricStatistics",
                    "cloudwatch:Describe*"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": "autoscaling:Describe*",
                "Resource": "*"
            }
        ]
}
```

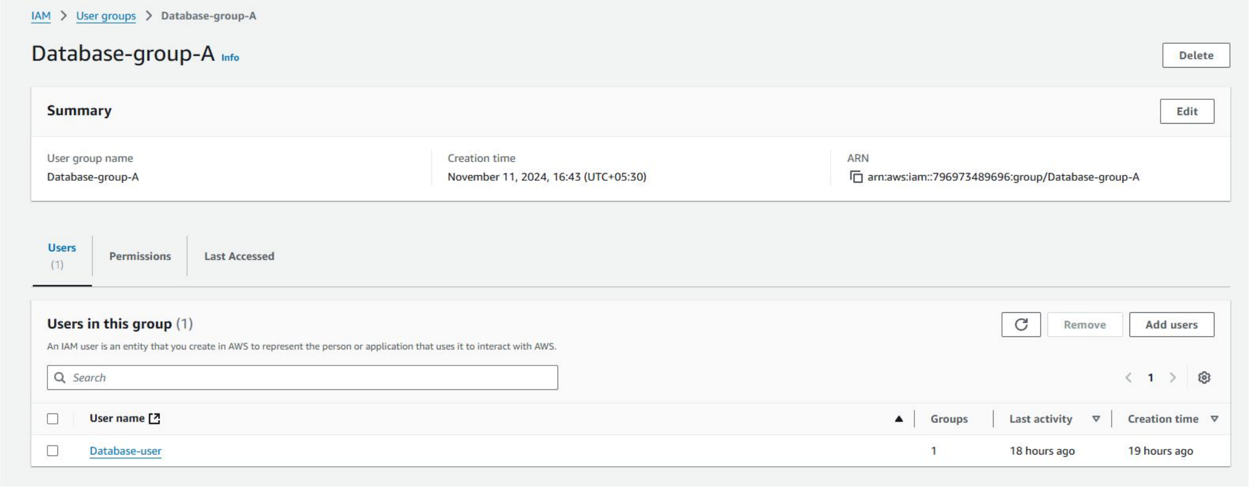
□	Backend-group-policy	Customer managed	1

Backend-group-policy

allowing backend group user to have access to the backend instance

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "arn:aws:ec2:ap-south-1:796973489696:instance/i-00ee9de2f513f4e19",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/Name": "Fundoo-Backend"
                }
            }
        }
    ]
}
```

2. Database administrator group should have access to database only he/she can only able to manage RDS



Database-group-A Info

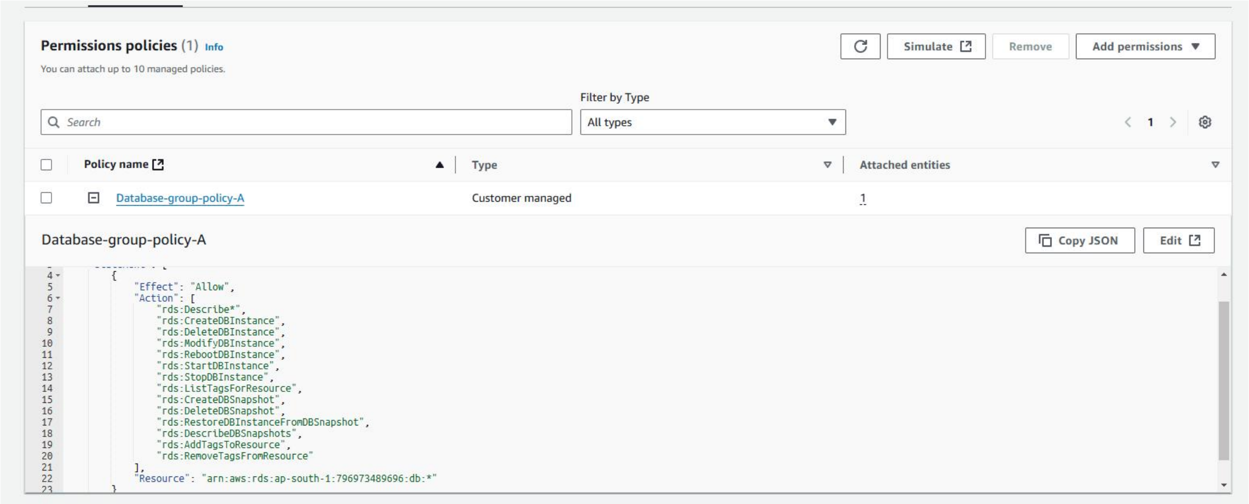
Summary

User group name	Creation time	ARN
Database-group-A	November 11, 2024, 16:43 (UTC+05:30)	arn:aws:iam::796973489696:group/Database-group-A

Users in this group (1)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

□	User name	Groups	Last activity	Creation time
□	Database-user	1	18 hours ago	19 hours ago



Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

□	Policy name	Type	Attached entities
□	⊟ Database-group-policy-A	Customer managed	1

Database-group-policy-A

```
        {
            "Effect": "Allow",
            "Action": [
                "rds:Describe*",
                "rds:CreateDBInstance",
                "rds:DeleteDBInstance",
                "rds:ModifyDBInstance",
                "rds:RebootDBInstance",
                "rds:StartDBInstance",
                "rds:StopDBInstance",
                "rds:ListTagsForResource",
                "rds:CreateDBSnapshot",
                "rds:DeleteDBSnapshot",
                "rds:RestoreDBInstanceFromDBSnapshot",
                "rds:DescribeDBSnapshots",
                "rds:AddTagsToResource",
                "rds:RemoveTagsFromResource"
            ],
            "Resource": "arn:aws:rds:ap-south-1:796973489696:db:*"
        }
```

- Policies

AWS managed policies , User managed policies Identity-based policies, Resource-based policies.

***Types**:

AWS Managed Policies: Predefined by AWS, suitable for common use cases.

User Managed Policies: Custom policies created and managed by users.

Identity-Based Policies: Attach to users, groups, or roles to control what actions they can perform.

Resource-Based Policies: Attach directly to AWS resources (e.g., S3 bucket policies) to control access.

B) Understanding and deep dive to IAM policies and their structure.

- Topics:

- JSON structure of IAM policies: Statements, Actions, Resources, Conditions.

Statements: Core components of the policy, containing:

Effect: Specifies if the statement allows or denies access (e.g., "Effect": "Allow" or "Effect": "Deny").

Action: Lists the operations permitted (e.g., "s3:ListBucket" or "elasticloadbalancing:DescribeLoadBalancers").

Resource: Specifies the AWS resources the actions apply to (e.g., "arn:aws:s3:::example-bucket").

Condition: Defines specific circumstances under which the policy grants permission (e.g., using "IpAddress" or "StringEquals" conditions).

Inline policies vs. managed policies.

- **Inline Policies**: Embedded directly within an IAM user, group, or role.
- **Managed Policies**: Standalone policies that can be attached to multiple IAM entities.
- **AWS Managed Policies**: Created and maintained by AWS.
- **Customer Managed Policies**: Custom policies created by the user.

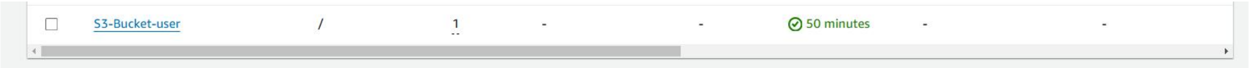
Policy evaluation logic and how IAM determines whether to allow or deny a request.

Policy Evaluation Logic

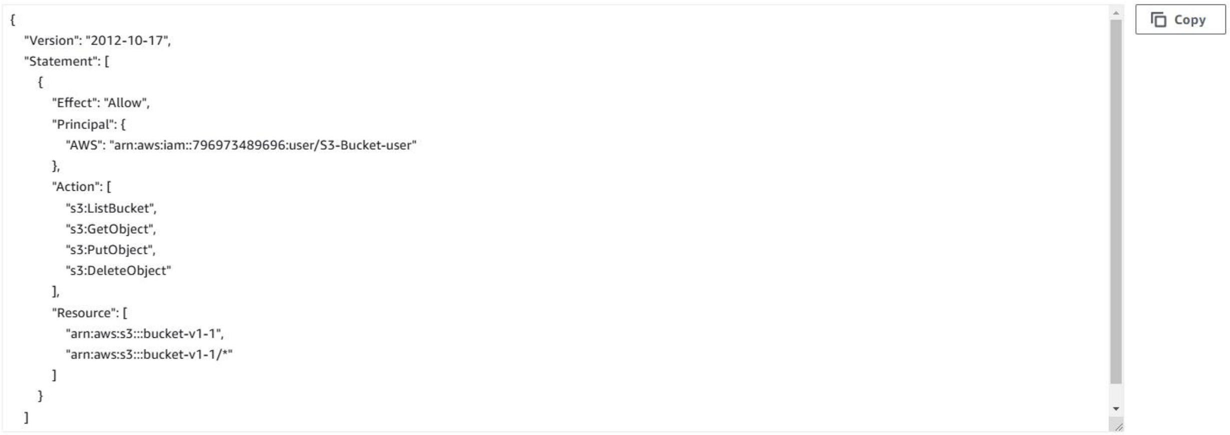
- IAM uses **policy evaluation logic** to determine if a request is allowed or denied:
    1. By default, all requests are **denied**.
    2. An **explicit allow** in a policy overrides the default deny.
    3. An **explicit deny** overrides any allows.

Resource-based policies for S3. Create an S3 bucket policy to control access to specific users or roles

- Resource-based policies are attached to resources like **S3 buckets**. They control access at the resource level and can specify users, roles, or accounts.



□	S3-Bucket-user	/	1	-	-	✓ 50 minutes	-	-



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::796973489696:user/S3-Bucket-user"
      },
      "Action": [
        "s3:ListBucket",
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-v1-1",
        "arn:aws:s3:::bucket-v1-1/*"
      ]
    }
  ]
```

- Hands-On:

Create a custom IAM policy using the JSON editor.

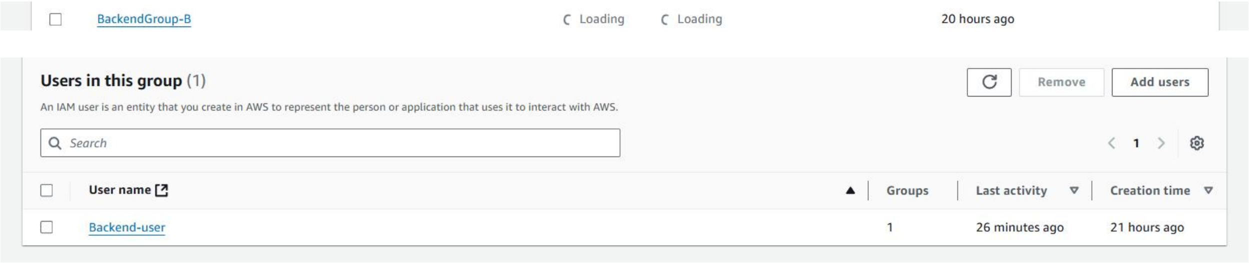
Create 2 custom policies for 2 groups backend developers and database administrator with below permission

Attach above the custom policy to a user or group.

1. for backend group policy -

   S3 bucket Read/Write access but not admin access

   Load balancer administrator access

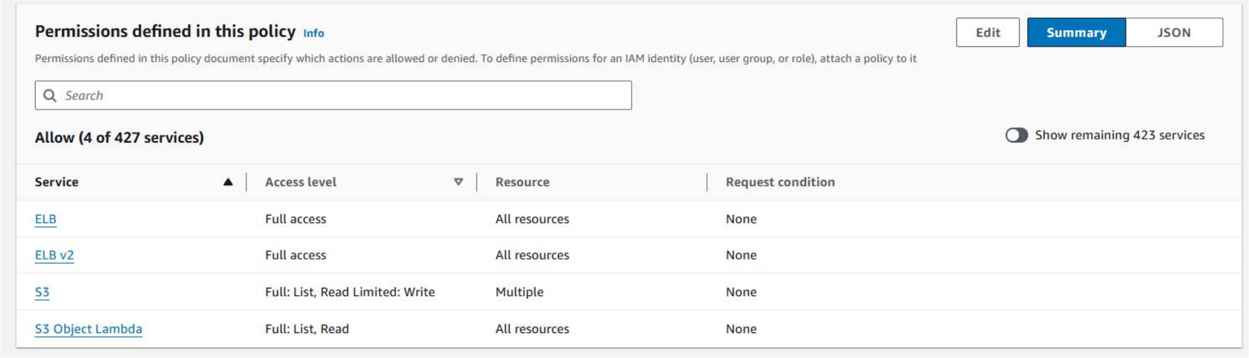


□ BackendGroup-B	Loading	Loading	20 hours ago

Users in this group (1)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

□	User name	Groups	Last activity	Creation time
□	Backend-user	1	26 minutes ago	21 hours ago



Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

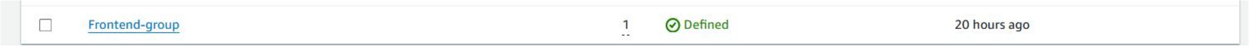
Allow (4 of 427 services)

Show remaining 423 services

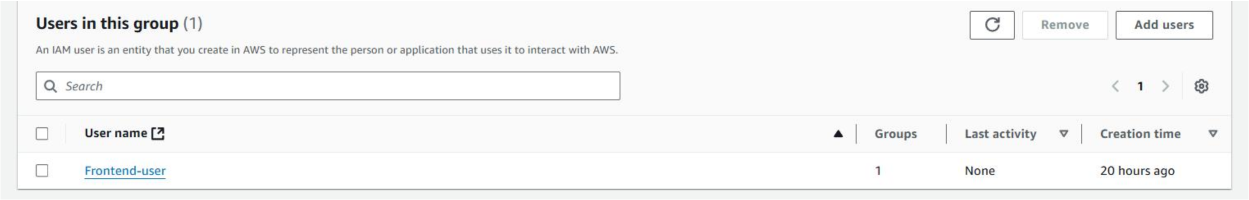
Service ▲	Access level ▽	Resource	Request condition
ELB	Full access	All resources	None
ELB v2	Full access	All resources	None
S3	Full: List, Read Limited: Write	Multiple	None
S3 Object Lambda	Full: List, Read	All resources	None

B) for frontend group policy -

EC2 access but for specific frontend server only



□	Frontend-group	1	✓ Defined	20 hours ago

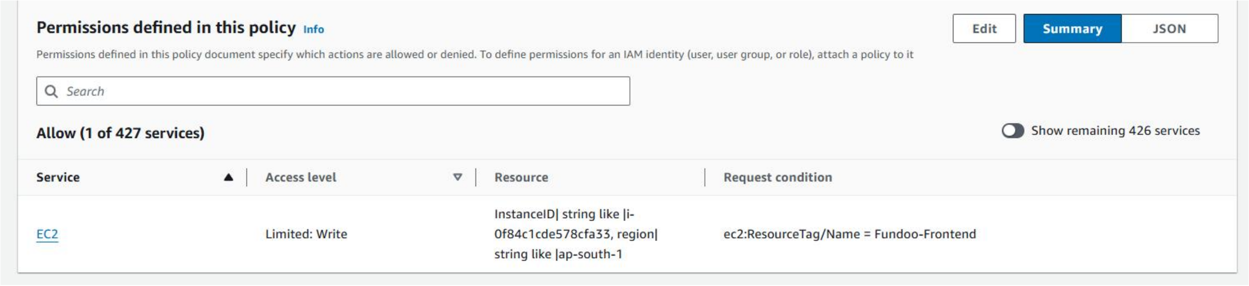


Users in this group (1)

An IAM user is an entity that you create in AWS to represent the person or application that uses it to interact with AWS.

Search

□	User name ▲	Groups	Last activity ▽	Creation time ▽
□	Frontend-user	1	None	20 hours ago



Permissions defined in this policy Info

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

Search

Allow (1 of 427 services)

Show remaining 426 services

Service ▲	Access level ▽	Resource	Request condition
EC2	Limited: Write	InstanceID| string like |i-0f84c1cde578cfa33, region| string like |ap-south-1	ec2:ResourceTag/Name = Fundoo-Frontend

C) Managing IAM Roles - Learn to create and manage IAM roles for various use cases

• Topics:

- What are IAM roles and when to use them.

IAM Roles in AWS are entities that define a set of permissions for making AWS service requests. Unlike IAM users, IAM roles are not associated with a specific person or application; instead, they are assumed by trusted entities like AWS services, users, or applications. Roles enable secure, temporary access to AWS resources.

Use cases:

Roles for EC2,

When you want an **EC2 instance** to interact with AWS services, you can assign it an **IAM role**. This allows the instance to perform actions on behalf of the user without needing to embed sensitive credentials like access keys in your code.

Use Case:

- **Example:** An EC2 instance that needs to interact with an S3 bucket to download files.
- Instead of hardcoding credentials, you assign an IAM role to the EC2 instance that grants it permissions to access S3.

How it Works:

- When you launch an EC2 instance, you can assign a role to it in the instance settings.
- The instance will automatically have temporary credentials provided by AWS (via **STS - Security Token Service**), which it can use to perform allowed actions.

Example:

If you want an EC2 instance to be able to access an S3 bucket, you would assign a role with permissions

Cross-account access

Cross-account access is a way to allow one AWS account to access resources in another AWS account. This is achieved by using IAM roles and defining a **trust relationship** between the accounts.

Use Case:

- **Example:** Account A wants an EC2 instance in Account B to access an S3 bucket in Account A.
- In this scenario, Account A creates an IAM role that grants access to the S3 bucket and allows **Account B's EC2 instance** to assume the role.

How it Works:

- **Account A** creates an IAM role with a trust relationship policy that specifies that **Account B** (or a user/service in Account B) is allowed to assume the role.
- When an EC2 instance in Account B needs to access the S3 bucket in Account A, it assumes the role defined in Account A.

Example of Cross-Account Role Trust Relationship (Account A to Account B):

Lambda execution roles.

A **Lambda execution role** is an IAM role that gives AWS Lambda functions the permissions to access other AWS services during their execution.

Use Case:

- **Example:** A Lambda function needs to read from DynamoDB, write logs to CloudWatch, or send messages to an SNS topic.
- The execution role grants the Lambda function the required permissions to perform these actions without embedding credentials in the Lambda code.

How it Works:

- When creating a Lambda function, you assign it an execution role that contains the required permissions.
- The Lambda service assumes this role whenever the function is triggered, allowing it to access the necessary AWS resources.

Example of Lambda Execution Role Permissions:

Here’s an example policy attached to a Lambda execution role that allows reading from DynamoDB and writing to CloudWatch logs:

Key Differences:

- **EC2 Role:** Grants permissions to EC2 instances to interact with other AWS resources.
- **Cross-Account Role:** Allows resources in one AWS account to access resources in another account by assuming a role.
- **Lambda Execution Role:** Grants AWS Lambda functions permissions to interact with AWS resources during function execution.

Trust Relationships in Roles:

A **trust relationship** defines which entities can assume the role. It is specified as a policy in the role that grants permission to trusted users or services to assume the role. Trust relationships are crucial for cross-account access or delegating permissions to services.

Service-linked roles and their use cases

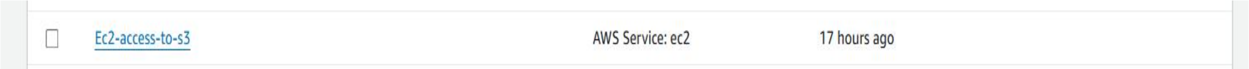
A **Service-Linked Role** is a type of role that is predefined by an AWS service to perform tasks on your behalf. These roles are linked to a specific service and allow that service to interact with AWS resources.

- **Example:** AWS Elastic Load Balancing automatically creates a service-linked role called AWS Service Role For ElasticLoadBalancing to manage resources required for load balancing.
- **Use case:** Service-linked roles are used for specific services like Amazon RDS, ECS, Lambda, etc., where AWS services need specific permissions.

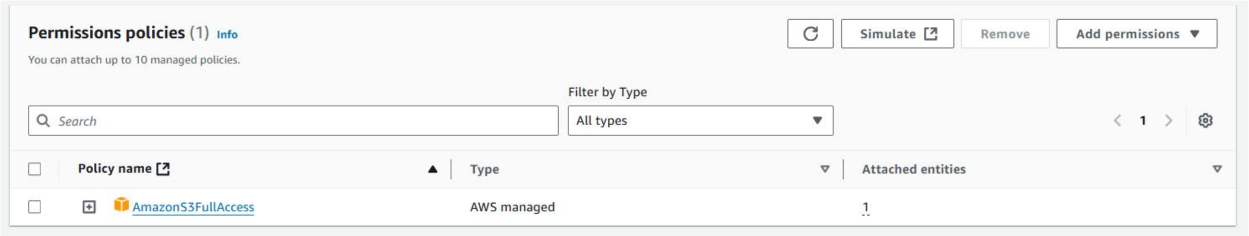
Summary:

- IAM roles provide temporary and flexible access management for AWS services.
- Trust relationships define which entities can assume roles.
- Service-linked roles are predefined roles for specific AWS services to manage resources.

- Hands-On:
  - Create a role(instance profile role) for an EC2 instance to access S3.



□	Ec2-access-to-s3	AWS Service: ec2	17 hours ago

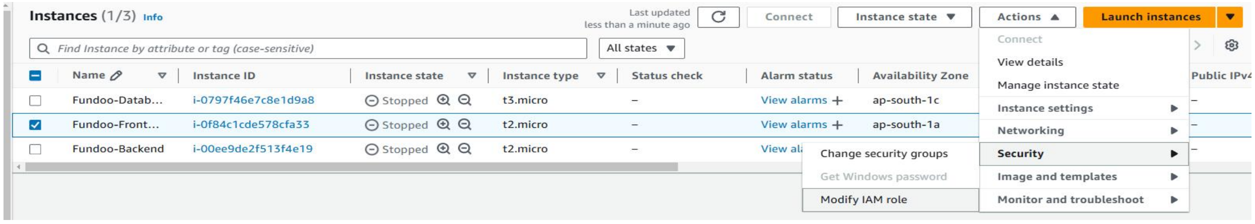


Permissions policies (1) Info

You can attach up to 10 managed policies.

Filter by Type: All types

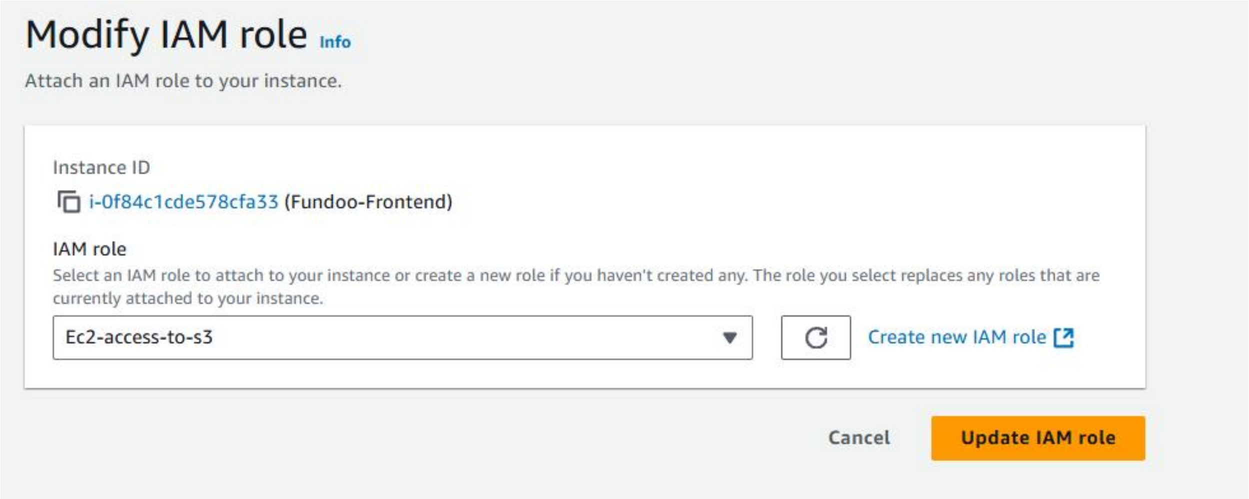
☐	Policy name	Type	Attached entities
☐	AmazonS3FullAccess	AWS managed	1



Instances (1/3) Info

Last updated less than a minute ago

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
☐	Fundoo-Datab...	i-0797f46e7c8e1d9a8	Stopped	t3.micro	–	View alarms +	ap-south-1c	–
☑	Fundoo-Front...	i-0f84c1cde578cfa33	Stopped	t2.micro	–	View alarms +	ap-south-1a	–
☐	Fundoo-Backend	i-00ee9de2f513f4e19	Stopped	t2.micro	–	View al		–



Modify IAM role Info

Attach an IAM role to your instance.

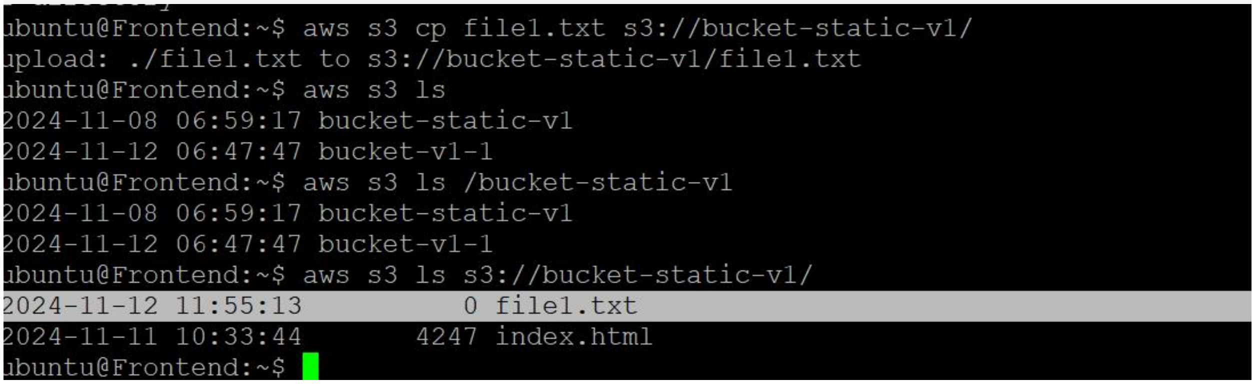
Instance ID

i-0f84c1cde578cfa33 (Fundoo-Frontend)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

Ec2-access-to-s3 ▼

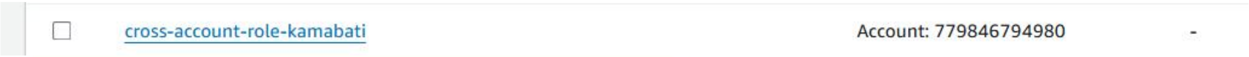


```
ubuntu@Frontend:~$ aws s3 cp file1.txt s3://bucket-static-v1/
upload: ./file1.txt to s3://bucket-static-v1/file1.txt
ubuntu@Frontend:~$ aws s3 ls
2024-11-08 06:59:17 bucket-static-v1
2024-11-12 06:47:47 bucket-v1-1
ubuntu@Frontend:~$ aws s3 ls /bucket-static-v1
2024-11-08 06:59:17 bucket-static-v1
2024-11-12 06:47:47 bucket-v1-1
ubuntu@Frontend:~$ aws s3 ls s3://bucket-static-v1/
2024-11-12 11:55:13          0 file1.txt
2024-11-11 10:33:44       4247 index.html
ubuntu@Frontend:~$
```

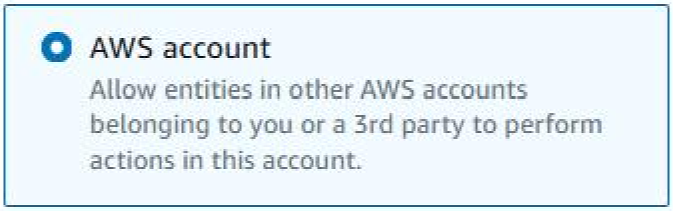
- Create a role for cross-account access and understand trust policy configuration.

Setup cross account access to connect resources on another aws account

ON Trusted account:-

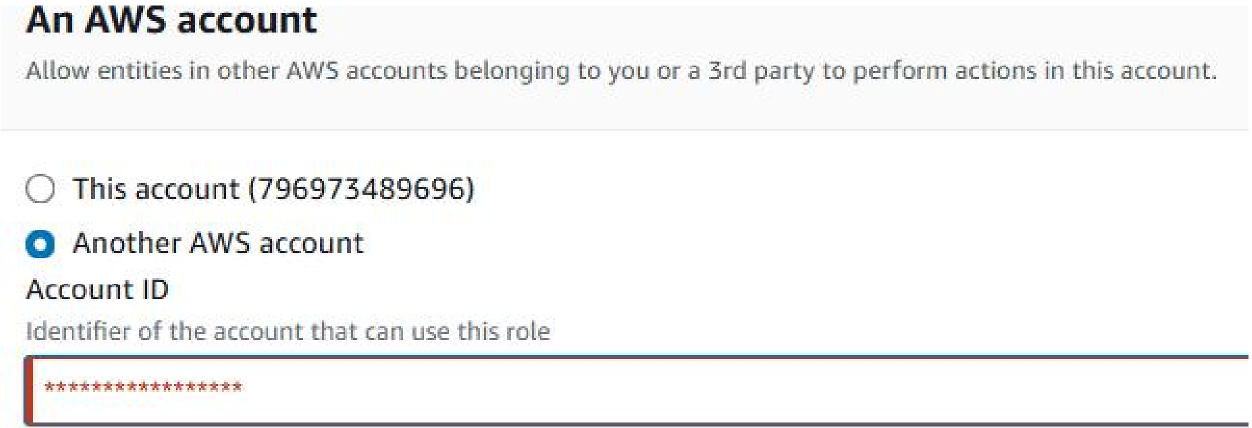


□ cross-account-role-kamabati Account: 779846794980 -



● AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.



An AWS account

Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

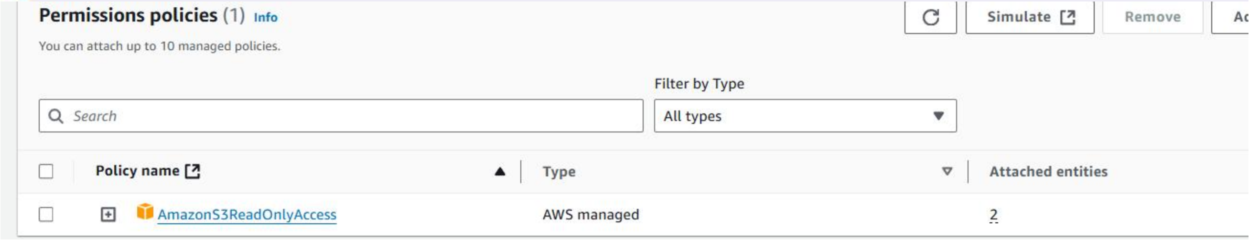
○ This account (796973489696)

● Another AWS account

Account ID

Identifier of the account that can use this role

****

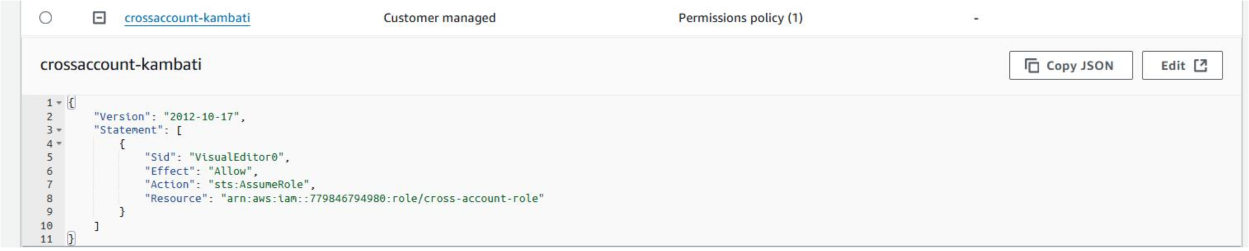


Permissions policies (1) Info

You can attach up to 10 managed policies.

□	Policy name	Type	Attached entities
□	AmazonS3ReadOnlyAccess	AWS managed	2

On cross account:-



crossaccount-kambati

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::779846794980:role/cross-account-role"
        }
    ]
}
```

crossaccount-kambati

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::779846794980:role/cross-account-role"
        }
    ]
}
```

Note:- Attach the policy to a user and login with the credentials and switch the role

With (alias or trusted acc) with acc id and role path

D) Understanding IAM Best Practices

- Topics:

  - Principle of least privilege.

Definition: Grant each user, group, or role only the permissions they need to perform their work and nothing more.

Practice: Avoid using AWS root user credentials for everyday tasks, and use IAM policies to restrict permissions.

- Enabling Multi-Factor Authentication (MFA).

Definition: Adding an extra layer of security by requiring not just a password but also a second form of authentication (like a code from a mobile app).

Why: It helps protect your AWS accounts from unauthorized access.

How: You can enable MFA on the AWS root user account and on IAM user accounts.

- Monitoring IAM activities using AWS CloudTrail.

• Hands-On:

• Enable MFA for the root user and an IAM user.

• Review CloudTrail logs for IAM activity.