CSCI 5902 Adv. Cloud Architecting
Fall 2023
Instructor: Dr. Lu Yang

Module 7 Connecting Networks (Section 6) &
Module 8 Securing User and Application Access (Sections 1-2)
Oct 30, 2023

Housekeeping items and feedback

PIER tour

1. Start recording

- Questions

Is route table similar to/the same as a load balancer?

The main purpose of a routing table is to help routers make effective routing decisions. Whenever a packet is sent through a router to be forwarded to a host on another network, the router consults the routing table to find the IP address of the destination device and the best path to reach it. The packet is then directed to a neighboring router -- or the next hop listed in the table -- until it reaches its final destination.

Recap of the last lecture

Module overview

Sections

1. Architectural need
2. Connecting to your remote network with AWS Site-to-Site VPN On-prem to VPC
3. Connecting to your remote network with AWS Direct Connect On-prem to VPC
4. Connecting VPCs in AWS with VPC peering VPC ⟺ VPC
5. Scaling your VPC network with AWS Transit Gateway VPC. VPN. DC Private
6. Connecting your VPC to supported AWS services

VPC → Service

Module 7: Connecting Networks

Section 6: Connecting your VPC to supported AWS services

VPC endpoints

- Enable you to privately connect your VPC to supported AWS services and to VPC endpoint services that are powered by AWS PrivateLink
- Enable traffic between your VPC and the other service without leaving the Amazon network
- Do not require an internet gateway, VPN, network address translation (NAT) devices, or firewall proxies
- Are horizontally scaled, redundant, and highly available

Two types of VPC endpoints

aws academy

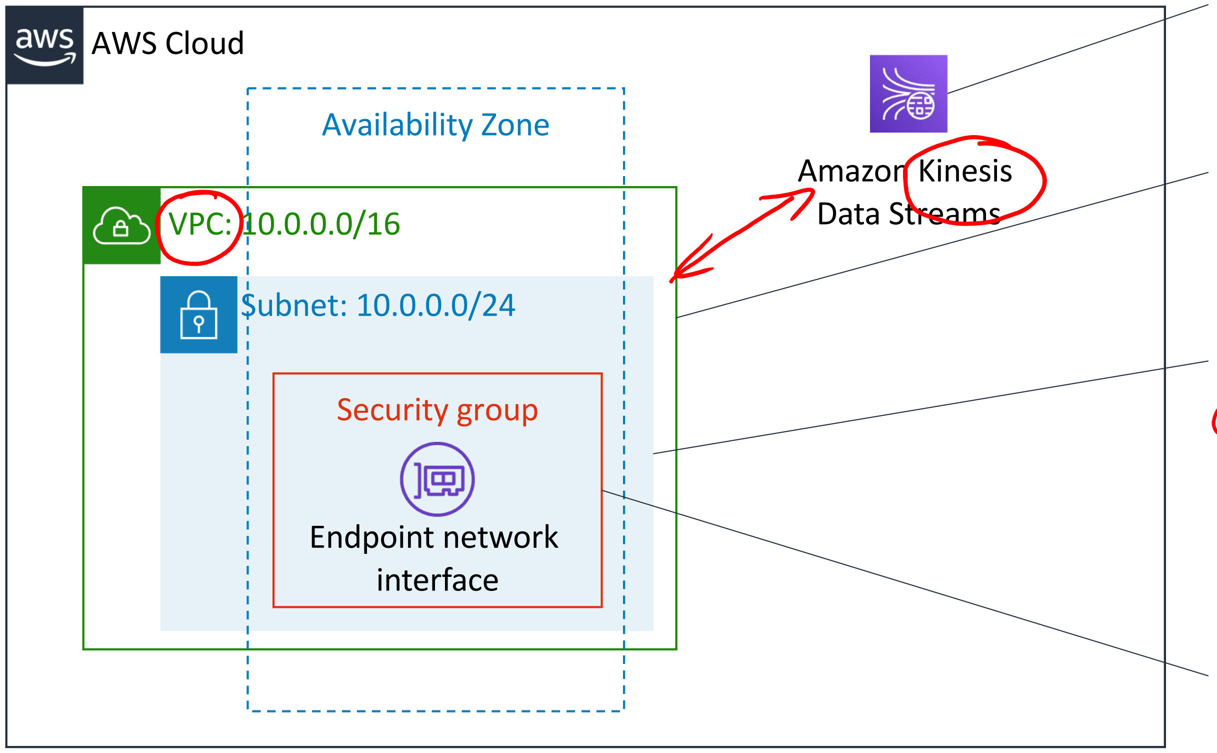
- Interface endpoint – An elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported service (all except DynamoDB)
- Powered by AWS PrivateLink
- Examples –
  - Amazon CloudWatch
  - Amazon EC2 API
  - Elastic Load Balancing
- Support cross-region access.

- Gateway endpoint – A gateway that you specify as a target for a route in your route table for traffic destined to a supported AWS service
- Supported AWS services –
  - Amazon S3
  - Amazon DynamoDB
- Doesn't support cross-region access.

Cheaper

Reference (optional):
https://aws.amazon.com/premiumsupport/knowledge-center/vpc-endpoints-cross-region-aws-services/

How to set up an interface endpoint



1. Specify the AWS service, endpoint service.

2. Choose the VPC where you want to create the interface endpoint.

3. Choose a subnet in your VPC that will use the interface endpoint. You can specify more than one subnet in different Availability Zones (as supported by the service).

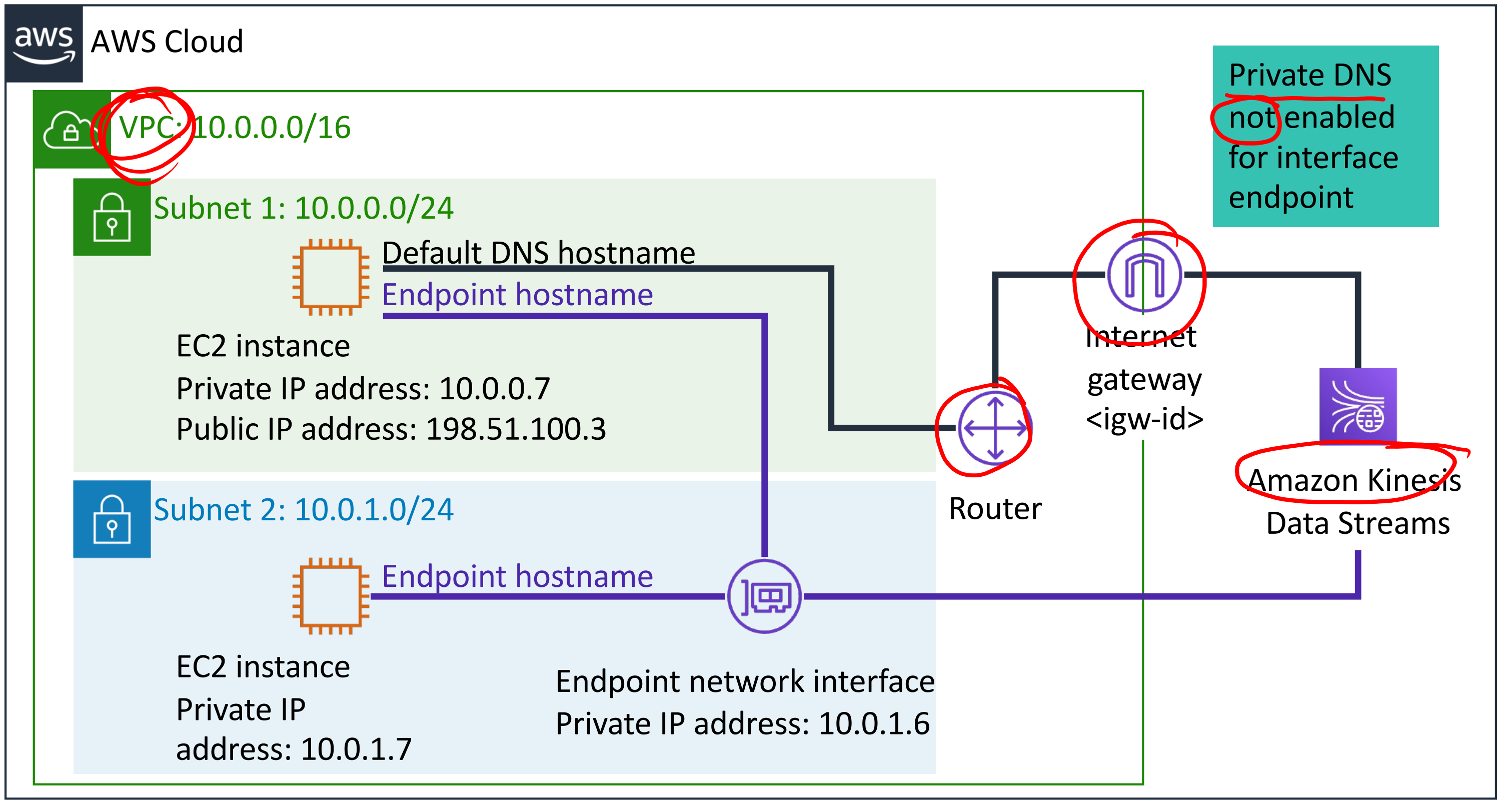
4. (Optional) Enable private Domain Name System (DNS) for the endpoint.

5. Specify the security groups to associate with the network interface.

Demonstration: How to create an AWS Interface VPC Endpoint

https://www.youtube.com/watch?v=zlFmrT2Do74 (00:50-16:25)

Example of using VPC endpoints (1 of 2)



Subnet 1 route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-id

Internet

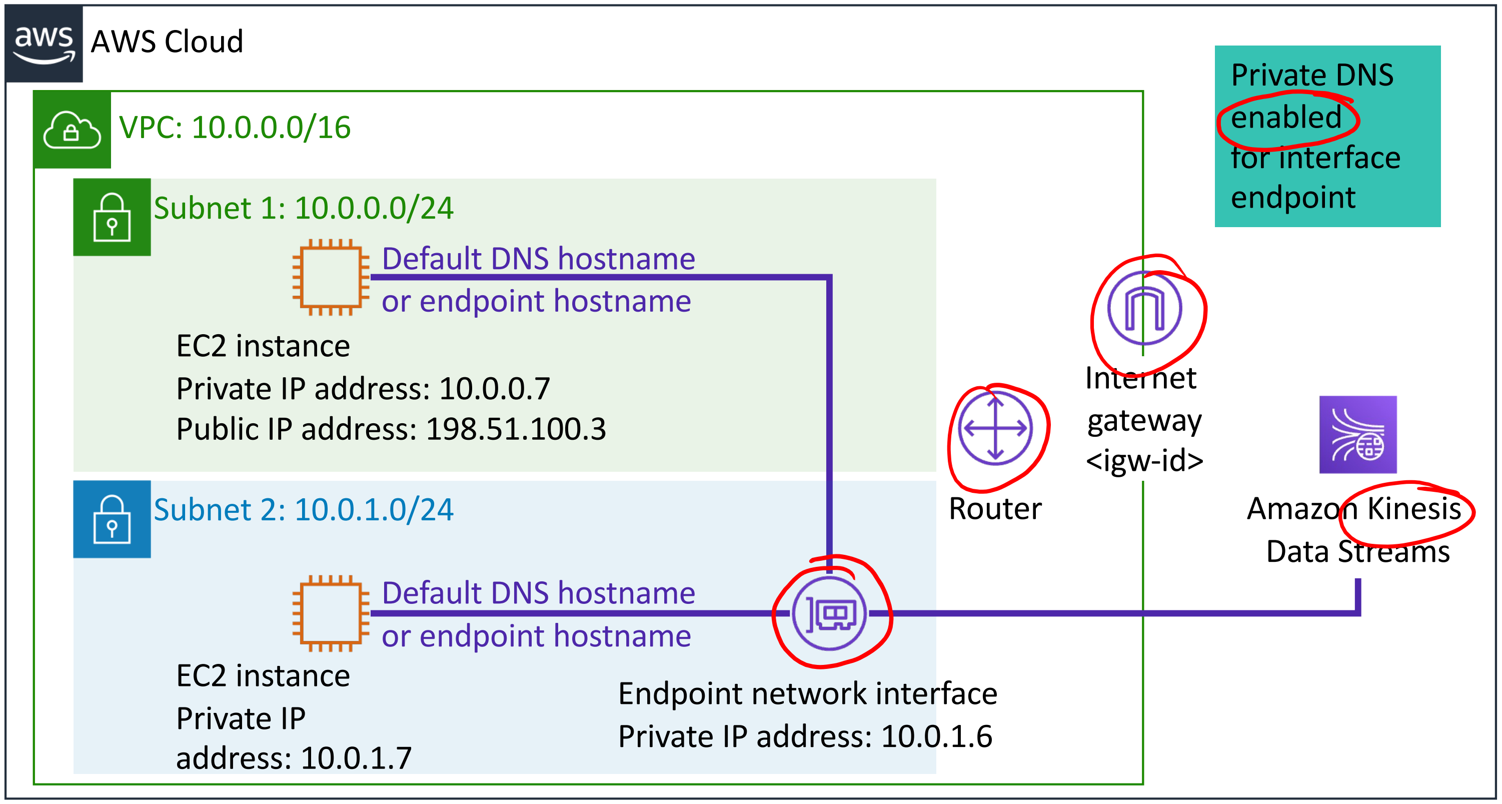
Subnet 2 route table

Destination	Target
10.0.0.0/16	local

Default DNS hostname: kinesis.us-east-1.amazonaws.com

Endpoint-specific DNS hostname: vpce-123-ab-kinesis.us-east-1.vpce.amazonaws.com

Example of using VPC endpoints (2 of 2)



Subnet 1 route table

Destination	Target
10.0.0.0/16	local
0.0.0.0/0	igw-id

Internet

Subnet 2 route table

Destination	Target
10.0.0.0/16	local

Default DNS hostname: kinesis.us-east-1.amazonaws.com

Endpoint-specific DNS hostname: vpce-123-ab-kinesis.us-east-1.vpce.amazonaws.com

Demonstration: How to create an AWS Gateway VPC Endpoint

https://www.youtube.com/watch?v=SMpK5SS7bLg

Section 6 key takeaways

- A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink
- VPC endpoints do not require an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection
- There are two types of VPC endpoints: interface endpoints and gateway endpoints

Module 7: Connecting Networks

Module wrap-up

Module summary

In summary, in this module, you learned how to:

- Describe how to connect an on-premises network to the AWS Cloud
- Describe how to connect VPCs in the AWS Cloud
- Connect VPCs in the AWS Cloud by using VPC peering
- Describe how to scale VPCs in the AWS Cloud
- Describe how to connect VPCs to supported AWS services

AWS Academy Cloud Architecting

Module 8: Securing User and Application Access

aws academy

Module overview

Sections

1. Architectural need
2. Account users and IAM
3. Organizing users
4. Federating users
5. Multiple accounts

Module objectives

At the end of this module, you should be able to:

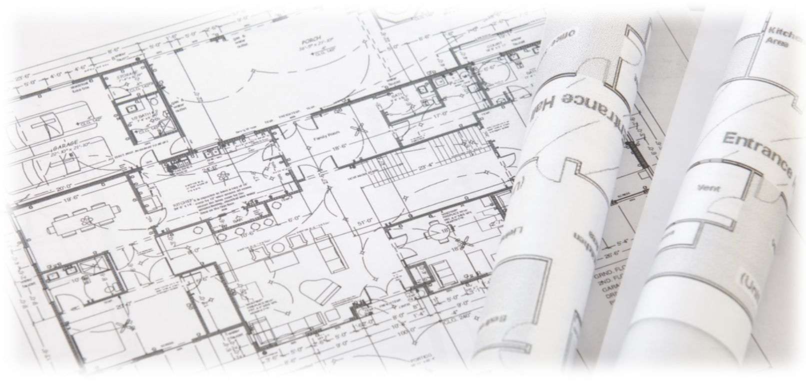
- Explain the purpose of AWS Identity and Access Management (IAM) users, groups, and roles
- Describe how to allow user federation within an architecture to increase security
- Recognize how AWS Organizations service control policies (SCPs) increase security within an architecture
- Describe how to manage multiple AWS accounts
- Configure IAM users

Module 8: Securing User and Application Access

Section 1: Architectural need

Café business requirement

The café needs to define what level of access users and systems should have across cloud resources and then put these access controls into place across the AWS account.



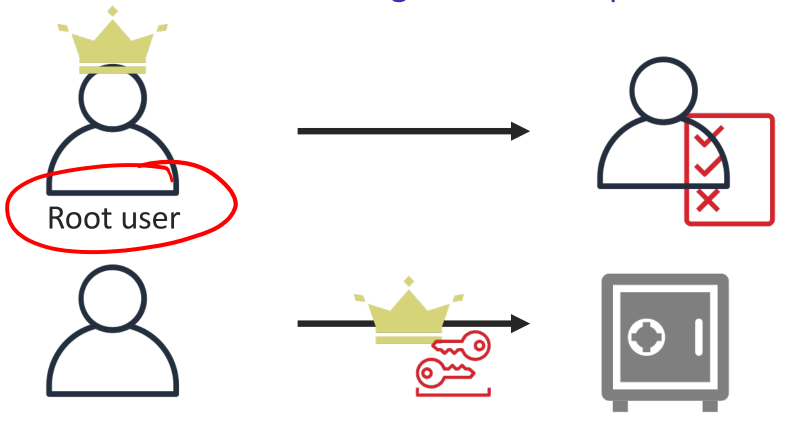
Module 8: Securing User and Application Access

Section 2: Account users and IAM

Secure the root account

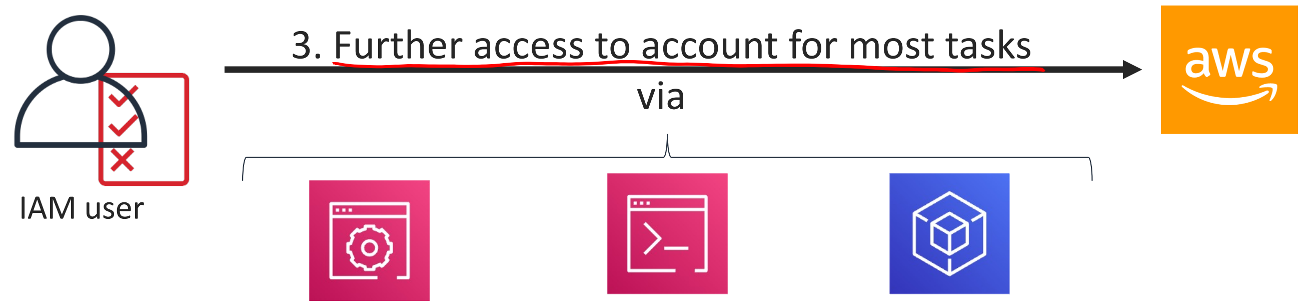
The account root user has a large amount of power. Recommended security steps:

Group



1. Create an IAM admin user

2. Lock away the root user credentials



AWS Management Console

AWS Command Line Interface (AWS CLI)

AWS Tools and software development kits (SDKs)

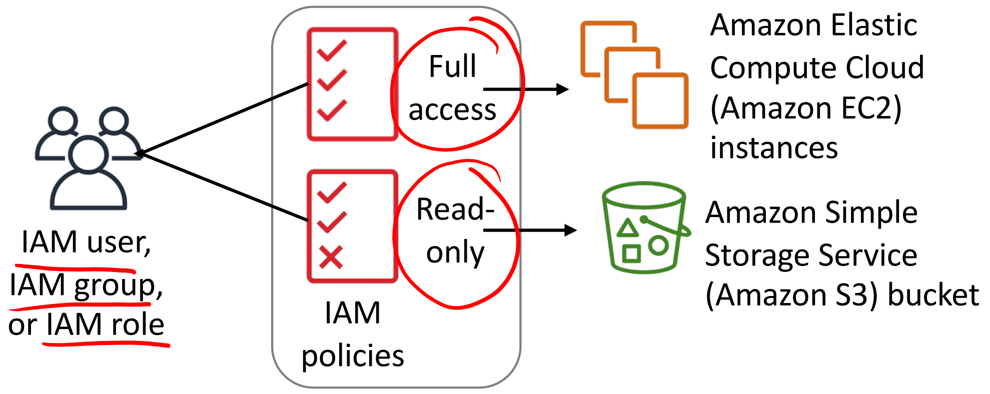
AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM)

- Securely control individual and group access to your AWS resources
- Integrates with other AWS services
- Federated identity management
- Granular permissions
- Support for multi-factor authentication

IAM components: Review

IAM user	Defined in your AWS account. Use credentials to authenticate programmatically or via the AWS Management Console.
IAM group	A collection of IAM users that are granted identical authorization.
IAM policy	Defines which resources can be accessed and the level of access to each resource.
IAM role	Mechanism to grant temporary access for making AWS service requests. *Assumable* by a user, application, or service.


IAM user, IAM group, or IAM role
Full access
Read-only
IAM policies
Amazon Elastic Compute Cloud (Amazon EC2) instances
Amazon Simple Storage Service (Amazon S3) bucket

IAM permissions

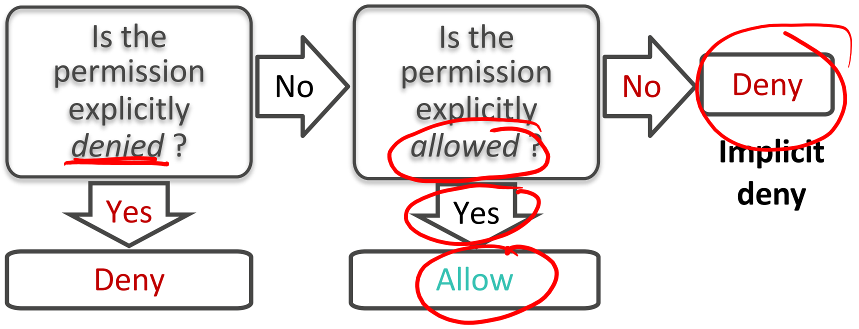
aws academy

IAM policy

Permissions are specified in an IAM policy:

- A document formatted in JavaScript Object Notation (JSON)
- It defines which resources and operations are allowed
- Best practice – follow the principle of least privilege
- Two types of policies –
  - Identity-based: Attach to an IAM principal (IAM users, IAM groups, IAM roles, or AWS services)
  - Resource-based: Attach to an AWS resource

How IAM determines permissions at the time of request:


Is the permission explicitly *denied* ?
No
Yes
Deny
Is the permission explicitly *allowed* ?
No
Yes
Allow
Deny
Implicit deny

Identity-based versus resource-based policies

Identity-based policies

- Attached to a user, group, role, or service
- Types of policies
  - AWS managed
  - Customer managed
  - Inline

Resource-based policies

- Attached to AWS resources
  - Example: Attach to an Amazon S3 bucket
- No managed resource-based policies
- Always an inline policy

IAM policy document structure (1/2)

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Effect": "effect",
      "Action": "action",
      "Resource": "arn",
      "Condition":{
         "condition":{
            "key": "value"
         }
      }
   }]
}
```

- Effect: Effect can be either *Allow* or *Deny*
- Action: Type of access that is allowed or denied

```
"Action": "s3:GetObject"
```

- Resource: Resources that the action will act on

```
"Resource": "arn:aws:sqs:us-west-2:123456789012:queue1"
```

- **Condition**: Conditions that must be met for the rule to apply

```
"Condition" : {
    "StringEquals" : {
            "aws:username" : "johndoe"
    }
}
```

IAM policy document structure (2/2)

- IAM JSON policy elements: Version
- IAM JSON policy elements: Id
- IAM JSON policy elements: Statement
- IAM JSON policy elements: Sid
- IAM JSON policy elements: Effect
- AWS JSON policy elements: Principal
- AWS JSON policy elements: NotPrincipal
- IAM JSON policy elements: Action
- IAM JSON policy elements: NotAction
- IAM JSON policy elements: Resource
- IAM JSON policy elements: NotResource
- IAM JSON policy elements: Condition
- Variables and tags
- Supported data type

Reference: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements.html

ARNs and wildcards

- Resources are identified by using Amazon Resource Name (ARN) format
  - Syntax – `arn:`*`partition`*`:`*`service`*`:`*`region`*`:`*`account`*`:`*`resource`*
  - Example – `"Resource": "arn:aws:iam::123456789012:user/mmajor"`

- You can use a wildcard (*) to give access to all actions for a specific AWS service
  - Examples –
    - `"Action": "s3:*"`
    - `"Action": "iam:*AccessKey*"`

IAM policy example

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Action":["DynamoDB:*","s3:*"],
    "Resource":[
      "arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
      "arn:aws:s3:::bucket-name",
      ~~"arn:aws:s3:::bucket-name/~~*"]
    },
    {
    "Effect":"Deny",
    "Action":["dynamodb:*","s3:*"],
    "NotResource":["arn:aws:dynamodb:region:account-number-without-hyphens:table/table-name",
      "arn:aws:s3:::bucket-name",
      "arn:aws:s3:::bucket-name/*"]
    }
  ]
}
```

Explicit allow gives users access to a specific DynamoDB table and…

…Amazon S3 buckets.

Explicit deny ensures that the users cannot use any other AWS actions or resources other than that table and those buckets.

An explicit deny statement **takes precedence** over an allow statement.

Thank you, and Kahoot!

aws academy