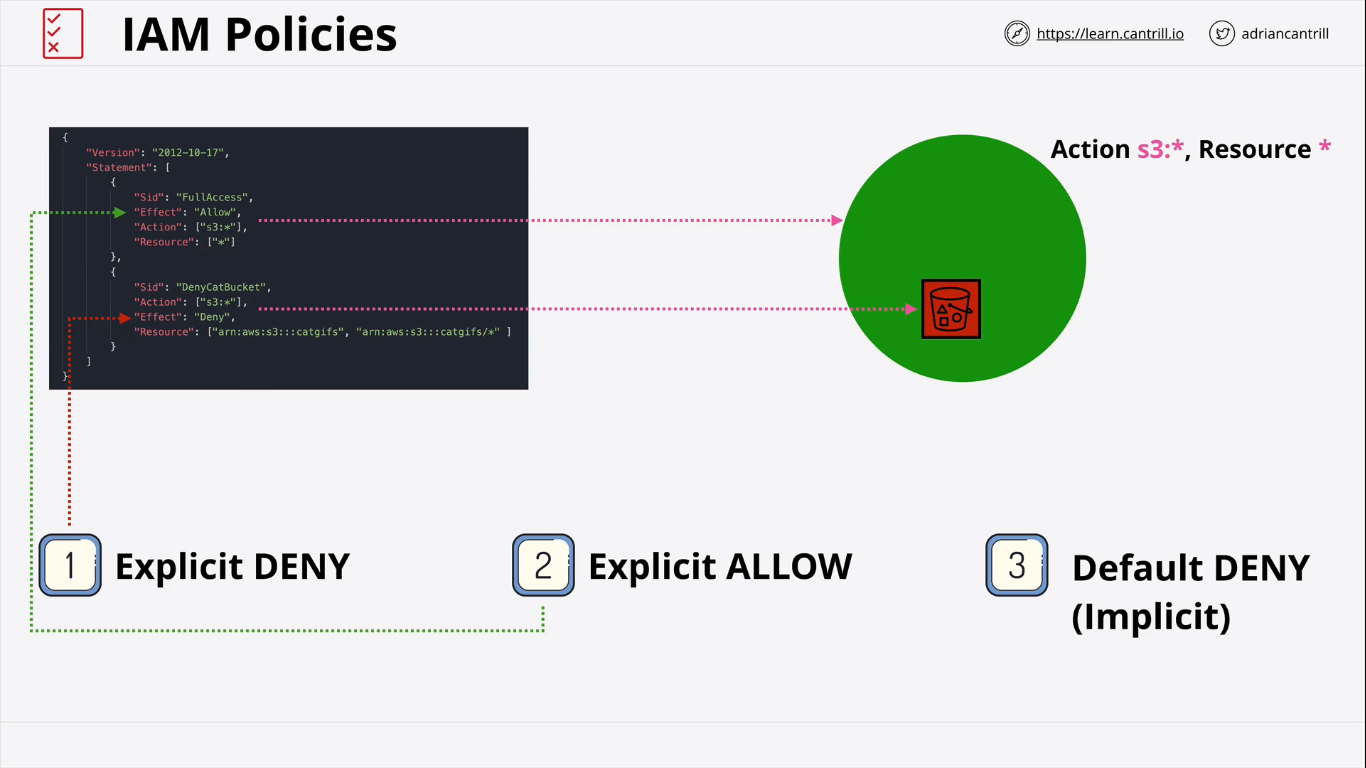
# IAM

## Policies



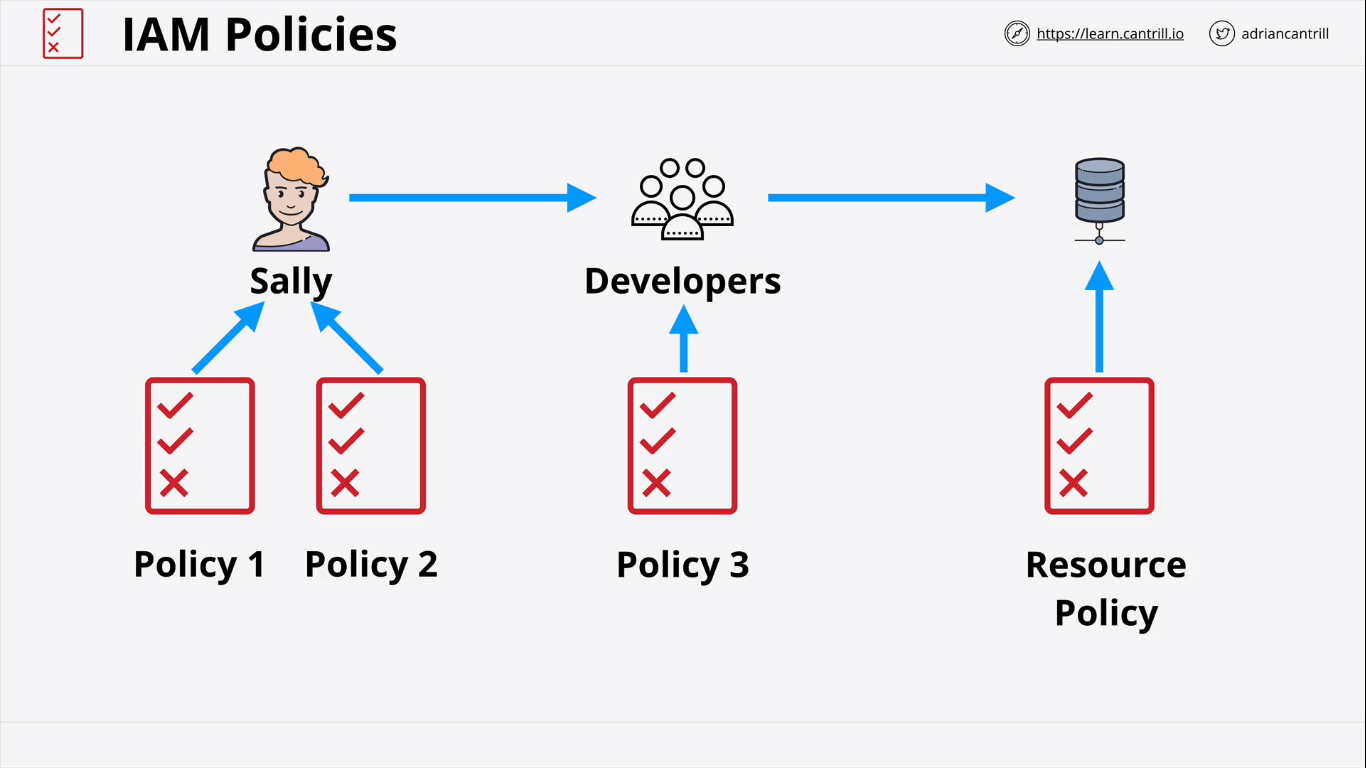


Consider the above diagrams

* It has one allow and one deny statement (explicitly)
* If a policy document contains
* Explicit deny – then it always takes first priority
* Explicit allow – takes second priority
* Default deny – third (If we don’t have allow or deny mentioned).

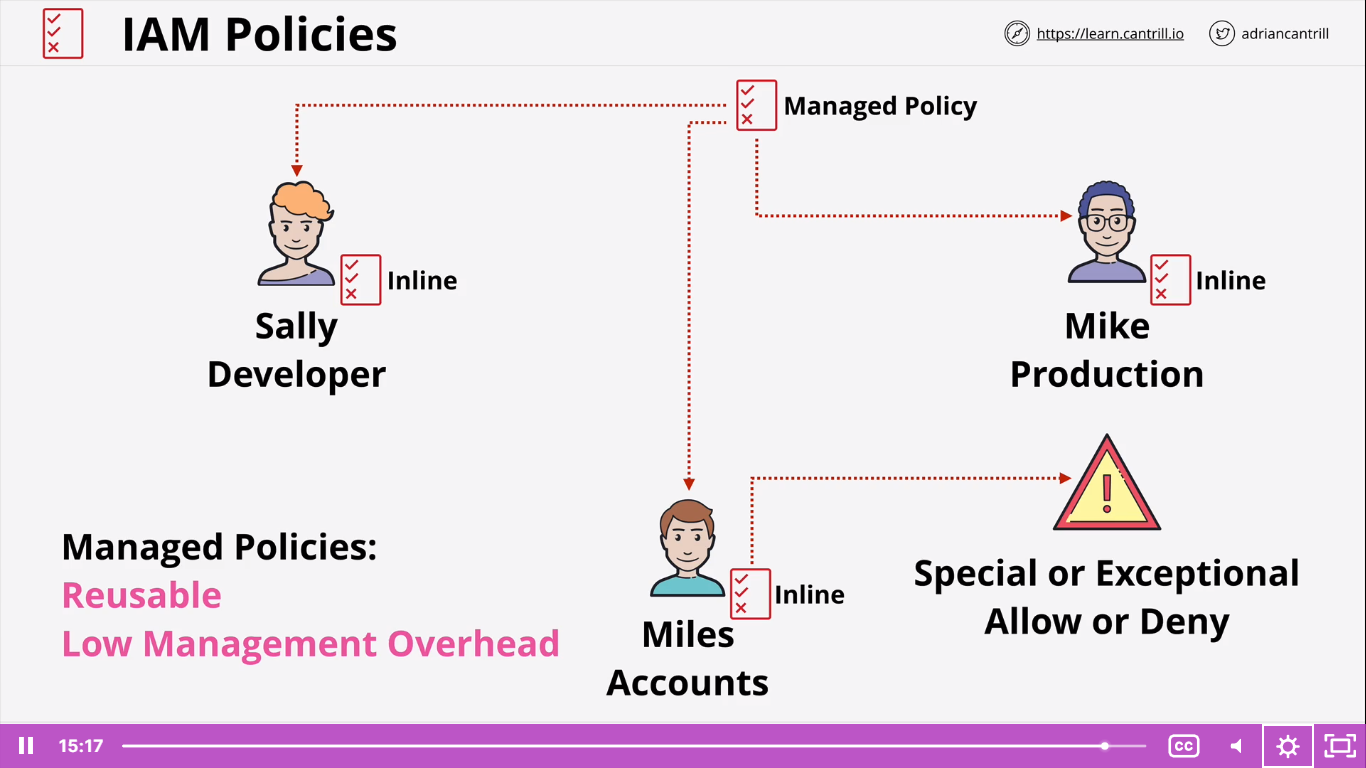
### How does it work if we have multiple policy documents associated as mentioned below?

* Always same rule (explicit deny ; explicit allow ; default deny)

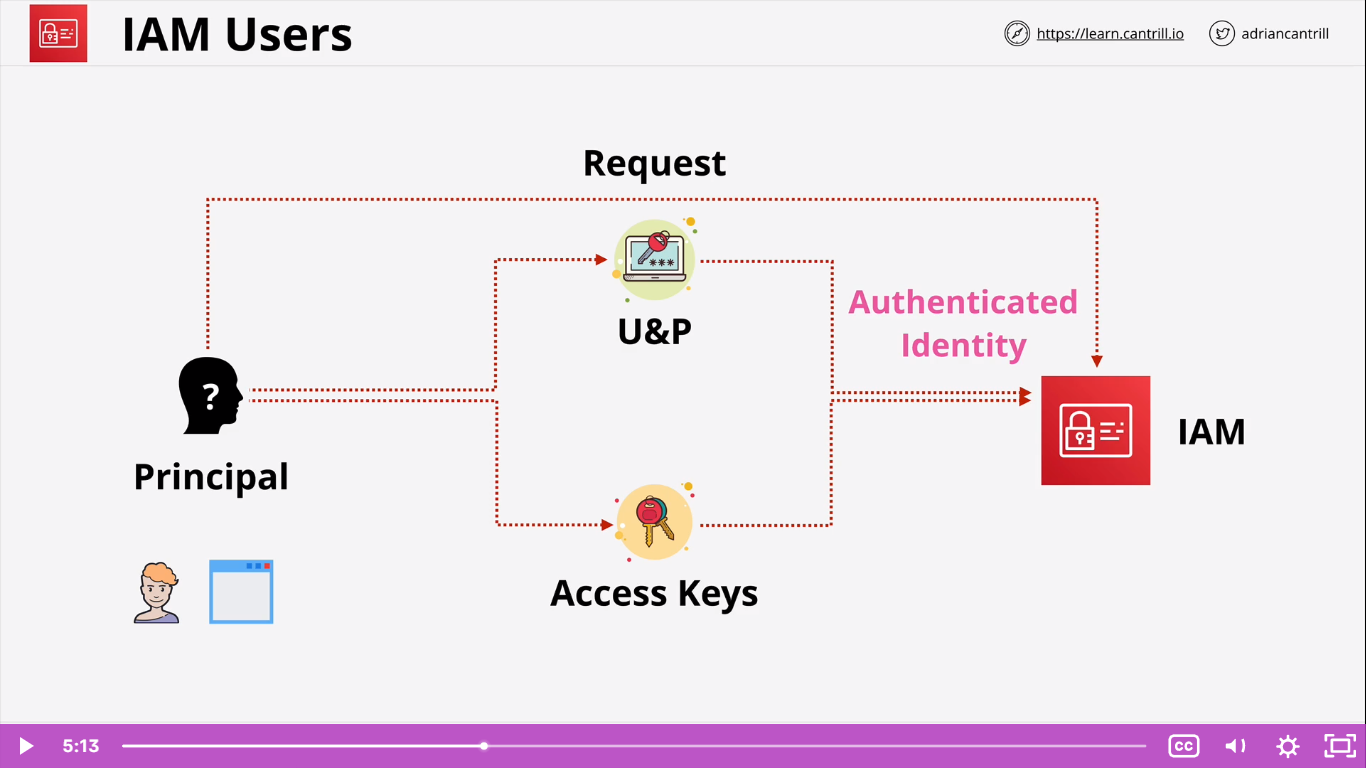


### Use of managed policy

* Instead of creating and assigning policies for individual (Inline policies), create managed policies and assign it to users
* Use inline policy when we need to explicitly deny permission



## Users

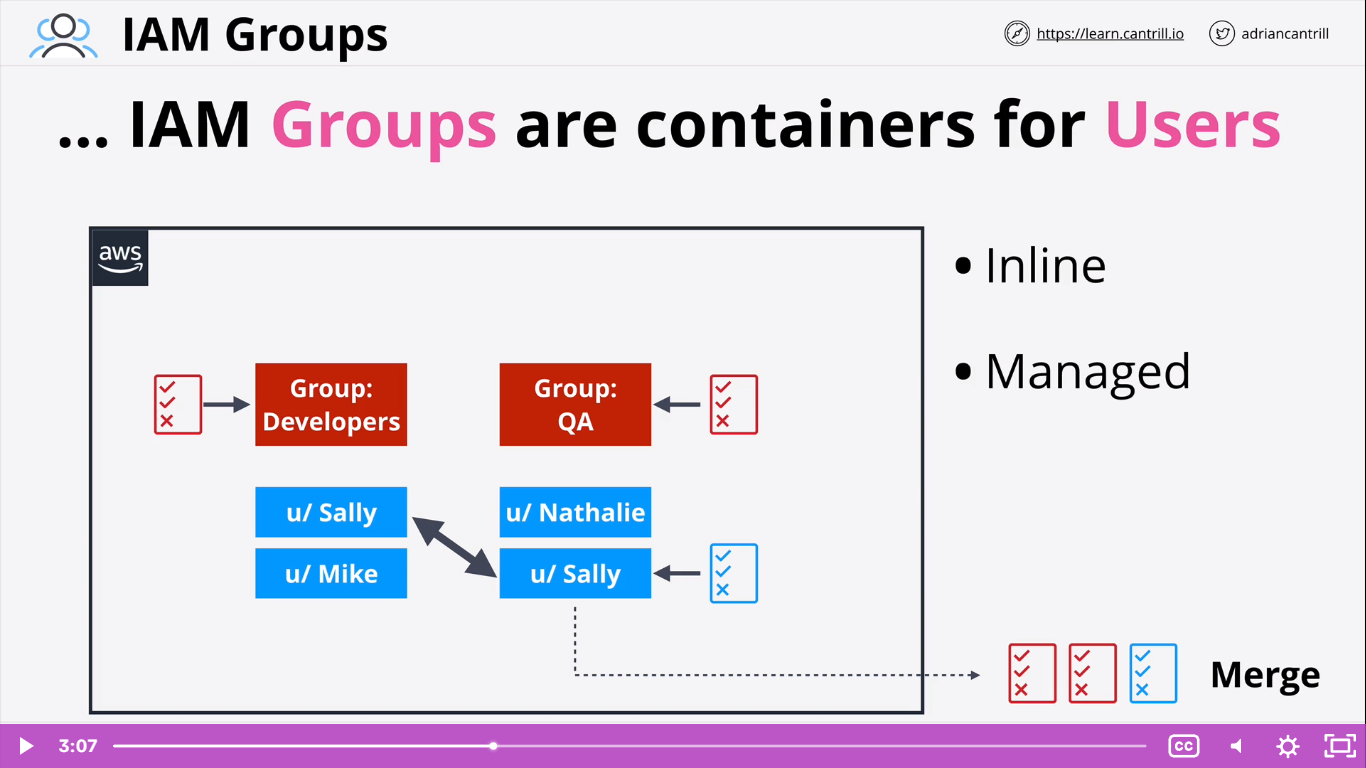


User:

* Any person / application who needs to login into AWS account
* Must have username/ password to login through console
* Must have secret key to login through programmatic way

## Group

* Policies (Inline/ Managed) can be attached to Groups



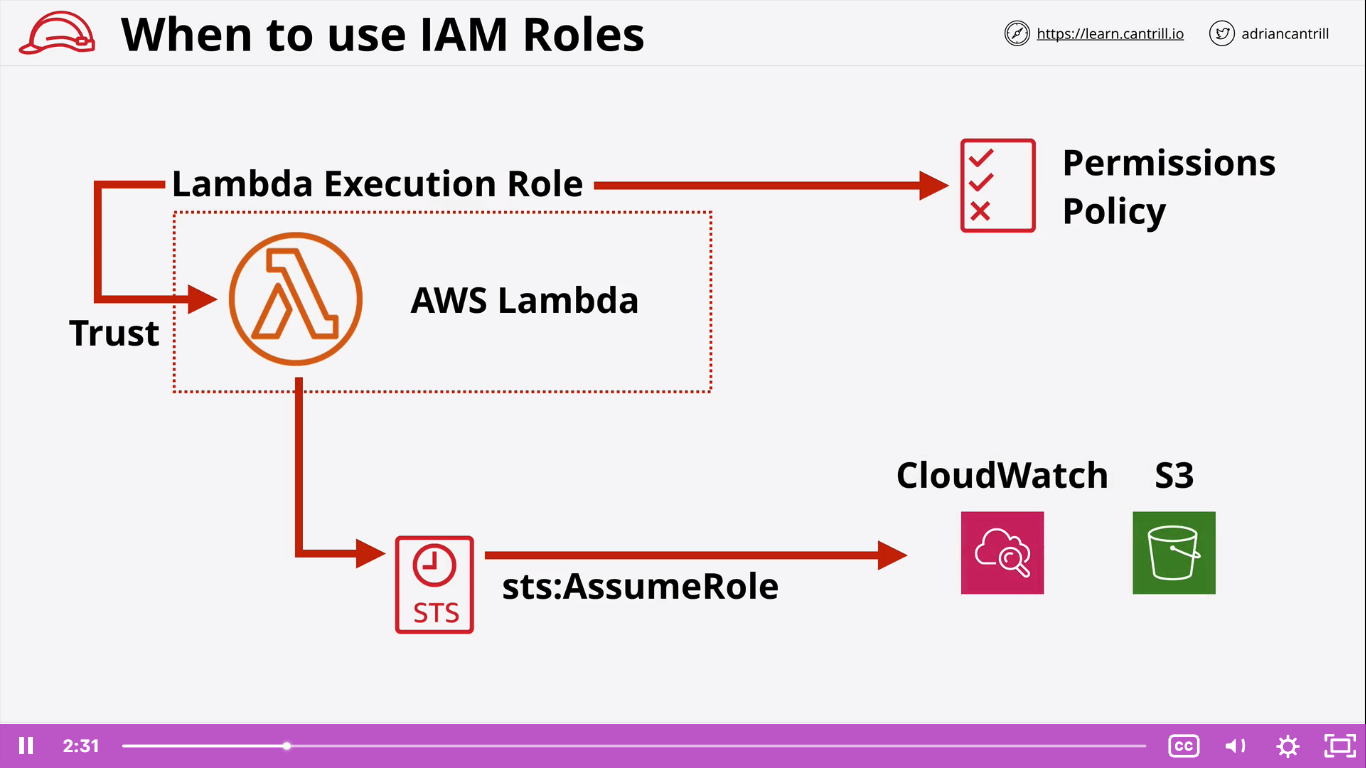
### Hard limits

* 5000 users
* Am IAM user can be a member of 10 groups
* Cannot have group within another group
* Groups cannot be a true identity (unlike users & Resources)

# Roles

* One type of identity to AWS Account
* Anything that is considered as a single entity – create IAM user
* When the # of users. Applications are not quantifiable / multiple principals
* Used for short period of time
* Example:
  + Consider an application which assumes a role in your account, then for the short period of time, it has all the provisions that the user has in this account
* Sts:AssumeRole service provides the temporary credential for the roles

When should we use role:



Consider the above scenario:

* You have logged in to AWS account and create Lambda function
* AWS Lambda service which needs to use cloud watch and S3.
* As AWS Lambda is not an (AWS Identity) user/application/principal to access , We need to create a role to assume the permission
* Trust policy which should trust Lambda to assume the role
* Permissions policy should allow AWS Lambda to access the other services
  + - It is success ‘AWS:sts:AssumeRole’ operation will generate temporary credentials to access the AWS services based on permission policy

Scenario 2:

