NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Permissions & Policies

Cloud Infrastructure Engineering

Nanyang Technological University
& Skills Union - 2022/2023

Course Content

- Quick Check-In
- Dive into the basics of Permissions & Policies
- Explore the types of Permissions & Policies
- Explore creating Permissions & Policies on AWS

Self Study Check-In

What will be the default permissions attached to a newly created IAM user?
a. Read only permissions
b. No permissions at all
c. Permissions sufficient to login to view their IAM user information
d. Permissions sufficient to login to view their IAM user information and generate access keys for cli

Q1) Which of the following actions can be performed by an IAM user with the "ReadOnlyAccess" policy attached?

a. Create an S3 bucket
b. Modify an EC2 instance
c. Create an IAM user
d. View an S3 bucket's contents

Which of the following is a recommended best practice for securing IAM user credentials?

a. Store access keys in plaintext on local machines

b. Rotate access keys regularly

c. Use the same access key and secret key for all IAM users

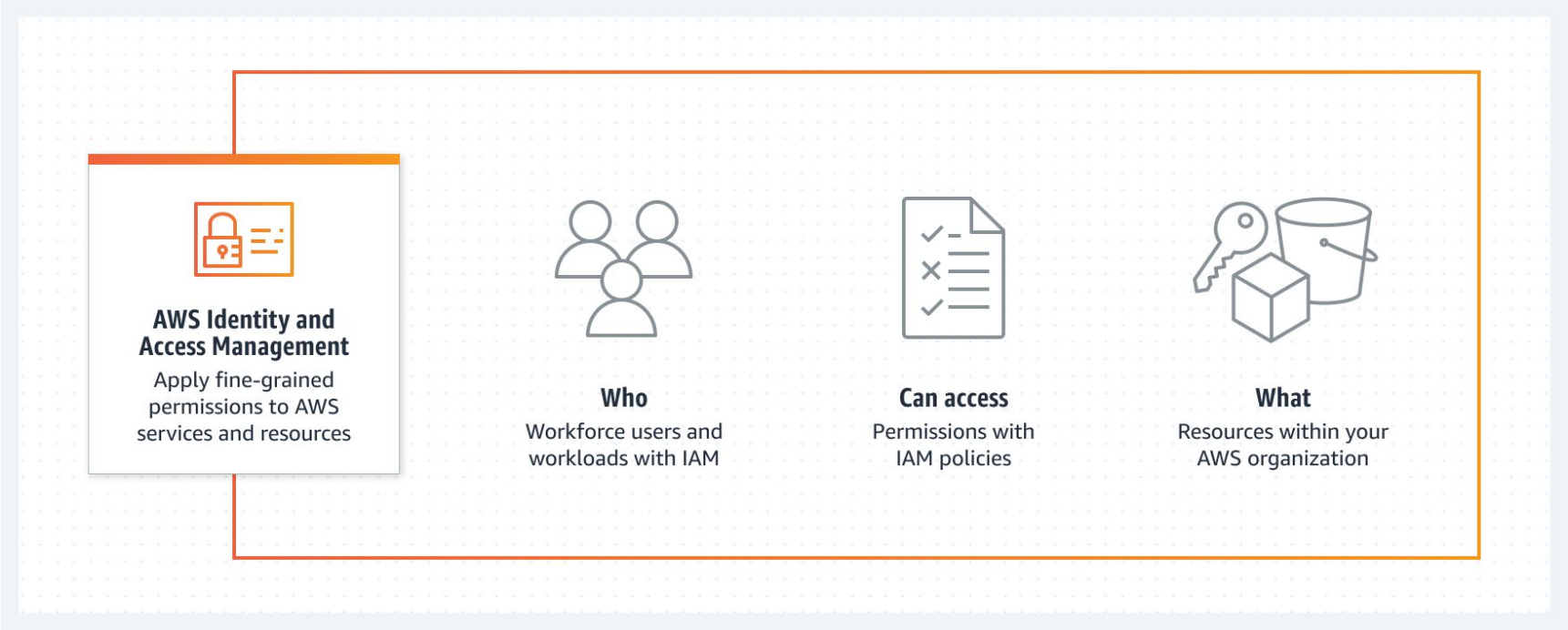
d. Use long and complex passwords for access keys

Lesson Overview

Lesson Overview

IAM - Identity Access Management

Lesson Overview



Lesson Overview - Video

Lesson Overview

Users - Specific human individuals, can have personal logins

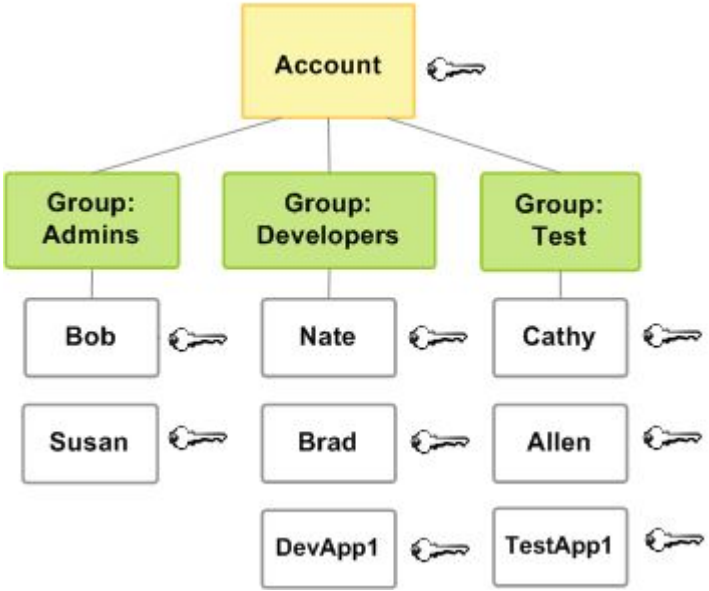
Groups - Collection of users e.g. HR, Developers, Security

Roles - Collection of Policies (DB Read, DB Write, S3 Admin)

Policies - Low-level permission to resources i.e. ALLOW or DENY

Lesson Overview

Groups - Collection of users e.g. HR, Developers, Security



Lesson Overview

Policies - Low-level permission to resources i.e. ALLOW or DENY

```
{
    "Version":"2012-10-17",
    "Statement":[
       {
          "Sid": "add-read-only-perm",
          "Principal": "*",
          "Effect": "Allow",
          "Action": [
             "glacier:InitiateJob",
             "glacier:GetJobOutput"
          ],
          "Resource": [
             "arn:aws:glacier:us-west-2:999999999999:vaults/examplevault"
          ]
       }
    ]
}
```

Lesson Overview

By **creating policies and tying them to IAM identities** (users, groups of users, or roles) or AWS resources, you can **manage access in AWS**.

In AWS, a policy is an object that, when linked to an identity or resource, **defines the permissions for those objects**.

Why IAM?

- **Security** - Securely manage identities and access to AWS services and resources
- **Compliance** - Embedding IAM security strategies into your business operations helps you keep up with regulatory compliance
- **Confidentiality** - Only allowing the right users, groups or applications access to private data.

Policy Types

Overview

- Identity-Based Policies
- Resource-Based Policies
- Permissions Boundaries
- Organizations' SCPs
- Access Control Lists (ACLs)
- Session Policies

Identity-Based Policies

Attach **managed** or **inline policies** to IAM identities (users, groups to which users belong, or roles).

Identity-based policies grant permissions to an identity.

Identity-Based Policies

Identity-based policies are **JSON permissions policy documents that control what actions an identity** (users, groups of users, and roles) **can perform**, on **which resources**, and **under what conditions**. Identity-based policies can be further categorized:

Managed policies – Standalone identity-based policies that you can attach to multiple users, groups, and roles in your AWS account. There are two types of managed policies:

1. **AWS managed policies** – Managed policies that are created and managed by AWS.

Identity-Based Policies

Managed policies – Standalone identity-based policies that you can **attach to multiple users, groups, and roles** in your AWS account. There are two types of managed policies:

2. Customer managed policies – Managed policies that you create and manage in your AWS account. Customer managed policies provide more precise control over your policies than AWS managed policies.

Inline policies – Policies that you add directly to a single user, group, or role. Inline policies **maintain a strict one-to-one relationship** between a policy and an identity. They are deleted when you delete the identity.

Resource-Based Policies

Attach **inline policies to resources.** The most common examples of resource-based policies are **Amazon S3 bucket policies** and **IAM role trust policies**.

Resource-based policies grant permissions to the principal that is specified in the policy. Principals can be in the same account as the resource or in other accounts.

Resource-Based Policies

Resource-based policies are **JSON policy documents that you attach to a resource such as an Amazon S3 bucket**.

These policies grant the specified principal permission to **perform specific actions on that resource** and **defines under what conditions this applies**.

Resource-based policies are **inline policies.** There are no managed resource-based policies.

Resource-Based Policies

To enable **cross-account access**, you can specify an **entire account or IAM entities in another account as the principal** in a resource-based policy.

Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship.

When the principal and the resource are in separate AWS accounts, you must also use an identity-based policy to grant the principal access to the resource.

However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required.

Resource-Based Policies

The IAM service supports only **one type of resource-based policy** called a **role trust policy**, which is attached to an IAM role.

An IAM role is both an identity and a resource that supports resource-based policies.

For that reason, you must **attach both a trust policy and an identity-based policy to an IAM role**.

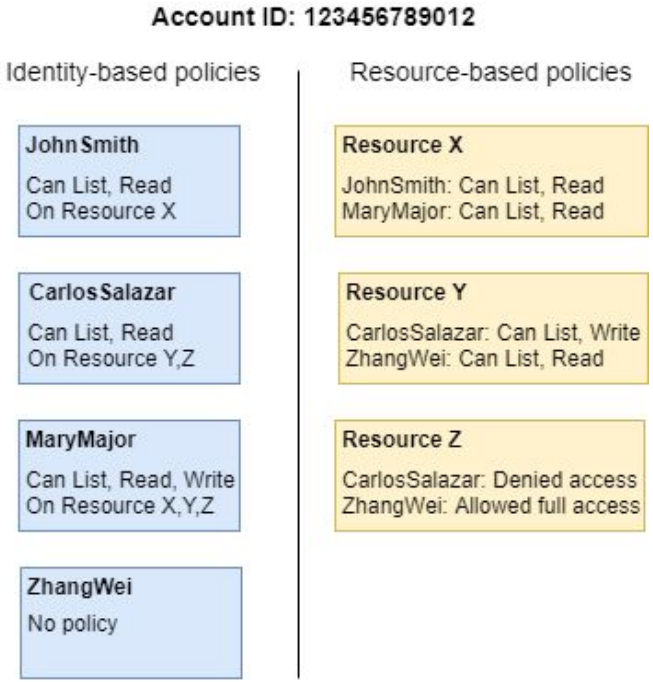
Trust policies define which principal entities (accounts, users, roles, and federated users) can assume the role.

Trust Policy Examples

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::255945442255:user/jazeel-temp-user"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Comparing IBP & RBP



Permissions Boundaries

Use a **managed policy** as the permissions boundary for an **IAM entity** (user or role).

That policy defines the **maximum permissions** that the **identity-based policies can grant to an entity**, but **does not grant permissions**.

Permissions boundaries do not define the maximum permissions that a resource-based policy can grant to an entity.

Permissions Boundaries

A permissions boundary is an **advanced feature** in which you set the **maximum permissions that an identity-based policy can grant to an IAM entity**.

When you set a permissions boundary for an entity, the entity can perform only the actions that are **allowed by both its identity-based policies and its permissions boundaries**.

Resource-based policies that specify the user or role as the principal are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow.

Session Policies

Pass advanced session policies when you use the AWS CLI or AWS API to **assume a role or a federated user**.

Session policies **limit the permissions that the role or user's identity-based policies grant to the session**.

Session policies limit permissions for a created session, but do not grant permissions.

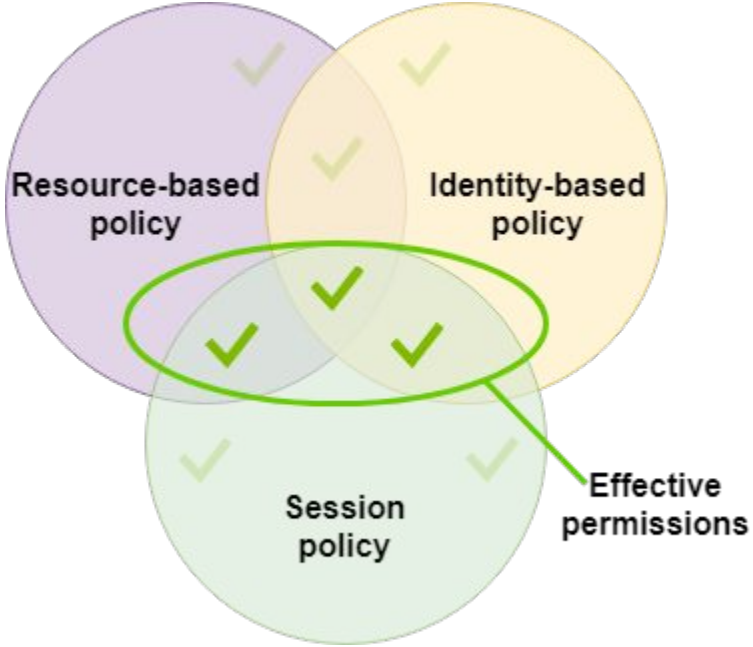
Session Policies

Session policies are **advanced policies** that you pass as a parameter when you programmatically create a **temporary session for a role or federated user.**

The permissions for a session are the **intersection of the identity-based policies for the IAM entity (user or role) used to create the session and the session policies**. Permissions can also come from a resource-based policy.

An explicit deny in any of these policies overrides the allow.

Session Policies



Session Policies

You can create role session and pass session policies programmatically using the AssumeRole, AssumeRoleWithSAML, or AssumeRoleWithWebIdentity API operations.

You can pass a single JSON inline session policy document using the Policy parameter. You can use the PolicyArns parameter to specify up to 10 managed session policies.

Session Policies

A resource-based policy can specify the ARN of the user or role as a principal.

In that case, the permissions from the resource-based policy are added to the role or user's identity-based policy before the session is created.

The session policy limits the total permissions granted by the resource-based policy and the identity-based policy.

The resulting session's permissions are the intersection of the session policies and the resource-based policies plus the intersection of the session policies and identity-based policies.

S3 Access Control Lists (ACLs)

S3 ACLs **allow you to control which principals in another account can access objects** and buckets.

Therefore every s3 bucket and object in a s3 bucket has a ACL attached to it as a sub resource. (Disabled by Default)

ACLs are similar to resource-based policies, although they are the only policy type that **does not use the JSON policy document format**.

However, S3 ACLs are a legacy feature, and **AWS recommends using Bucket Policies** to control access to bucket and objects going forward.

Not to be confused with: AWS WAF and AWS VPC also use the term “Access Control Lists”, however that is a entirely different feature provided by those services.

Organization SCP

Use an AWS Organizations Service Control Policy (SCP) to **define the maximum permissions for account members of an organization or organizational unit** (OU).

SCPs **limit permissions that identity-based policies or resource-based policies grant to entities** (users or roles) within the account, but do not grant permissions.

Policy Evaluation Logic

[Policy evaluation logic - AWS Identity and Access Management (amazon.com)]

Summary

- Identity-Based Policies ***
- Resource-Based Policies ***
- Permissions Boundaries **
- Session Policies **
- Access Control Lists (ACLs) **
- Organizations' SCPs **

More On IAM

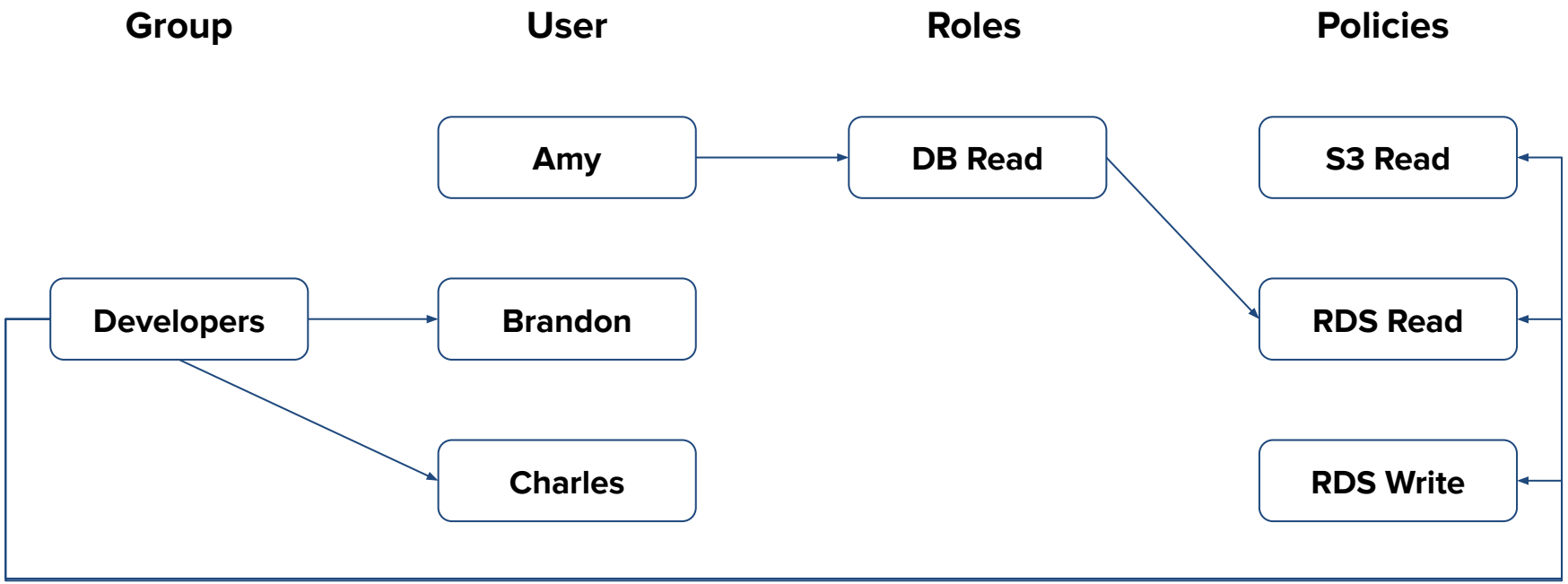
Examples of Permission & Policies

- Allows access during a specific range of dates.
- Allows enabling and disabling AWS Regions.
- Allows MFA-authenticated users to manage their own credentials on the My Security Credentials page.
- Allows specific access when using MFA during a specific range of dates.
- Allows users to manage their own credentials on the My Security Credentials page.

Examples of Permission & Policies

- Allows users to manage their own MFA device on the My Security Credentials page.
- Allows users to manage their own password on the My Security Credentials page.
- Allows users to manage their own password, access keys, and SSH public keys on the My Security Credentials page.
- Denies access to AWS based on the requested Region.
- Denies access to AWS based on the source IP address.

Tying It Up


Group
User
Roles
Policies
Amy
DB Read
S3 Read
Developers
Brandon
RDS Read
Charles
RDS Write

Best Practices

Rotate access keys regularly for use cases that require long-term credentials

Require **Multi-Factor Authentication**

Grant **Least Privilege**

https://aws.amazon.com/iam/resources/best-practices/?nc=sn&loc=4&dn=1

Something to think about

Let's say in your organization, you have the following teams:

- Cloud Engineers
- Developers
- Data Scientists/ Engineer
- Security / Audit Team

Think of how you will segregate the access between the various teams

Activity

Learner:

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.

Instructor

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.
- Check the AWS account after learner clean up.

What’s Next?