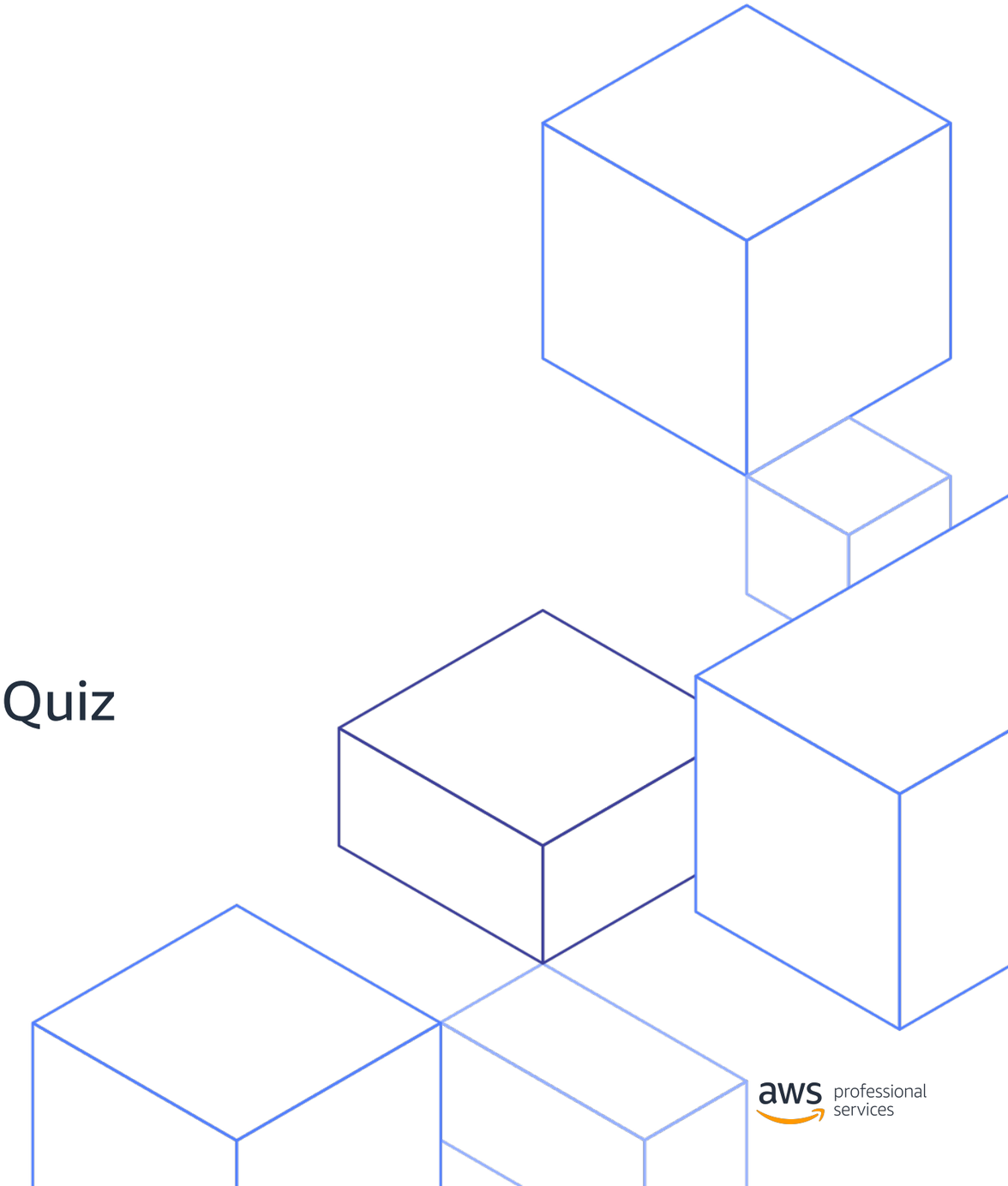
aws professional services

Identity & Access Management Permissions Quiz

Quiz time!

Identity & Access Management Permissions Quiz



aws professional services

When to use each type of permission policy?

AWS Organizations
Service control policies (SCPs)

Guardrails on the account to disable access to services

AWS IAM
Inline policies
Managed policies

Set granular permissions based on functions that users or applications need to perform

AWS STS
Scoped-down policies

Reduce general shared permissions further

Specific AWS services
Resource-based policies

Example: S3 bucket policies

Cross-account access and to control access from the resource

#1 - Set permission guardrails across accounts

#1 - Situation

Your team has gone through and set up Cloudtrail in all accounts. Your company also requires users to authenticate with their existing identity provider.

#1 - Challenge

Ensure developers cannot turn off Cloudtrail, create IAM users, or set up AWS Directory Services.

#2 - Control creation of resources to specific regions

#2 - Situation

You've learned that you can trust your development team to create resources in AWS, however your leadership is concerned about creating resources in unapproved regions.

#2 - Challenge

Ensure your developers can create resources, but only in approved regions.

#3 - Enable developers to create roles safely

#3 - Situation

Your developers know their stuff! They mentioned they can build on AWS more quickly if they can create their own roles without going through your central security team.

#3 - Challenge

Enable your developers to create IAM roles to pass to EC2 and Lambda, but ensure they cannot exceed their own permissions.

#4 - Use tags to scale permissions management

#4 - Situation

The Unicorns project has been split into two projects. Dorky Unicorns and Sneaky Unicorns. They still share an account and keep stepping on each other toes.

#4 - Challenge

Update permissions to enable developers working on Dorky Unicorns and Sneaky Unicorns to manage their own resources without managing the other project's.

Match each tool to the correct use-case

Service control policies (SCPs)

Permission Boundaries

IAM Permission Policy

Scoped-down policies

Resource-based policies

Endpoint Policies

Set Permission Guardrails Across Accounts

Control Creation of Resources to Regions

Enable Developers to Create Roles Safely

Use Tags to Scale Permissions Management

aws professional services

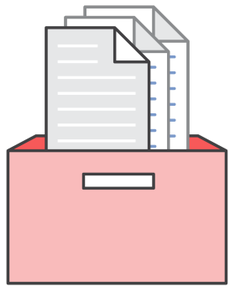
Challenge #1

Ensure developers cannot turn off Cloudtrail, create IAM users, or set up AWS Directory Services.

Pro Tip: Rely on deny statements when restricting access to accounts to reduce blast radius.

SCP for challenge #1

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyUnapprovedAction",
            "Effect": "Deny",
            "Action": [
                "ds:*",
                "iam:CreateUser",
                "cloudtrail:StopLogging"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```



Don’t Forget!

*We also have an Allow *.* policy attached to this OU*

Challenge #2

Ensure your developers can create resources, but only in approved regions.

Pro Tip: Use the RequestedRegion AWS condition

Policy for challenge #2

```
{
          "Effect": "Allow",
           "Action": [
               "secretsmanager:*",
               "lambda:*",
               "s3:PutObject",
               "s3:GetObject",
               "s3:DeleteObject"
           ],
           "Resource": "*",
           "Condition": {
               "StringEquals": {
                   "aws:RequestedRegion": [
                       "us-west-1",
                       "us-west-2"
                   ]
               }
           }
       }
```

Policy for challenge #2

```
{
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*:*:launch-template/*",
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:placement-group/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*::image/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": ["us-west-1","us-west-2"]
                }
            }
        }
```

Policy for challenge #2

```
{
          "Effect": "Allow",
           "Action": [
               "ec2:Describe*",
               "ec2:Get*",
               "s3:ListBucket",
               "s3:ListAllMyBuckets",
               "iam:list*"
           ],
           "Resource": "*"
       }
```

Challenge #3

Enable your developers to create IAM roles to pass to EC2 and

Lambda, but ensure they cannot exceed their own permissions.

Pro Tip: Require and use role naming conventions to control the roles developers can manage.

Four parts required for permission boundaries

Allow create managed policies

Allow create role, but only with a specific permission boundary

- *This is a condition with a pointer to an existing managed policy*

Allow attach managed policies, but only to roles with a specific boundary

Allow passRole for these roles using a naming requirement

Policy for Challenge #3

Allow create managed policies

```
 {
     "Effect": "Allow",
     "Action": [
          "iam:CreatePolicy",
          "iam:CreatePolicyVersion",
          "iam:DeletePolicyVersion"
     ],
     "Resource": "arn:aws:iam::128609111811:policy/unicorns-*"
}
```

Policy for Challenge #3

Allow create role, but only with a specific permission boundary.
Allow attach managed policies, but only to roles with a specific boundary

```
{
     "Effect": "Allow",
     "Action": [
          "iam:DetachRolePolicy",
          "iam:CreateRole",
          "iam:AttachRolePolicy"
     ],
     "Resource": "arn:aws:iam::128609111811:role/unicorns-*",
     "Condition": {
          "StringEquals": {
               "iam:PermissionsBoundary": "arn:aws:iam::128609111811:policy/region-
restriction"
          }
     }
}
```

Permission boundary workflows

Admin creates maximum permissions

Admin allows developers to create role with maximum permissions

Developer creates role with maximum permissions and specific permissions

Developers passes the role to application resources

aws professional services

Challenge #4

Enable developers working on project X and project Y to manage their own resources without also managing the other project's.

Pro Tip: Carefully consider the tag keys you want to use for authorization

Three parts required for tag-based access control

- Allow users to create tags when creating resources, but require specific tags when users create resources
  - *RequestTag condition to require specific tag value during create actions*
- Control which existing resources and values developers can tag
  - *Use a combination of RequestTag and ResourceTag control access*
- Control resources users can manage based on tag values
  - ResourceTag to control access to resources based on a tag that exists on a resource

Policy for challenge #4

```
"Effect": "Allow",
          "Action": [
              "ec2:RunInstances"
          ],
          "Resource": [
              "arn:aws:ec2:*:*:subnet/*",
              "arn:aws:ec2:*:*:key-pair/*",
              "arn:aws:ec2:*::snapshot/*",
              "arn:aws:ec2:*:*:launch-template/*",
              "arn:aws:ec2:*:*:volume/*",
              "arn:aws:ec2:*:*:security-group/*",
              "arn:aws:ec2:*:*:placement-group/*",
              "arn:aws:ec2:*:*:network-interface/*",
              "arn:aws:ec2:*::image/*"
          ],
          "Condition": {
              "StringEquals": {
                  "aws:RequestedRegion": ["us-west-1", "us-west-2" ]
              }
          }
```

Policy for Challenge #4

Allow for creation of tags when creating new resources, but…

```
"Effect": "Allow",
"Action": "ec2:CreateTags",
"Resource": "*",
"Condition": {
     "StringEquals": {
          "ec2:CreateAction": "RunInstances"
     }
}
```

Allows creation of tags

But only during RunInstances calls

Policy for Challenge #4

…require specific tags when users create new resources

```
"Effect": "Allow",
"Action": [
     "ec2:RunInstances"
],
"Resource": [
     "arn:aws:ec2:*:*:instance/*"],
"Condition": {
     "ForAllValues:StringEquals": {
          "aws:TagKeys": ["project","name"]
     },
     "StringEquals": {
          "aws:RequestTag/project": ["dorky"],
          "aws:RequestedRegion": ["us-west-1","us-west-2"]
     }}
```

Allows project and/or name, but nothing else

Requires project tag and must be this value

Requires instance to be in approved region

Policy for Challenge #4

Control which existing resources and values developers can tag

```
"Effect": "Allow",
"Action": "ec2:CreateTags",
"Resource": "*",
"Condition": {
     "StringEquals": {
          "ec2:ResourceTag/project": ["dorky"]
     },
     "ForAllValues:StringEquals": {
          "aws:TagKeys": ["project","name"]
     },
     "StringEqualsIfExists": {
          "aws:RequestTag/project": ["dorky"]
     }
}
```

Only tag resources with these tags

Tag with either of these keys

For *project*, you specify only these values

Policy for Challenge #4

Control resources users can manage based on tag values

```
"Effect": "Allow",
"Action": [
     "ec2:StartInstances",
     "ec2:StopInstances"],
"Resource": "*",
"Condition": {
     "StringEquals": {
          "ec2:ResourceTag/project": "dorky"
     }
}
}
```

Only manage resources with these tags

Bonus challenge

New! You can tag IAM users and roles

Create a general policy that allows read access to secrets tagged with a role tag.

Pro Tip: Any condition key can also be used as a variable as a condition value (the right hand side)

```
["${aws:PrincipalTag/project}"]
```

Policy for Challenge #5

…require specific tags when users create new resources

```
"Effect": "Allow",
"Action": [
     "ec2:RunInstances"
],
"Resource": [
     "arn:aws:ec2:*:*:instance/*"],
"Condition": {
     "ForAllValues:StringEquals": {
          "aws:TagKeys": ["project","name"]
     },
     "StringEquals": {
          "aws:RequestTag/project": ["${aws:PrincipalTag/project}"],
          "aws:RequestedRegion": ["us-west-1","us-west-2"]
     }}
```

Allows project and/or name, but nothing else

Requires project tag and must be my project tag

Requires instance to be in approved region

Policy for Challenge #5

Control which existing resources and values developers can tag

```
"Effect": "Allow",
"Action": "ec2:CreateTags",
"Resource": "*",
"Condition": {
     "StringEquals": {
          "ec2:ResourceTag/project": ["${aws:PrincipalTag/project}"]
     },
     "ForAllValues:StringEquals": {
          "aws:TagKeys": ["project","name"]
     },
     "StringEqualsIfExists": {
          "aws:RequestTag/project": ["${aws:PrincipalTag/project}"]
     }
}
```

Only tag resources with my project tag

Tag with either of these keys

For *project*, you specify your project tag

Policy for Challenge #5

Control resources users can manage based on tag values

```
"Effect": "Allow",
"Action": [
     "ec2:StartInstances",
     "ec2:StopInstances"],
"Resource": "*",
"Condition": {
     "StringEquals": {
          "ec2:ResourceTag/project": "${aws:PrincipalTag/project}"
     }
}
}
```

Only manage resources with my project tag

aws professional services

The End