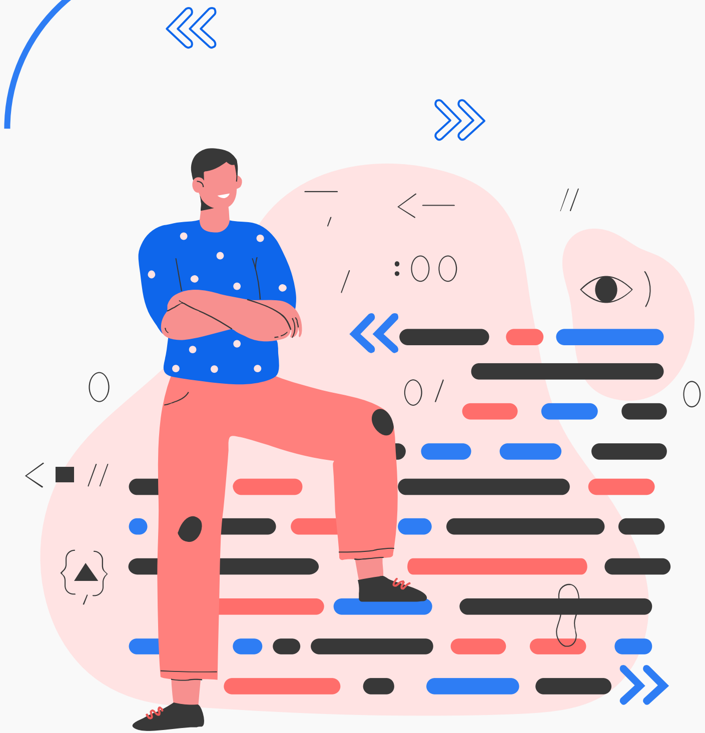
AWS IAM

Good & Bad Practices

Michał Mikołajczyk | Knowit
AWS Meetup Silesia 3/2024



Cześć!

Michał Mikołajczyk
AWS Solutions Architect / Cloud Engineer
Knowit

knowit

knowit.pl

aws certified
Solutions Architect
PROFESSIONAL
aws certified
DevOps Engineer
PROFESSIONAL
aws certified
Solutions Architect
ASSOCIATE
aws certified
Security
SPECIALTY
Terraform
CERTIFIED ASSOCIATE
HCTA0-002
CERTIFIED
kubernetes
ADMINISTRATOR

Table of contents

01

Introduction

Why are we here

02

Theory

AWS IAM Recap,
Best & Worst Practices

03

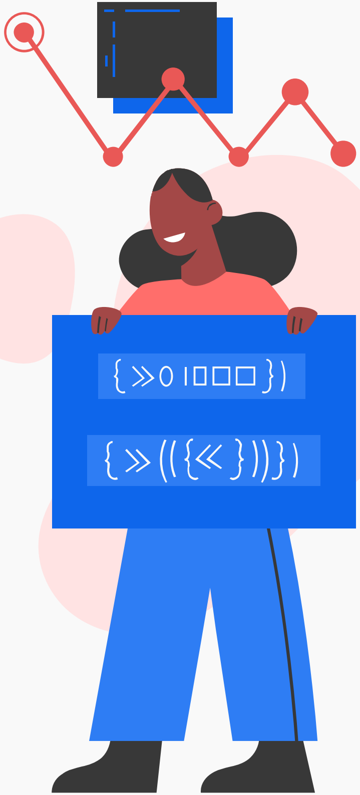
Demos

Secret

04

Conclusions

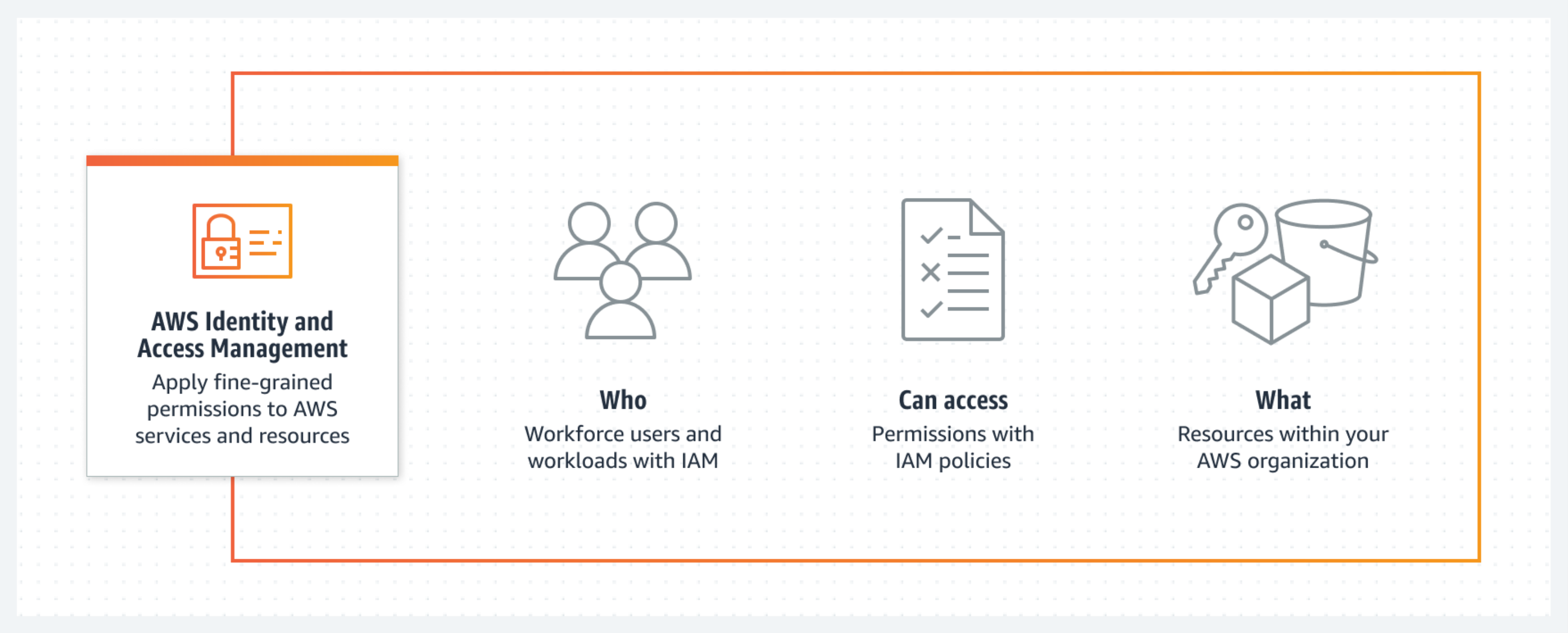
Summary & Discussion



01

Introduction

Identity and Access Management



AWS Identity and Access Management

Apply fine-grained permissions to AWS services and resources

Who

Workforce users and workloads with IAM

Can access

Permissions with IAM policies

What

Resources within your AWS organization

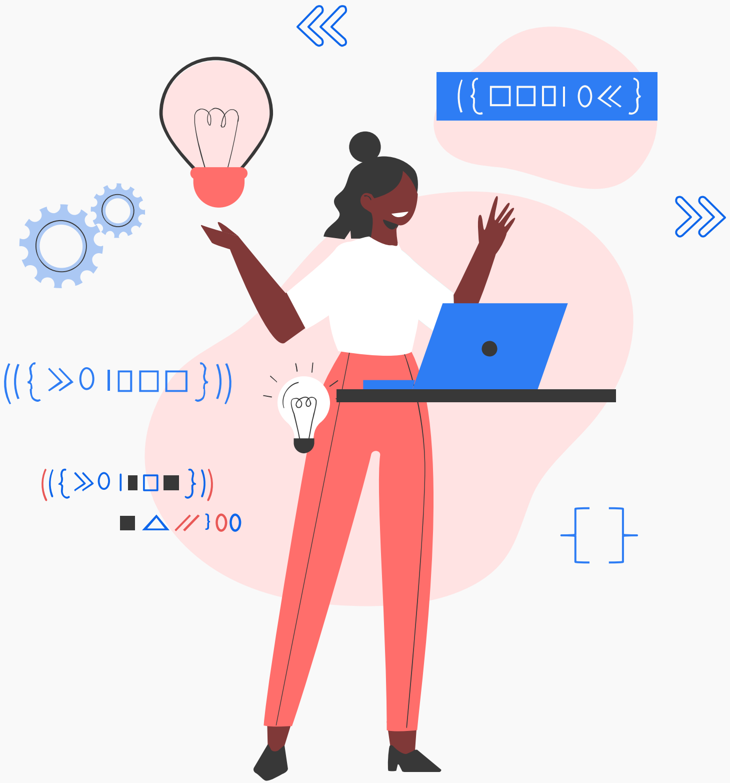
“DevOps is a meaningful term. You understand DevOps because you use it everyday”

—Positive Affirmations for SREs

https://www.youtube.com/watch?v=ia8Q51ouA_s

“You understand AWS IAM because you use it everyday”

—Me



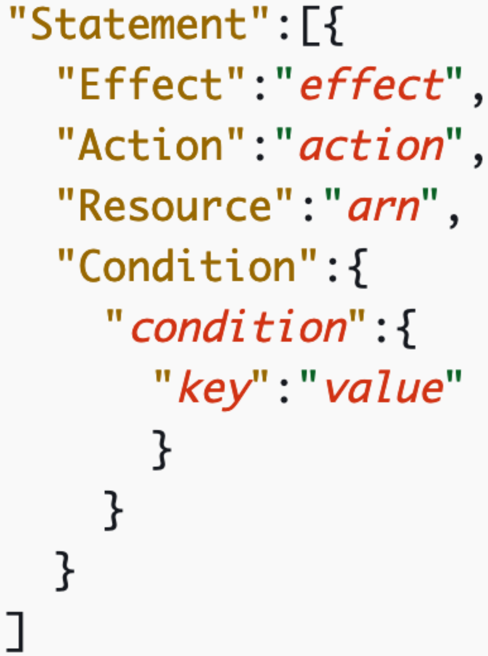
02

Theory

AWS IAM Recap

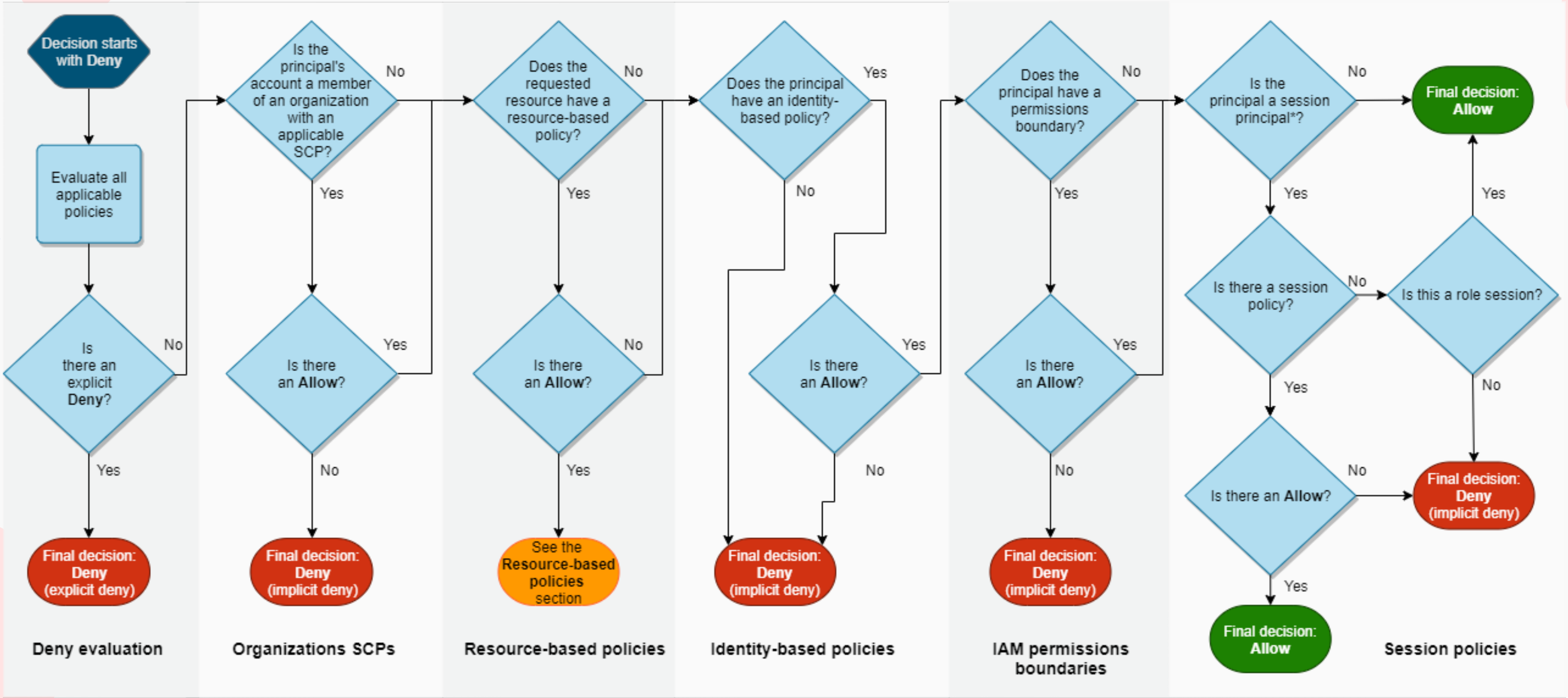
Policy Syntax

- **Effect:** Allow or Deny.
  An explicit deny overrides any allows.
- **Action**: Specific API action for which you are granting or denying permission.
- **Resource**: The resource that's affected by the action.
- **Condition**: (Optional). Used to control when your policy is in effect.



```
"Statement":[{
  "Effect":"effect",
  "Action":"action",
  "Resource":"arn",
  "Condition":{
    "condition":{
      "key":"value"
      }
    }
  }
]
```

Policy Evaluation



Decision starts with Deny

Evaluate all applicable policies

Is there an explicit Deny?

Yes

Final decision: Deny (explicit deny)

No

Deny evaluation

Is the principal's account a member of an organization with an applicable SCP?

Yes

Is there an Allow?

No

Final decision: Deny (implicit deny)

Yes

No

Organizations SCPs

Does the requested resource have a resource-based policy?

Yes

Is there an Allow?

Yes

See the Resource-based policies section

No

No

Resource-based policies

Does the principal have an identity-based policy?

No

Final decision: Deny (implicit deny)

Yes

Is there an Allow?

No

Yes

Identity-based policies

Does the principal have a permissions boundary?

Yes

Is there an Allow?

No

Final decision: Deny (implicit deny)

Yes

No

IAM permissions boundaries

Is the principal a session principal*?

No

Final decision: Allow

Yes

Is there a session policy?

Yes

Is there an Allow?

Yes

Final decision: Allow

No

Final decision: Deny (implicit deny)

No

Is this a role session?

Yes

No

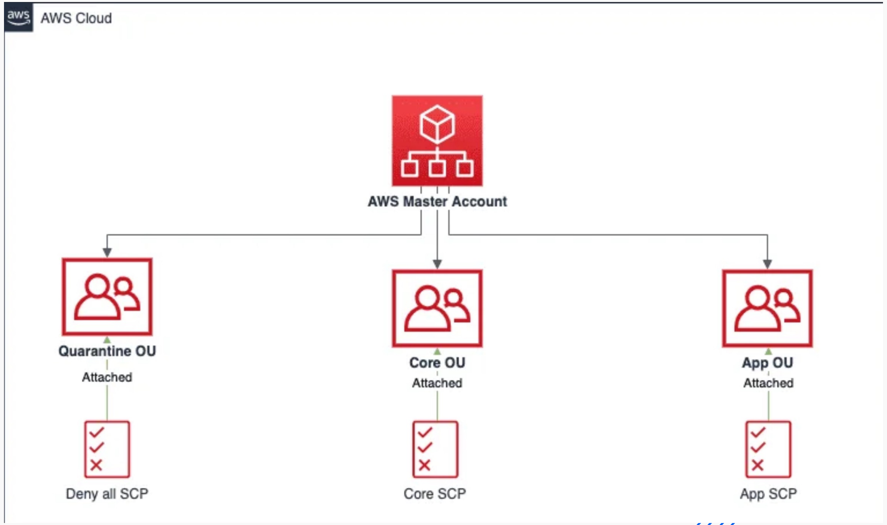
Session policies

*A session principal is either a role session or an IAM federated user session.

Better Policy Evaluation

https://www.youtube.com/watch?v=71-Gjo6a5Cs

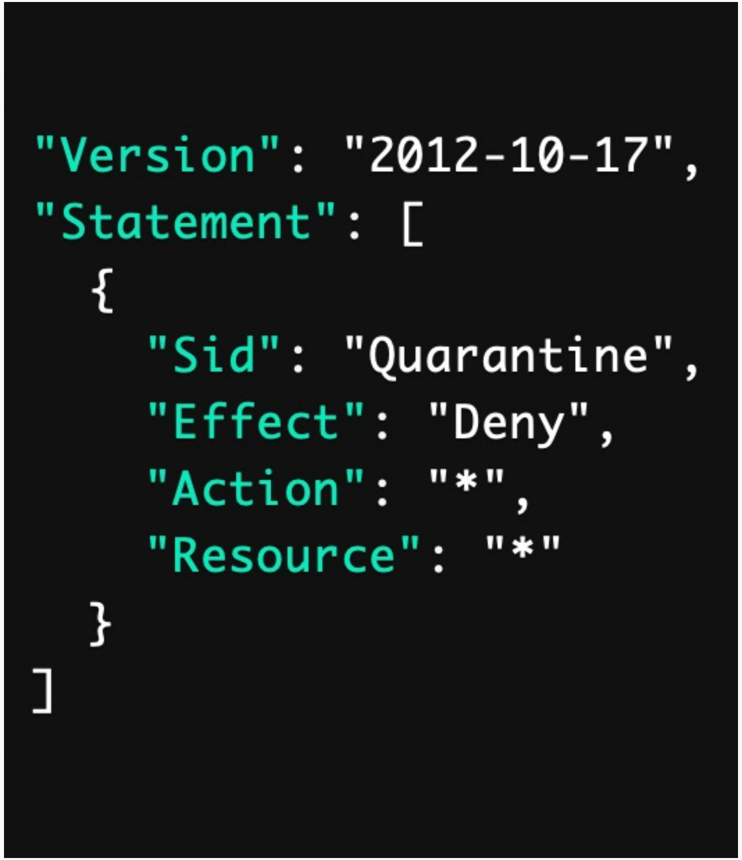
Service Control Policies(SCPs)



- SCPs cannot restrict principals outside of the Organization
- Applied to Organization, OU or Account
- 5 levels
- 5 SCPs per level

Example SCP

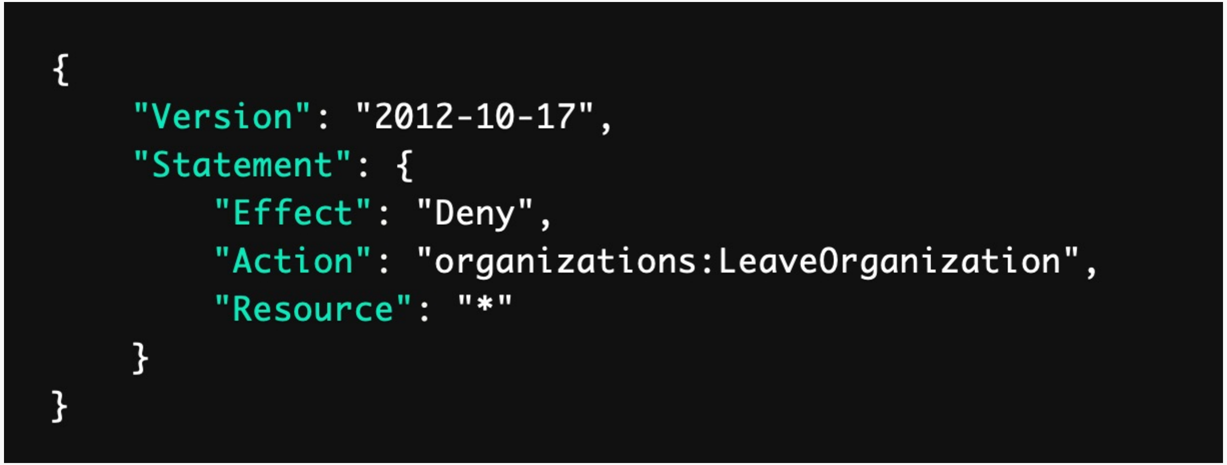
- Deny All Actions



```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "Quarantine",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*"
  }
]
```

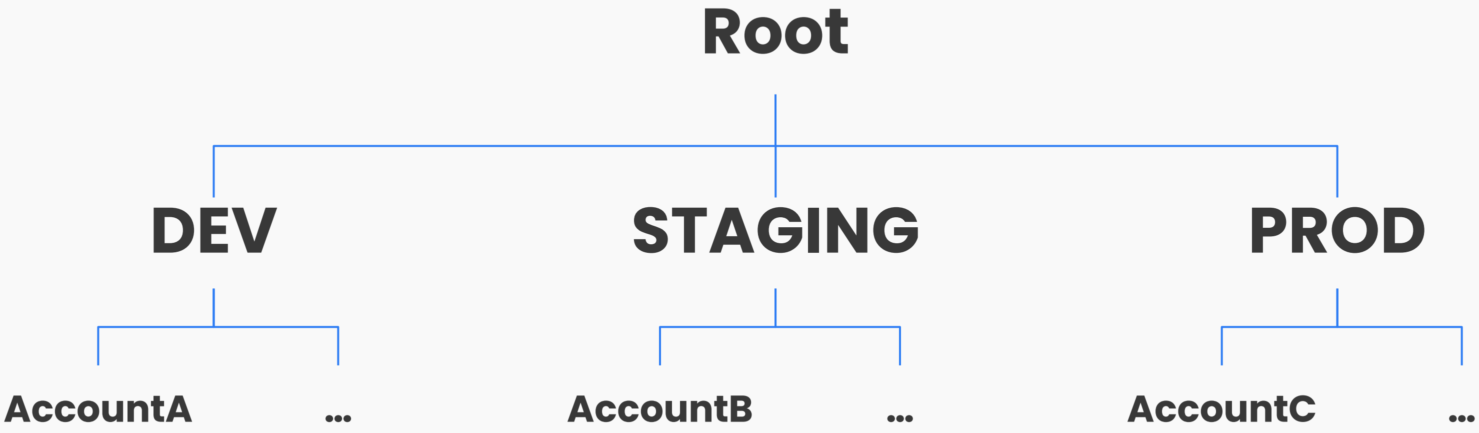
Example SCP

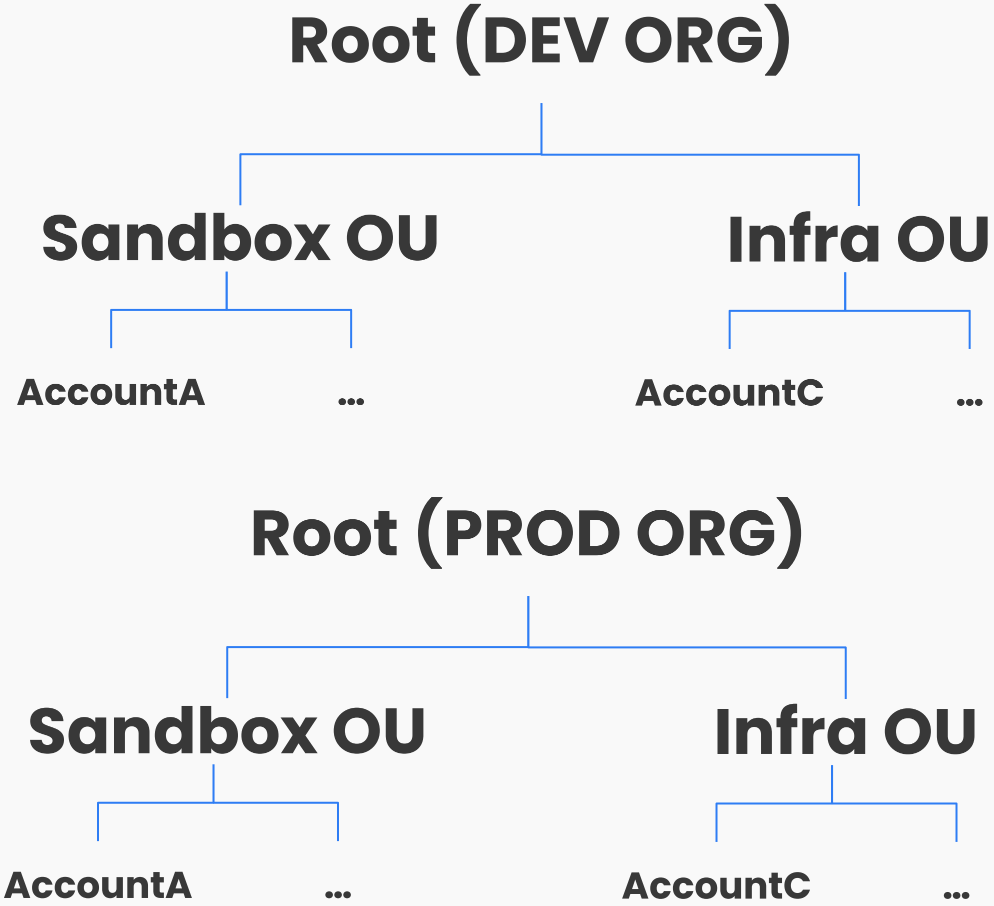
- Prevent Account from leaving the Organization



```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "organizations:LeaveOrganization",
        "Resource": "*"
    }
}
```

Organization Structure





Root (DEV ORG)

Sandbox OU

AccountA

...

Infra OU

AccountC

...

Root (PROD ORG)

Sandbox OU

AccountA

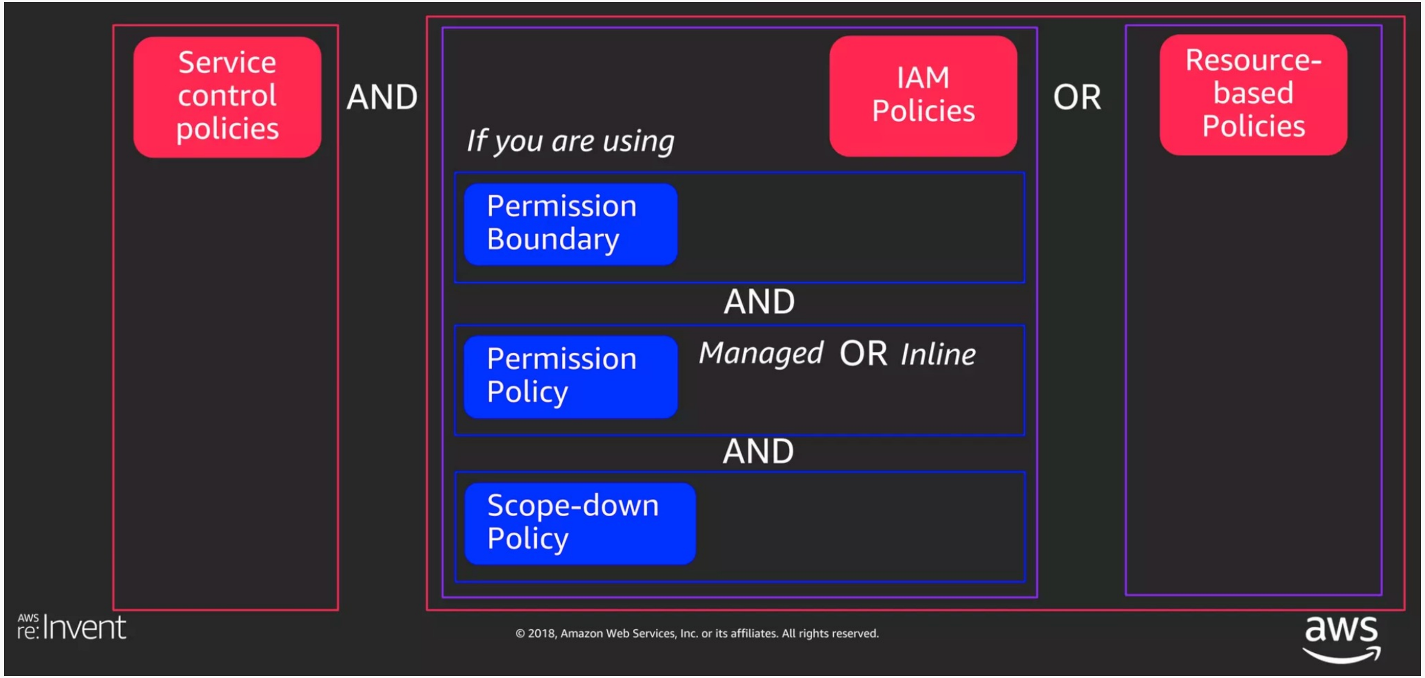
...

Infra OU

AccountC

...

Policy Evaluation – Same Account



Service control policies

AND

IAM Policies

If you are using

Permission Boundary

AND

Permission Policy *Managed* OR *Inline*

AND

Scope-down Policy

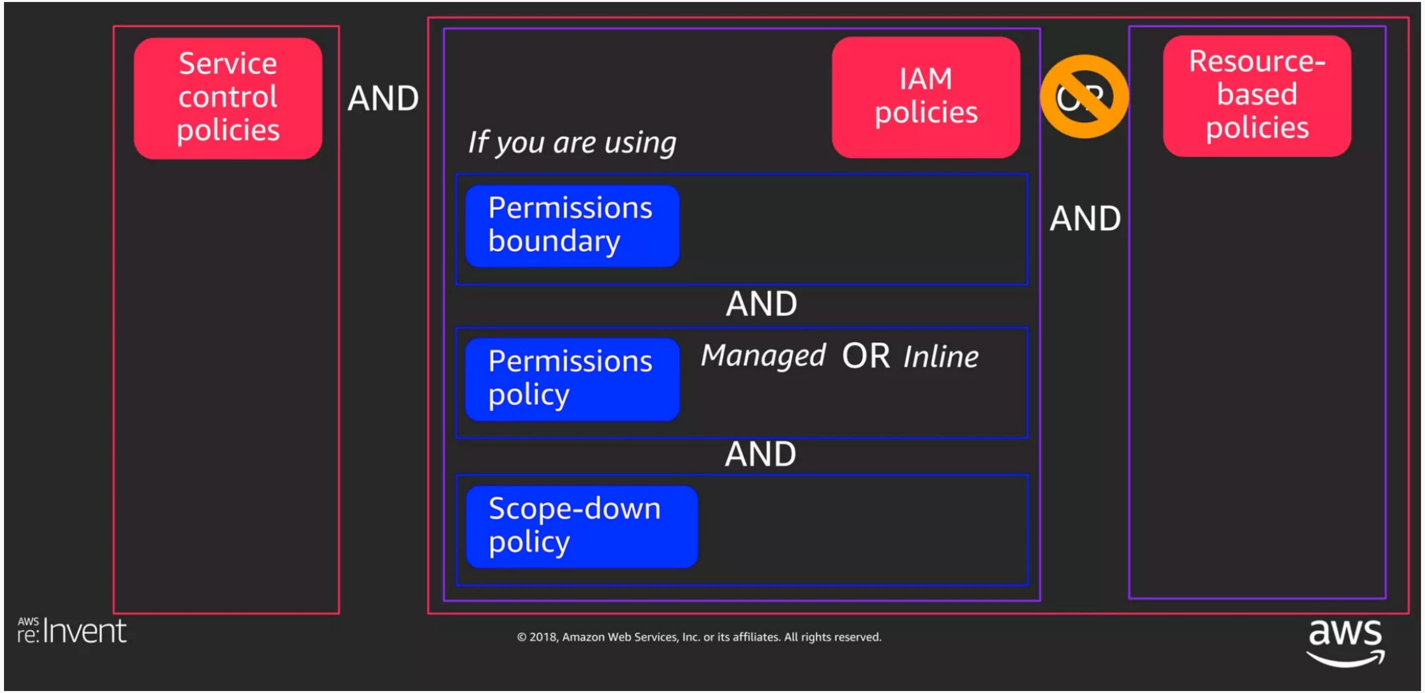
OR

Resource-based Policies

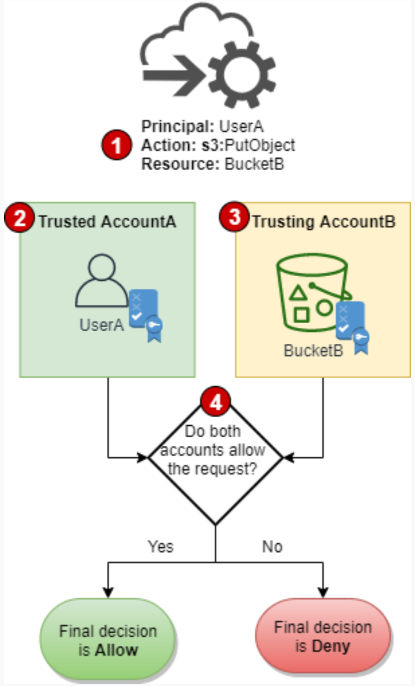
AWS re:Invent

aws

Policy Evaluation – Cross Account



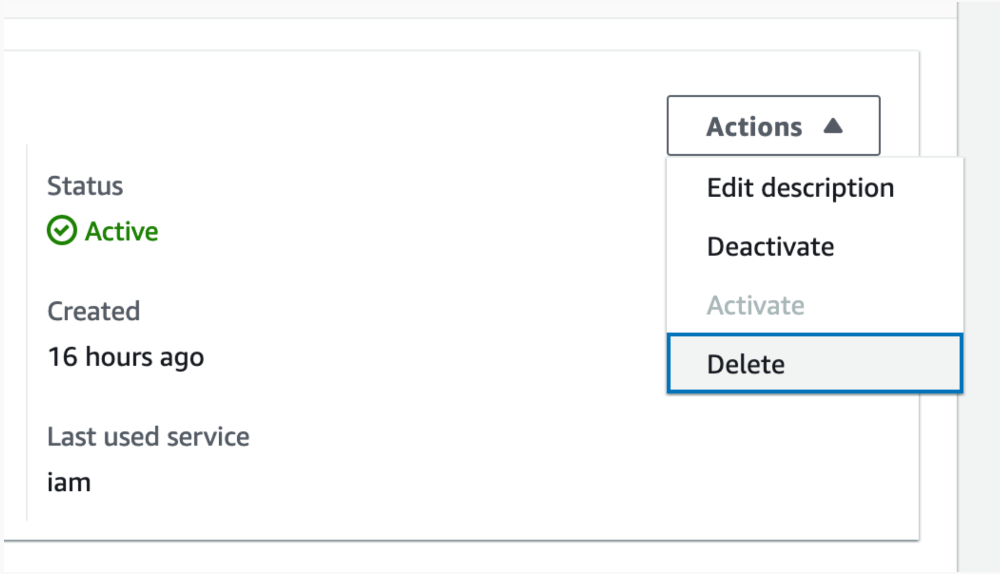
Policy Evaluation – Cross Account



Good & Bad Practices

Access Keys

- Rotate Keys
- Delete deactivated keys
- Notify Users
- Store Credentials Securely



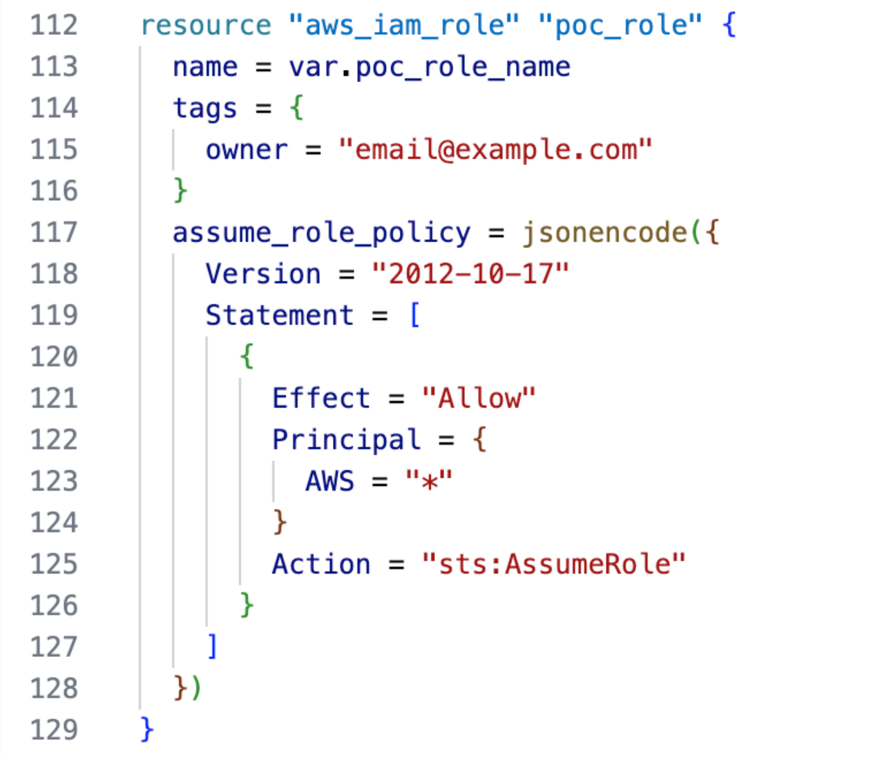
Status
✓ Active

Created
16 hours ago

Last used service
iam

All Principals can assume a role

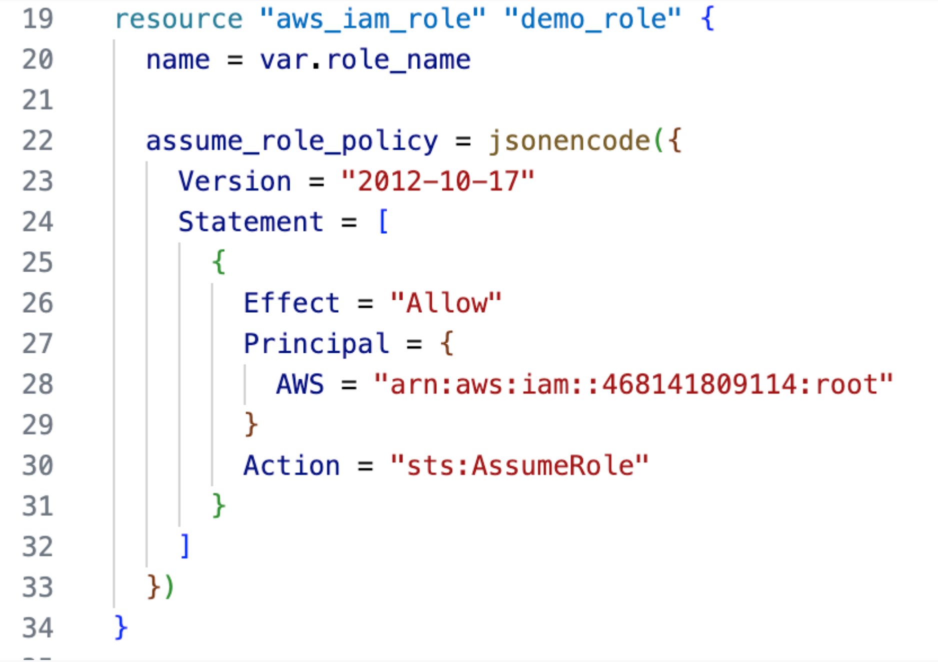
- Allows Any Principal in Any AWS Account to assume a role



```
resource "aws_iam_role" "poc_role" {
  name = var.poc_role_name
  tags = {
    owner = "email@example.com"
  }
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Principal = {
          AWS = "*"
        }
        Action = "sts:AssumeRole"
      }
    ]
  })
}
```

“ACCOUNTNUMBER/root” in trust policy

- Allows Any Principal from the AWS Account to assume a role

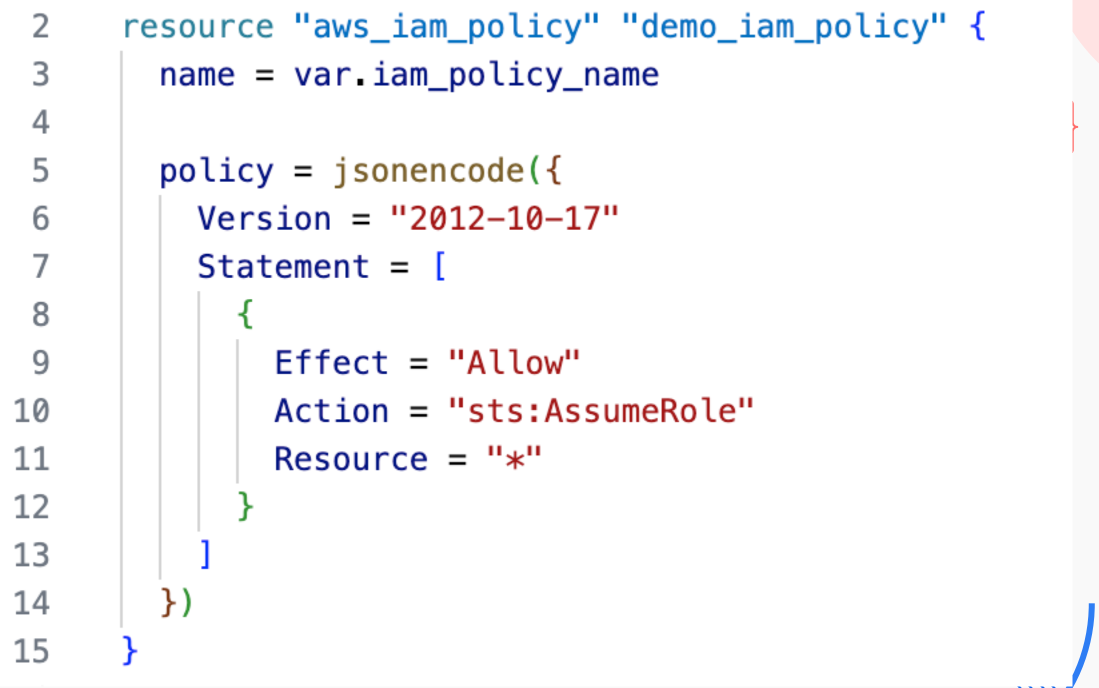


```
resource "aws_iam_role" "demo_role" {
  name = var.role_name

  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Principal = {
          AWS = "arn:aws:iam::468141809114:root"
        }
        Action = "sts:AssumeRole"
      }
    ]
  })
}
```

Too broad AssumeRole permissions

- Principal can Assume ANY ROLE

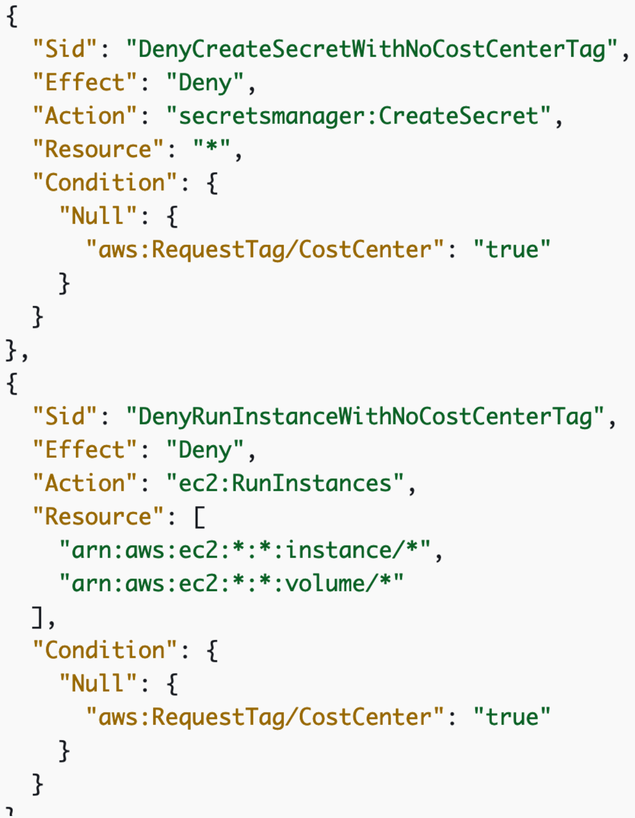


```
resource "aws_iam_policy" "demo_iam_policy" {
  name = var.iam_policy_name

  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Action = "sts:AssumeRole"
        Resource = "*"
      }
    ]
  })
}
```

Not using SCPs enough

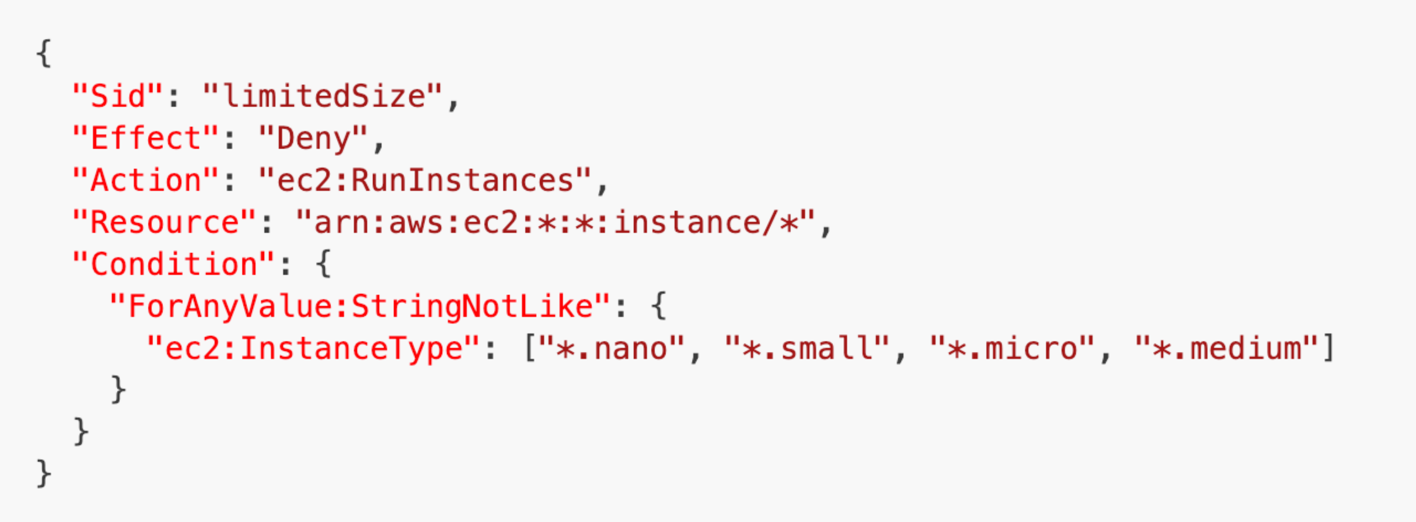
- Trusting users too much



```
{
  "Sid": "DenyCreateSecretWithNoCostCenterTag",
  "Effect": "Deny",
  "Action": "secretsmanager:CreateSecret",
  "Resource": "*",
  "Condition": {
    "Null": {
      "aws:RequestTag/CostCenter": "true"
    }
  }
},
{
  "Sid": "DenyRunInstanceWithNoCostCenterTag",
  "Effect": "Deny",
  "Action": "ec2:RunInstances",
  "Resource": [
    "arn:aws:ec2:*:*:instance/*",
    "arn:aws:ec2:*:*:volume/*"
  ],
  "Condition": {
    "Null": {
      "aws:RequestTag/CostCenter": "true"
    }
  }
]
```

Not using SCPs enough

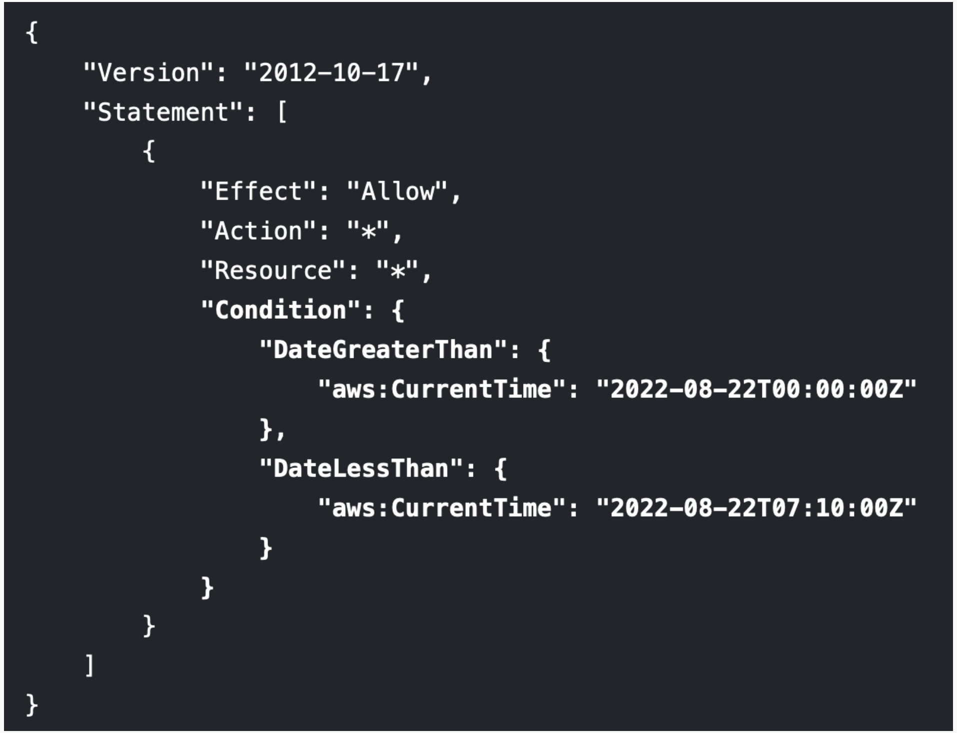
- Stop users from over-provisioning



```
{
  "Sid": "limitedSize",
  "Effect": "Deny",
  "Action": "ec2:RunInstances",
  "Resource": "arn:aws:ec2:*:*:instance/*",
  "Condition": {
    "ForAnyValue:StringNotLike": {
      "ec2:InstanceType": ["*.nano", "*.small", "*.micro", "*.medium"]
    }
  }
}
```

Leveraging IAM Condition Keys

Time Based Access

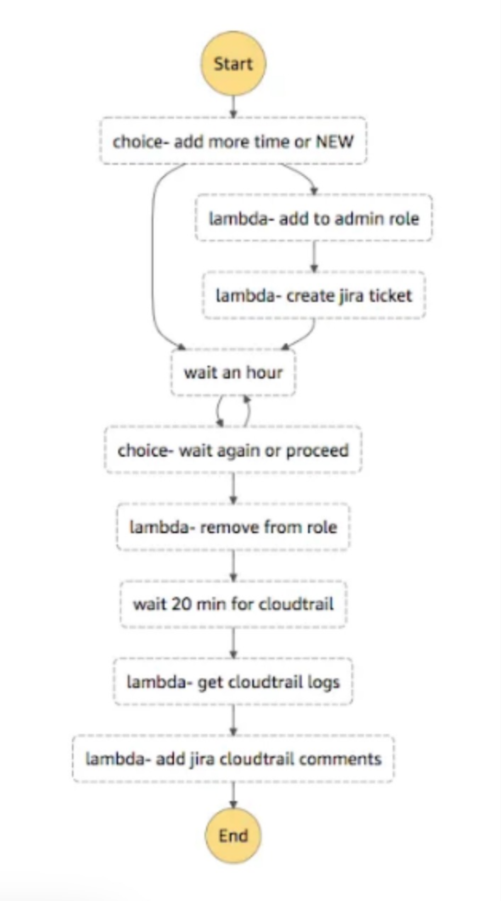


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "DateGreaterThan": {
                    "aws:CurrentTime": "2022-08-22T00:00:00Z"
                },
                "DateLessThan": {
                    "aws:CurrentTime": "2022-08-22T07:10:00Z"
                }
            }
        }
    ]
}
```

IAM Condition Keys

Permissions Workflow

StepFunctions, Lambda


Start
choice- add more time or NEW
lambda- add to admin role
lambda- create jira ticket
wait an hour
choice- wait again or proceed
lambda- remove from role
wait 20 min for cloudtrail
lambda- get cloudtrail logs
lambda- add jira cloudtrail comments
End

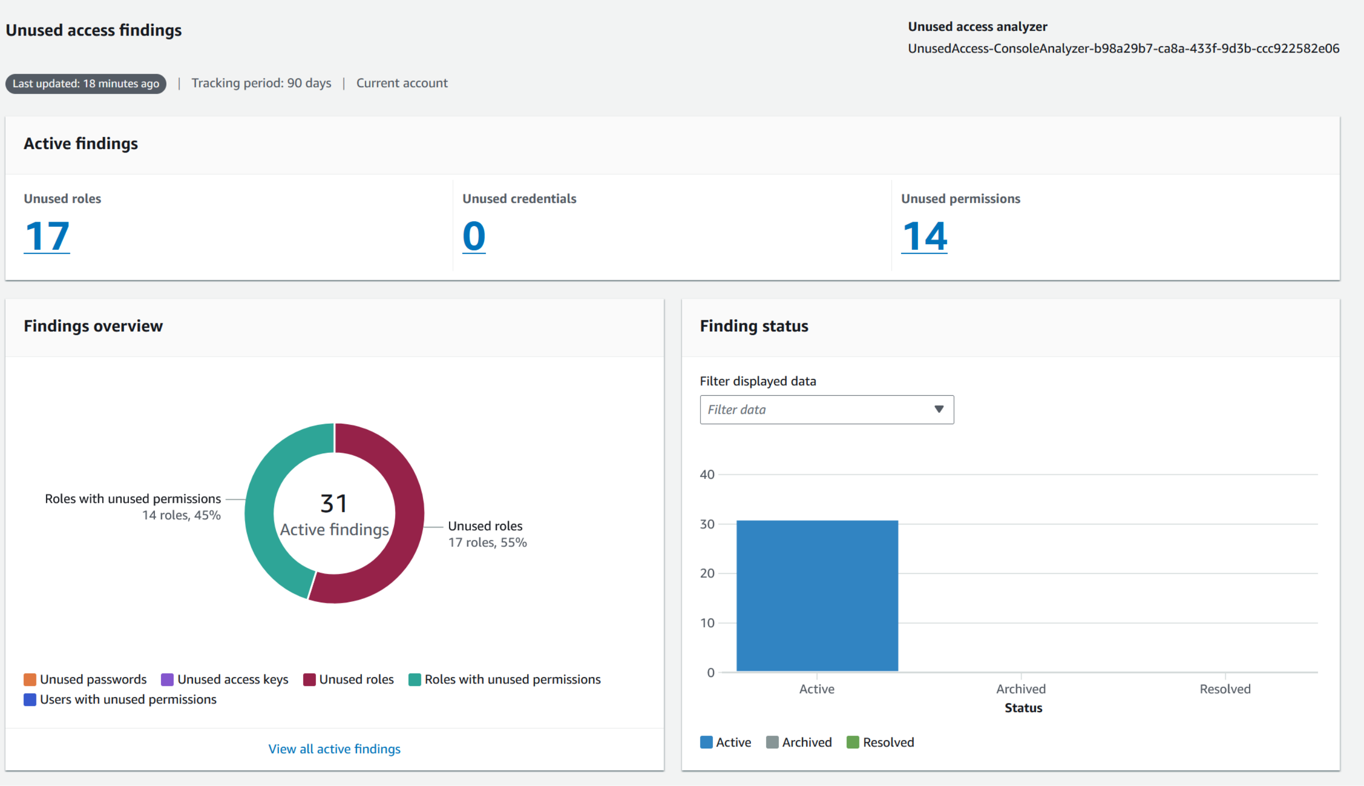
https://medium.com/rapidsos-engineering/building-a-permissions-workflow-using-aws-step-functions-cc987d20aa09

Helpful Tools

AWS IAM helpers

- IAM Access Analyzer
- Generate policy based on CloudTrail events
- Unit Test IAM (Sort of)
- Trust, but Verify

Access Analyzer



Unused access findings

Unused access analyzer
UnusedAccess-ConsoleAnalyzer-b98a29b7-ca8a-433f-9d3b-ccc922582e06

Last updated: 18 minutes ago | Tracking period: 90 days | Current account

Active findings

Unused roles	Unused credentials	Unused permissions
17	0	14

Findings overview

🟧 Unused passwords 🟪 Unused access keys 🟥 Unused roles 🟩 Roles with unused permissions
🟦 Users with unused permissions

View all active findings

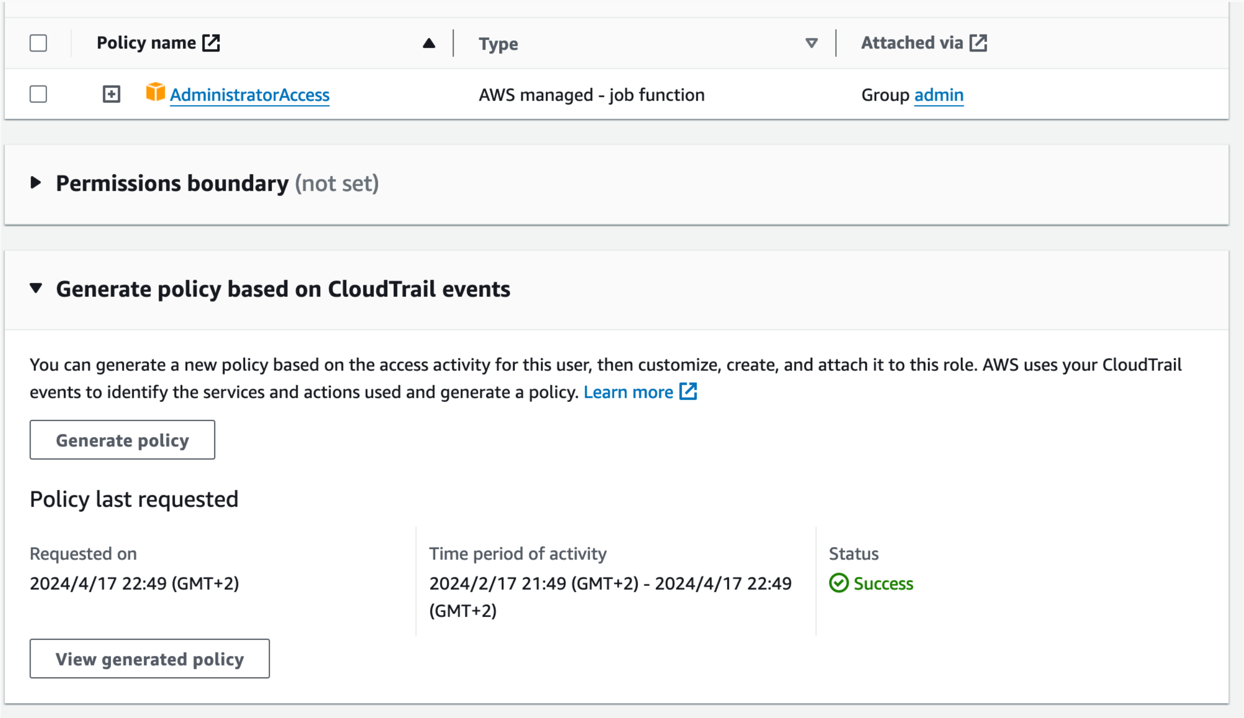
Finding status

Filter displayed data

Filter data ▼

🟦 Active ⬛ Archived 🟩 Resolved

Generate policy based on CloudTrail events



□	Policy name	Type	Attached via
□	AdministratorAccess	AWS managed - job function	Group admin

▶ **Permissions boundary** (not set)

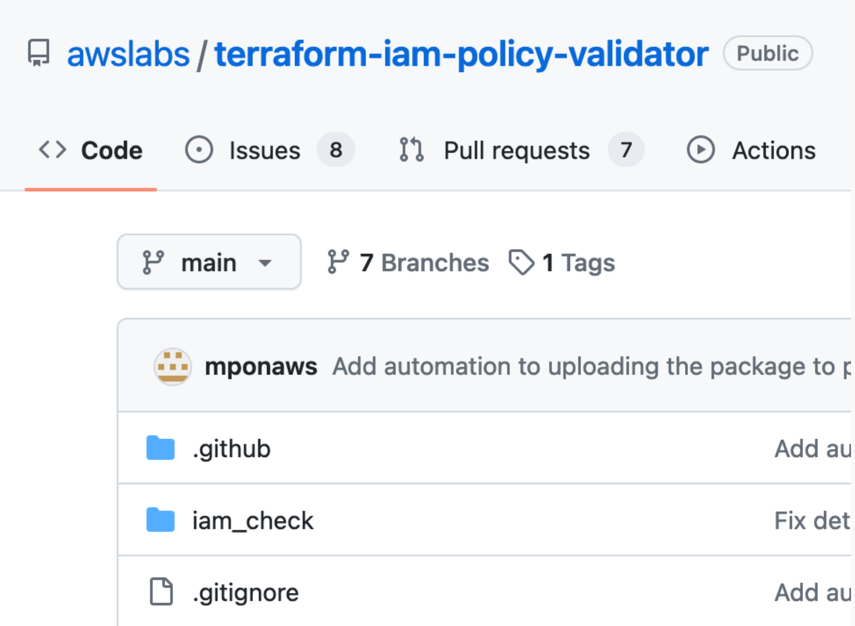
▼ **Generate policy based on CloudTrail events**

You can generate a new policy based on the access activity for this user, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. Learn more

Policy last requested

Requested on	Time period of activity	Status
2024/4/17 22:49 (GMT+2)	2024/2/17 21:49 (GMT+2) - 2024/4/17 22:49 (GMT+2)	✓ Success

Unit Test IAM



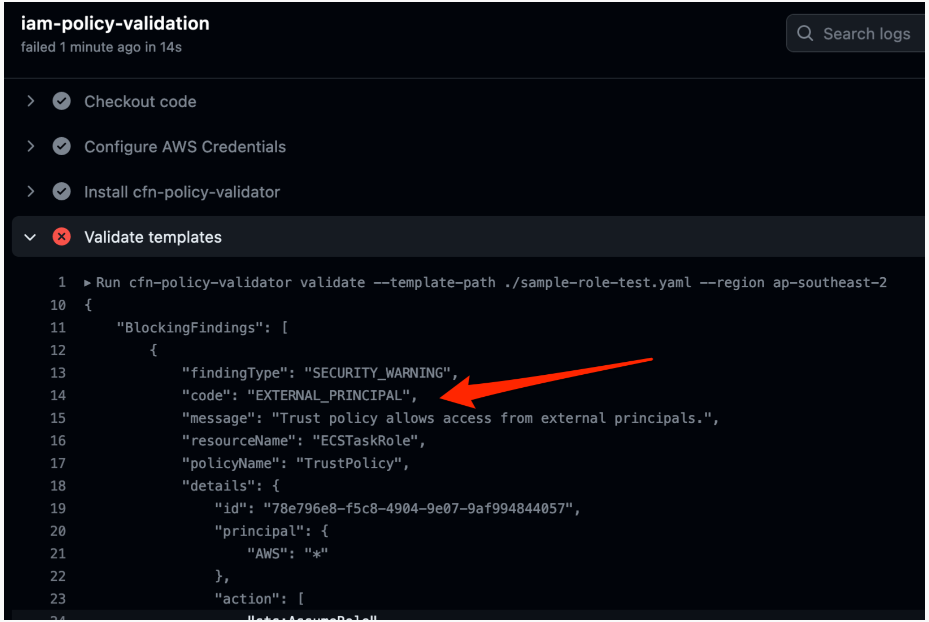
awslabs / terraform-iam-policy-validator Public

Code Issues 8 Pull requests 7 Actions

main 7 Branches 1 Tags

mponaws Add automation to uploading the package to p

.github	Add au
iam_check	Fix det
.gitignore	Add au



iam-policy-validation
failed 1 minute ago in 14s

- ✓ Checkout code
- ✓ Configure AWS Credentials
- ✓ Install cfn-policy-validator
- ✗ Validate templates

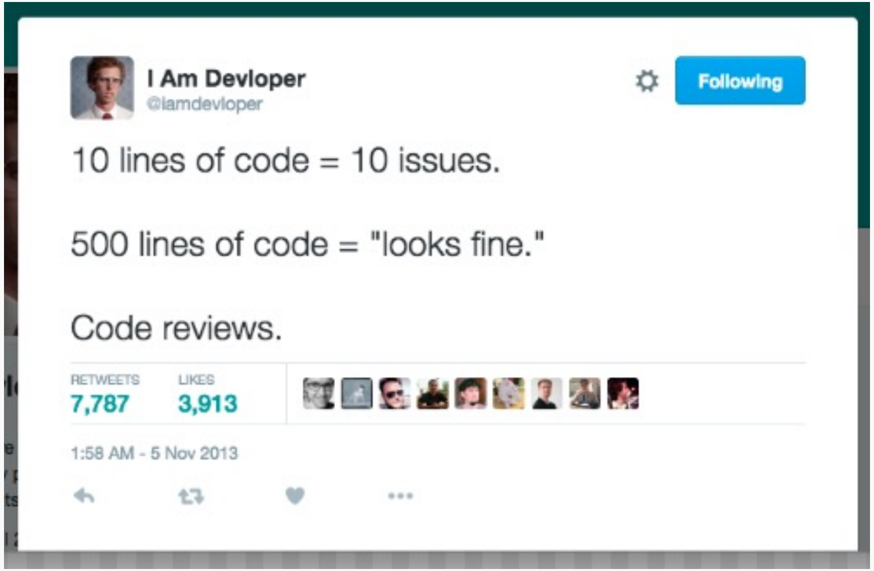
```
▸ Run cfn-policy-validator validate --template-path ./sample-role-test.yaml --region ap-southeast-2
{
    "BlockingFindings": [
        {
            "findingType": "SECURITY_WARNING",
            "code": "EXTERNAL_PRINCIPAL",
            "message": "Trust policy allows access from external principals.",
            "resourceName": "ECSTaskRole",
            "policyName": "TrustPolicy",
            "details": {
                "id": "78e796e8-f5c8-4904-9e07-9af994844057",
                "principal": {
                    "AWS": "*"
                },
                "action": [
                    "sts:AssumeRole"
```

https://github.com/awslabs/terraform-iam-policy-validator?tab=readme-ov-file

https://aws.amazon.com/blogs/security/validate-iam-policies-by-using-iam-policy-validator-for-aws-cloudformation-and-github-actions/

Trust but Verify

- Not everyone in the Team needs admin access
- Default Production access should be equal to ReadOnly
- IaC > ClickOps



I Am Devloper
@iamdevloper

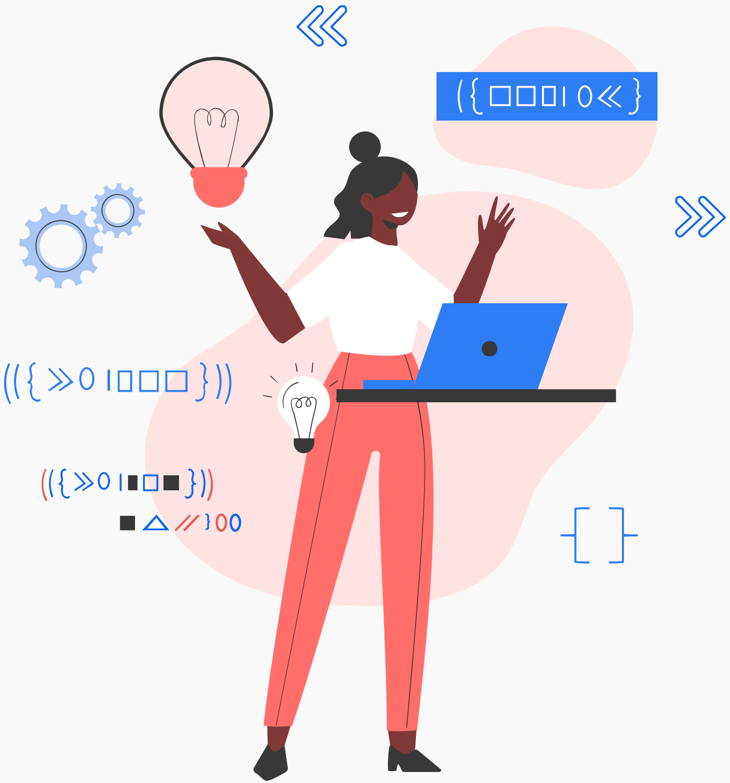
10 lines of code = 10 issues.

500 lines of code = "looks fine."

Code reviews.

RETWEETS 7,787 LIKES 3,913

1:58 AM - 5 Nov 2013



03

Demo Time

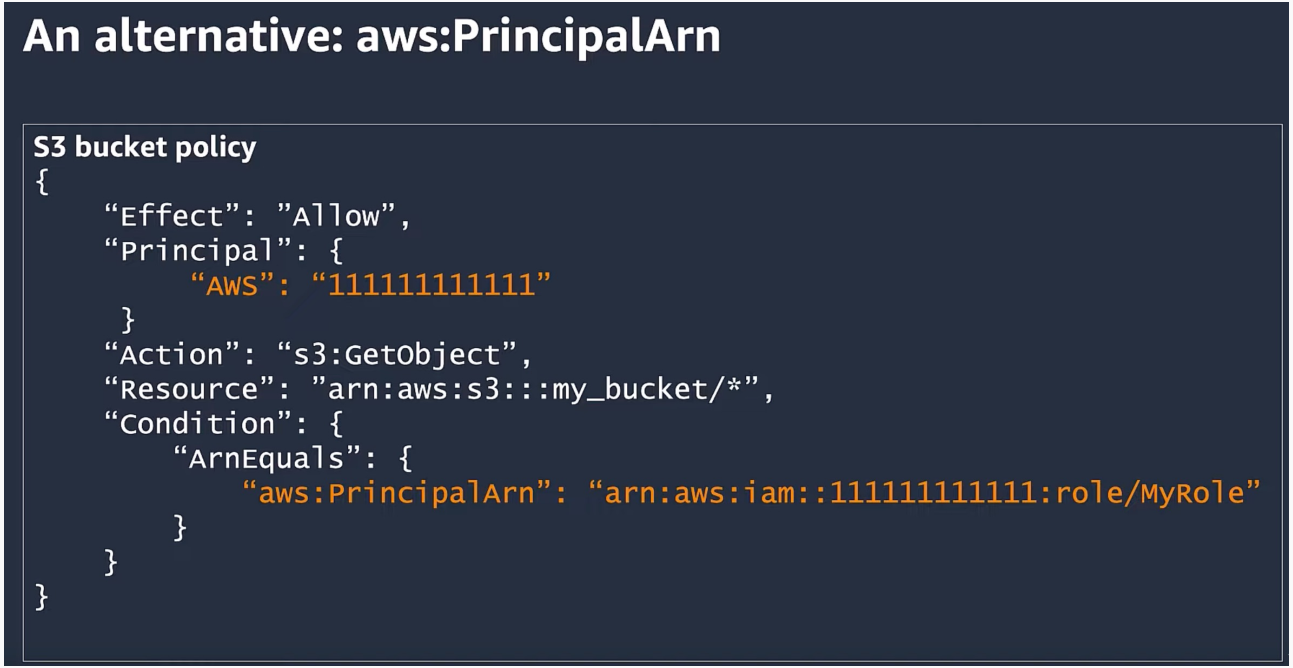
Let the Demo Gods be with Us

Demo Summary

What we’ve seen:

- IAM Unique Identifiers
- Privilege Escalation using Policy Versions
- CloudFormation IaC Generator

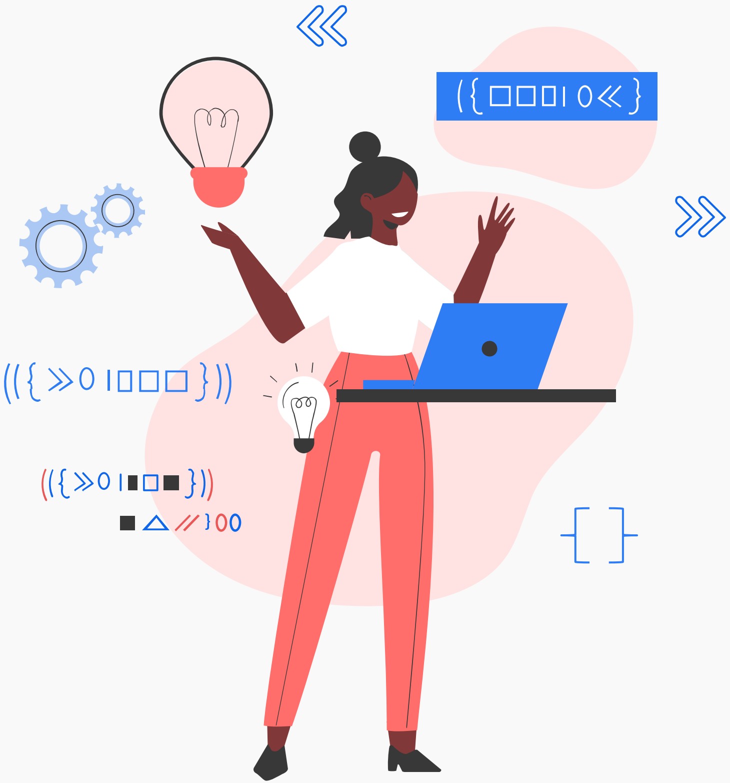
Condition Keys to the rescue



An alternative: aws:PrincipalArn

S3 bucket policy

```
{
    “Effect”: ”Allow”,
    “Principal”: {
         “AWS”: “111111111111”
     }
    “Action”: “s3:GetObject”,
    “Resource”: ”arn:aws:s3:::my_bucket/*”,
    “Condition”: {
        “ArnEquals”: {
            “aws:PrincipalArn”: “arn:aws:iam::111111111111:role/MyRole”
        }
    }
}
```



04

Summary

AWS Best & Worst Practices

Thanks!

Reach out:
MichalMikolajczyk.pl@gmail.com

Resources:
CloudNinja.pl
github.com/MichalMiko/aws_meetup

CloudNinja

AWS IAM Tips, Tricks & Links

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**