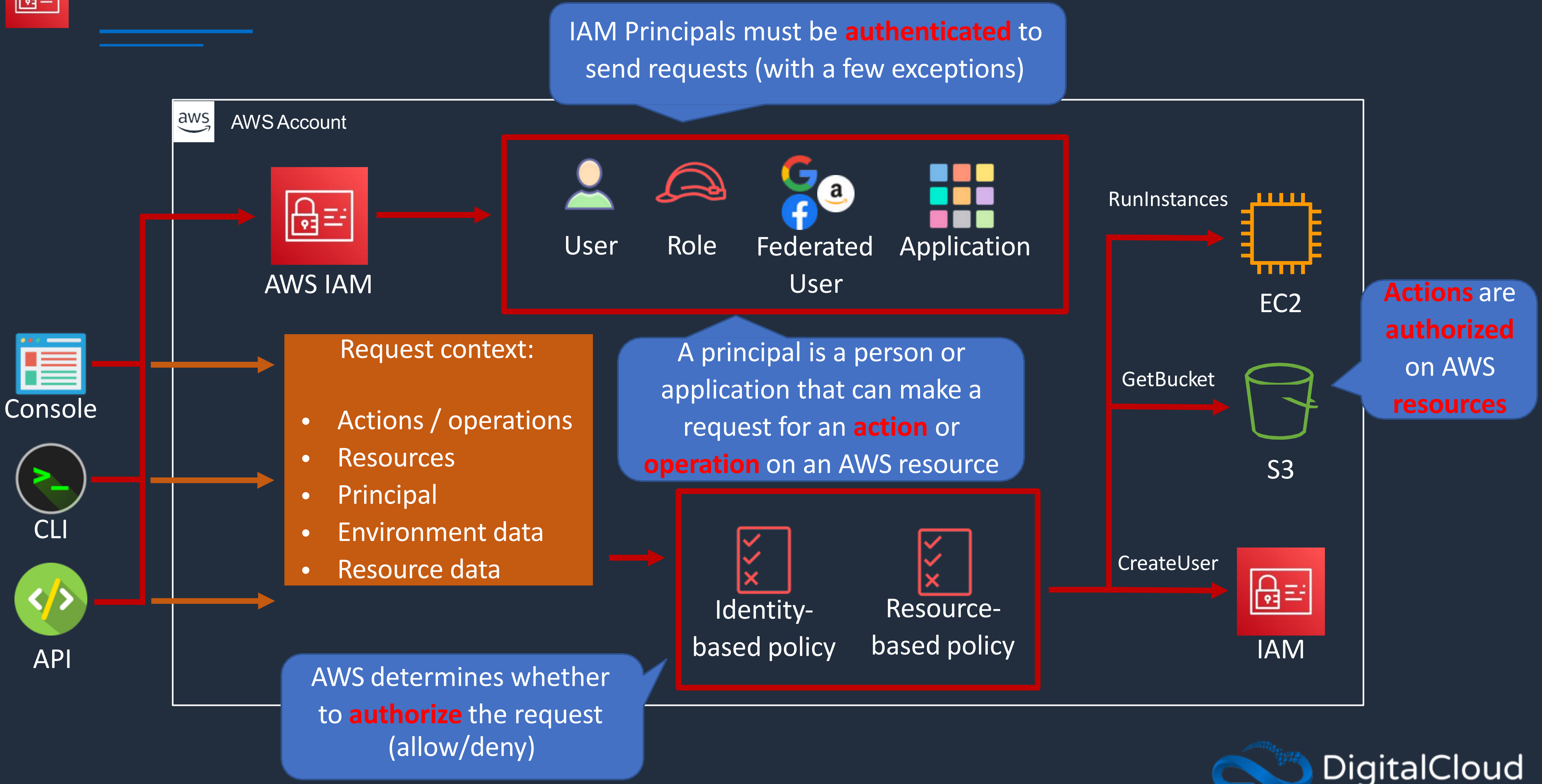
SECTION 3

Identity Management and Permissions

How IAM Works

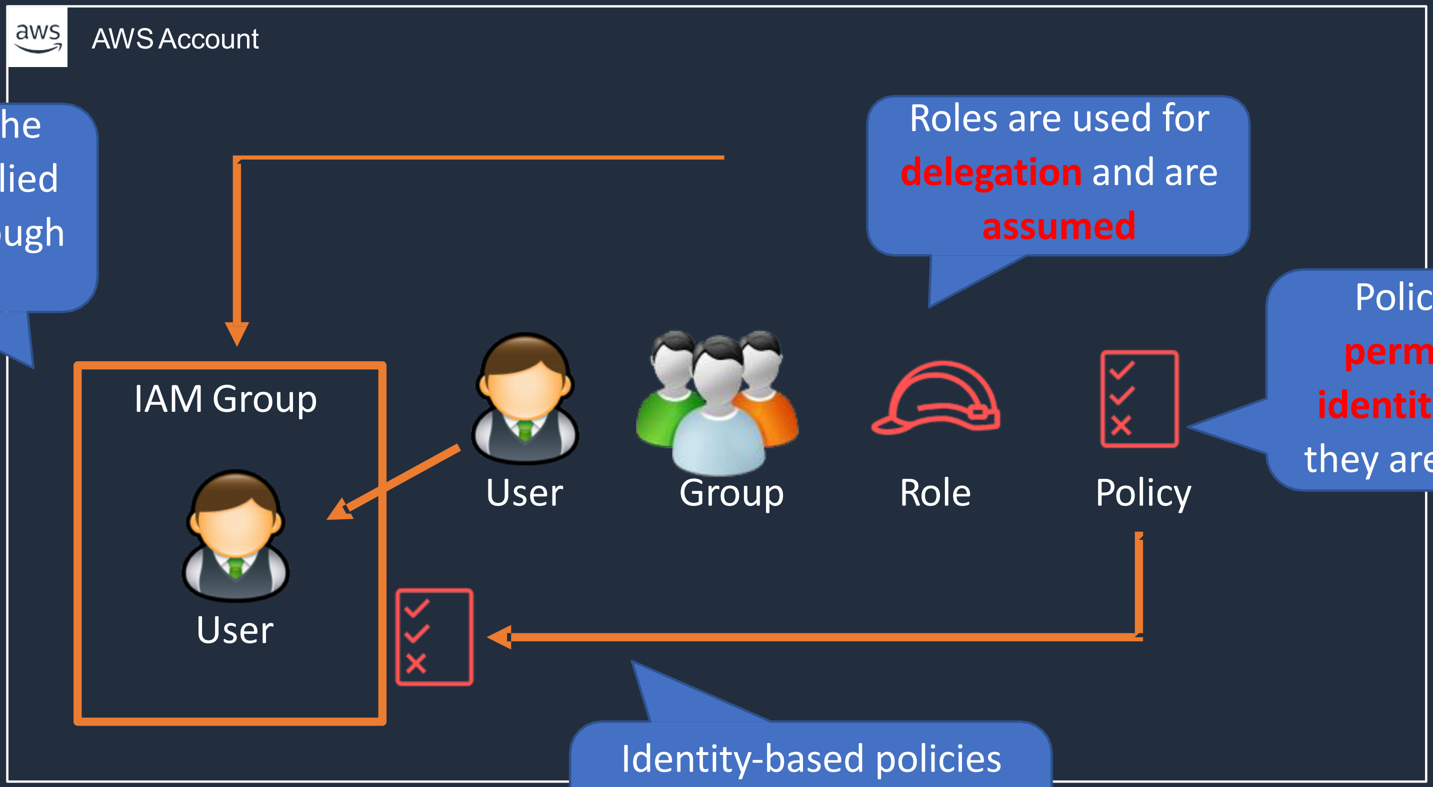
How IAM Works


IAM Principals must be authenticated to send requests (with a few exceptions)
AWS Account
AWS IAM
User
Role
Federated User
Application
Console
CLI
API
Request context:
• Actions / operations
• Resources
• Principal
• Environment data
• Resource data
A principal is a person or application that can make a request for an action or operation on an AWS resource
Identity-based policy
Resource-based policy
AWS determines whether to authorize the request (allow/deny)
RunInstances
EC2
GetBucket
S3
CreateUser
IAM
Actions are authorized on AWS resources

Overview of Users, Groups, Roles and Policies

DigitalCloud TRAINING

Users, Groups, Roles and Policies



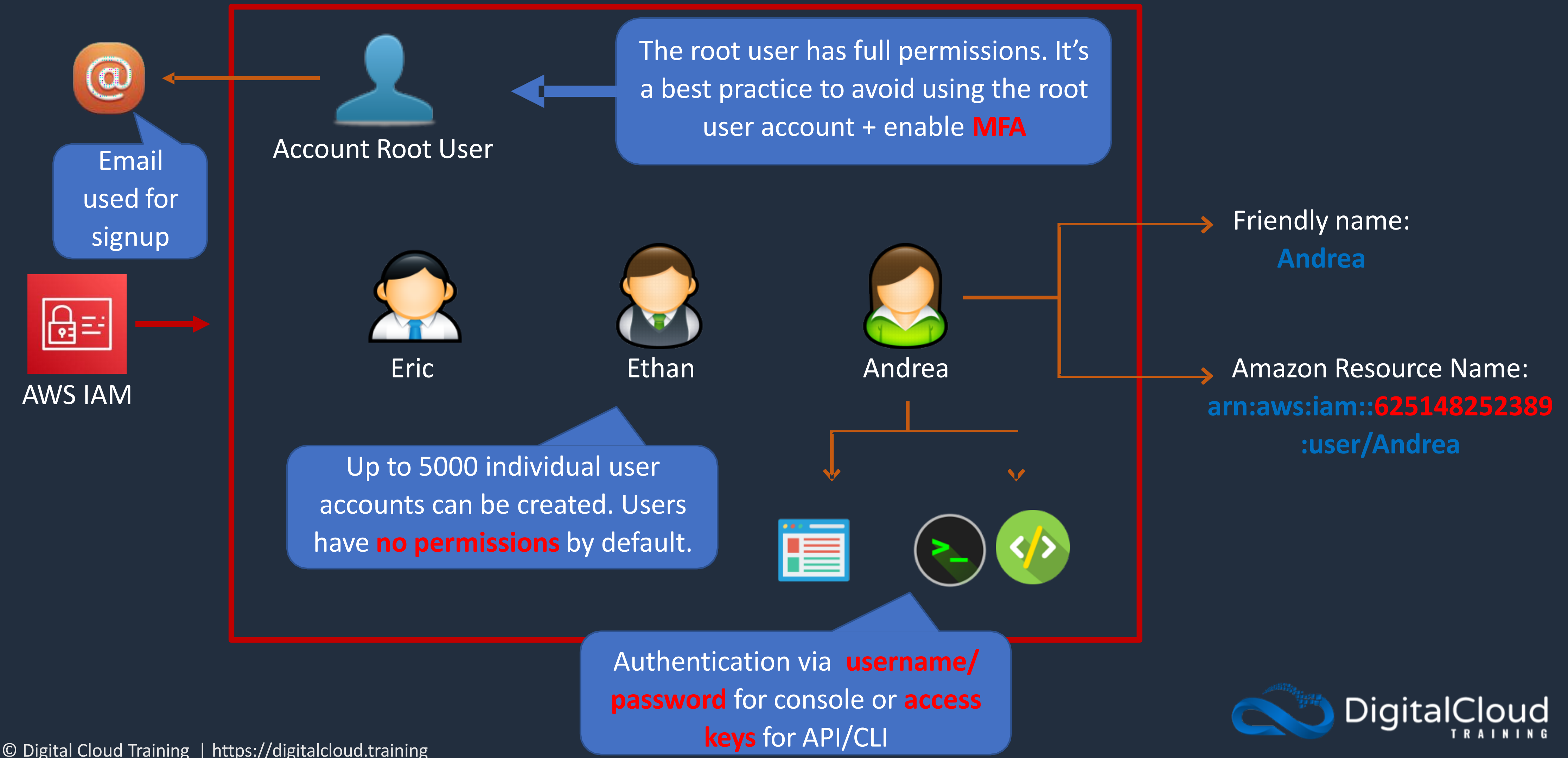
The user gains the **permissions** applied to the **group** through the **policy**

Roles are used for **delegation** and are **assumed**

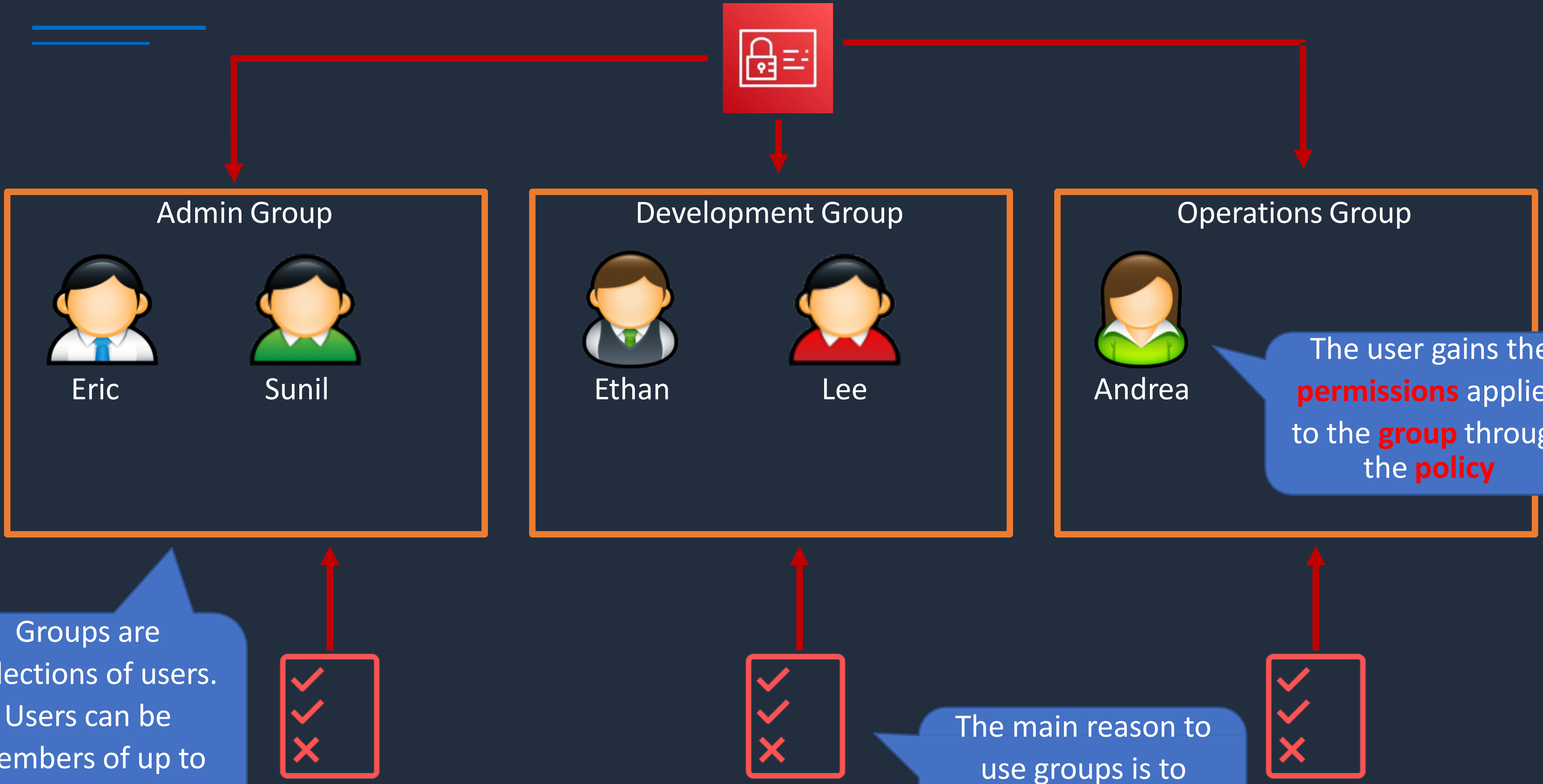
Policies define the **permissions** for the **identities** or **resources** they are associated with

Identity-based policies can be applied to users, groups, and roles

IAM Users


Email used for signup
Account Root User
The root user has full permissions. It's a best practice to avoid using the root user account + enable MFA
AWS IAM
Eric
Ethan
Andrea
Friendly name: Andrea
Amazon Resource Name: arn:aws:iam::625148252389:user/Andrea
Up to 5000 individual user accounts can be created. Users have no permissions by default.
Authentication via username/password for console or access keys for API/CLI

IAM Groups



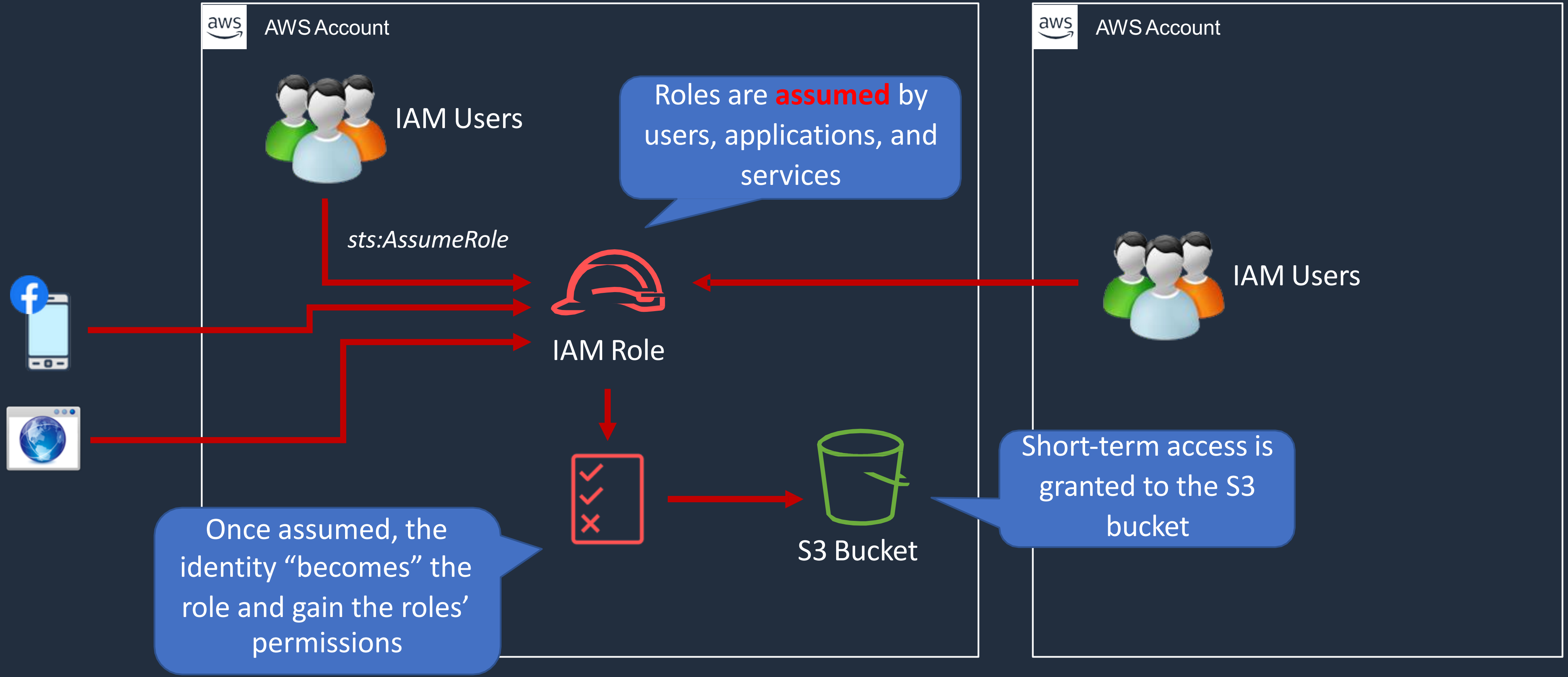
The user gains the **permissions** applied to the **group** through the **policy**

Groups are collections of users. Users can be members of up to 10 groups

The main reason to use groups is to apply **permissions** to users using **policies**

IAM Roles

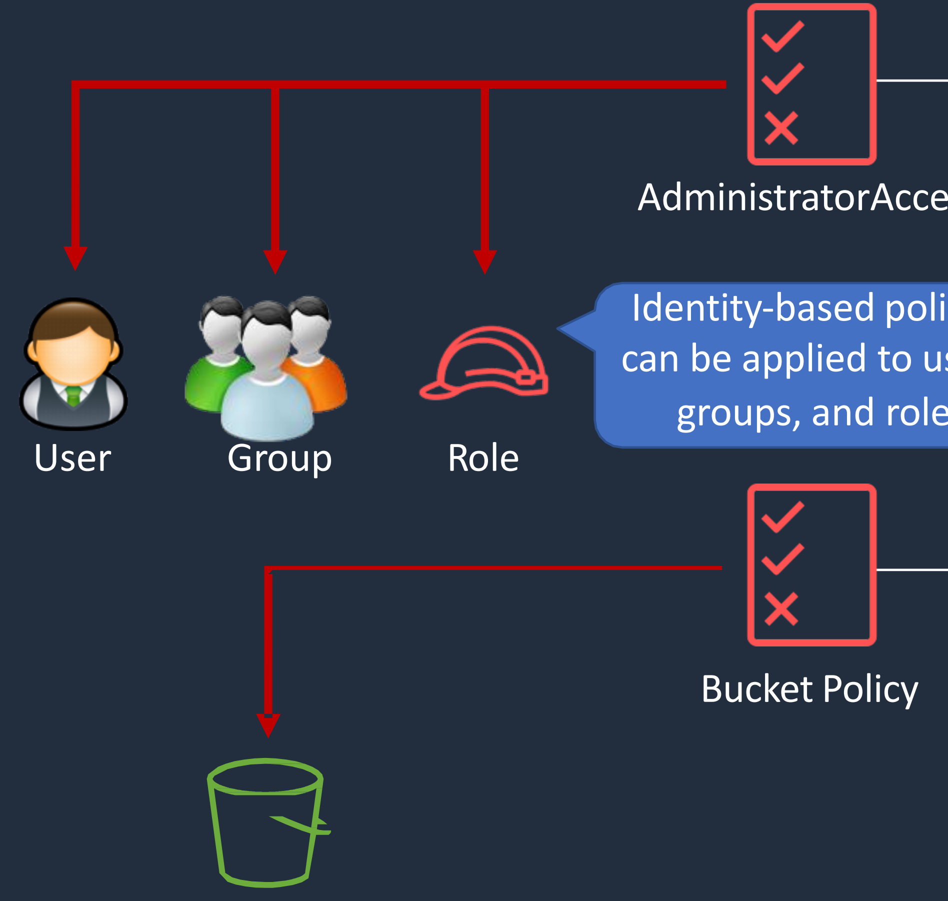
An IAM role is an IAM identity that that has specific permissions



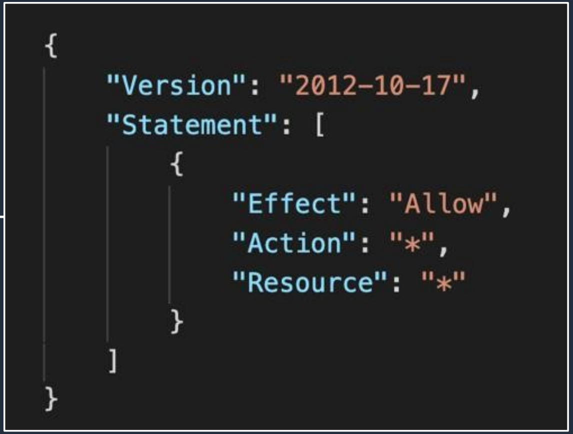
AWS Account
IAM Users
Roles are assumed by users, applications, and services
sts:AssumeRole
IAM Role
AWS Account
IAM Users
S3 Bucket
Short-term access is granted to the S3 bucket
Once assumed, the identity "becomes" the role and gain the roles' permissions

IAM Policies

Policies are documents that define permissions and are written in JSON

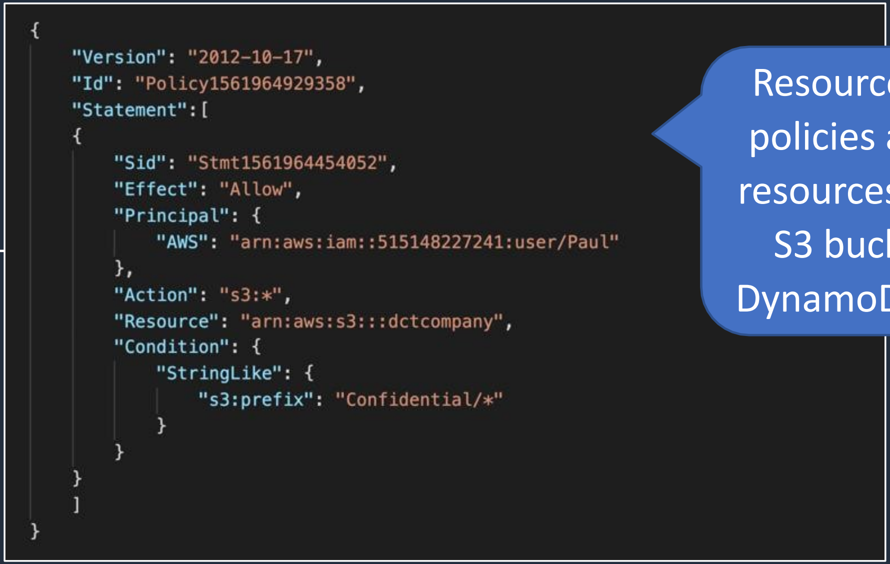


Identity-based policies can be applied to users, groups, and roles



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

All permissions are implicitly denied by default

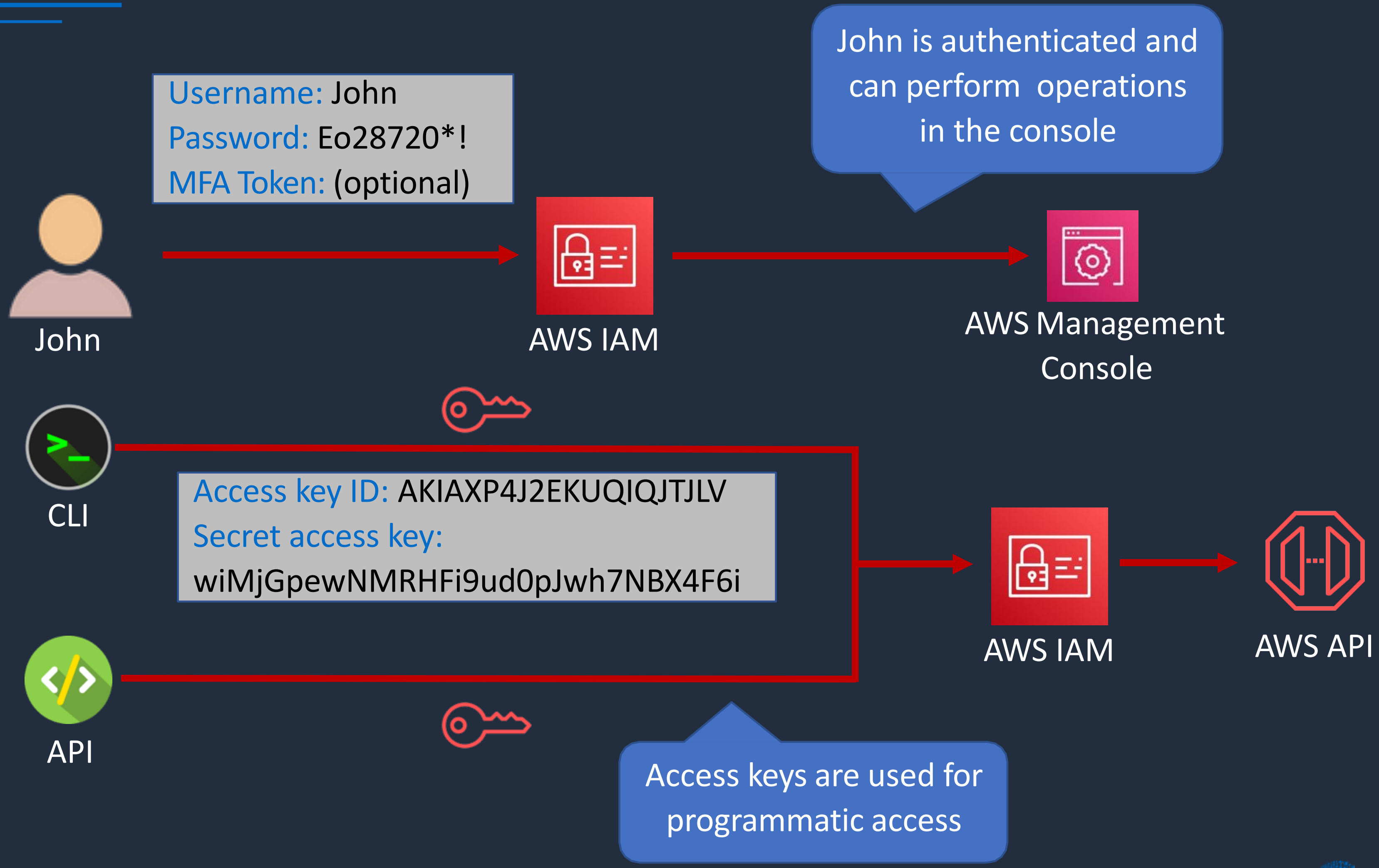


```
{
    "Version": "2012-10-17",
    "Id": "Policy1561964929358",
    "Statement":[
    {
        "Sid": "Stmt1561964454052",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::515148227241:user/Paul"
        },
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::dctcompany",
        "Condition": {
            "StringLike": {
                "s3:prefix": "Confidential/*"
            }
        }
    }
    ]
}
```

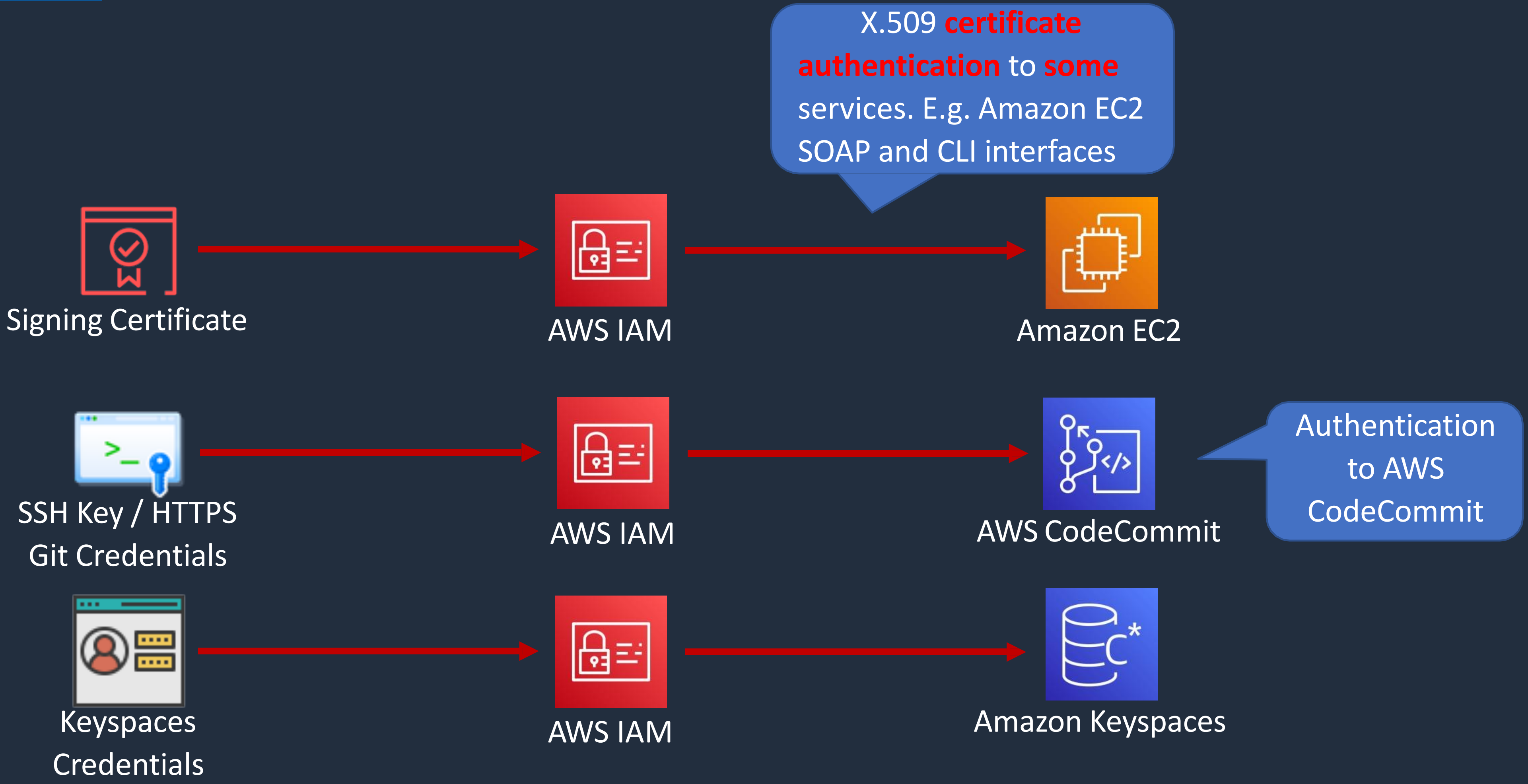
Resource-based policies apply to resources such as S3 buckets or DynamoDB tables

IAM Authentication Methods

IAM Authentication Methods


Username: John
Password: Eo28720*!
MFA Token: (optional)
John is authenticated and can perform operations in the console
John
AWS IAM
AWS Management Console
CLI
Access key ID: AKIAXP4J2EKUQIQJTJLV
Secret access key: wiMjGpewNMRHFi9ud0pJwh7NBX4F6i
AWS IAM
AWS API
API
Access keys are used for programmatic access

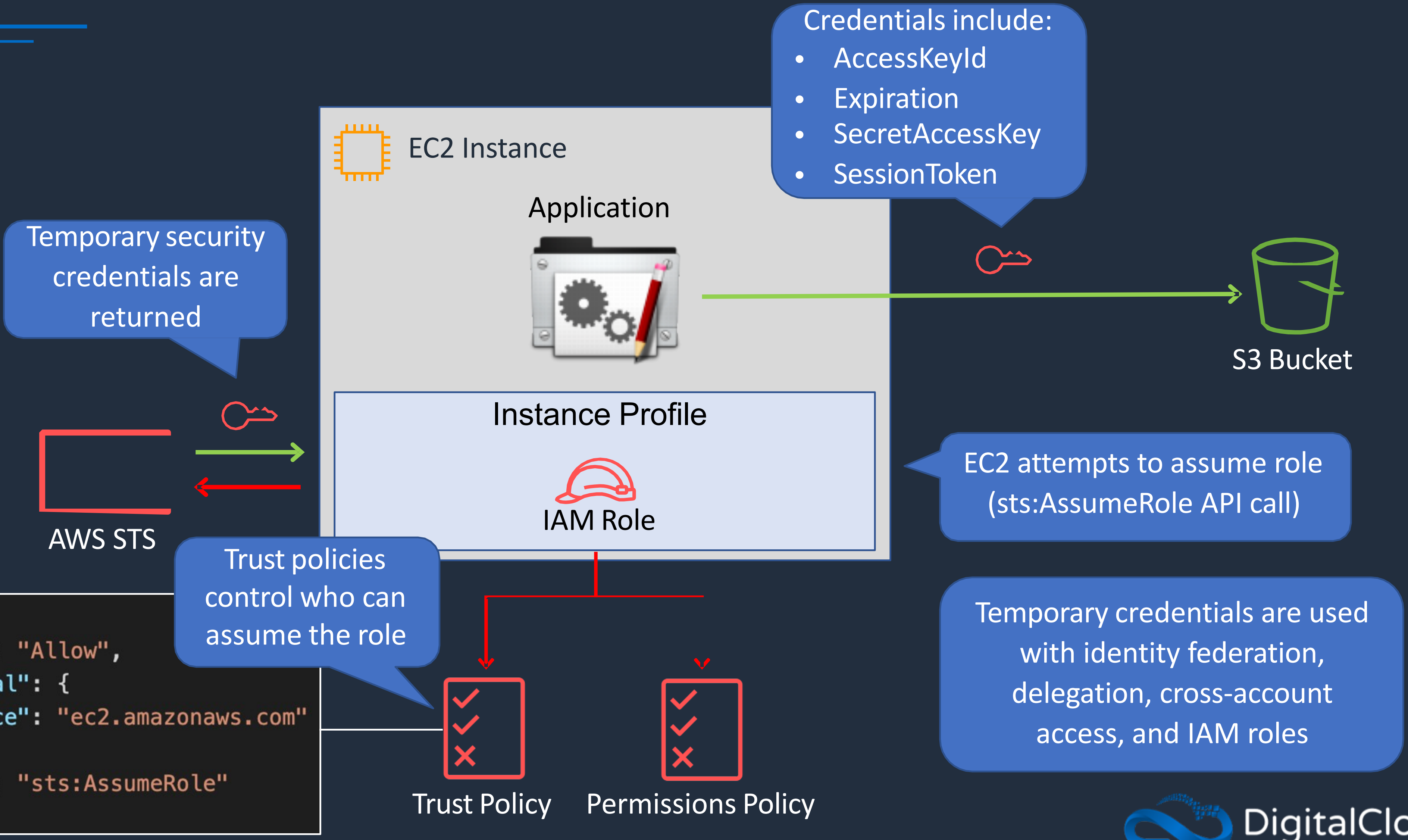
IAM Authentication Methods



X.509 certificate authentication to some services. E.g. Amazon EC2 SOAP and CLI interfaces
Signing Certificate
AWS IAM
Amazon EC2
SSH Key / HTTPS Git Credentials
AWS IAM
AWS CodeCommit
Authentication to AWS CodeCommit
Keyspaces Credentials
AWS IAM
Amazon Keyspaces

AWS Security Token Service (STS)

AWS Security Token Service (STS)


Credentials include:
AccessKeyId
Expiration
SecretAccessKey
SessionToken
EC2 Instance
Application
S3 Bucket
Temporary security credentials are returned
Instance Profile
IAM Role
AWS STS
EC2 attempts to assume role (sts:AssumeRole API call)
Trust policies control who can assume the role
Temporary credentials are used with identity federation, delegation, cross-account access, and IAM roles
Trust Policy
Permissions Policy

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "ec2.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
```

Multi-Factor Authentication (MFA)

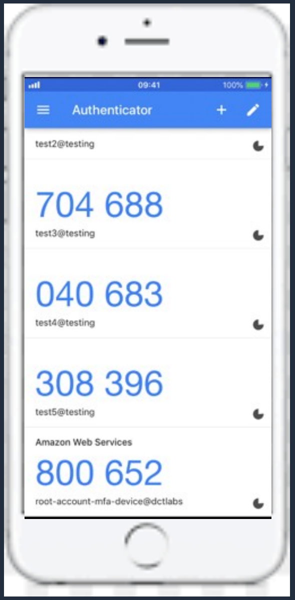
`1

Something you know:

EJPx!*21p9%

Password

Something you have:



09:41

100%

Authenticator

test2@testing

704 688

test3@testing

040 683

test4@testing

308 396

test5@testing

Amazon Web Services

800 652

root-account-mfa-device@dctlabs

Something you are:

Multi-Factor Authentication

Something you know:

IAM User

EJPx!*21p9%

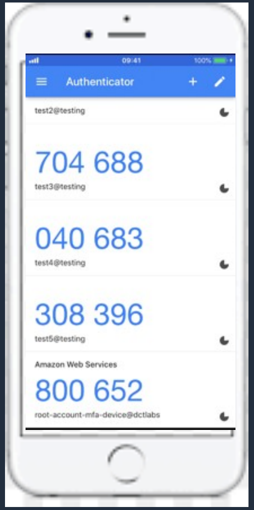
Password

Something you have:

MFA Virtual MFA

e.g. Google Authenticator on your smart phone

MFA Physical MFA



Setup Multi-Factor Authentication (MFA)

DigitalCloud TRAINING

Identity-Based Policies and Resource-Based Policies

DigitalCloud TRAINING

Identity-Based IAM Policies

Identity-based policies are JSON permissions policy documents that control what actions an identity can perform, on which resources, and under what conditions

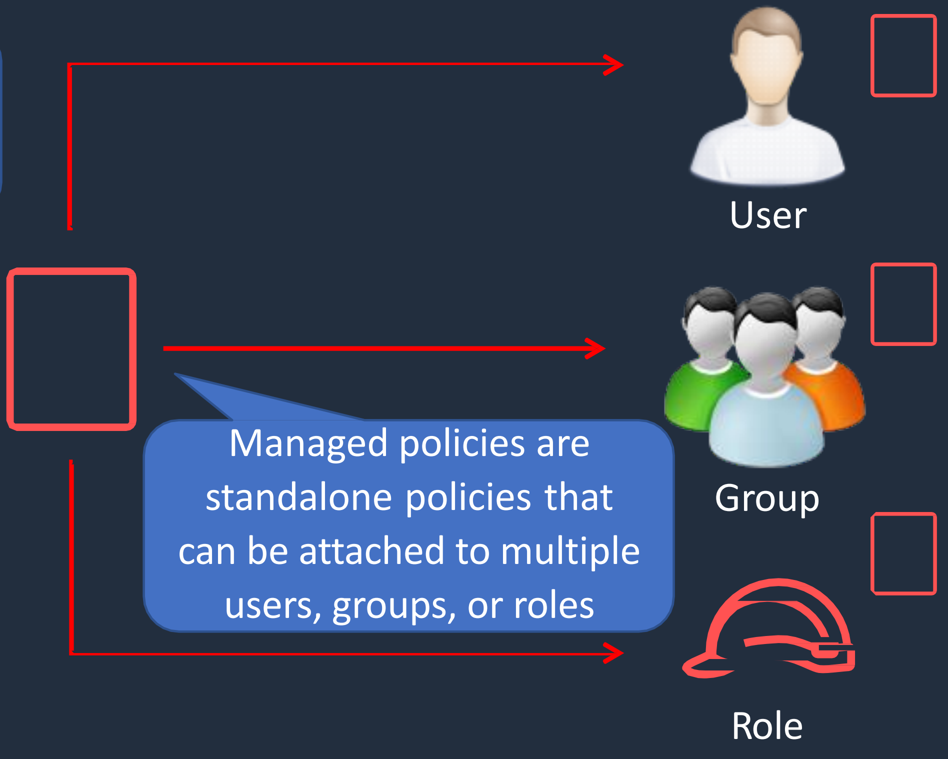
Managed policy. Either AWS managed or customer managed

AWS managed are created and managed by AWS; customer managed are created and managed by you

Managed policies are standalone policies that can be attached to multiple users, groups, or roles

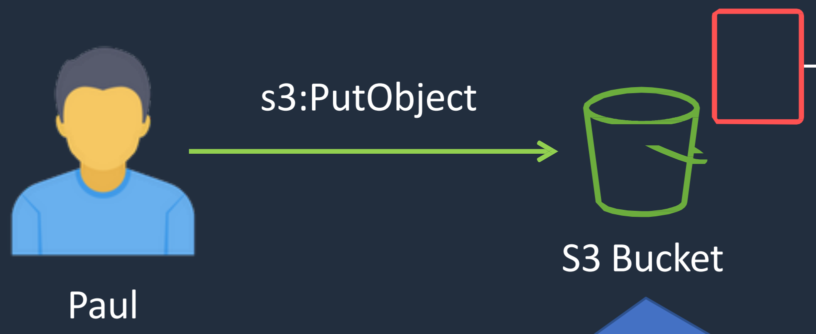
Inline policy

Inline policies have a 1-1 relationship with the user, group, or role

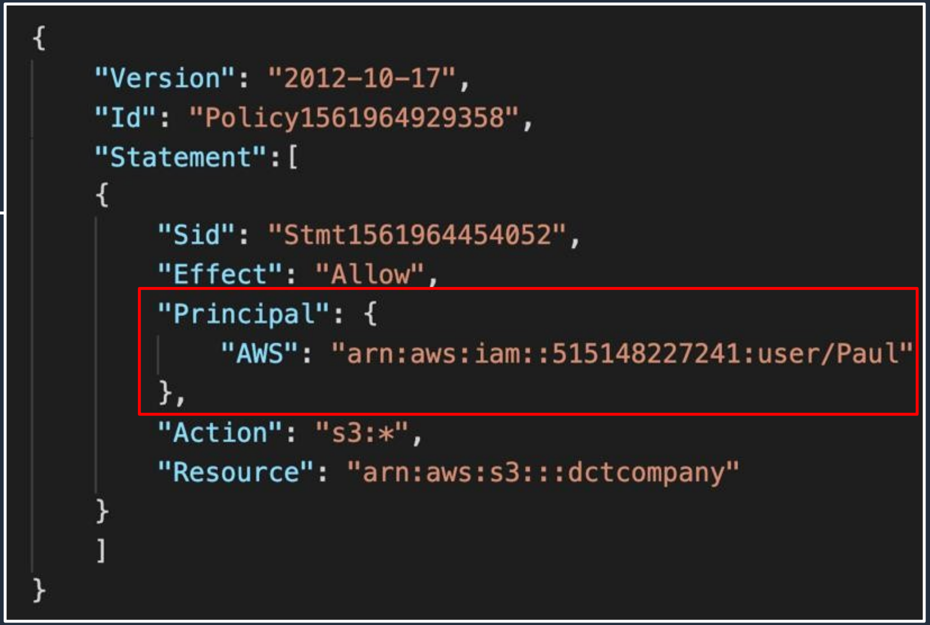


Resource-Based Policies

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

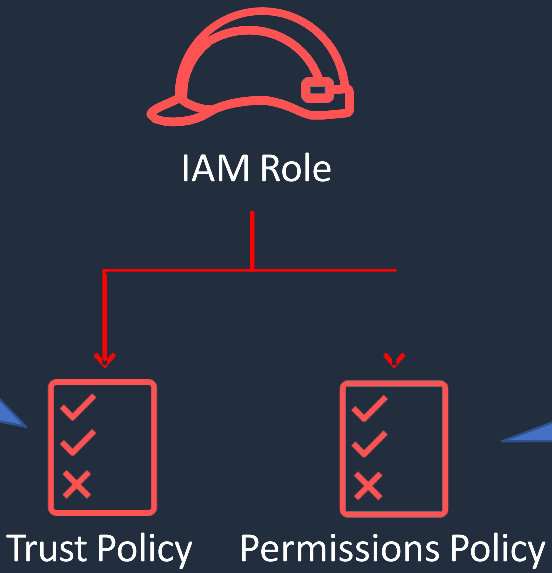


Resource-based policies grant the specified **principal** (Paul) **permission** to perform specific **actions** on the **resource**



```
{
    "Version": "2012-10-17",
    "Id": "Policy1561964929358",
    "Statement":[
    {
        "Sid": "Stmt1561964454052",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::515148227241:user/Paul"
        },
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::dctcompany"
    }
    ]
}
```

Resource-Based Policies

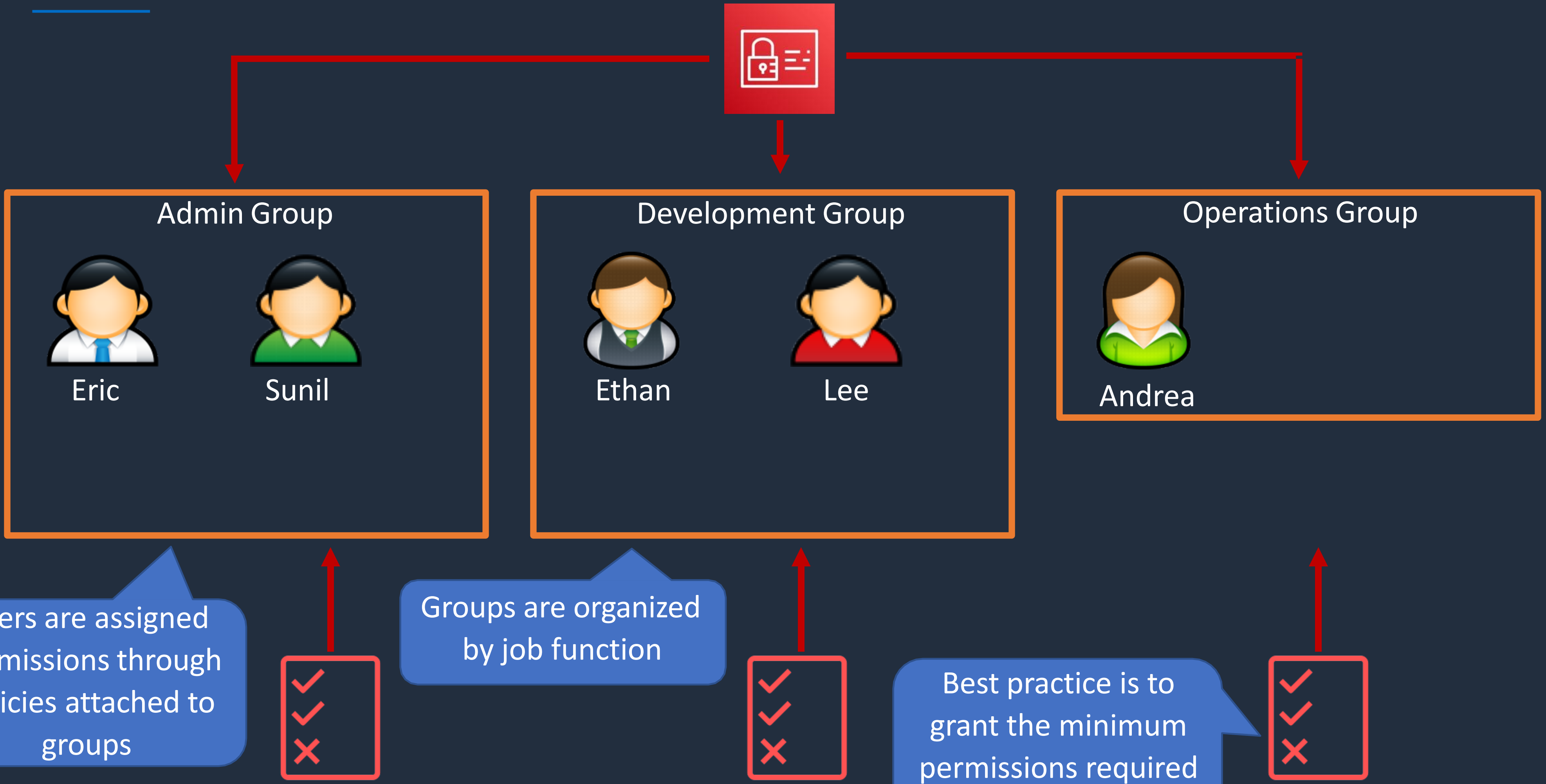


A trust policy is also an example of a **resource-based** policy

Whereas a permissions policy is an **identity-based** policy

Access Control Methods - RBAC & ABAC

Role-Based Access Control (RBAC)



Users are assigned permissions through policies attached to groups

Groups are organized by job function

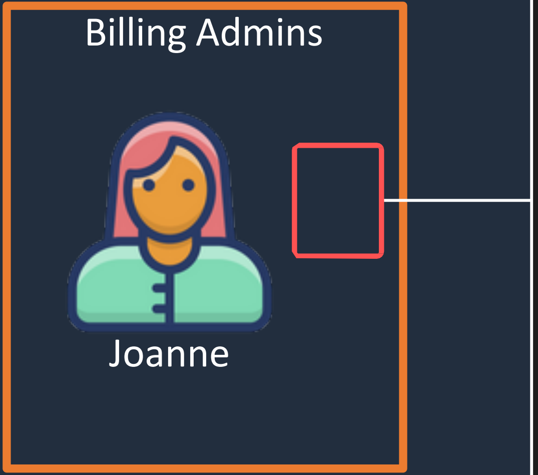
Best practice is to grant the minimum permissions required to perform the job

Role-Based Access Control (RBAC)

Job function policies:

- Administrator
- Billing
- Database administrator
- Data scientist
- Developer power user
- Network administrator
- Security auditor
- Support user
- System administrator
- View-only user

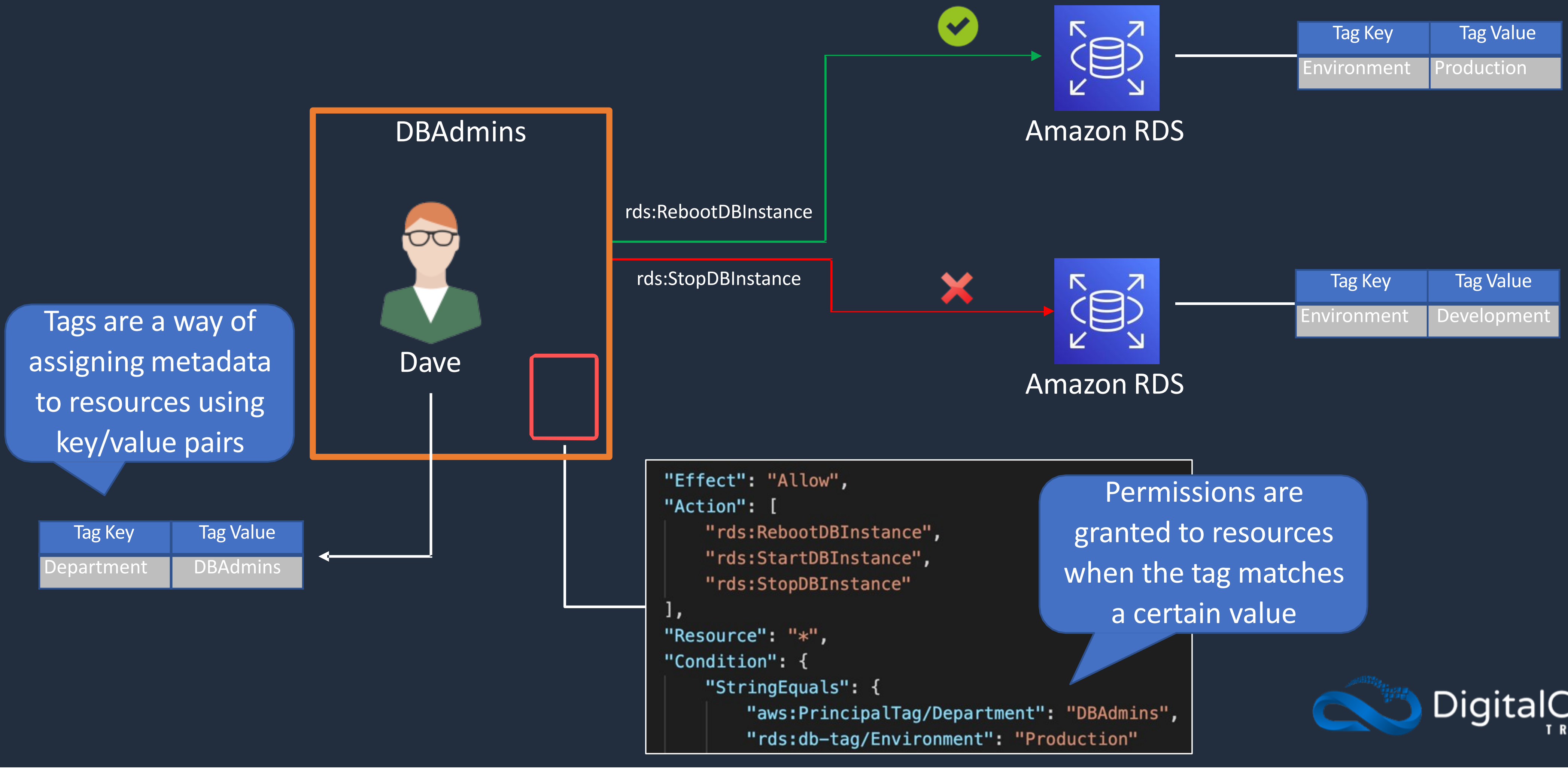
The Billing managed policy is attached to the group



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aws-portal:*Billing",
                "aws-portal:*Usage",
                "aws-portal:*PaymentMethods",
                "budgets:ViewBudget",
                "budgets:ModifyBudget",
                "ce:UpdatePreferences",
                "ce:CreateReport",
                "ce:UpdateReport",
                "ce:DeleteReport",
                "ce:CreateNotificationSubscription",
                "ce:UpdateNotificationSubscription",
                "ce:DeleteNotificationSubscription",
                "cur:DescribeReportDefinitions",
                "cur:PutReportDefinition",
                "cur:ModifyReportDefinition",
                "cur:DeleteReportDefinition",
                "purchase-orders:*PurchaseOrders"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS managed policies for job functions are designed to closely align to common job functions in the IT industry

Attribute-Based Access Control (ABAC)



Tag Key	Tag Value
Environment	Production

Tag Key	Tag Value
Environment	Development

Tags are a way of assigning metadata to resources using key/value pairs

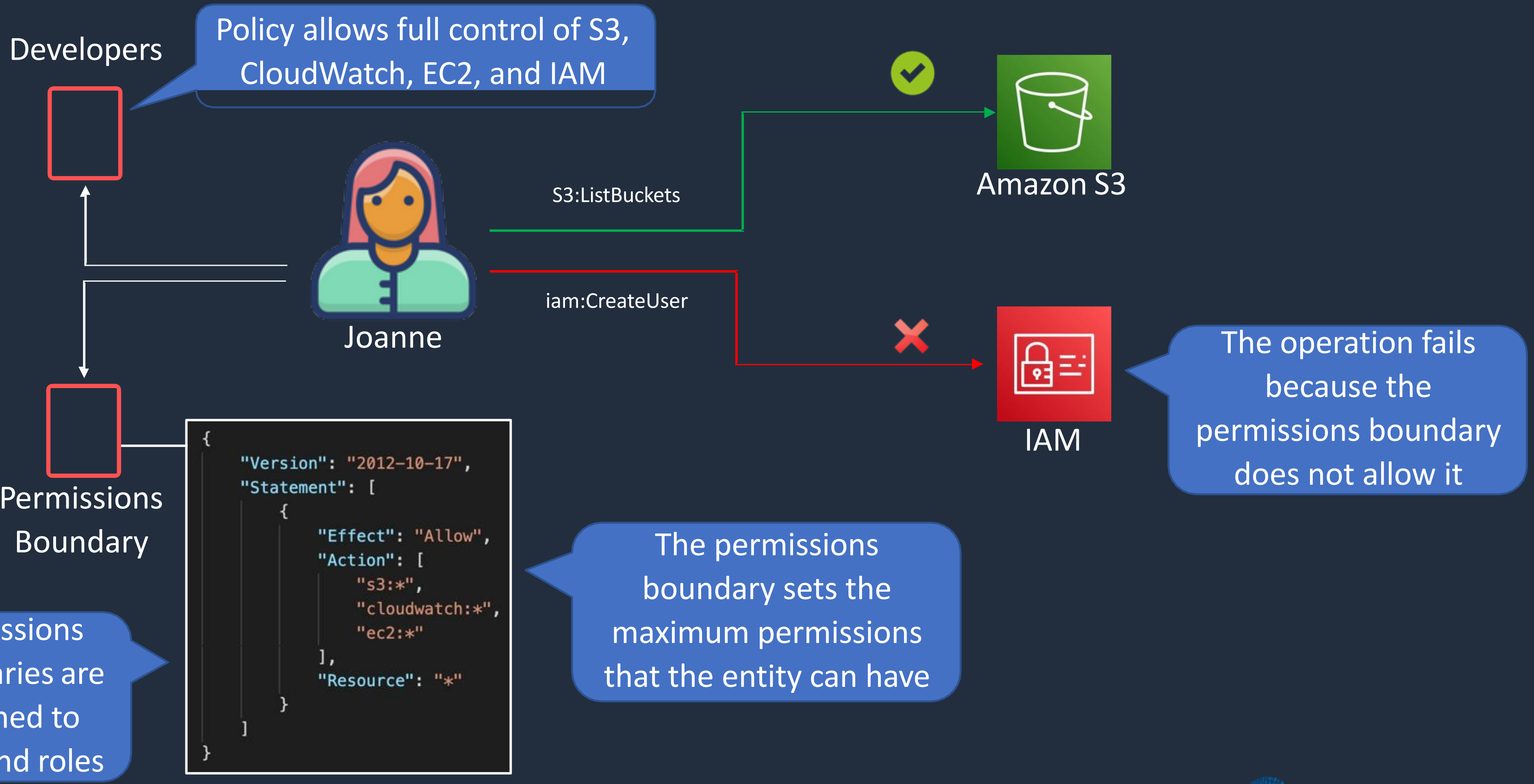
Tag Key	Tag Value
Department	DBAdmins

```
"Effect": "Allow",
"Action": [
    "rds:RebootDBInstance",
    "rds:StartDBInstance",
    "rds:StopDBInstance"
],
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:PrincipalTag/Department": "DBAdmins",
        "rds:db-tag/Environment": "Production"
```

Permissions are granted to resources when the tag matches a certain value

Permissions Boundaries

Permissions Boundaries

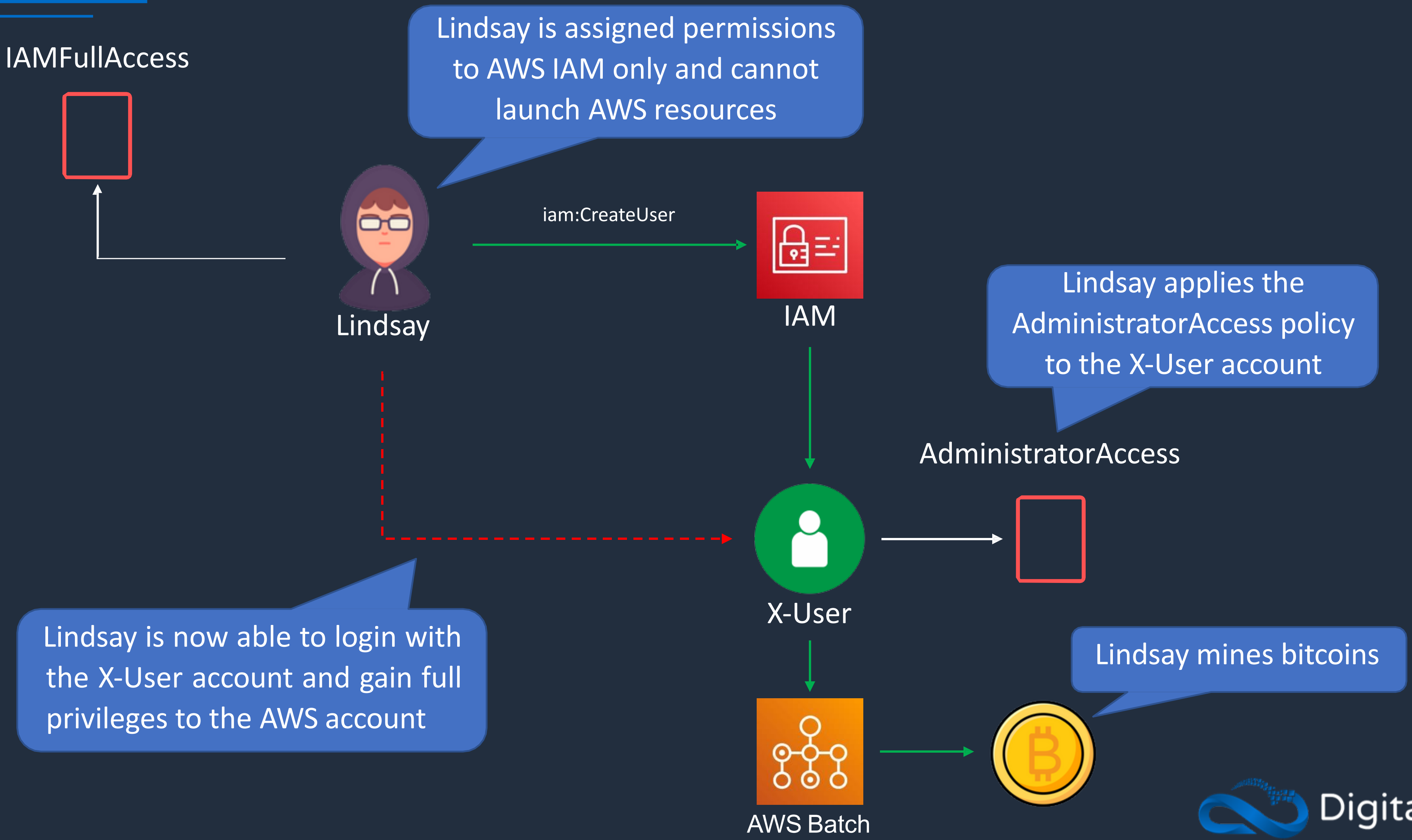


Developers
Policy allows full control of S3, CloudWatch, EC2, and IAM
Joanne
S3:ListBuckets
Amazon S3
iam:CreateUser
IAM
The operation fails because the permissions boundary does not allow it
Permissions Boundary
The permissions boundary sets the maximum permissions that the entity can have

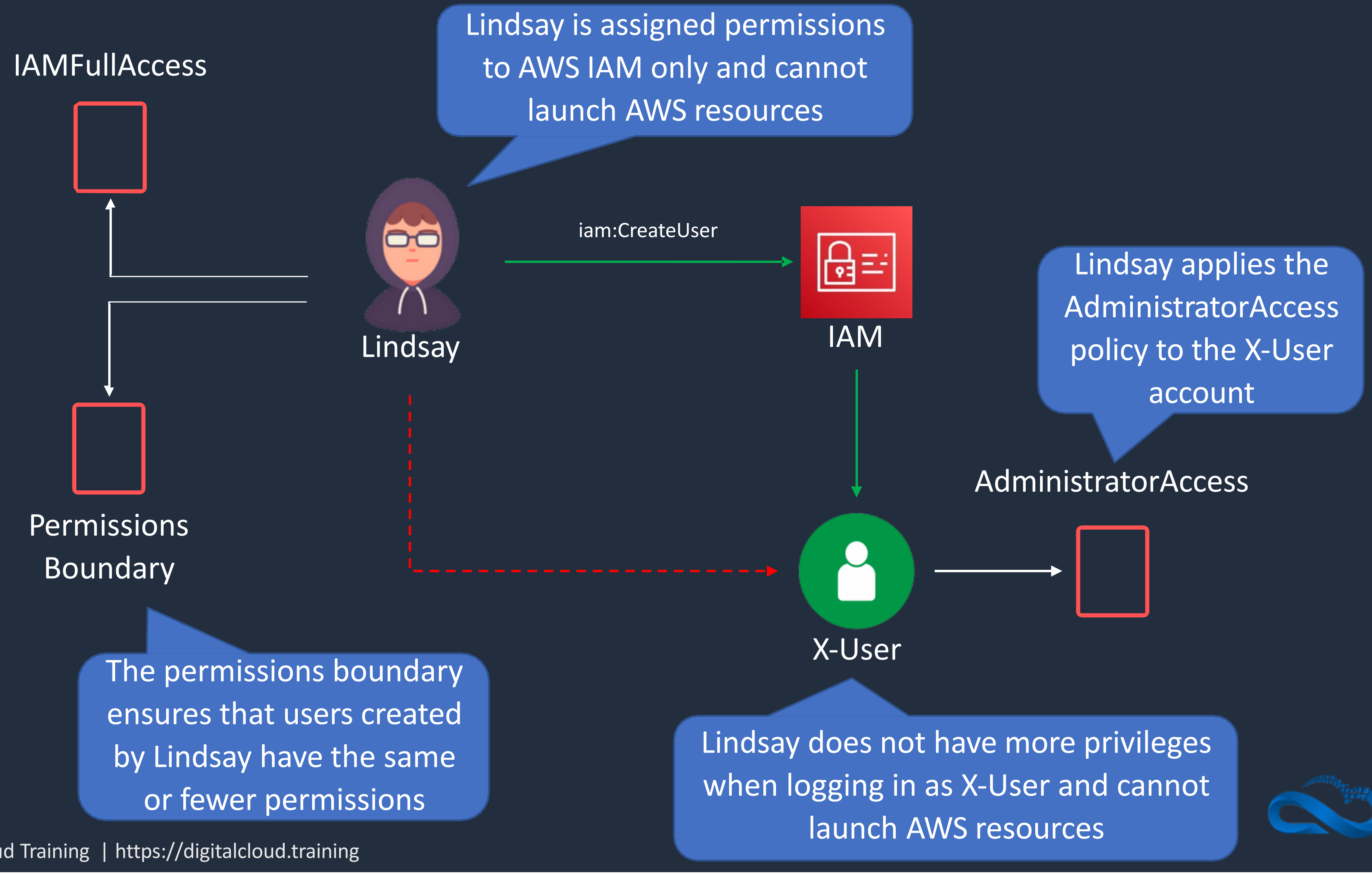
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Permissions boundaries are attached to users and roles

Privilege Escalation

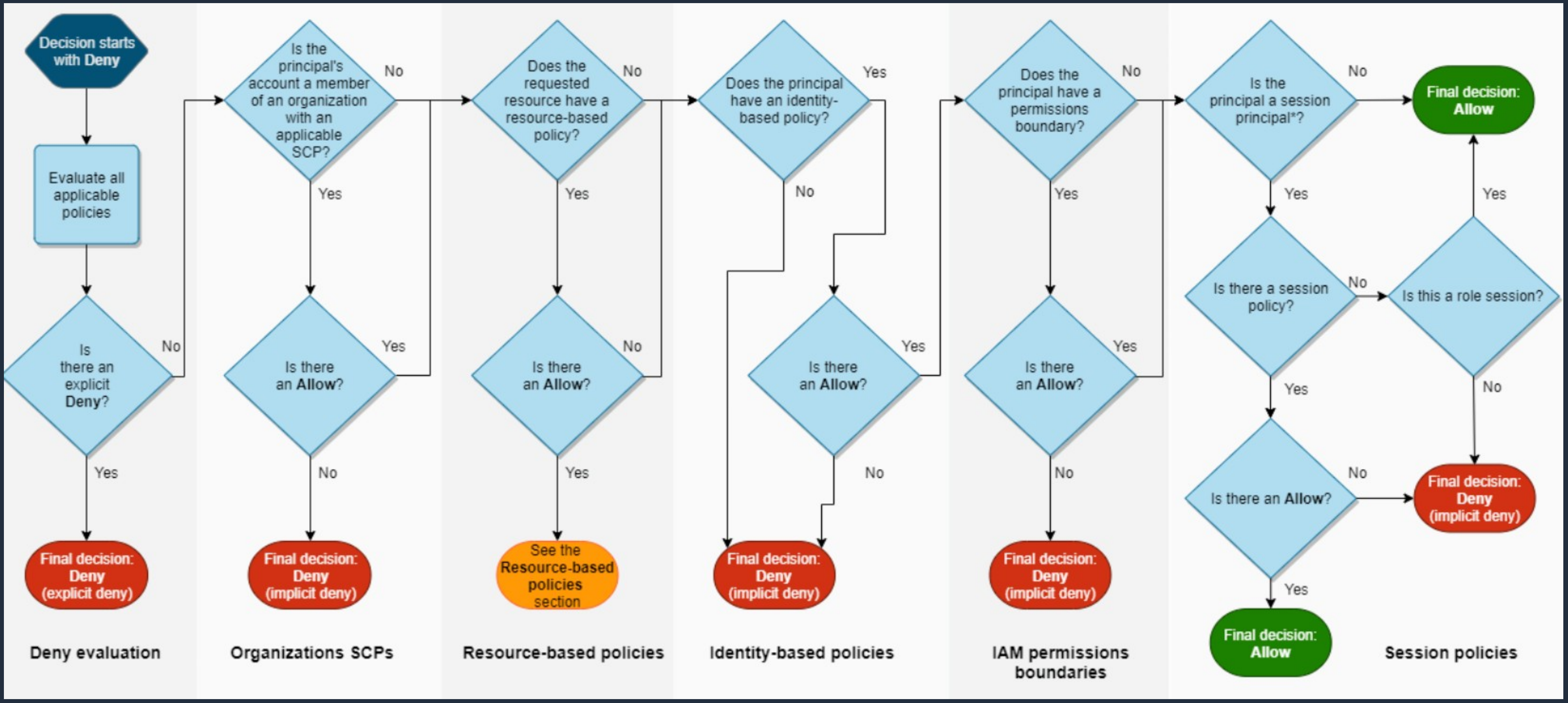

IAMFullAccess
Lindsay is assigned permissions to AWS IAM only and cannot launch AWS resources
Lindsay
iam:CreateUser
IAM
Lindsay applies the AdministratorAccess policy to the X-User account
AdministratorAccess
X-User
Lindsay is now able to login with the X-User account and gain full privileges to the AWS account
Lindsay mines bitcoins
AWS Batch

Preventing Privilege Escalation


IAMFullAccess
Lindsay is assigned permissions to AWS IAM only and cannot launch AWS resources
iam:CreateUser
Lindsay
IAM
Lindsay applies the AdministratorAccess policy to the X-User account
AdministratorAccess
Permissions Boundary
X-User
The permissions boundary ensures that users created by Lindsay have the same or fewer permissions
Lindsay does not have more privileges when logging in as X-User and cannot launch AWS resources

IAM Policy Evaluation

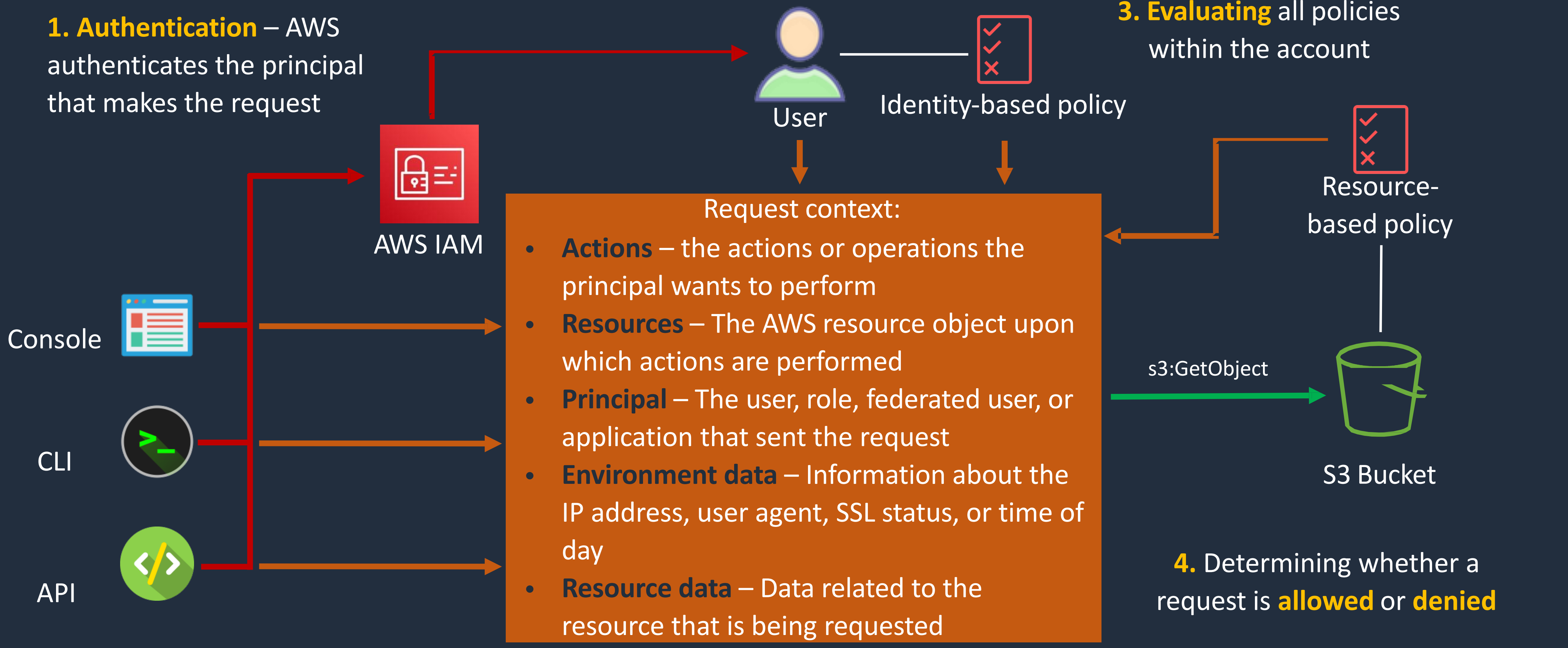
Evaluation Logic



Decision starts with Deny
Evaluate all applicable policies
Is there an explicit Deny?
No
Yes
Final decision: Deny (explicit deny)
Is the principal's account a member of an organization with an applicable SCP?
No
Yes
Is there an Allow?
Yes
No
Final decision: Deny (implicit deny)
Does the requested resource have a resource-based policy?
No
Yes
Is there an Allow?
No
Yes
See the Resource-based policies section
Does the principal have an identity-based policy?
Yes
No
Is there an Allow?
Yes
No
Final decision: Deny (implicit deny)
Does the principal have a permissions boundary?
No
Yes
Is there an Allow?
Yes
No
Final decision: Deny (implicit deny)
Is the principal a session principal*?
No
Yes
Final decision: Allow
Is there a session policy?
No
Yes
Is this a role session?
Yes
No
Is there an Allow?
No
Yes
Final decision: Deny (implicit deny)
Final decision: Allow
Deny evaluation
Organizations SCPs
Resource-based policies
Identity-based policies
IAM permissions boundaries
Session policies

Steps for Authorizing Requests to AWS

1. Authentication – AWS authenticates the principal that makes the request



Request context:

- **Actions** – the actions or operations the principal wants to perform
- **Resources** – The AWS resource object upon which actions are performed
- **Principal** – The user, role, federated user, or application that sent the request
- **Environment data** – Information about the IP address, user agent, SSL status, or time of day
- **Resource data** – Data related to the resource that is being requested

2. Processing the request context

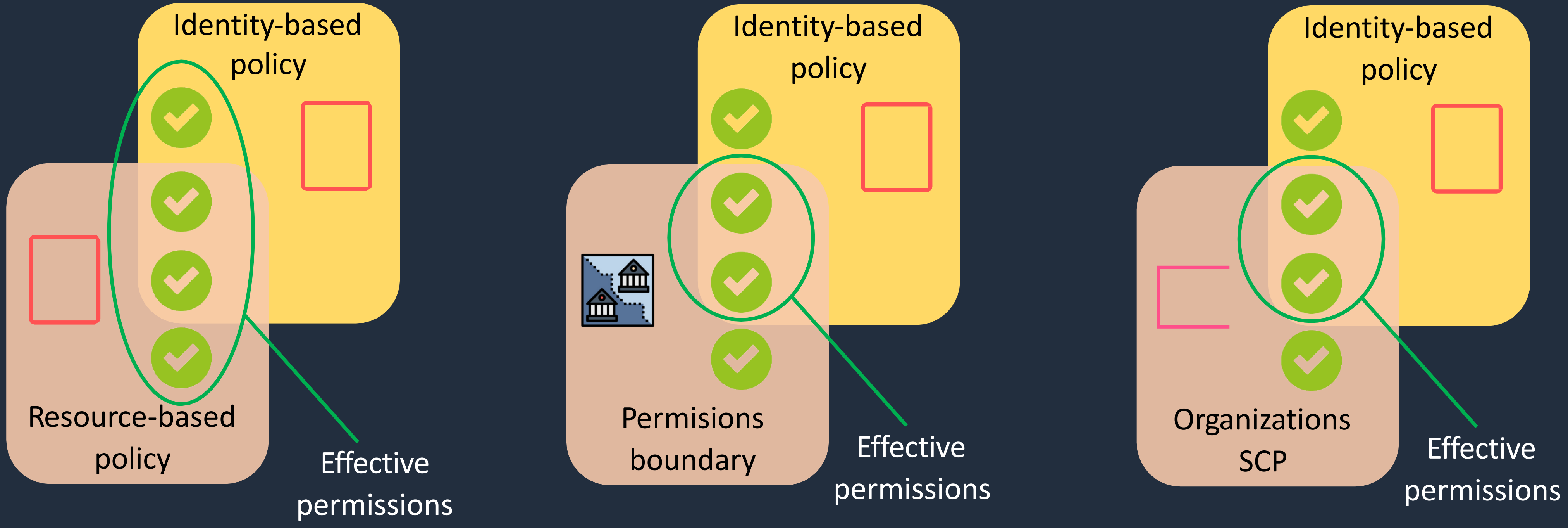
3. Evaluating all policies within the account

4. Determining whether a request is **allowed** or **denied**

Types of Policy

- **Identity-based policies** – attached to users, groups, or roles
- **Resource-based policies** – attached to a resource; define permissions for a principal accessing the resource
- **IAM permissions boundaries** – set the maximum permissions an identity-based policy can grant an IAM entity
- **AWS Organizations service control policies (SCP)** – specify the maximum permissions for an organization or OU
- **Session policies** – used with AssumeRole* API actions

Evaluating Policies within an AWS Account



Determination Rules

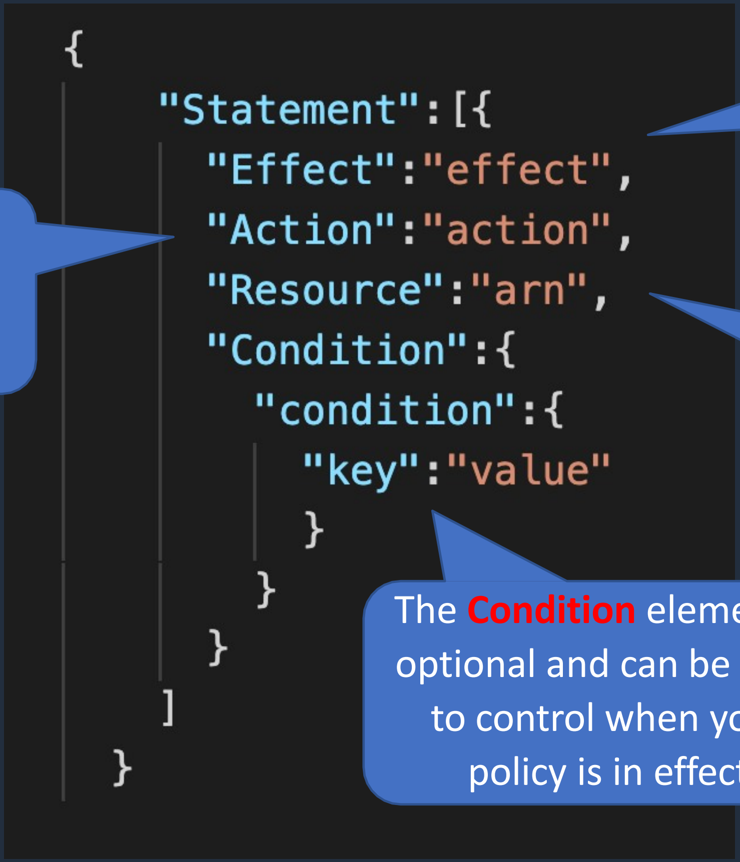
1. By default, all requests are implicitly denied (though the root user has full access)
2. An explicit allow in an identity-based or resource-based policy overrides this default
3. If a permissions boundary, Organizations SCP, or session policy is present, it might override the allow with an implicit deny
4. An explicit deny in any policy overrides any allows

IAM Policy Structure

DigitalCloud
TRAINING

IAM Policy Structure

An IAM policy is a JSON document that consists of one or more statements



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

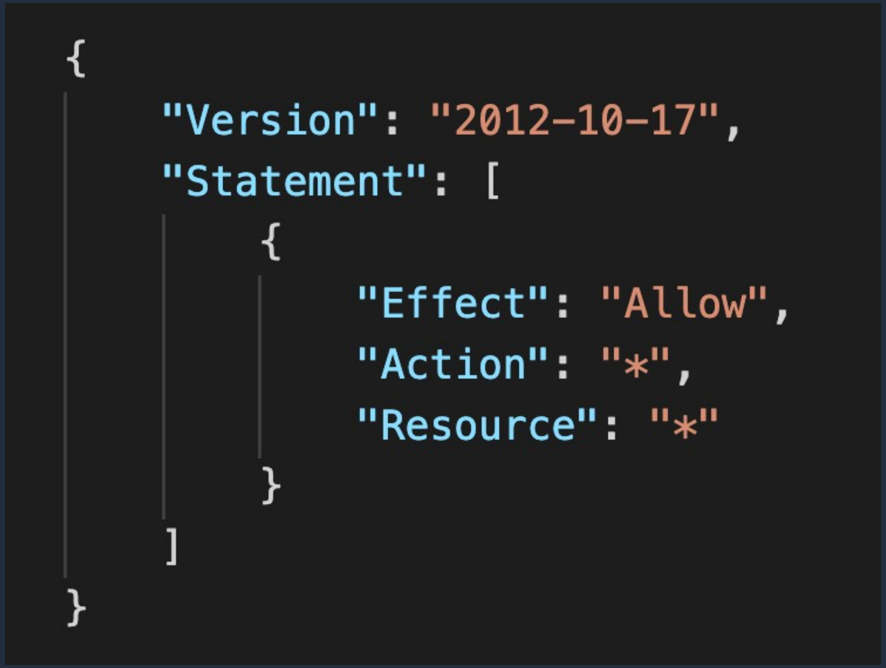
The **Effect** element can be Allow or Deny

The **Action** element is the specific API action for which you are granting or denying permission

The **Resource** element specifies the resource that's affected by the action

The **Condition** element is optional and can be used to control when your policy is in effect

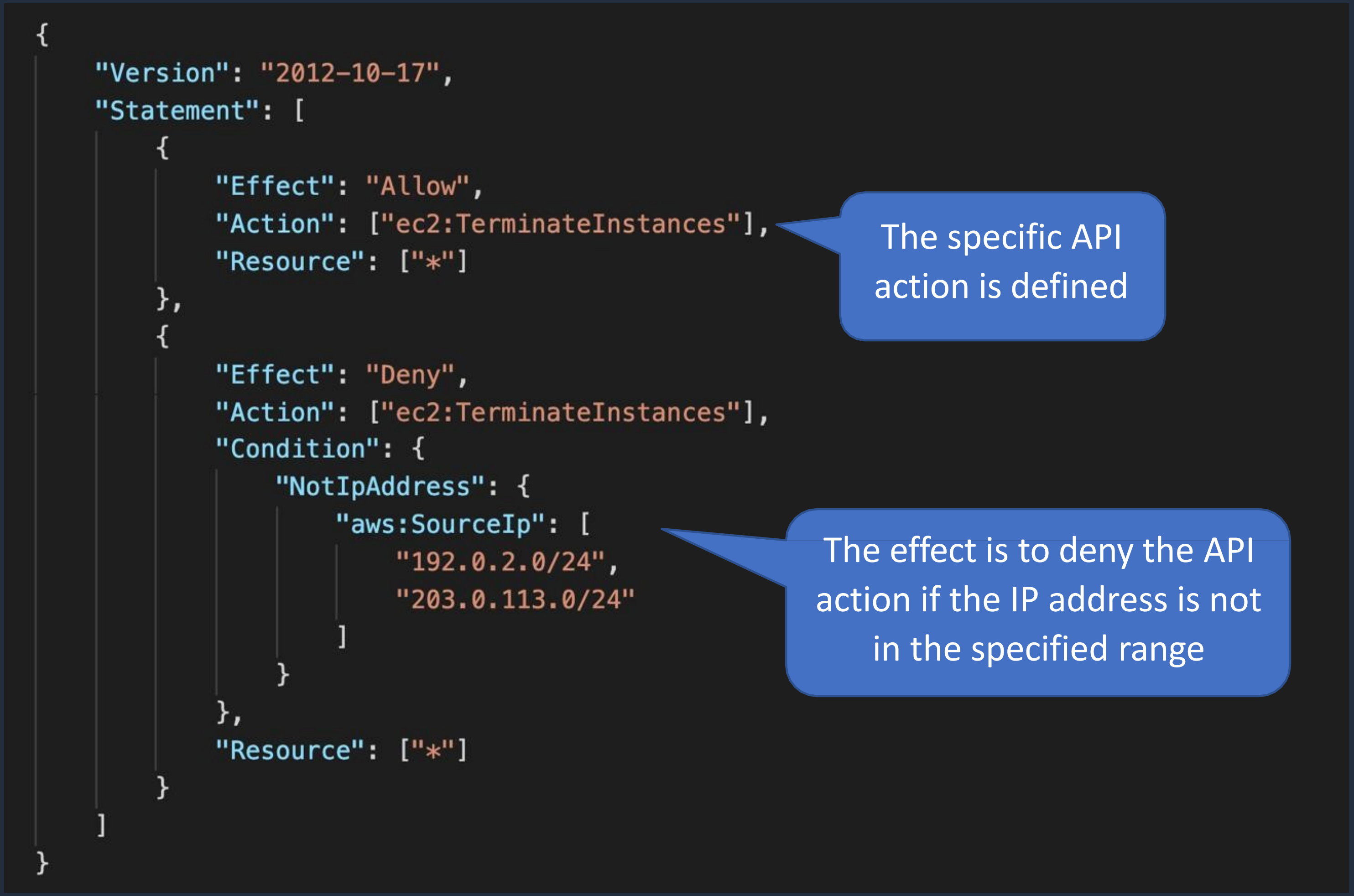
IAM Policy Example 1



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

The AdministratorAccess policy uses wildcards (*) to allow all actions on all resources

IAM Policy Example 2

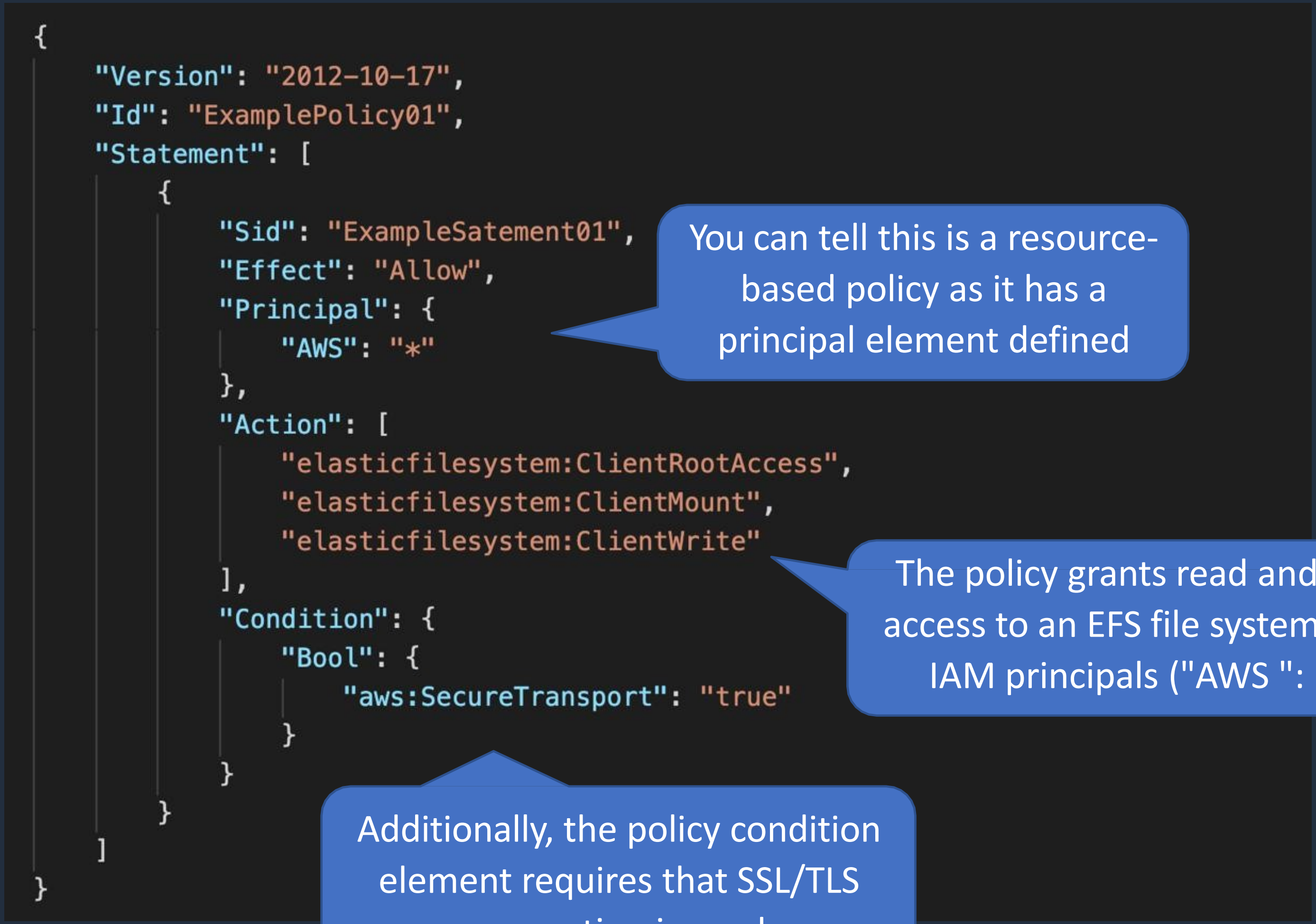


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ec2:TerminateInstances"],
            "Resource": ["*"]
        },
        {
            "Effect": "Deny",
            "Action": ["ec2:TerminateInstances"],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            },
            "Resource": ["*"]
        }
    ]
}
```

The specific API action is defined

The effect is to deny the API action if the IP address is not in the specified range

IAM Policy Example 3



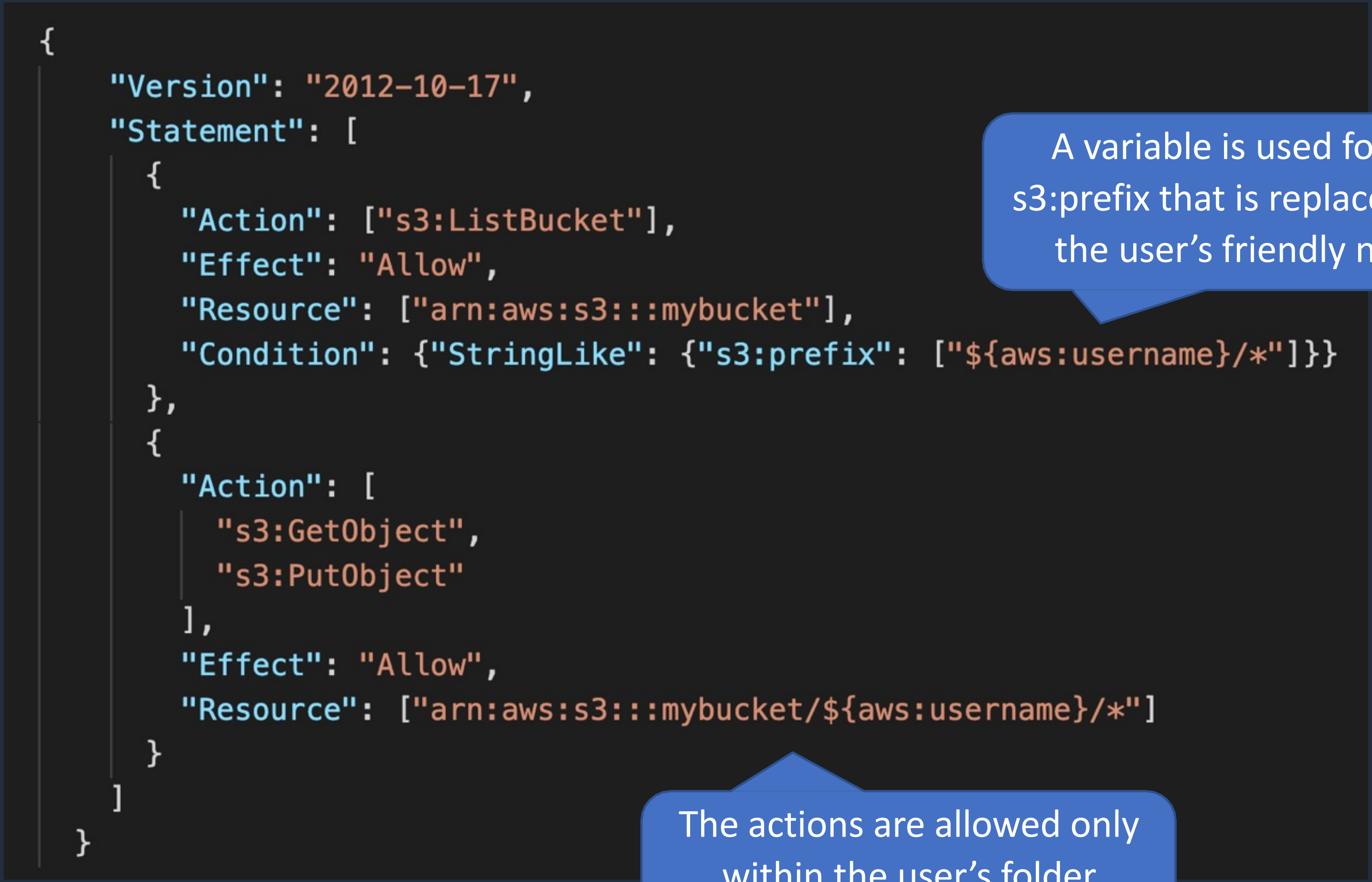
```
{
    "Version": "2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleSatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": [
                "elasticfilesystem:ClientRootAccess",
                "elasticfilesystem:ClientMount",
                "elasticfilesystem:ClientWrite"
            ],
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

You can tell this is a resource-based policy as it has a principal element defined

The policy grants read and write access to an EFS file systems to all IAM principals ("AWS ": "*")

Additionally, the policy condition element requires that SSL/TLS encryption is used

IAM Policy Example 4



```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Action": ["s3:ListBucket"],
        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::mybucket"],
        "Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
      },
      {
        "Action": [
          "s3:GetObject",
          "s3:PutObject"
        ],
        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::mybucket/${aws:username}/*"]
      }
    ]
  }
```

A variable is used for the s3:prefix that is replaced with the user's friendly name

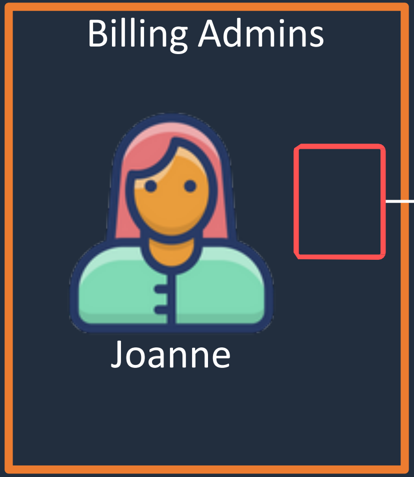
The actions are allowed only within the user's folder within the bucket

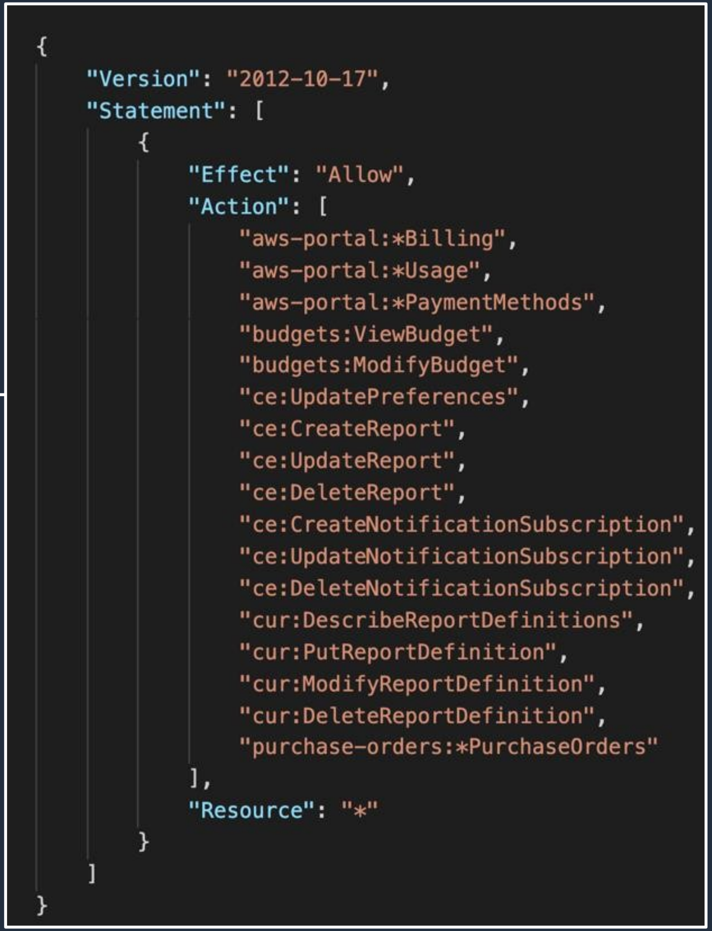
Using Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC)

Job function policies:

- Administrator
- Billing
- Database administrator
- Data scientist
- Developer power user
- Network administrator
- Security auditor
- Support user
- System administrator
- View-only user

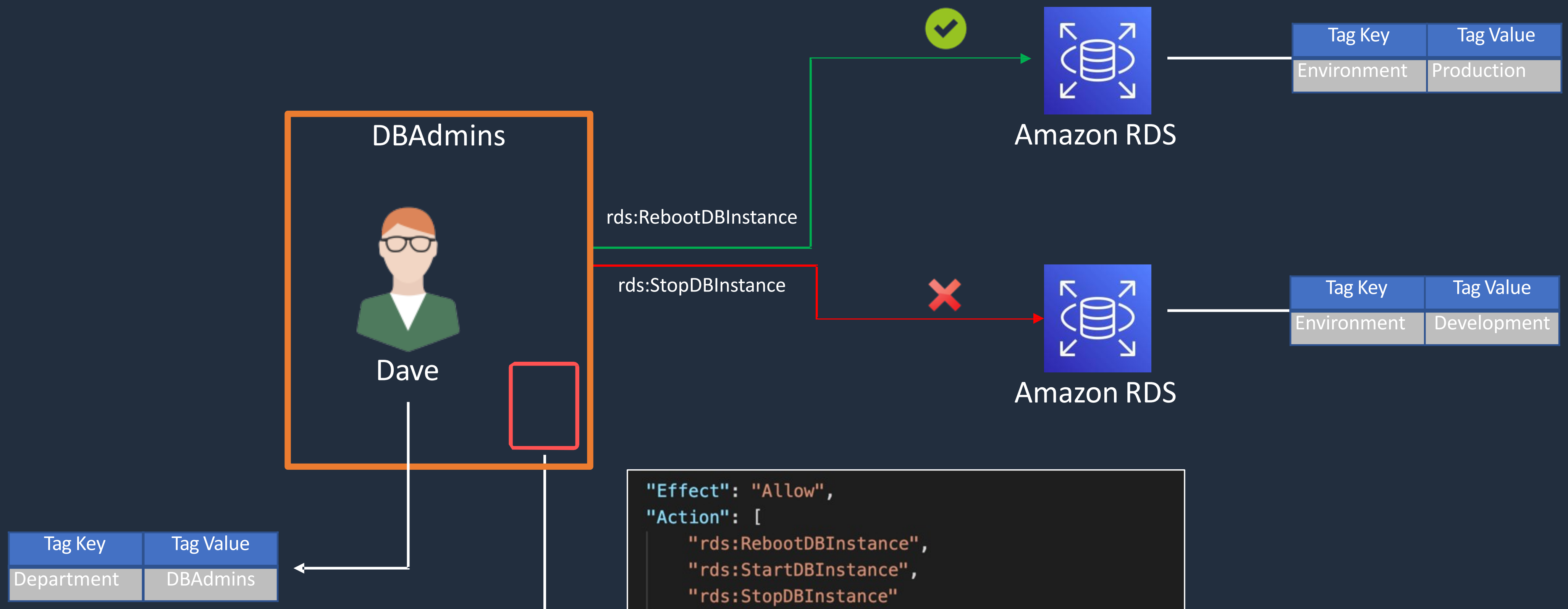




```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aws-portal:*Billing",
                "aws-portal:*Usage",
                "aws-portal:*PaymentMethods",
                "budgets:ViewBudget",
                "budgets:ModifyBudget",
                "ce:UpdatePreferences",
                "ce:CreateReport",
                "ce:UpdateReport",
                "ce:DeleteReport",
                "ce:CreateNotificationSubscription",
                "ce:UpdateNotificationSubscription",
                "ce:DeleteNotificationSubscription",
                "cur:DescribeReportDefinitions",
                "cur:PutReportDefinition",
                "cur:ModifyReportDefinition",
                "cur:DeleteReportDefinition",
                "purchase-orders:*PurchaseOrders"
            ],
            "Resource": "*"
        }
    ]
}
```

Using Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC)



Tag Key	Tag Value
Environment	Production

Tag Key	Tag Value
Environment	Development

Tag Key	Tag Value
Department	DBAdmins

```
"Effect": "Allow",
"Action": [
    "rds:RebootDBInstance",
    "rds:StartDBInstance",
    "rds:StopDBInstance"
],
"Resource": "*",
"Condition": {
    "StringEquals": {
        "aws:PrincipalTag/Department": "DBAdmins",
        "rds:db-tag/Environment": "Production"
```

Apply Permissions Boundary

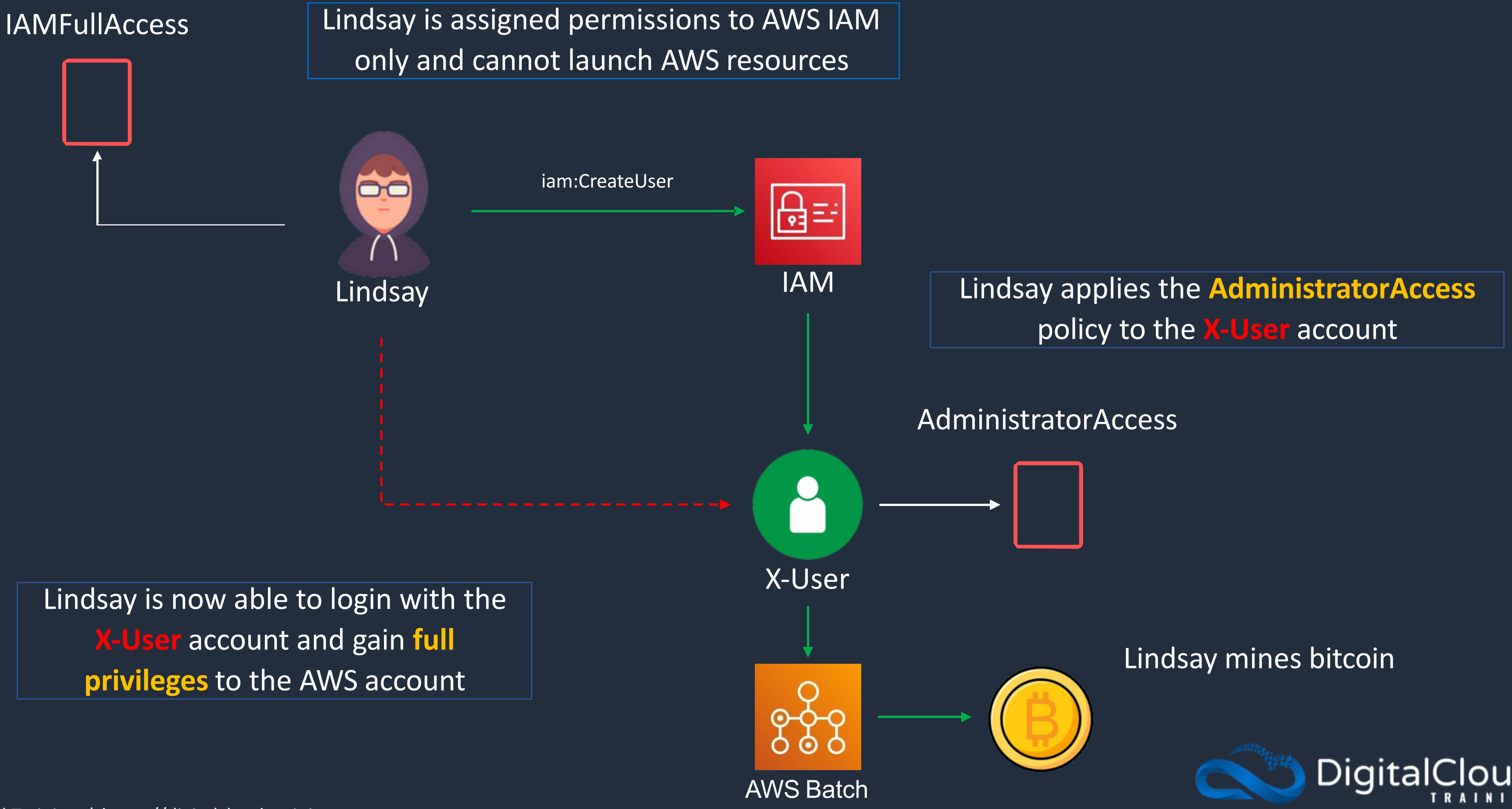
Permisions Boundary Hands-On Practice

*** Use the PermissionsBoundary.json file from the course download ***

The policy will enforce the following:

- IAM principals can't alter the permissions boundary to allow their own permissions to access restricted services
- IAM principals must attach the permissions boundary to any IAM principals they create
- IAM admins can't create IAM principals with more privileges than they already have
- The IAM principals created by IAM admins can't create IAM principals with more permissions than IAM admins

Privilege Escalation



Lindsay is assigned permissions to AWS IAM only and cannot launch AWS resources

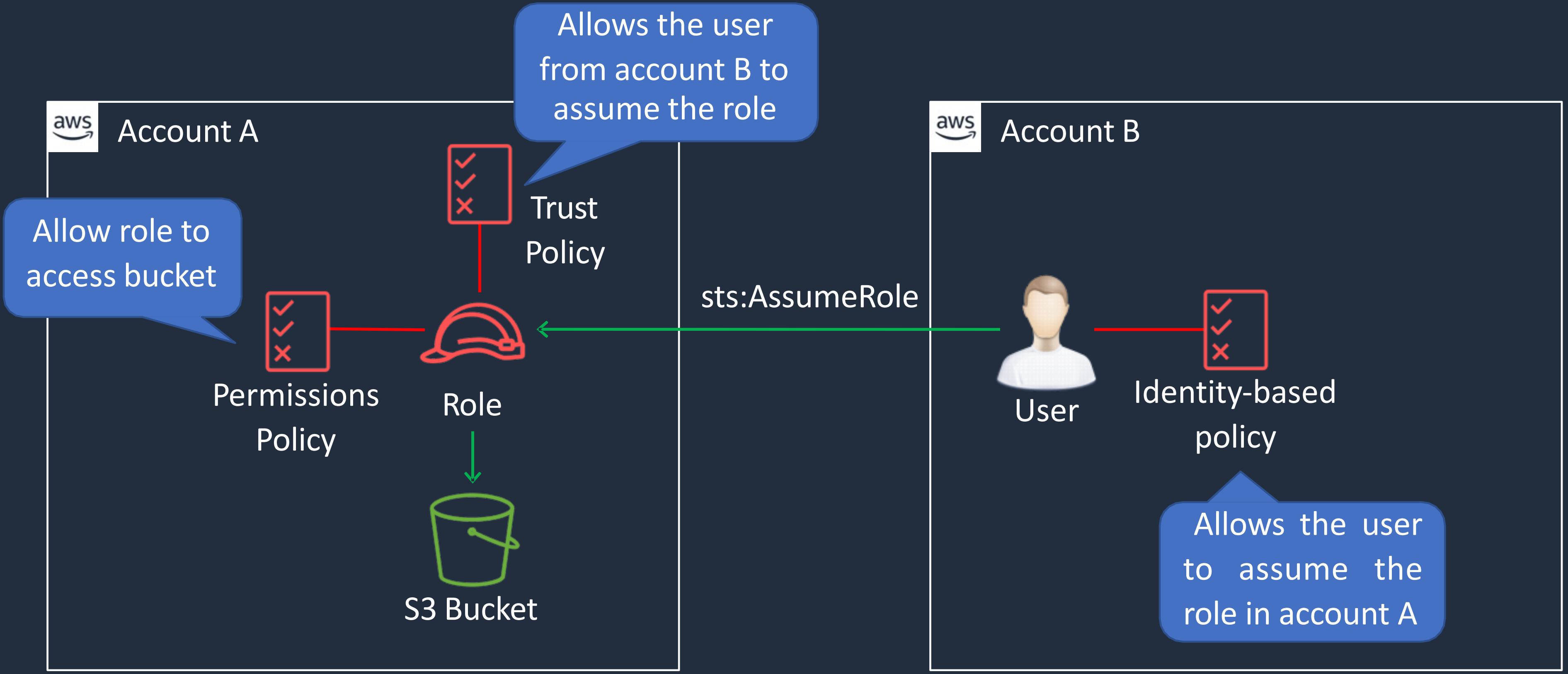
Lindsay applies the **AdministratorAccess** policy to the **X-User** account

Lindsay is now able to login with the **X-User** account and gain **full privileges** to the AWS account

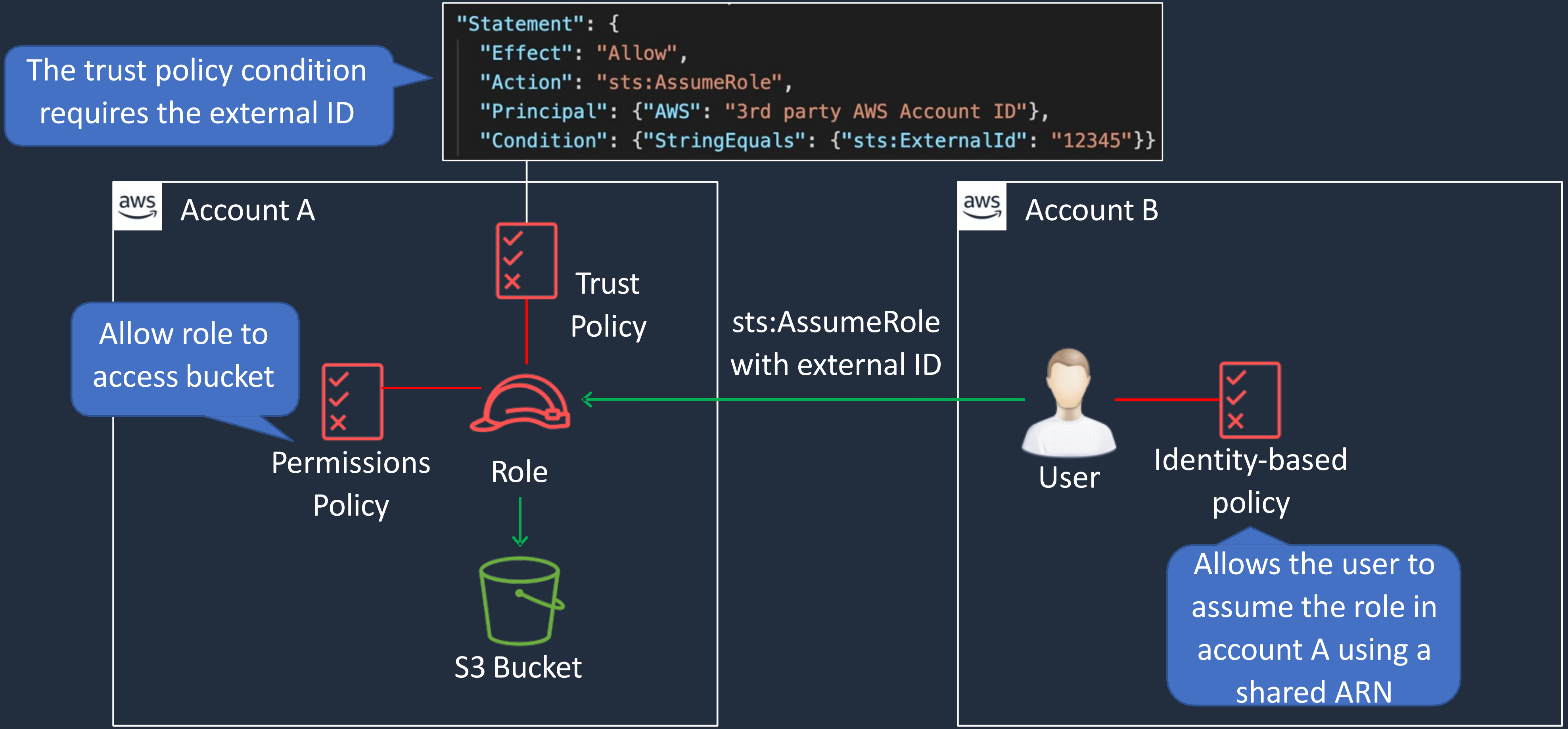
Lindsay mines bitcoin

Use Cases for IAM Roles

Use Case: Cross Account Access

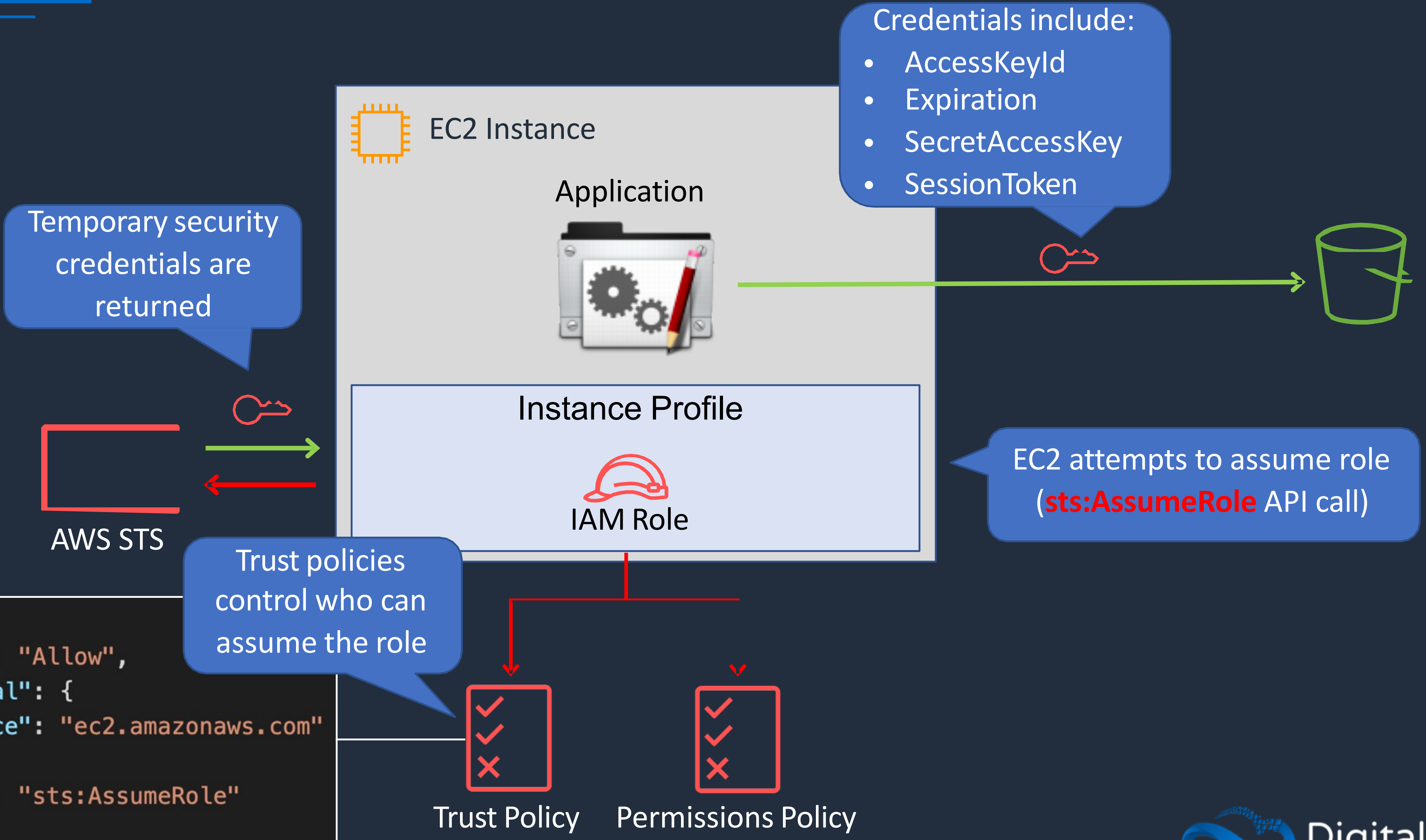

Allows the user from account B to assume the role
Account A
Account B
Allow role to access bucket
Trust Policy
sts:AssumeRole
Permissions Policy
Role
User
Identity-based policy
S3 Bucket
Allows the user to assume the role in account A

Use Case: Cross Account Access (3rd Party)


The trust policy condition requires the external ID
Account A
Trust Policy
Allow role to access bucket
Permissions Policy
Role
S3 Bucket
sts:AssumeRole with external ID
Account B
User
Identity-based policy
Allows the user to assume the role in account A using a shared ARN

```
"Statement": {
  "Effect": "Allow",
  "Action": "sts:AssumeRole",
  "Principal": {"AWS": "3rd party AWS Account ID"},
  "Condition": {"StringEquals": {"sts:ExternalId": "12345"}}
```

Use Case: Delegation to AWS Services



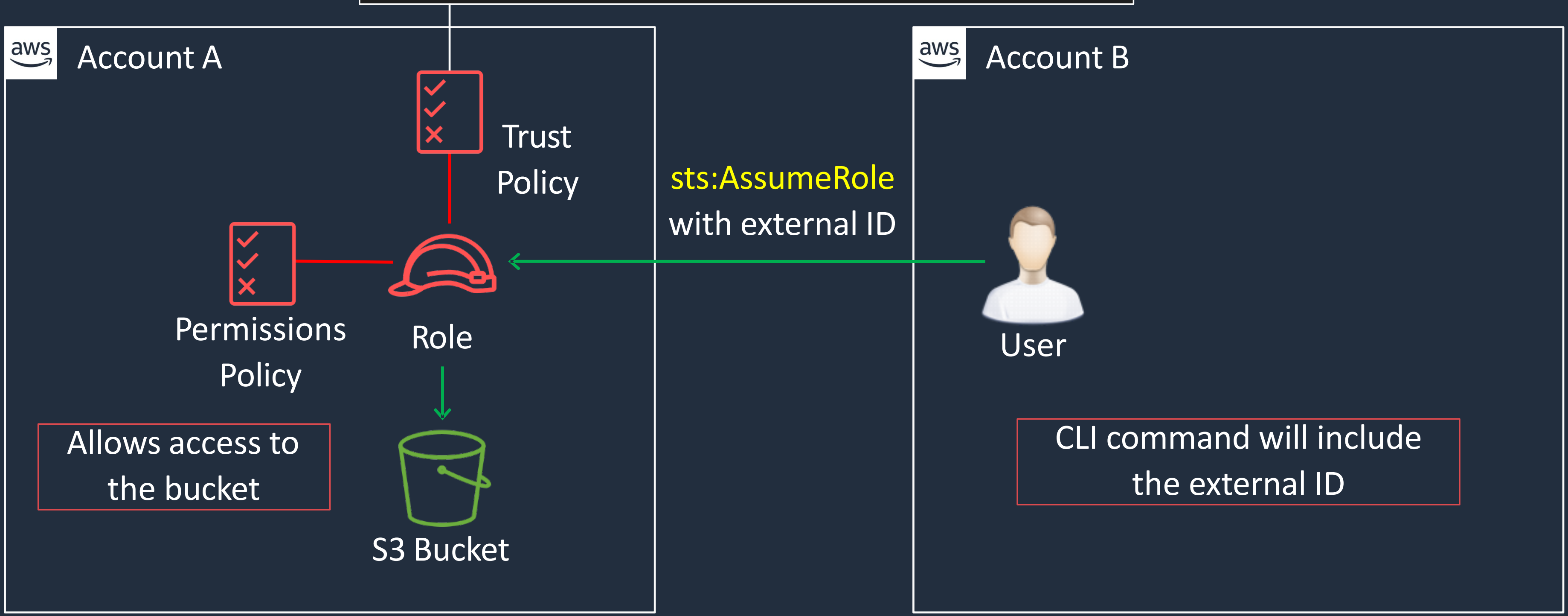
```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "ec2.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
```

Cross-Account Access

Cross Account Access (IAM Role)

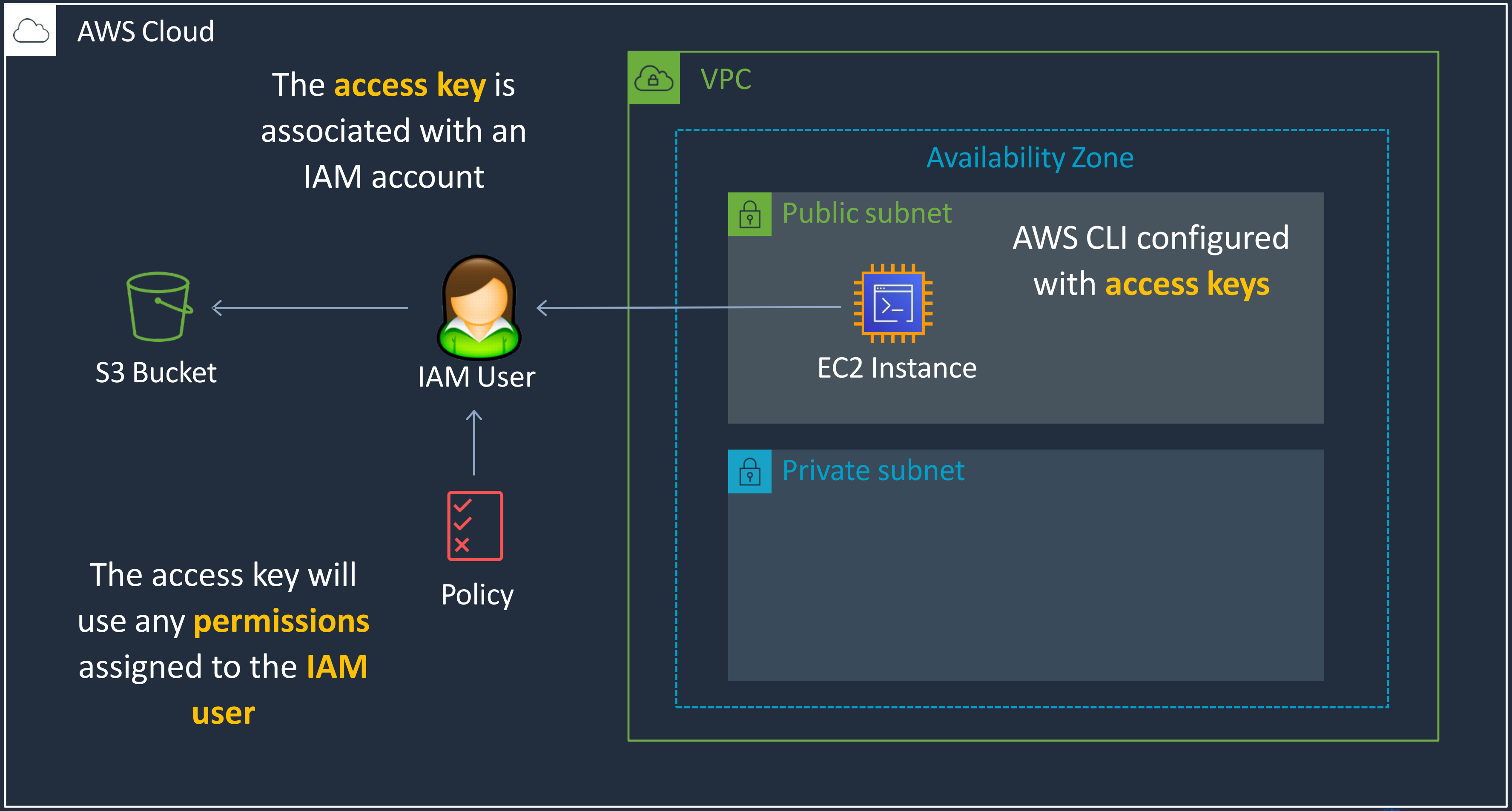
The trust policy condition requires the external ID

```
"Statement": {
  "Effect": "Allow",
  "Action": "sts:AssumeRole",
  "Principal": {"AWS": "3rd party AWS Account ID"},
  "Condition": {"StringEquals": {"sts:ExternalId": "12345"}}
```


Account A
Trust Policy
Permissions Policy
Role
Allows access to the bucket
S3 Bucket
sts:AssumeRole with external ID
Account B
User
CLI command will include the external ID

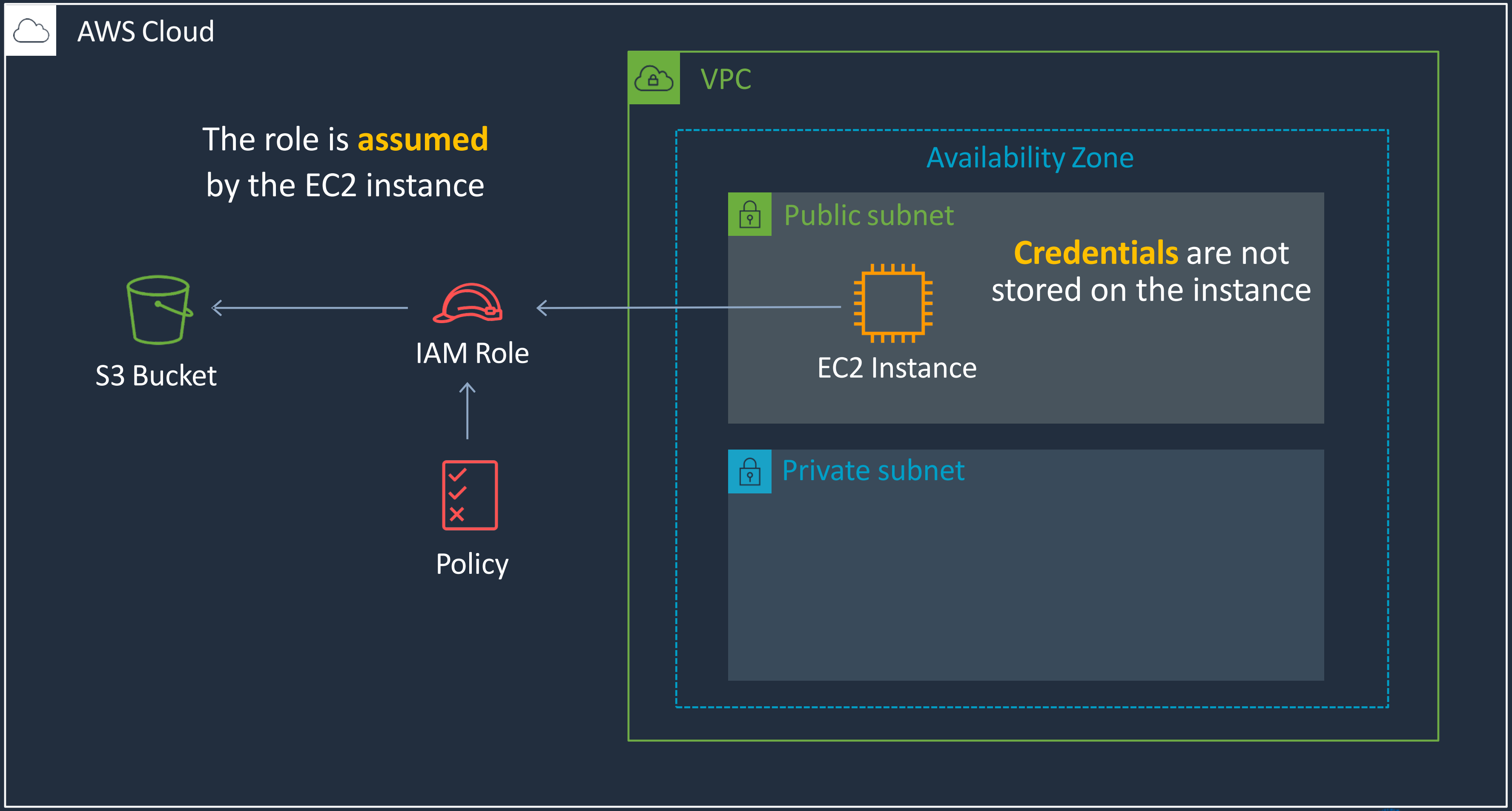
Access Keys and IAM Roles with EC2

Using Access Keys with Amazon EC2


AWS Cloud
The access key is associated with an IAM account
VPC
Availability Zone
Public subnet
AWS CLI configured with access keys
S3 Bucket
IAM User
EC2 Instance
Private subnet
Policy
The access key will use any permissions assigned to the IAM user

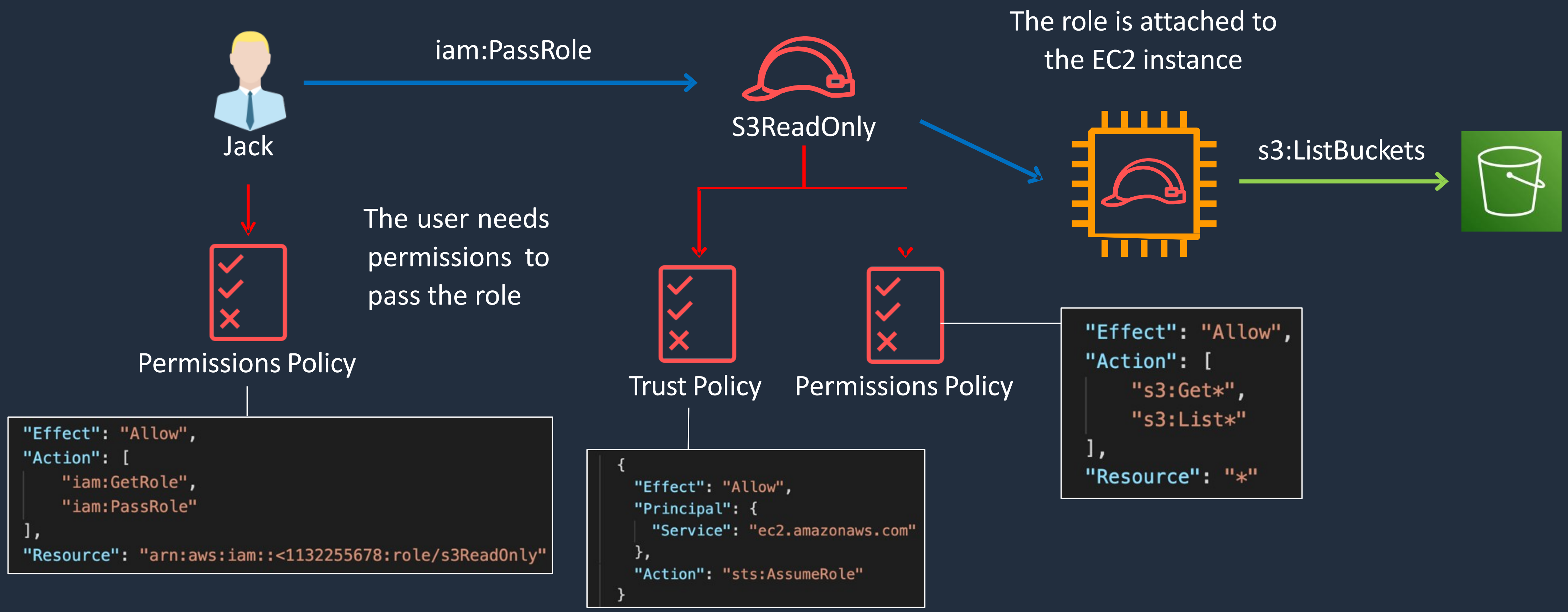
DigitalCloud TRAINING

Using Roles with Amazon EC2



Amazon EC2 Instance Profile

Attach Role to EC2 Instance



```
"Effect": "Allow",
"Action": [
    "iam:GetRole",
    "iam:PassRole"
],
"Resource": "arn:aws:iam::<1132255678:role/s3ReadOnly"
```

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "ec2.amazonaws.com"
  },
  "Action": "sts:AssumeRole"
}
```

```
"Effect": "Allow",
"Action": [
    "s3:Get*",
    "s3:List*"
],
"Resource": "*"
```

IAM Best Practices

AWS IAM Best Practices

- Require human users to use federation with an identity provider to access AWS using temporary credentials
- Require workloads to use temporary credentials with IAM roles to access AWS
- Require multi-factor authentication (MFA)
- Rotate access keys regularly for use cases that require long-term credentials
- Safeguard your root user credentials and don't use them for everyday tasks
- Apply least-privilege permissions
- Get started with AWS managed policies and move toward least-privilege permissions

AWS IAM Best Practices

- Use IAM Access Analyzer to generate least-privilege policies based on access activity
- Regularly review and remove unused users, roles, permissions, policies, and credentials
- Use conditions in IAM policies to further restrict access
- Verify public and cross-account access to resources with IAM Access Analyzer
- Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions
- Establish permissions guardrails across multiple accounts
- Use permissions boundaries to delegate permissions management within an account