aws SUMMIT ONLINE

S E C 0 9

Federated access and authorisation made simple

Louay Shaat

Senior Solutions Architect
Amazon Web Services

aws SUMMIT ONLINE

Agenda

Attribute Based Access (ABAC) in AWS and why?

ABAC in depth

Demo

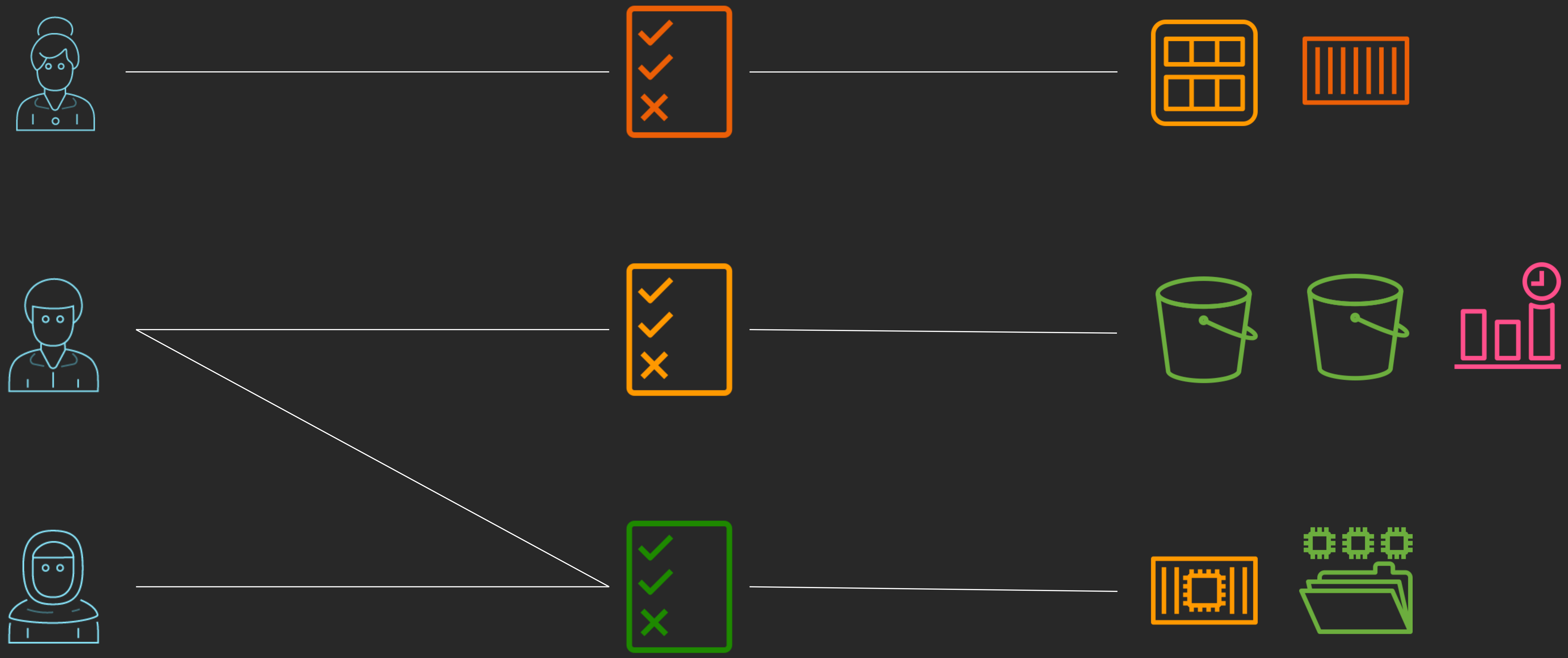
ABAC best practices

Conclusion

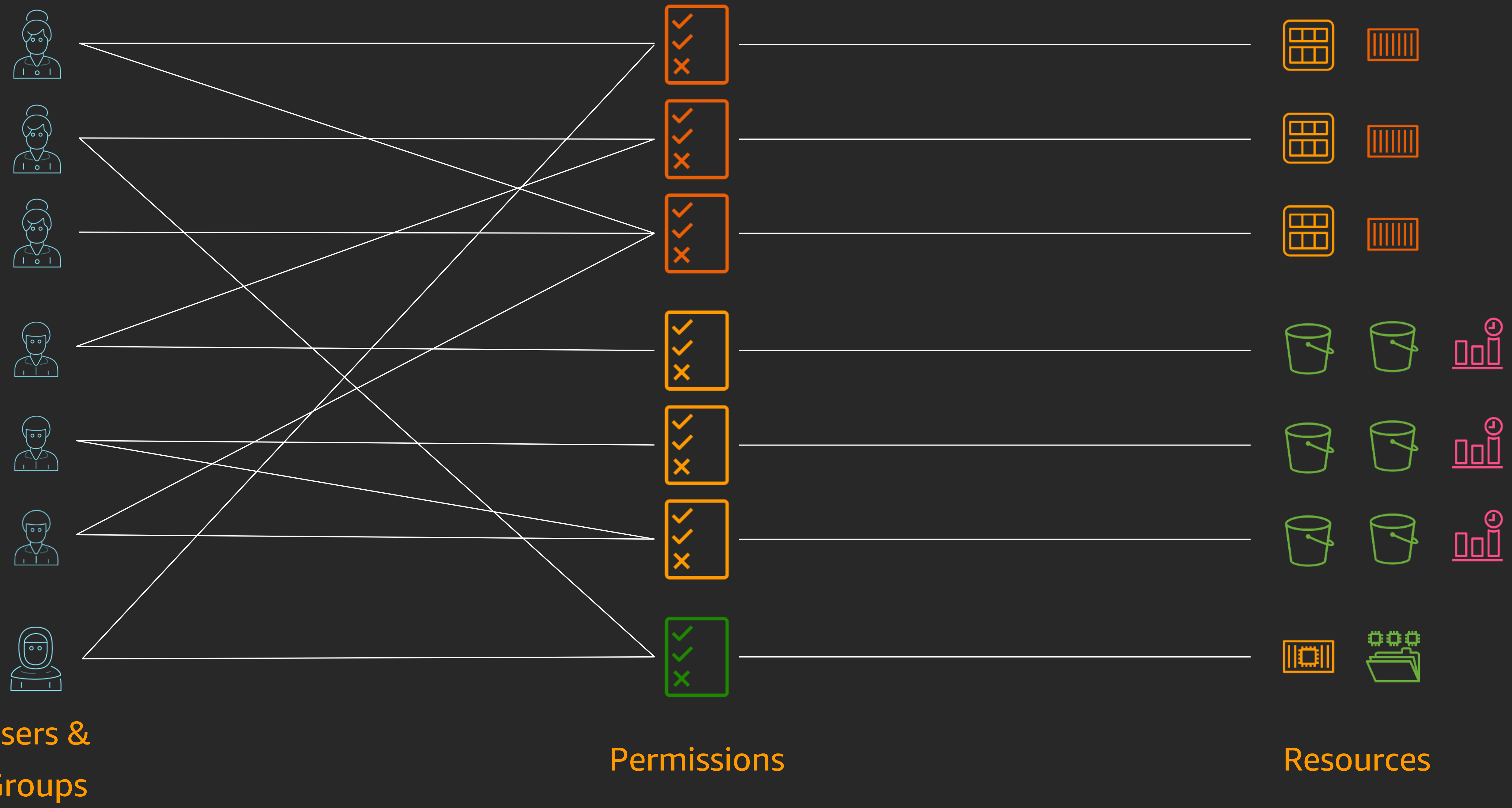
Identity in AWS

aws SUMMIT ONLINE

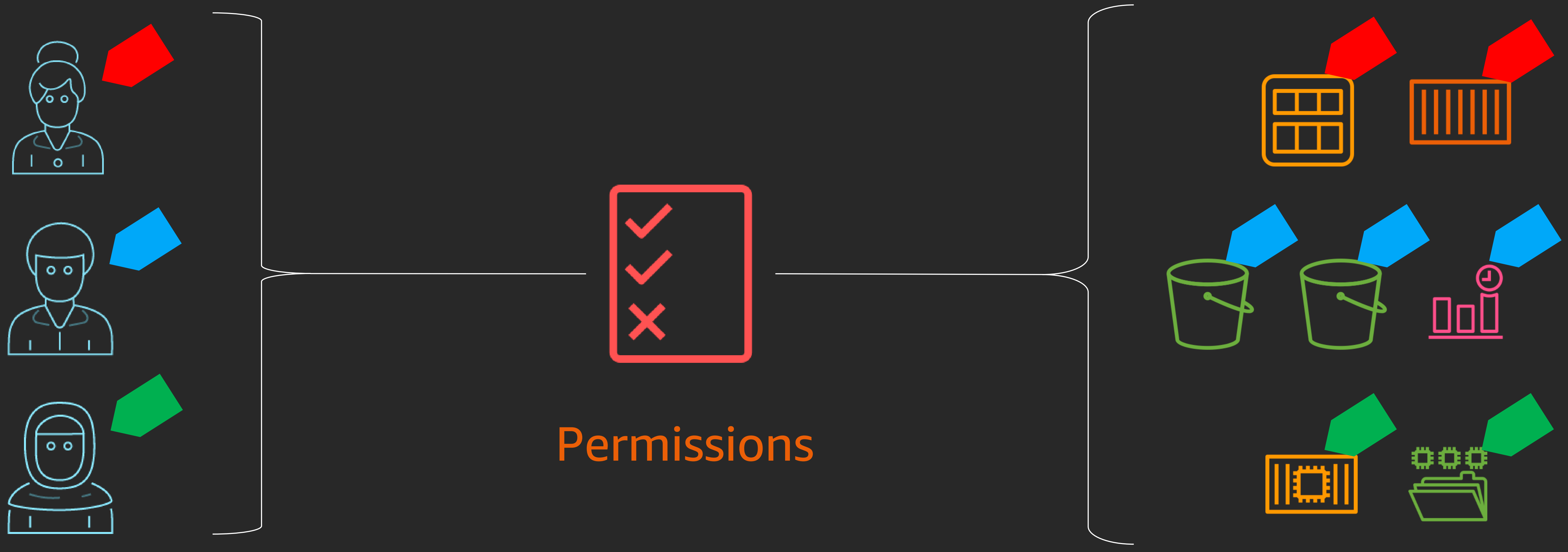
Role-based access control (RBAC)



What ends up happening



A scalable permissions model based on attributes



Workforce teams

Resources

A little bit about attributes in AWS

Attributes are used as tags in AWS

Attributes are a key or a key and value pair

Use attributes for cost-allocation, discovery, and access control

Examples

Team = Engineering
Project = Serverless

Team = Engineering
Project = Serverless

Benefits of ABAC

Permissions scale with innovation, enabling developers to build

Teams move fast, as permissions automatically apply based on attributes

Granular permissions are possible without requiring a permissions update for every new user or resource

Audit attributes are available to determine access

Examples of Attribute based permissions

Engineering Department requires read and write access to their project resources

Engineering Department requires developers to assign their project to new resources

Grant Project Serverless read access to their team resources

Manage only the resources that I own

How does it work?

Attributes in AWS session

Attributes in the AWS session are called session tags

Session tags are temporary tags, not stored anywhere in AWS

These are tags (key:value pairs) in the AWS session

Logged in CloudTrail during assume role

Session tags override tags on role

Session tag control

New Permission

sts:TagSession

Required to pass tags in the session

API support

AssumeRole

AssumeRoleWithSAML

AssumeRoleWithWebIdentity

GetFederationToken

Additional Controls

aws:RequestTag/key

aws:TagKeys

aws:PrincipalTag/key

Transitivity

Persist session tags during role-chaining

sts:TransitiveTagKeys

Passing session tags during SAML federation

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:project">
      <AttributeValue>Serverless<AttributeValue>
</Attribute>

 <Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:jobfunction">
      <AttributeValue>Engineering<AttributeValue>
</Attribute>
```

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
    <AttributeValue>project<AttributeValue>
</Attribute>
```

Identity Partners for session tags

Auth0

Ping Identity

ForgeRock

RSA

onelogin

okta

IBM Security

Sample ABAC policy for session tags

```
{
  "Effect":"Allow",
  "Action":[
    "secretsmanager:GetResourcePolicy",
    "secretsmanager:GetSecretValue",
    "secretsmanager:UpdateSecret"
  ],
  "Resource":"*",
  "Condition":{
    "StringEquals":{
      "secretsmanager:ResourceTag/project":"${aws:PrincipalTag/project}"
    }
  }
}
```

Only manage resources which match user’s project

Sample role trust policy for session tags

```
{
  "Effect": "Allow",
  "Principal": {
    "Federated": "arn:aws:iam::<accountID>:saml-provider/ExampleProvider"
  },
  "Action": [
    "sts:AssumeRoleWithSAML",
    "sts:TagSession"
  ],
  "Condition": {
    "StringLike": {
      "aws:RequestTag/project": "*",
      "aws:RequestTag/department": [
        ”Engineering",
        ”Serverless"
      ]
    },
    "StringEquals": {
      "SAML:aud": "https://signin.aws.amazon.com/saml"
    }
  }
}
```

Trusted IdP

New Permission

Must pass Project and JobFunction Tags

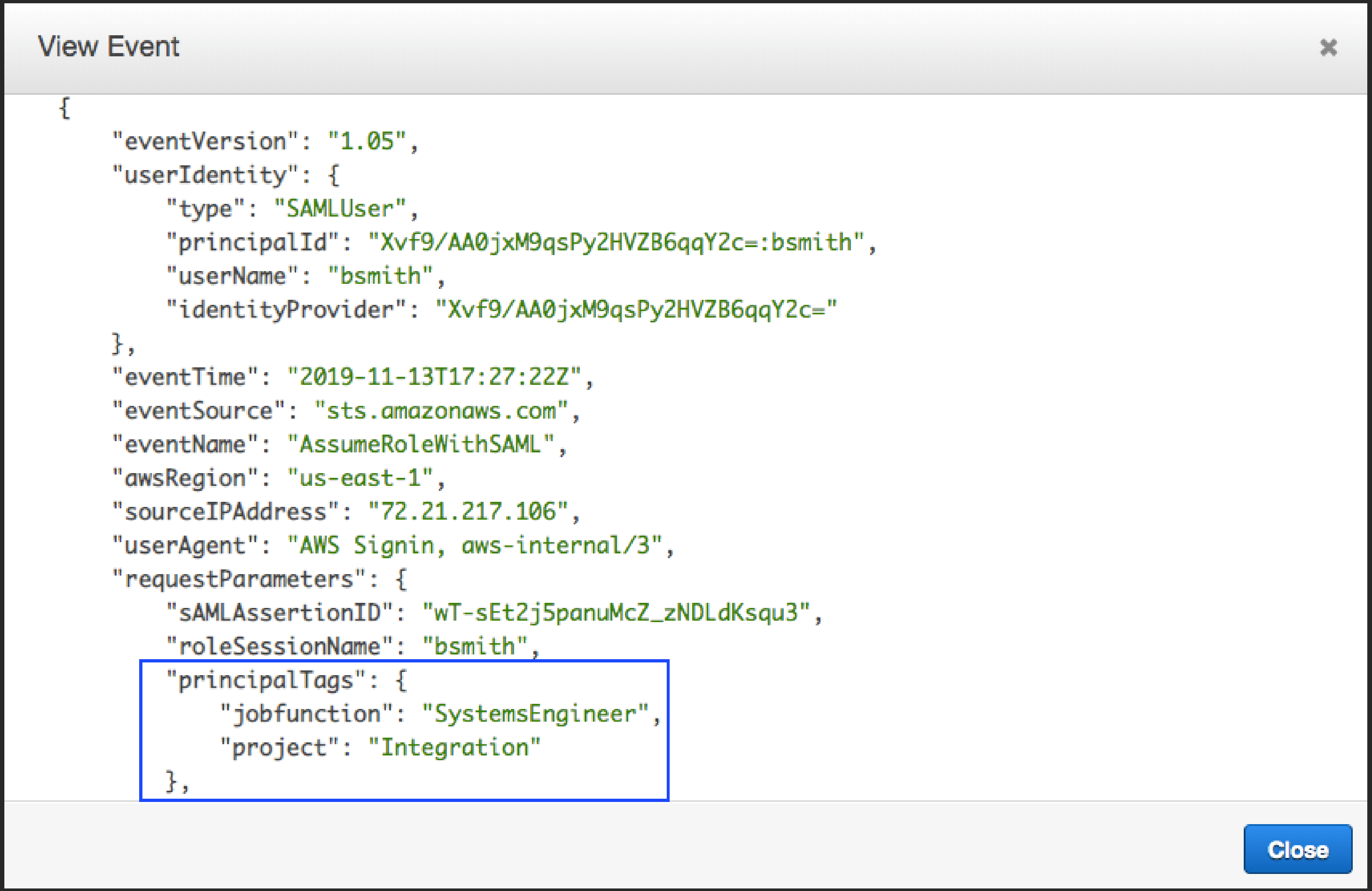
Using ABAC controls in policies

```
"Condition": {
  "StringEquals": {
    "secretsmanager:ResourceTag/project": "${aws:PrincipalTag/project}"
  }
}
```

```
"Condition": {
  "StringEquals": {
    "aws:PrincipalTag/jobfunction": “Engineering”
  }
}
```

```
"Resource": "arn:aws:s3:::${aws:PrincipalTag/BucketName}/*”
```

Logged in AWS CloudTrail



```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "SAMLUser",
        "principalId": "Xvf9/AA0jxM9qsPy2HVZB6qqY2c=:bsmith",
        "userName": "bsmith",
        "identityProvider": "Xvf9/AA0jxM9qsPy2HVZB6qqY2c="
    },
    "eventTime": "2019-11-13T17:27:22Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoleWithSAML",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.217.106",
    "userAgent": "AWS Signin, aws-internal/3",
    "requestParameters": {
        "sAMLAssertionID": "wT-sEt2j5panuMcZ_zNDLdKsqu3",
        "roleSessionName": "bsmith",
        "principalTags": {
            "jobfunction": "SystemsEngineer",
            "project": "Integration"
        },
```

Demo

aws SUMMIT ONLINE

Demo Setup

Project Automation

Secrets Manager

Set up users in your IdP with attributes

1 Identities with attributes

2 Pass user attributes during federation

3 Require attributes for new resources

4 Set permissions based on attributes

Demo Steps

1. Workforce identity working on *project Serverless*
2. *Configure IdP* to pass session tags
3. Create a policy to set access permissions based on *project attributes* for Secrets Manager

Required Attributes

- *project* of the identity that created the resource

Create new resources and demonstrate permissions

5

Create new resources

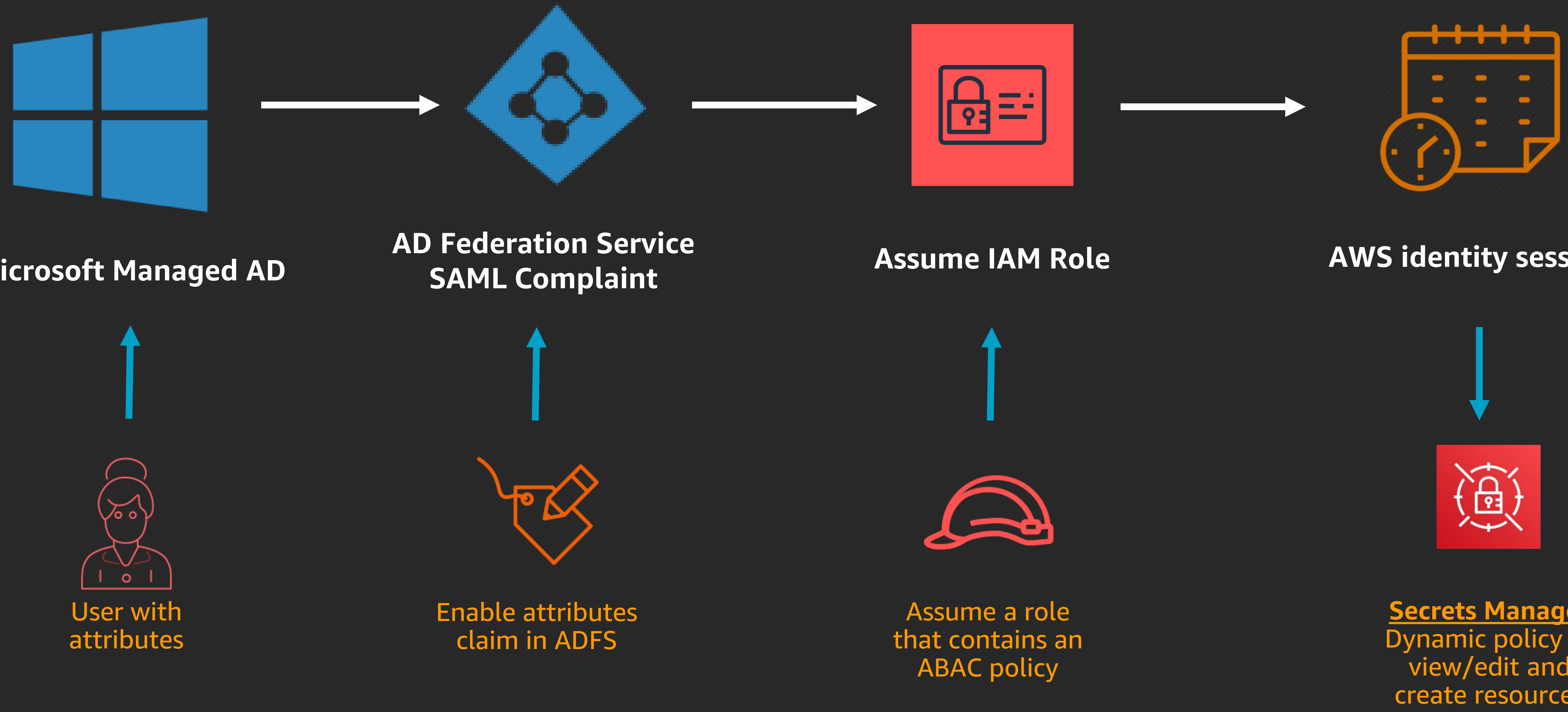
6

Permissions automatically apply

Demo Steps

1. Use the AWS-DEV to access an existing secret with *project=Serverless* tag
2. Use the AWS-DEV to create a new secret with *project=Serverless* tag

Authentication flow


Microsoft Managed AD
AD Federation Service SAML Complaint
Assume IAM Role
AWS identity session
User with attributes
Enable attributes claim in ADFS
Assume a role that contains an ABAC policy
Secrets Manager
Dynamic policy to view/edit and create resources

ABAC Best Practices

aws SUMMIT ONLINE

Five ABAC best practices to take with you

1. Only approved principals can set or modify attributes

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "Statement1",
    "Effect": "Deny",
    "Action": [
      "iam:TagRole",
      "iam:UntagRole",
      "ec2:CreateTags",
      "ec2:DeleteTags"
    ],
    "Resource": "*",
    "Condition": {
      "StringNotEquals": {
        "aws:PrincipalArn”: "arn:aws:iam::<accountID>:user/<username>"
      }
    }
  }]
}
```

Use Service Control Policies to restrict who can add or modify tags

Five ABAC best practices continued

2. Reserve a subset of attributes for access control
3. Tag everything during creation so permissions apply immediately
4. Rely on attributes to grant permissions to manage resources
5. Periodically audit to ensure resources and users are tagged appropriately

Summary: session tags for ABAC

Identity Provider is the source of truth

Permissions apply automatically

Track user activity

Where do I get started

ABAC documentation –
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction_attribute-based-access-control.html

Session tags documentation –
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_session-tags.html

Rely on attributes from your corporate directory to create fine-grained permissions in AWS –
https://aws.amazon.com/blogs/security/rely-employee-attributes-from-corporate-directory-create-fine-grained-permissions-aws/

Partner resources –
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml_3rd-party.html

Thank you!

Louay Shaat

lshaat

lshaat