AWS Solutions Architect Associate

Session 201

Security, Id &Compliance: IAM

July/2024

Identity and Access Management

- (..) is a web service that helps you securely control access to AWS resources. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use cloud resources

IAM for:
Identity = Authentication
Access = Authorization
&
Cloudtrail for
Logs = Accounting

Taken from https://docs.aws.amazon.com/IAM/latest/UserGuide/access.html (01/07/2024)

Shared access to your AWS account

- You can grant other people permission to administer and use resources in your AWS account without having to share your password or access key.

Granular permissions

- You can grant different permissions to different people for different resources.

Taken from https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies-cross-account-resource-access.html (01/07/2024)

Multi-factor authentication (MFA)

- You can add two-factor authentication to your account and to individual users for extra security. With MFA you or your users must provide not only a password or access key to work with your account, but also a code from a specially configured device.

Identity Federation

- You can allow users who already have passwords elsewhere—for example, in your corporate network or with an internet identity provider—to get temporary access to your AWS account.

Taken from https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers.html (01/07/2024)

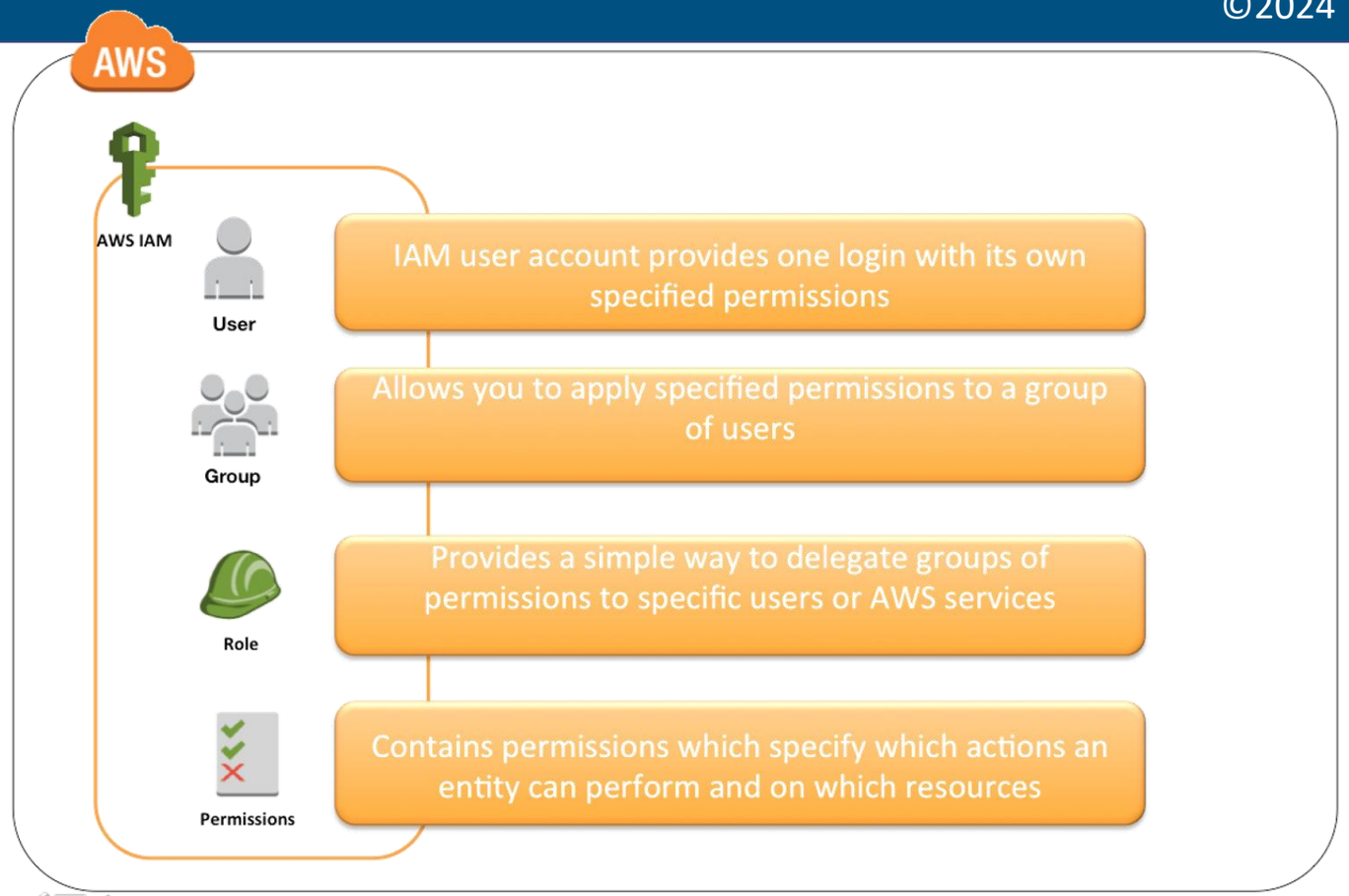
To sump up all,

- Centralized account management.
- Granular permissions.
- Identity Federation → Facebook, LDAP, AD, Google, LinkedIn...
- Multifactor Authentication (MFA).
- Temporal Access for users (STS).
- Password rotation policy.
- PCI-DSS Compliance.
- Free to Use.

Key Terms

Role

Policies



- **User**: Final User.
- **Group**: Collection of users that share a permissions set.
- **Role:** IAM entity that defines a set of permissions for making AWS service requests. Delegate access to a Trusted Entity.
- **Policy**: It is a JSON document that define a set of permissions.

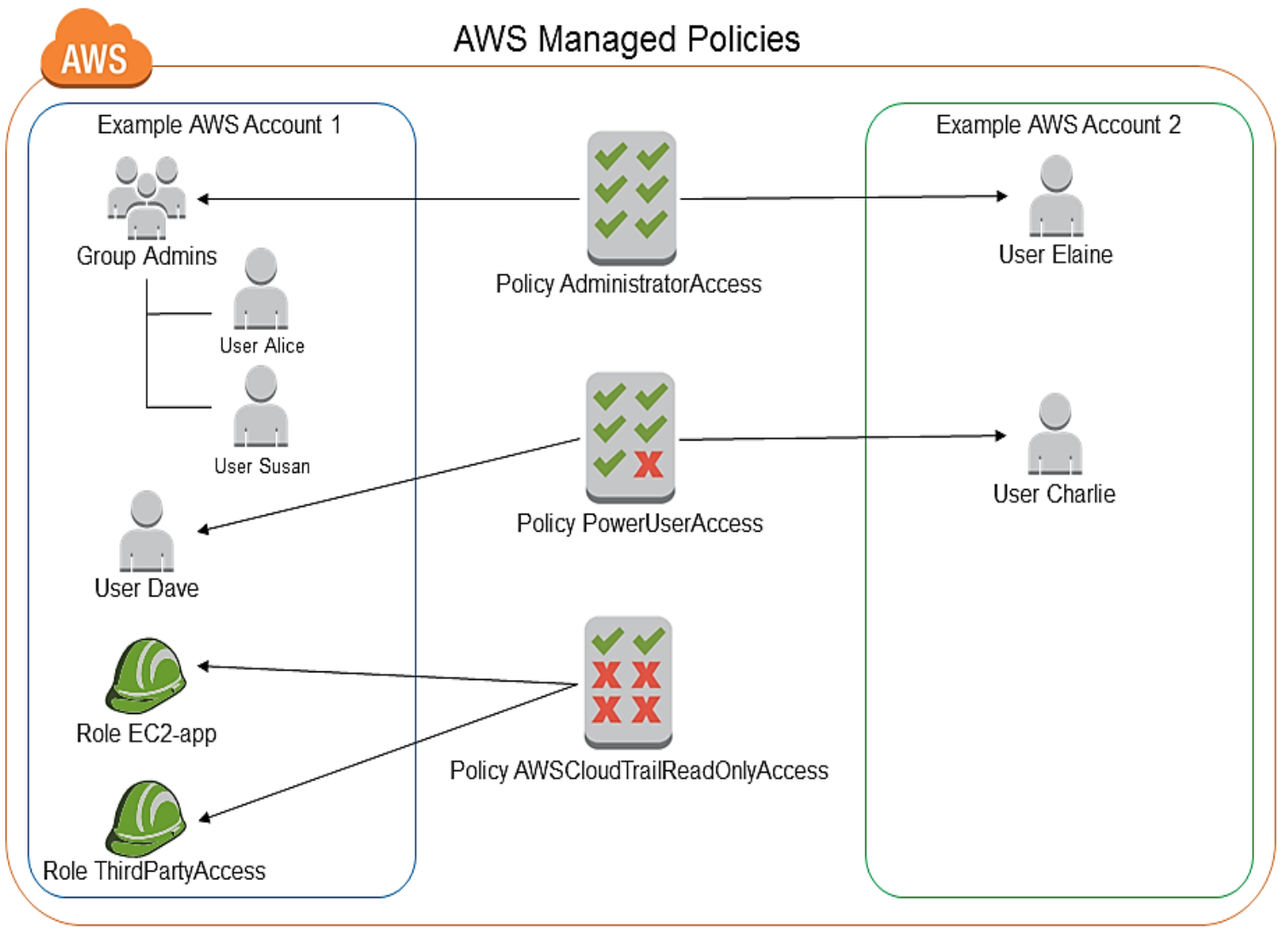
Concepts

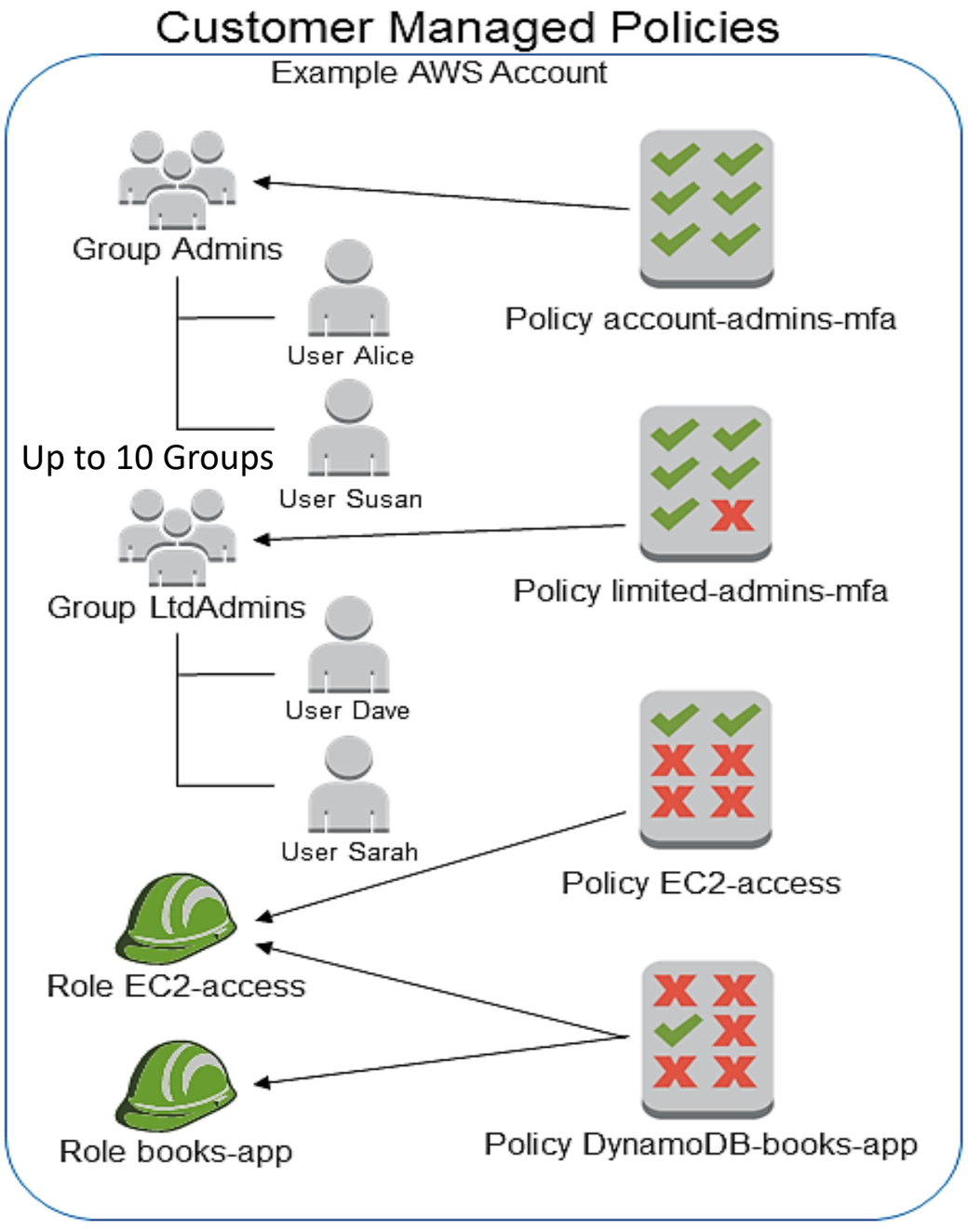
- **<u>IAM → Global</u>** (Eventually Consistent).
- When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user*.
- Options to log in to New Users: Programmatic Access (Access Key, Secret Access Key and Session Token –Opt-) and/or AWS Management Console Access.

IAM Policy

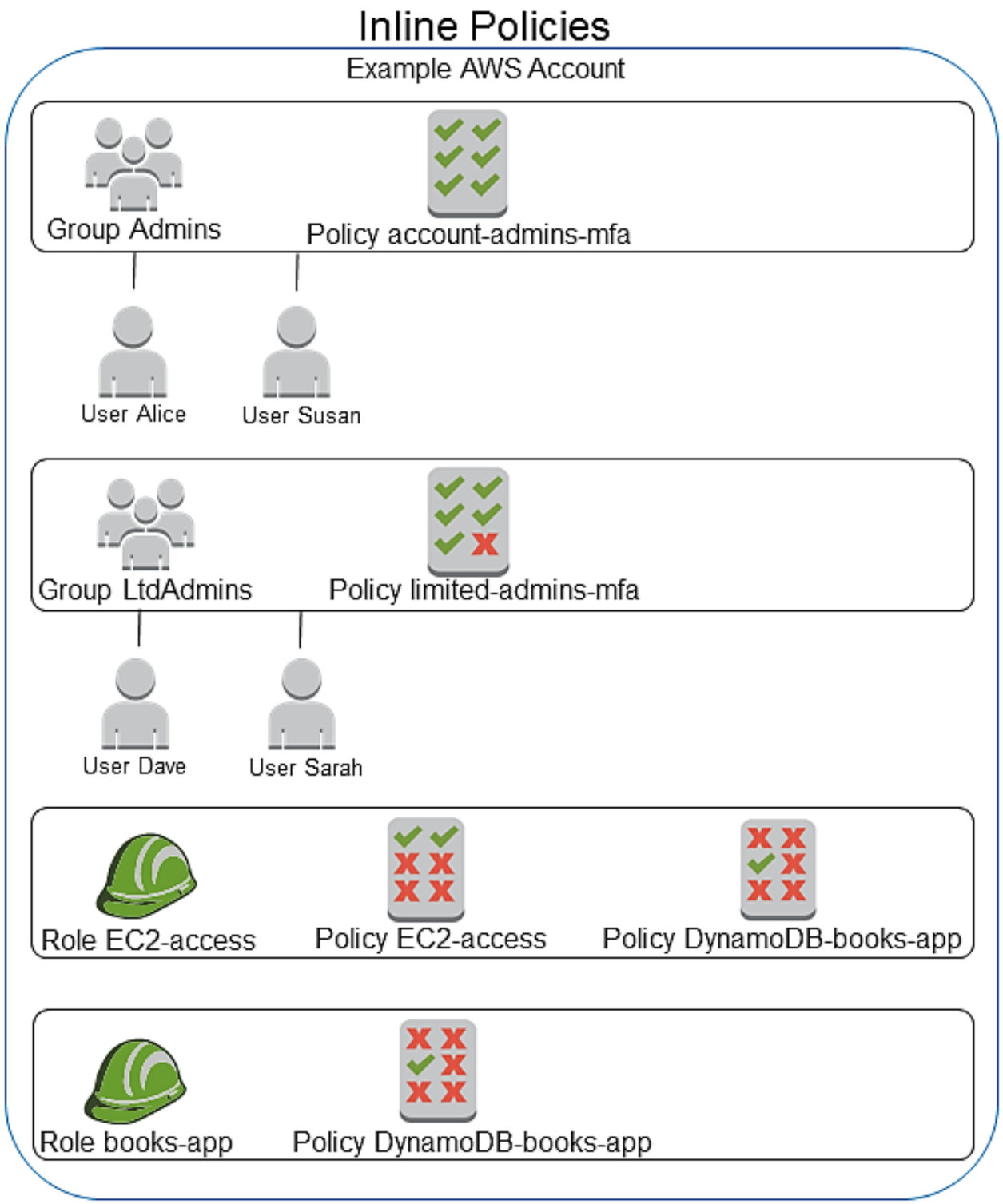
- JSON format.
- Could be associate to a user, role or group.
- Policies control the actions that IAM could do in other resources based in certain conditions.
- Policy types:
  - AWS managed policies.
  - Customer managed policies.
  - Inline policies.

Types: AWS & Customer Managed Policy





Inline vs Managed Policy



- Reusability.
- Central change management
- Versioning and rolling back
- Delegating permissions management
- Automatic updates for AWS managed policies

Components of a Policy

- Version.
- Statement.
- Sid.
- Effect.
- Action.
- Resource.
- Condition: i.e. Source IP

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": [
            "sqs:DeleteMessage",
            "sqs:ListDeadLetterSourceQueues",
            "sqs:DeleteMessageBatch",
            "sqs:SendMessageBatch",
            "sqs:ReceiveMessage",
            "sqs:SendMessage",
            "sqs:GetQueueAttributes"
         ],
         "Resource": [
            "arn:aws:sqs:us-east-1:XXXXXXXXXXX:
         ]
      }
   ]
}
```

```
{
  "Version": "2012-10-
  "Statement": [
    {
      "Sid": "FirstSta
      "Effect": "Allow
      "Action": ["iam:
      "Resource": "*"
    },
    {
      "Sid": "SecondSt
      "Effect": "Allow
      "Action": "s3:Li
      "Resource": "*"
    },
    {
      "Sid": "ThirdSta
      "Effect": "Allow
      "Action": [
        "s3:List*",
        "s3:Get*"
      ],
      "Resource": [
        "arn:aws:s3:::
        "arn:aws:s3:::
      ],
      "Condition": {"I
    }
  ]
}
```

arn:partition:service:region:account-id:resource

Components

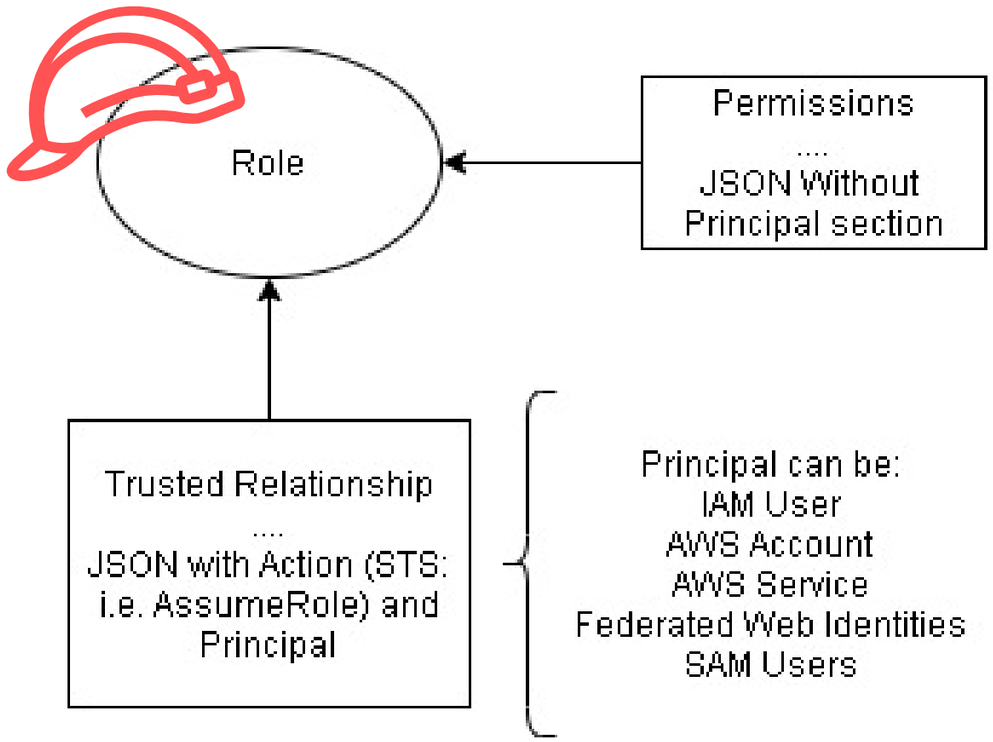
- **Version** – Specify the version of the policy language that you want to use. As a best practice, use the latest 2012-10-17 version.
- **Statement** – Use this main policy element as a container for the following elements.
- **Sid** – Include an optional statement ID to differentiate between your statements.
- **Effect** – Use **Allow or Deny** to indicate whether the policy allows or denies access.
- **Principal** – Indicate the account, user, role, or federated user to which you would like to allow or deny access. **If you are creating a policy to attach to a user or role, you cannot include this element**. The principal is implied as that user or role. See EC2 Profile as example.
- **Action** – Include a list of actions that the policy allows or denies.
- **Resource** – Specify a **list of resources** to which the actions apply.
- **Condition** (Optional) – Specify the circumstances under which the policy grants permission.

More information about Principal: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html (02/06/2021)

Principal in IAM Roles / Resource-Based Policies

fmorenod.co

In IAM Roles



In Resource-Based Policies

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"PublicRead",
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject","s3:GetObjectVersion"],
      "Resource":["arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"]
    }
  ]
}
```

More information about Principal: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_terms-and-concepts.html and https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html (14/06/2021)

IAM vs Resource-Based Permissions

Identity-Based (IAM) Permissions

Larry

Can Read, Write, List

On Resource X

Sam

Can Read

On Resources Y, Z

Managers

Can List

On Resources X, Y, Z

Admins

Can do All Actions

On All Resources

Resource-Based Permissions

Resource X

Bob: Can Read, Write, List
Jim: Can Read, List
Sara: Can List
Doug: Can Read, Write, List
etc...

Resource Y

Bob: Can Read, Write, List
Larry: Can Read
Sam: Can Write, List

etc...

Rationale:

- Central Permission Repository
- Multi-account Government
- Different Identity management instead of IAM

- Resource-Based Permissions: i.e. S3.
- Better option to have a central gov over IAM Groups and users to access S3 ?

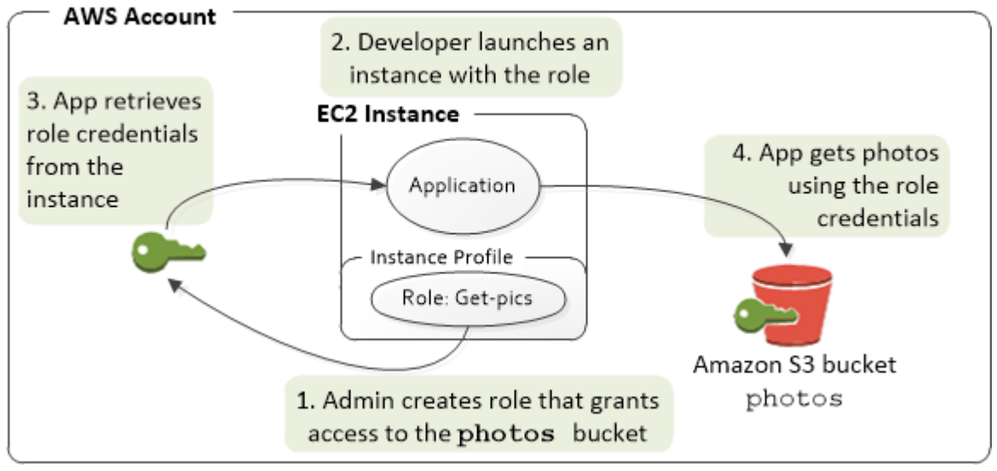
Services that work with IAM

Service	Actions	Resource-level permissions	Resource-based policies	ABAC	Temporary credentials	Service-linked roles
Amazon Elastic Compute Cloud (Amazon EC2)	Yes	Partial	No	Yes	Yes	Partial (Info)
Amazon EC2 Auto Scaling	Yes	Yes	No	Yes	Yes	Yes
EC2 Image Builder	Yes	Yes	No	Yes	Yes	Yes
Amazon EC2 Instar						
Amazon ElastiCach						
AWS Elastic Beanst						
Amazon Elastic Blc (Amazon EBS)						
Amazon Elastic Co Registry (Amazon I						
Amazon Elastic Co Registry Public (An Public)						
Amazon Elastic Co (Amazon ECS)						

Service	Actions	Resource-level permissions	Resource-based policies	ABAC	Temporary credentials	Service-linked roles
Amazon Simple Queue Service (Amazon SQS)	Yes	Yes	Yes	Partial	Yes	No
Amazon Simple Storage Service (Amazon S3)	Yes	Yes	Yes	Partial (Info)	Yes	Partial (Info)
Amazon Simple Storage Service (Amazon S3) Object Lambda	Yes	Yes	No	No	Yes	No
Amazon Simple Storage Service (Amazon S3) on AWS Outposts	Yes	Yes	Yes	No	Yes	Yes
Amazon Simple Workflow Service (Amazon SWF)	Yes	Yes	No	Yes	Yes	No
AWS SimSpace Weaver	Yes	Yes	No	Yes	Yes	No
AWS Site-to-Site VPN	Yes	Yes	No	No	Yes	Yes
AWS Snowball	Yes	No	No	No	Yes	No
AWS Snowball Edge	Yes	No	No	No	Yes	No
AWS Snow Device Management	Yes	Yes	No	Yes	Yes	No
AWS SQL Workbench						

More information on https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-services-that-work-with-iam.html (01/07/2024)

Use Case: EC2 Profile




```hcl
#IAM Role
resource "aws_iam_role" "worker-node-role" {
  name = "${var.common_name}-worker-nodes-role"

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

```hcl
resource "aws_iam_instance_profile" "worker-node" {
  name = "eks-worker-node"
  role = "${aws_iam_role.worker-node-role.name}"
}
```

```hcl
#Launch Configuration
resource "aws_launch_configuration" "worker" {
  iam_instance_profile = "${aws_iam_instance_profile.worker-node.name}"
  image_id             = "${data.aws_ami.eks-worker.id}"
  instance_type        = "${var.worker_instance_type}"
  name_prefix          = "worker-node"
  security_groups      = ["${aws_security_group.worker-node-sg.id}"]
  user_data_base64     = "${base64encode(local.worker-node-userdata)}"

  lifecycle {
    create_before_destroy = true
  }
}
```

More information on https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2_instance-profiles.html (01/07/2024)

IAM Best Practices

1. Require human users to use federation with an identity provider to access AWS using temporary credentials
2. Require workloads to use temporary credentials with IAM roles to access AWS
3. Require multi-factor authentication (MFA)
4. Update access keys when needed for use cases that require long-term credentials
5. Follow best practices to protect your root user credentials
6. Apply least-privilege permissions
7. Get started with AWS managed policies and move toward least-privilege permissions
8. Use IAM Access Analyzer to generate least-privilege policies based on access activity
9. Regularly review and remove unused users, roles, permissions, policies, and credentials
10. Use conditions in IAM policies to further restrict access
11. Verify public and cross-account access to resources with IAM Access Analyzer
12. Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions
13. Establish permissions guardrails across multiple accounts
14. Use permissions boundaries to delegate permissions management within an account

More information on https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html (16/07/2024)

- (…) is a web service that enables you to **request temporary, limited-privilege credentials** for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users).

- Federation Identity is “..linking identity ..across multiple ID management Systems…”.

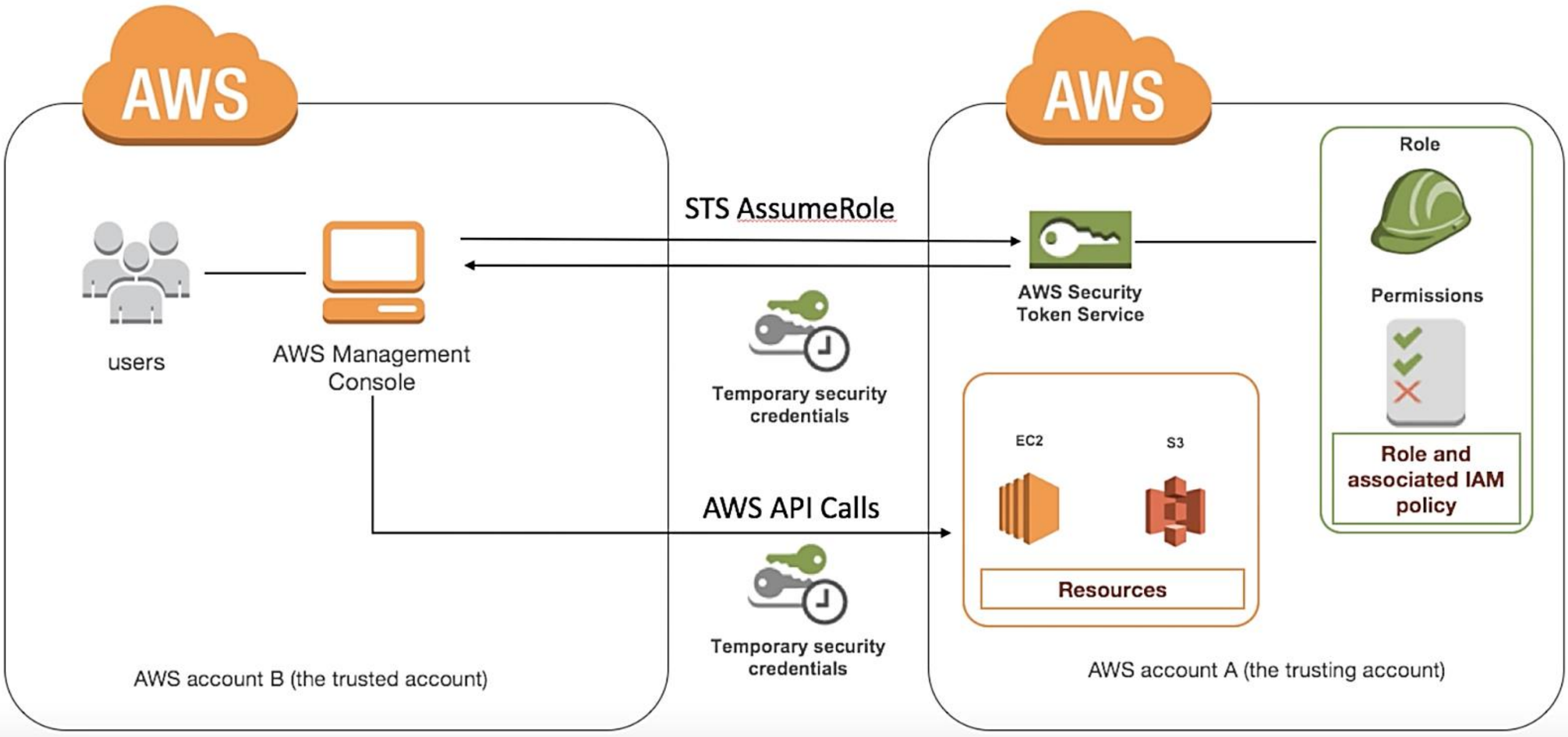
Taken from https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html (16/07/2024) and adapted from https://en.wikipedia.org/wiki/Federated_identity (09/06/2021)

STS API: AssumeRole

- Returns a set of **temporary security credentials** that you can use to access AWS resources that you might not normally have access to.

- These temporary credentials consist of an **access key ID, a secret access key, and a security token**. Typically, you use AssumeRole for cross-account access or federation.

STS API: AssumeRole



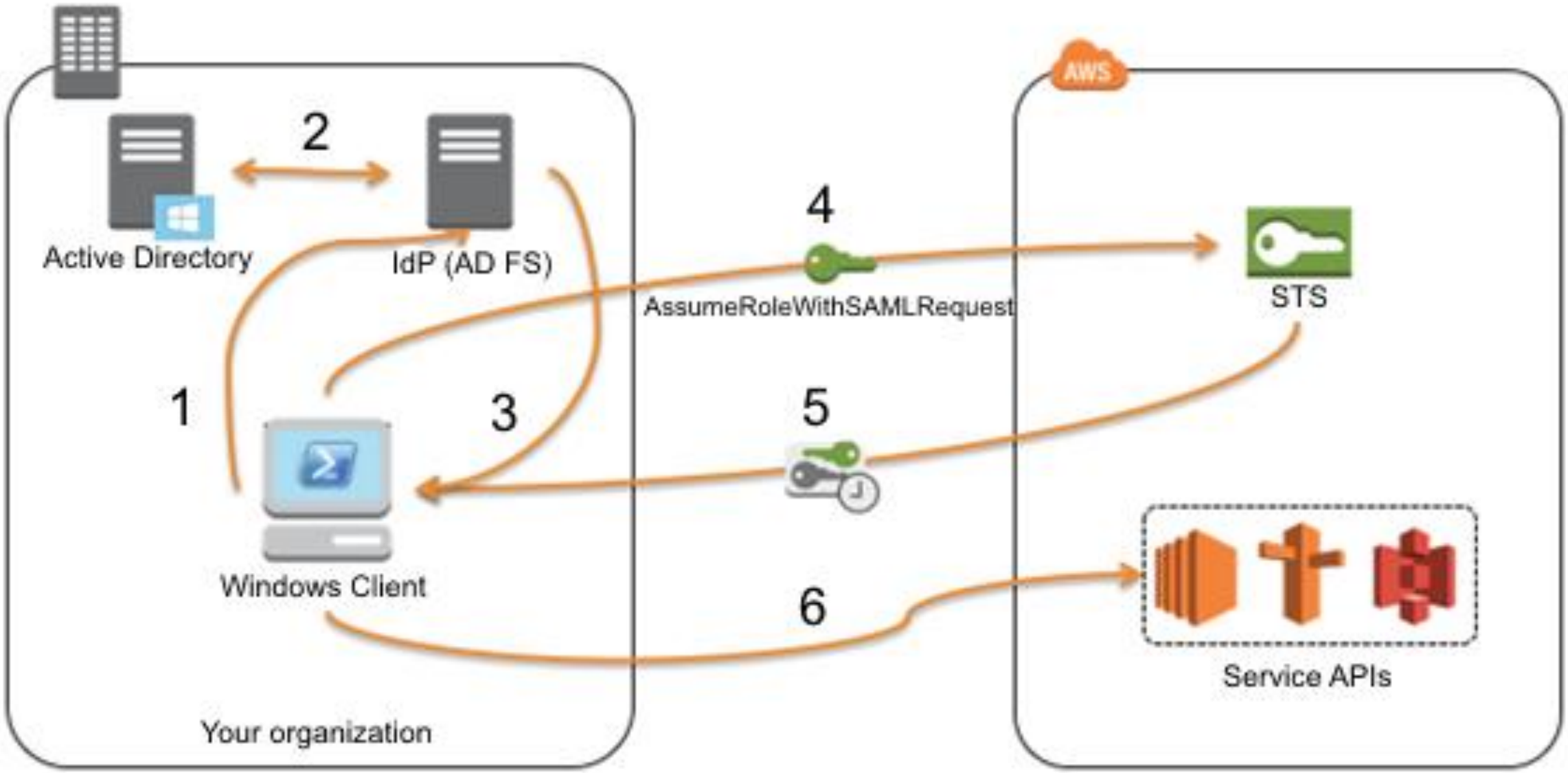
Taken from AWS IAM & Its Features — Day 1 (https://medium.com/everythingatonce/aws-iam-its-features-day-1-5c5bb6a075c6) (08/05/2020)

STS API: AssumeRoleWithSAML

- Returns a set of **temporary security credentials** for users who have been authenticated via a SAML authentication response.

- This operation provides a mechanism for tying an **enterprise identity store or directory to role-based AWS access** without user-specific credentials or configuration.

STS API: AssumeRoleWithSAML



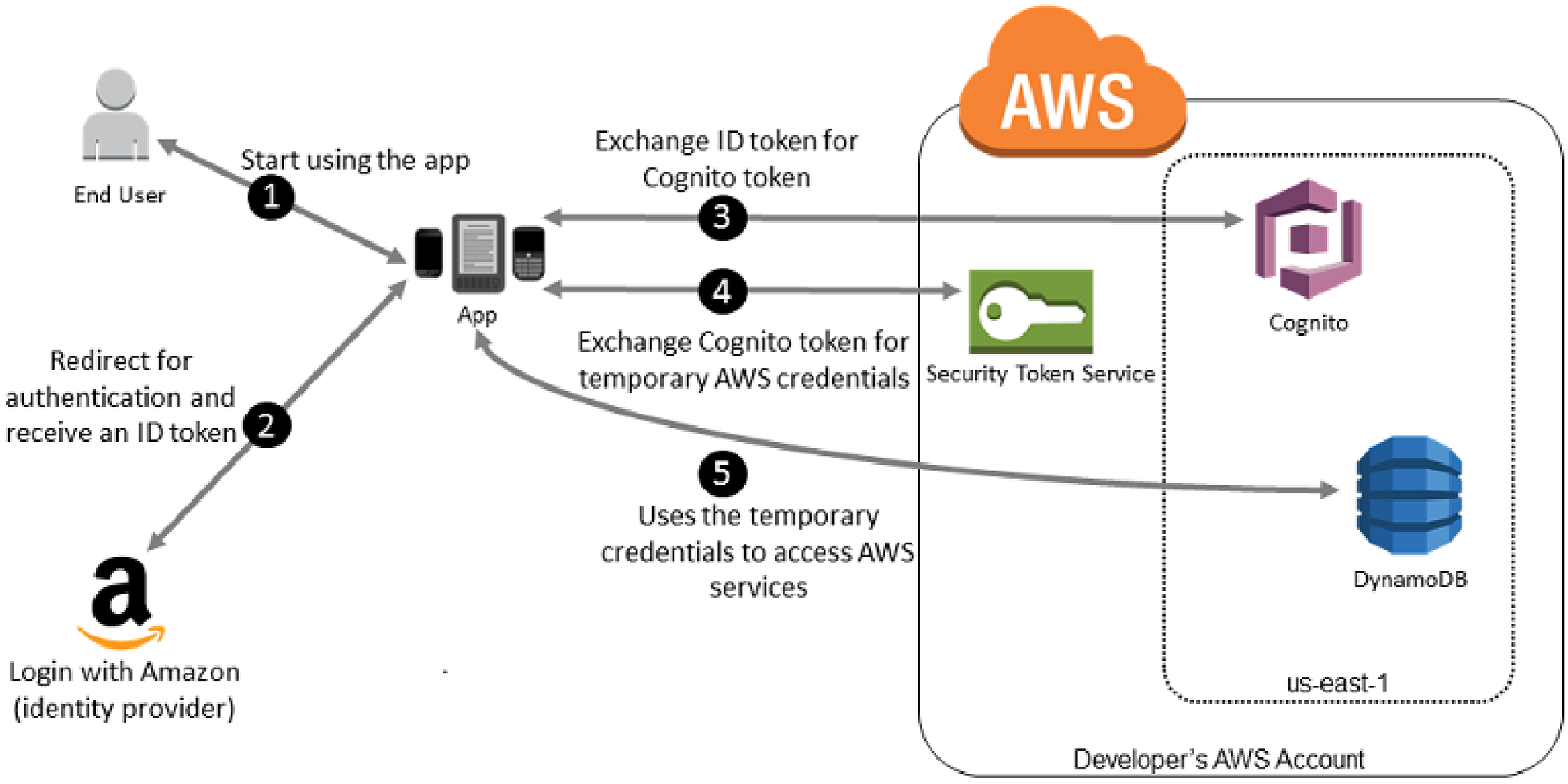
For an exercise using Powershell, visit https://aws.amazon.com/blogs/security/how-to-set-up-federated-api-access-to-aws-by-using-windows-powershell/ (16/07/2024)

STS API: AssumeRoleWithWebIdentity

- Returns a set of temporary security credentials for users who have been authenticated in a mobile or web application with a web identity provider.

- Example providers include Amazon Cognito, Login with **Amazon, Facebook, Google, or any OpenID** Connect-compatible identity provider.

STS API: AssumeRoleWithWebIdentity



Amazon Cognito has 2 Flows: Basic and Enhanced. For more information, visit
https://docs.aws.amazon.com/cognito/latest/developerguide/authentication-flow.html and
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_oidc_cognito.html (16/07/2024)