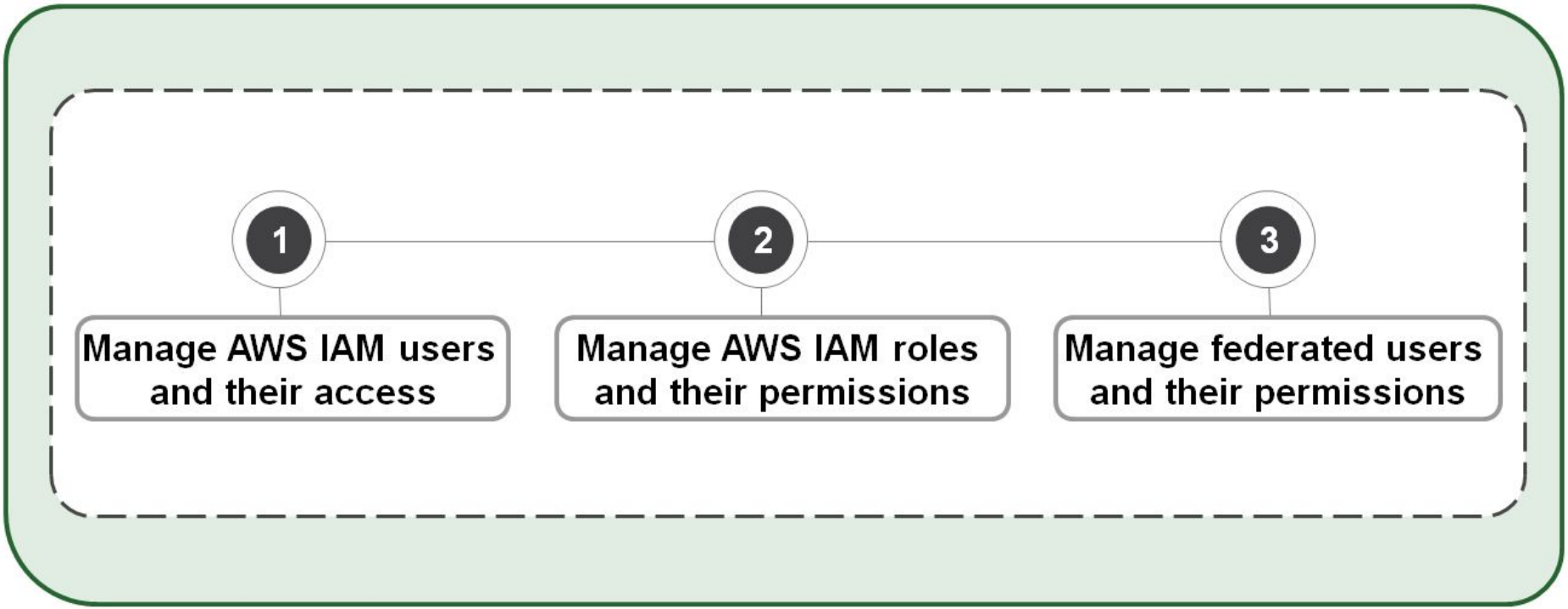
Amazon Identity and Access Management (IAM)

Nov - 2017

AWS IAM


1
Manage AWS IAM users and their access
2
Manage AWS IAM roles and their permissions
3
Manage federated users and their permissions

IAM Authentication

Authentication

- AWS Management Console
  - User Name and Password

IAM User

IAM Authentication for API or CLI

Authentication

- **AWS CLI or SDK API**
  - Access Key and Secret Key

IAM User

Access Key ID: AKIAIOSFODNN7EXAMPLE
Secret Access Key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

AWS CLI

```
[redacted]:~ [redacted]$ aws configure
AWS Access Key ID [****************O22A]:
AWS Secret Access Key [****************4m8i]:
Default region name [ap-southeast-1]:
Default output format [json]:
```

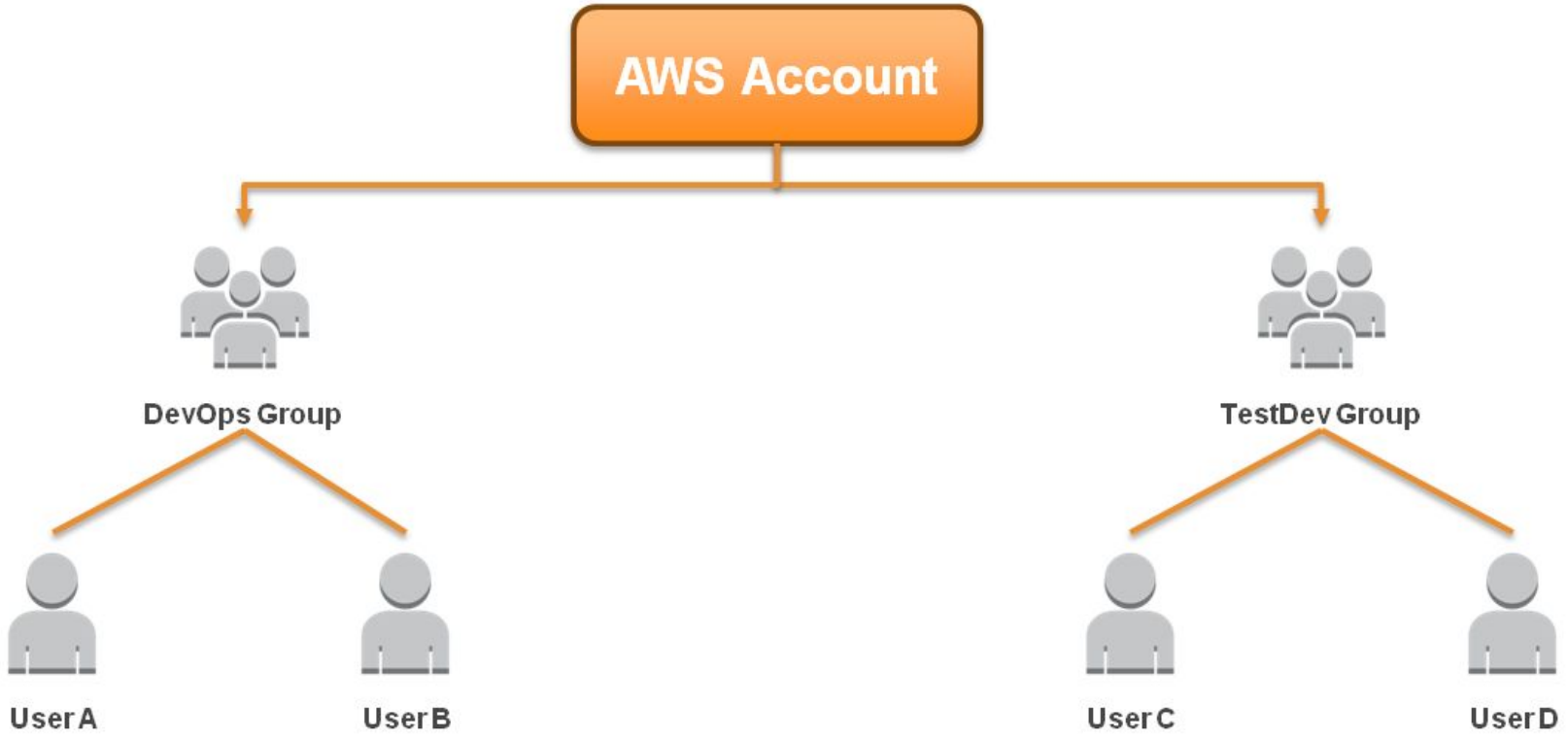
AWS SDK & API

Java

Python

.NET

AWS User and Groups



Principal

- Is an entity in AWS that can access resources and perform actions
  a. AWS Account (root user)
  b. IAM User
  c. IAM Role

Key concept

- User
  - Access Key and Secret Key
  - MFA
  - Password
  - 5000 per account

- Groups
  - Policies
  - Assigned user list
  - 100 per account

- Roles
  - Access to different AWS account user
  - Access to different AWS resources
  - 250 per account

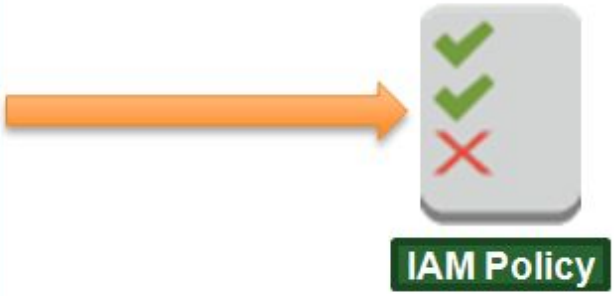
IAM Authorization

Authorization

- Policies:
  - Are JSON documents to describe permissions.
  - Are assigned to Users, Groups or Roles.

IAM Policy Elements

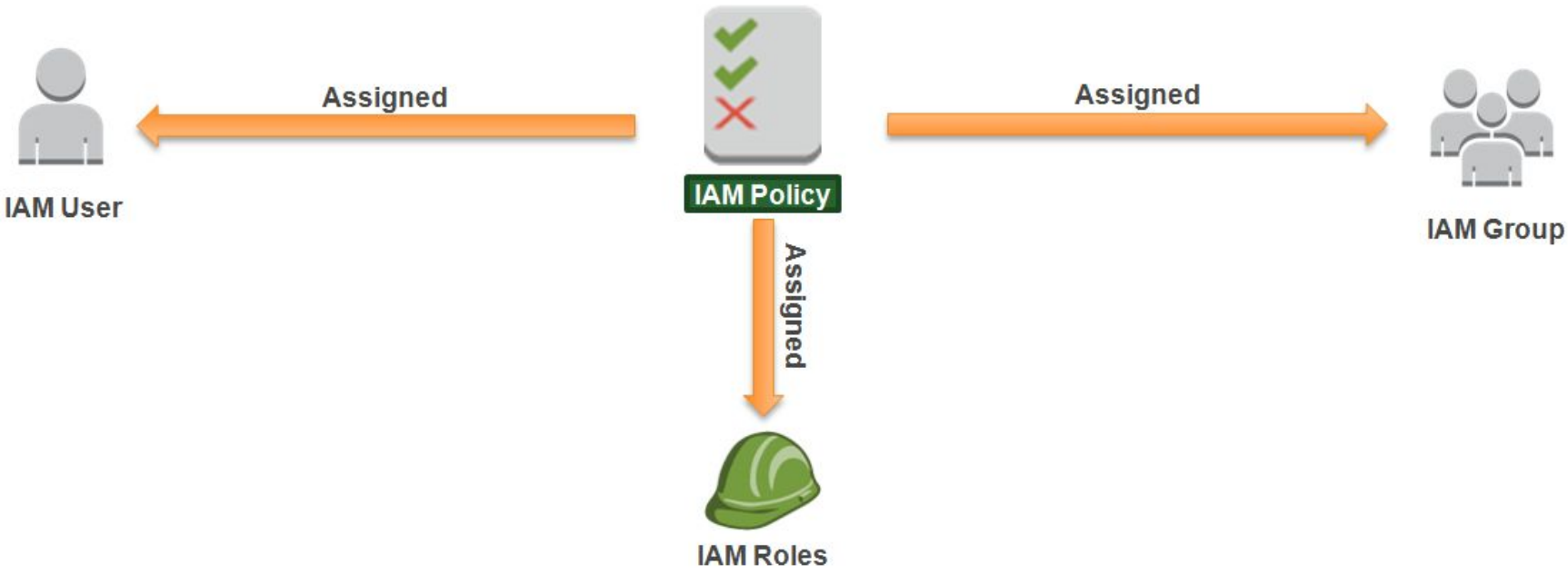
```
{
  "Version": "2012-10-17",
  "Statement": [
   {
     "Sid": "Stmt1453690971587",
            "Action": [
             "ec2:Describe*",
             "ec2:StartInstances",
             "ec2:StopInstances"
             ],
             "Effect": "Allow",
             "Resource": "*",
             "Condition": {
               "IpAddress": {
                      "aws:SourceIp": "54.64.34.65/32"
               }
             }
   },
   {
             "Sid": "Stmt1453690998327",
             "Action": [
             "s3:GetObject*"
            ],
       "Effect": "Allow",
       "Resource": "arn:aws:s3:::example_bucket\*"
          }
           ]
}
```



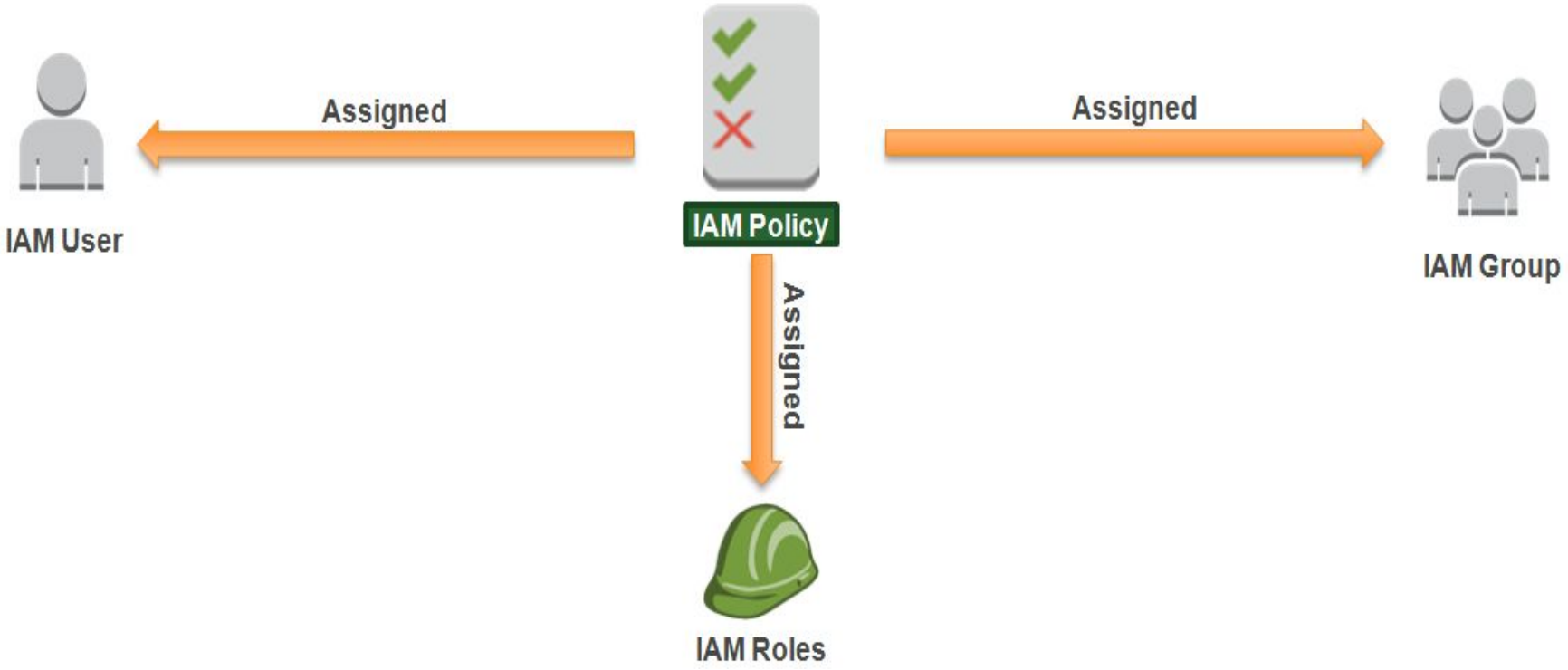
Policy

- Defines the permission for Role
- Two policies
  a. Trust Policy - specifies who is allowed to assume the role
  b. Permission Policy
     i. what permissions are allowed
     ii. which actions on which resources

AWS IAM Policy Assignment



AWS IAM Policy Assignment

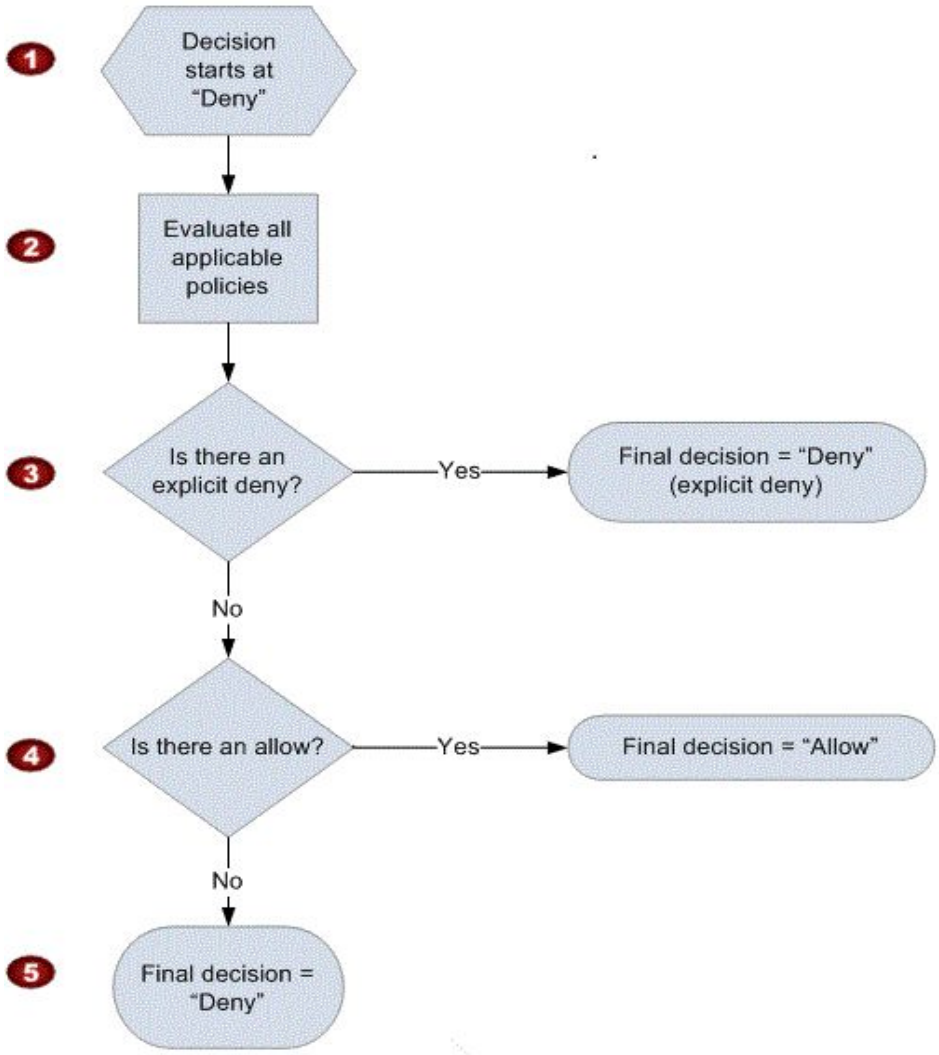


Permissions

- Permission
  - User-based
  - Resources-based
- Policy
  - Effects
  - Action
  - Resources

Image Resource: http://docs.aws.amazon.com/IAM/latest/UserGuide/policies_overview.html

IAM Policy Lifecycle Evaluation Logic



Mix of Allow and Deny Rules

Role

AWS IAM Roles

- An IAM role uses a policy.
- An IAM role has no associated credentials.
- IAM users, applications, and services may assume IAM roles.

Roles

A role is essentially a set of permissions that grant access to actions and resources in AWS. These permissions are attached to the role, not to an IAM user or group.

Roles can be used by the following:

- An IAM user in the same AWS account as the role
- An IAM user in a different AWS account as the role
- A web service offered by AWS such as Amazon Elastic Compute Cloud (Amazon EC2)
- An external user authenticated by an external identity provider (IdP) service that is compatible with SAML 2.0 or OpenID Connect, or a custom-built identity broker.

Roles and Use Cases

- Role is **intended to be assumed by** anyone who needs it
- Does not have its own password or access keys

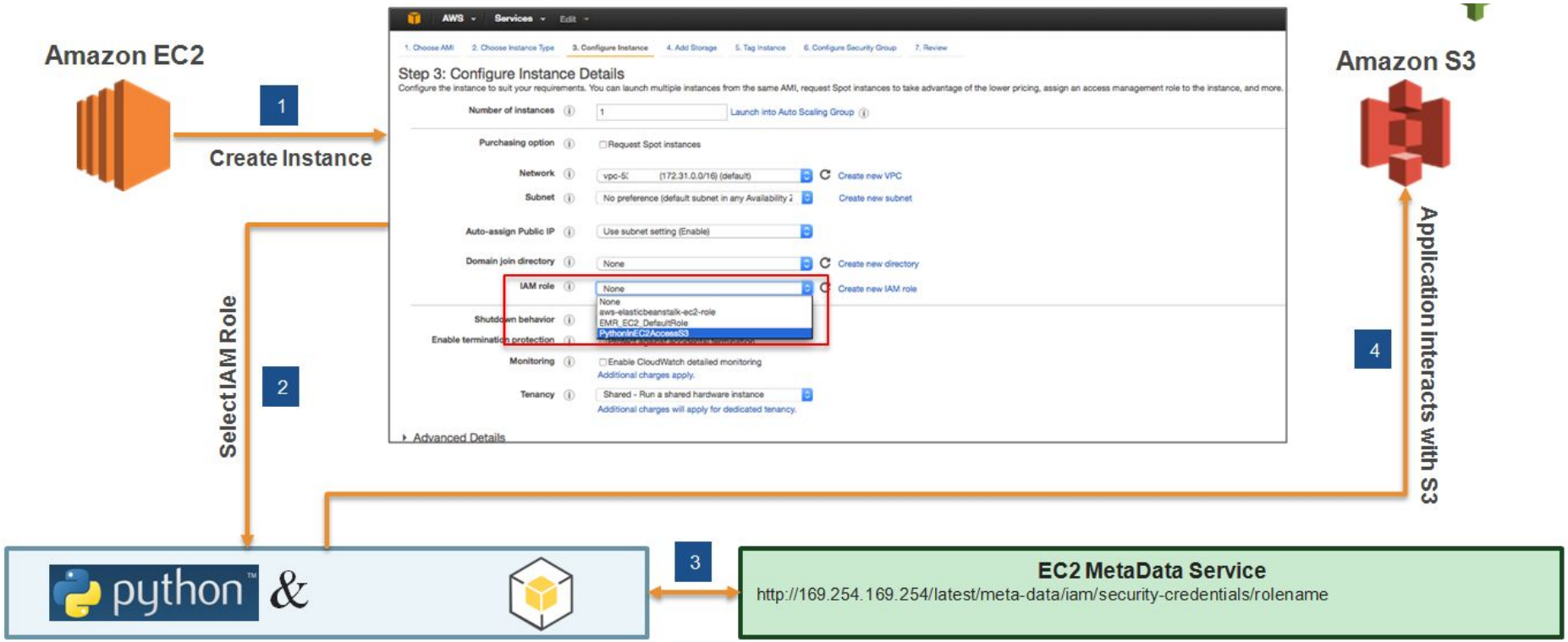
When to use Roles ?

- Use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources.
  - You do not want to share the aws access keys and secret keys
  - Provide access to users in different AWS account to access resources in your own account
  - Want to provide access to third party identity providers
    - To provide access to Active Directory Users (SAML 2.0)
    - Google Account , Amazon Account (OpenID Connect)
  - Third party account
  - Want to allow a mobile app to use AWS resources, but do not want to embed AWS keys within the app

Application Access to AWS Resources

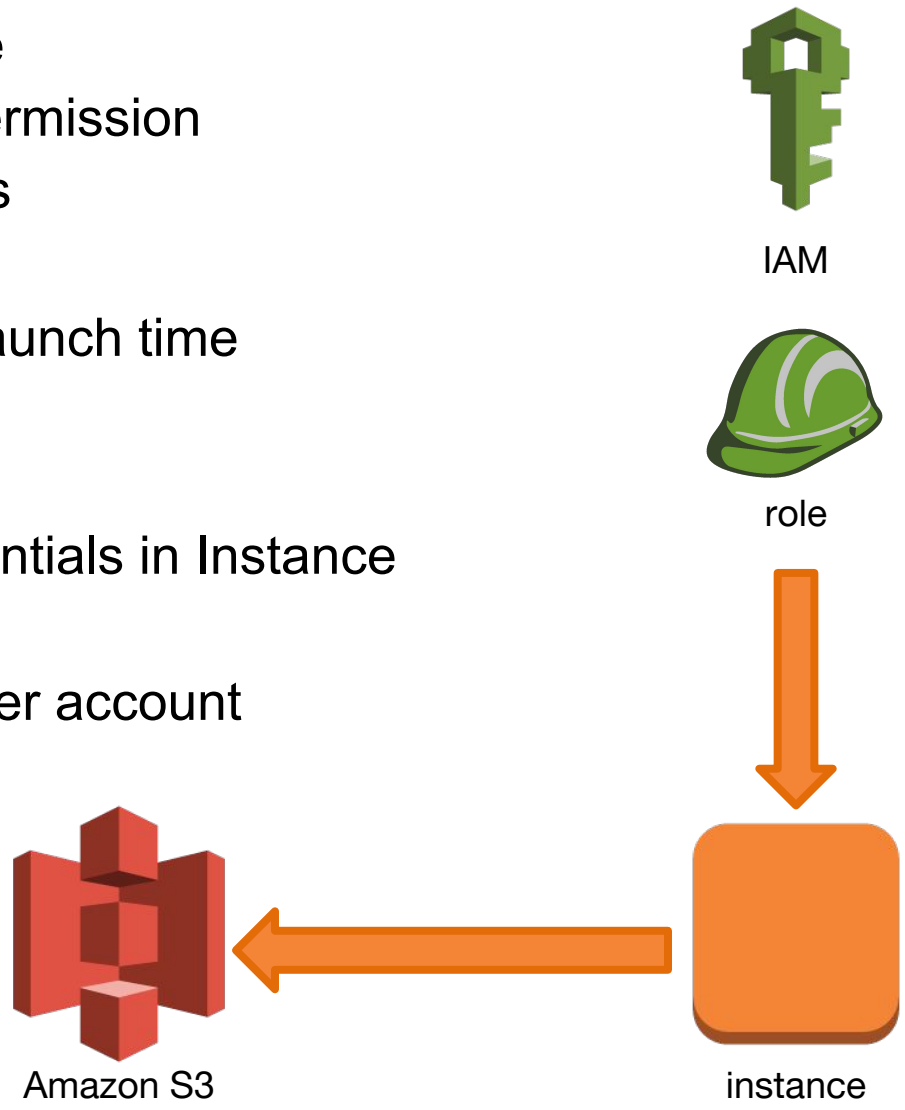
- Python application hosted on an Amazon EC2 Instance needs to interact with Amazon S3.
- AWS credentials are required:
  - ~~Option 1: Store AWS Credentials on the Amazon EC2 instance~~.
  - Option 2: Securely distribute AWS credentials to AWS Services and Applications.

IAM Role Instance Profile

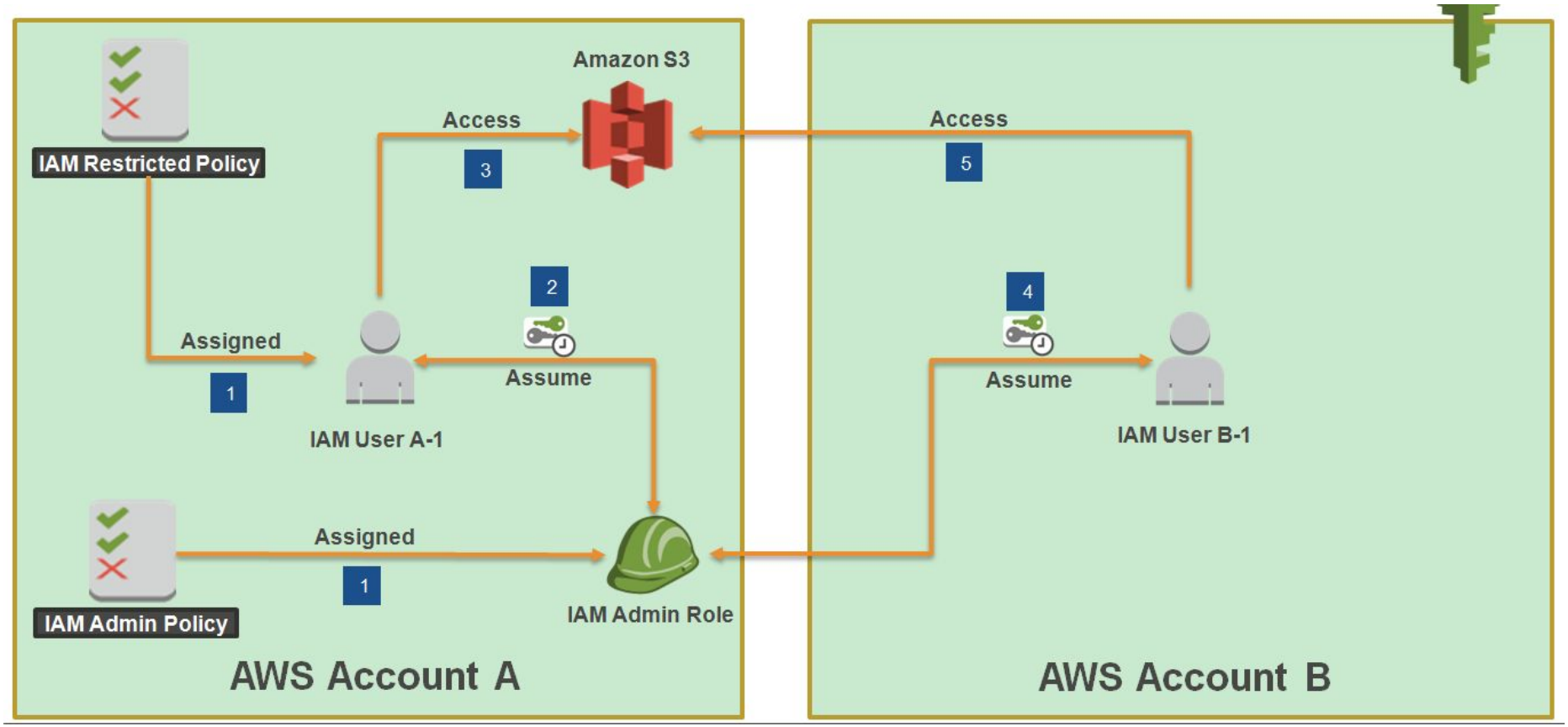


IAM Instance Profile

- Permission to EC2 instance
  - Assigning role based permission
  - Access to AWS services
- Assign Instance Profile at launch time
- Recommend by AWS
  - Avoid storing IAM credentials in Instance
- Limit 100 Instance Profile per account



AWS IAM Roles – Assume Role



Cross Account Access

Granting access to resources in one account to a trusted principal in a different account is often referred to as *cross-account access*.

- By using Role (acts as a proxy)
- Through Resource Based Policy

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"AddPerm",
      "Effect":"Allow",
      "Principal": "*",
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::examplebucket/*"]
    }
  ]
}
```

IAM Best Practices

- Reduce use of root account
  - Use IAM user
- Enable MFA
  - Extra layer of security
- Grant least privilege
  - Create custom policy
  - Remove permission if not needed
- Use IAM Roles for EC2
  - Avoid IAM user credentials
  - Recommended by AWS
- Enable Auditing using CloudTrail
  - Activity entry of each user
- Configure a strong password policy
- Remove unnecessary credentials

End of Module

Q & A