DAY – 10
AWS IDENTITY & ACCESS MANAGEMENT

AWS Architecture and Design

1. Day 1 Overview of Cloud Computing
2. Day 2 Overview of AWS
3. Day 3 Amazon EC2*
4. Day 4 Amazon EBS *
5. Day 5 Amazon CloudWatch *
6. Day 6 Amazon S3*
7. Day 7 Amazon Elastic Load Balancer *
8. Day 8 Amazon Auto Scaling *
9. Day 9 Amazon VPC *
10. **Day 10 Amazon IAM ***
11. Day 11 Amazon RDS
12. Day 12 Amazon Route 53 *
13. Day 13 Amazon DynamoDB* & Glacier
14. Day 14 Amazon Cloudfront* & Import Export & Amazon SES *
15. Day 15 Amazon ElasticBeanStalk & Amazon Cloudformation & Amazon OpsWorks
16. Day 16 AWS Economics & AWS Account Overview *
17. Day 17 AWS Architecture
18. Day 18 AWS Certification Preparation

[With Hands on Demo]

AWS Identity & Access Management

AWS IAM

- → What is IAM?
- → Key IAM Terminology
  - → Users
  - → Groups
  - → Roles
  - → STS
- → Demo

AWS IAM

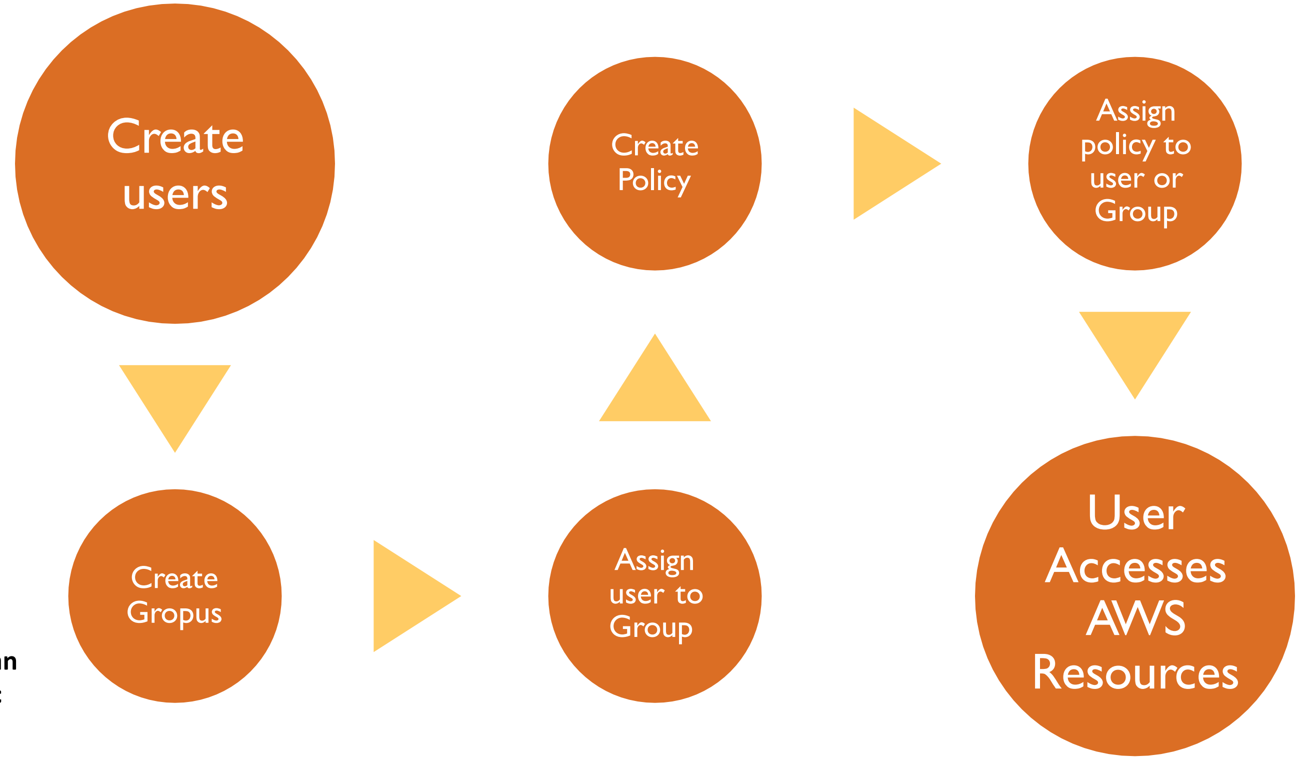
AWS Identity and Access Management (IAM) enables you to securely control access to AWS services and resources for your users.

- Create & manage Users
- Create & manage Groups
- Create & Manage Roles
- Create & Manage Policies
- Assign Policy to User / Group / Role
- Identity Federation between organization and AWS
- Fine grained Access to AWS Resources
- Free

http://aws.amazon.com/iam/

AWS IAM

- Create IAM user identities
- Organize IAM users in groups
- Create groups to easily manage permissions for multiple IAM users under your AWS account
- Policy: Centralize control of user access
- Control which operations users can perform on specific AWS resource
- Add conditions for finer access



http://aws.amazon.com/iam/

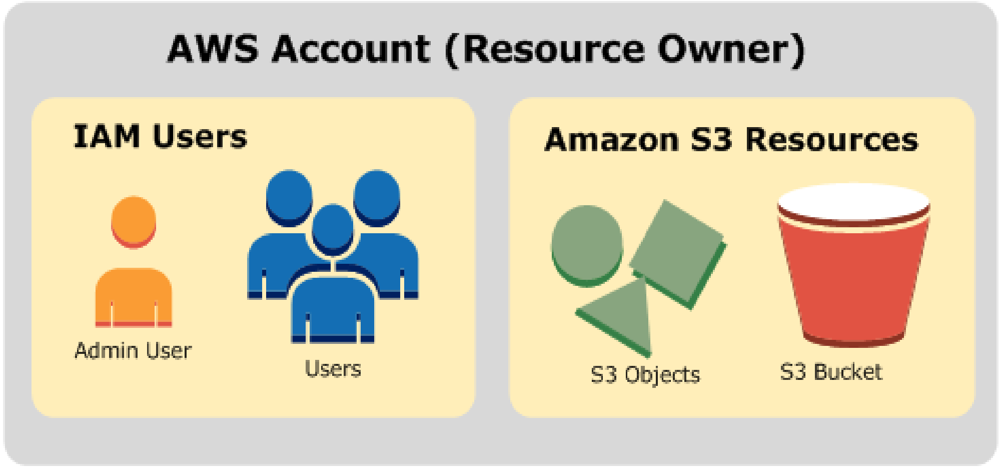
AWS Identity & Access Management (IAM)

→ The account you used to register with AWS is Root Account
  » Has full access to AWS

→ Create users for various AWS service access
  → The user can login with
    » Login/password (optional)
    » Access / secret keys (for APIs) (optional)
    » (V)MFA devices (optional)

→ Define policies to control access to AWS APIs

→ AWS Policy can be integrated with
  » S3: Policies on objects and buckets
  » Resource Action wise Access Control



→ Manage Credentials
  » Assign security credentials to your IAM users, and rotate and/or revoke these credentials as desired.
  » Define account-level: password complexity policies.
  » Request temporary security credentials with configurable expiration and permissions for your IAM users, federated users or applications.

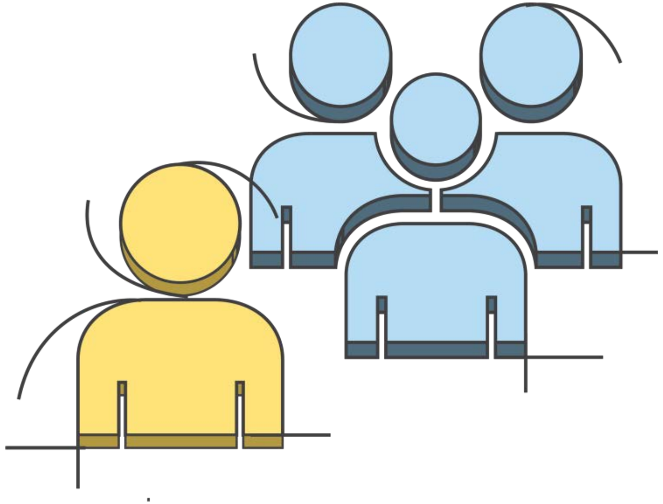
http://www.slideshare.net/AmazonWebServices/in-depth-aws-iam-and-vpc

IAM Users

Create Users as per Access Need

Create their keys, passwords, tokens

Login with IAM URL



ntity-and-access-management-best-practices

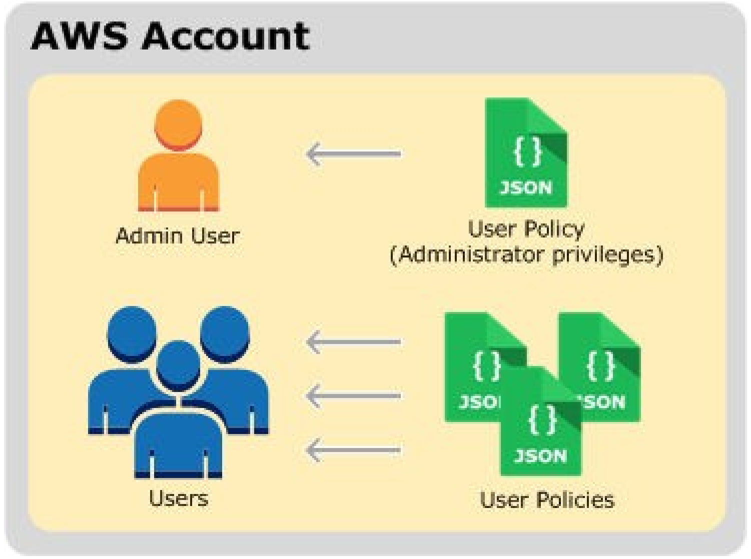
IAM Users (Contd.)

Each user has own credentials

Individual Permission

Can be part of group as well work independent

Can be as powerful as Administrator



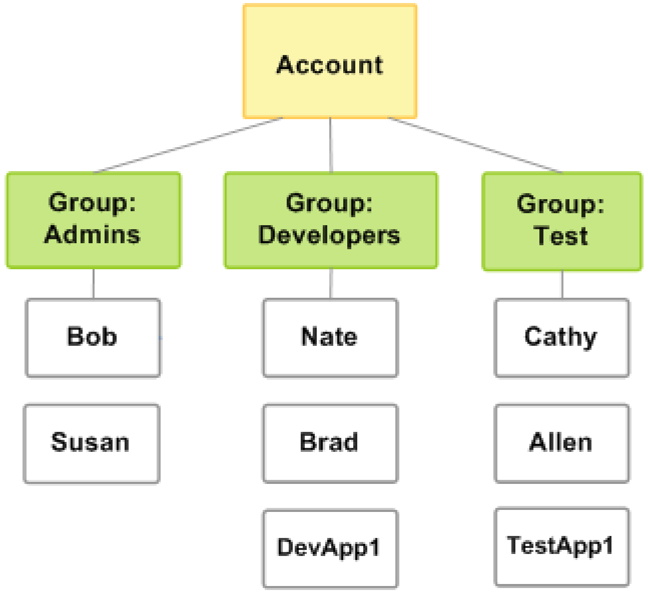
http://www.slideshare.net/AmazonWebServices/top-10-aws-identity-and-access-management-best-practices

IAM Groups

Create multiple Group as per need

Add users to groups

Define policy and attach to group



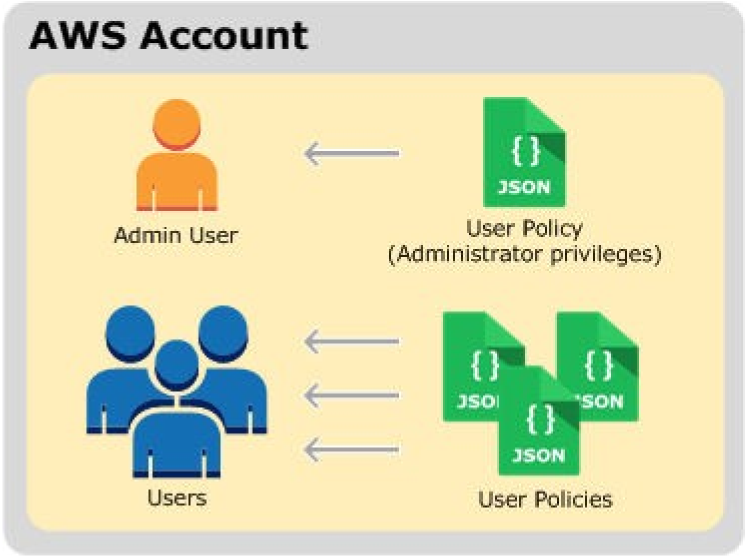
IAM Groups (Contd.)

Easily manage permissions

Permission is union of all

Single change applies to all users of groups

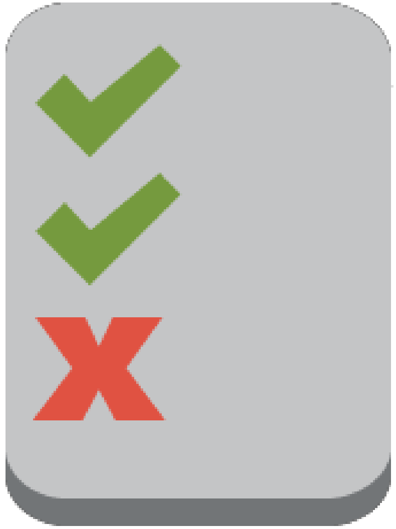
Create various groups based on department and assign access



http://www.slideshare.net/AmazonWebServices/top-10-aws-identity-and-access-management-best-practices

Policy

- Identify permissions to be assigned
- Create a Policy
- Use ready Policy Templates
- Assign Policy to User/ Group / Role



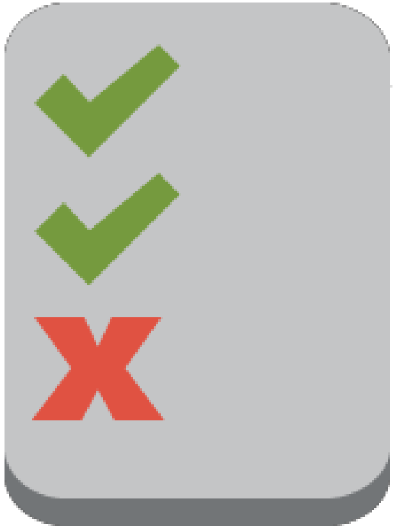
Policy

Granular Control

Easy to manage changes

Ready Templates

Works for S3, IAM, SQS



http://www.slideshare.net/AmazonWebServices/top-10-aws-identity-and-access-management-best-practices

Policy

IAM Policy is in JSON Format. Below are the attributes of IAM Policy.

→**Actions**: What action policy allows. E.g. ListBucket. If you do not allow any action then its denied.

→**Resources**: On which resource you allow action on. E.g. AWS S3 bucket

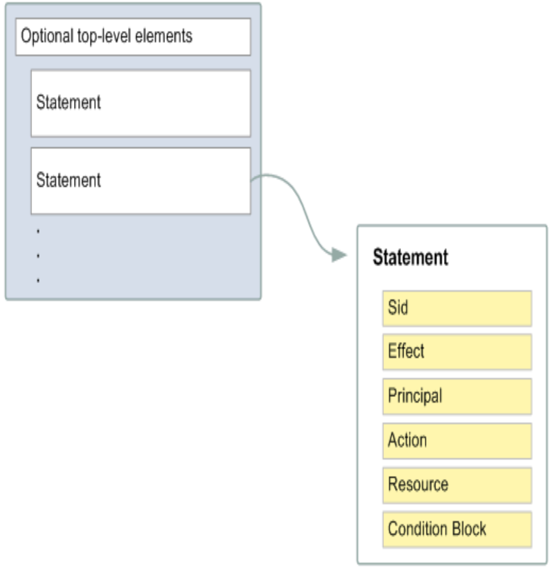
→**Effect** : What will be the effect when user access resource. [Allow / Deny]

→**Principals**: Defines who is allowed to access this resource such as users, account and roles, that is allowed or denied access to a resource. More for S Topic, S3 Buck Glacier Vault. For IAM user based policy principal is IAM user (Group.

→**Conditions**: Grants permissions to a different set of resources under a spec condition

» E.g. : time, transport, source ARN, source IP, UserAgent, Referrer

http://docs.aws.amazon.com/IAM/latest/UserGuide/AccessPolicyLanguage_ElementDescriptions.html



Policy

→ **Principal**

<!-- Everyone (All users) --> "Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}
Individual user --> "Principal":"AWS":"arn:aws:iam::123456789012:user/Username"

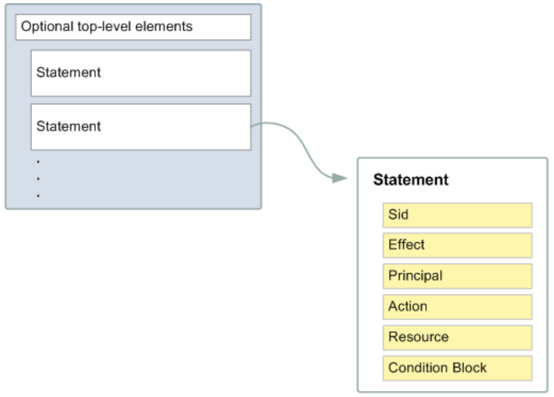
→ Resource

"Resource":"arn:aws:sqs:us-east-1:090807060504:queue1"
"Resource":"arn:aws:iam::989898989898:user/John"

```
"Resource": [
     "arn:aws:s3:::mybucket",
     "arn:aws:s3:::mybucket/*"
   ]
```

→ Effect

"Effect":"Allow"
"Effect":"Deny"



Policy

```
{
        "Version": "2008-10-17",
        "Id": "Policy1388811135561",
        "Statement": [
                {
                        "Sid": "Stmt1388811069831",
                        "Effect": "Allow",
                        "Principal": {
                                "AWS": "*"
                        },
                        "Action": [
                                "s3:GetObjectAcl",
                                "s3:GetObject",
                                "s3:ListBucket"

                        ],
                        "Resource": ["arn:aws:s3:::testintel-static/*.jpg",
                                "arn:aws:s3:::testintel-static"]
                }
        ]
}
```

→ Policies can be written/stored relative to identities, or relative to resources
   » http://docs.amazonwebservices.com/IAM/latest/UserGuide/PermissionsOverview.html

→ Full processing details in case where multiple policies apply:
   » http://docs.amazonwebservices.com/IAM/latest/UserGuide/AccessPolicyLanguageEvaluationLogic.html

```
{
 "Statement": [
  {
   "Sid": "Stmt1378617635600",
   "Action": [
    "ec2:AttachVolume",
    "ec2:CopySnapshot",
    "ec2:CreateSnapshot",
    "ec2:CreateVolume",
    "ec2:DeleteSnapshot",
    "ec2:DeleteVolume",
    "ec2:DescribeSnapshotAttribute",
    "ec2:DescribeSnapshots",
    "ec2:DescribeVolumeAttribute",
    "ec2:DescribeVolumeStatus",
    "ec2:DescribeVolumes",
    "ec2:DetachVolume",
    "ec2:ModifySnapshotAttribute",
    "ec2:ModifyVolumeAttribute",
            "ec2:DescribeAvailabilityZones",
            "ec2:DescribeRegions",
    "ec2:ResetSnapshotAttribute"
   ],
   "Effect": "Allow",
   "Resource": "*"
  }
 ]
}
```

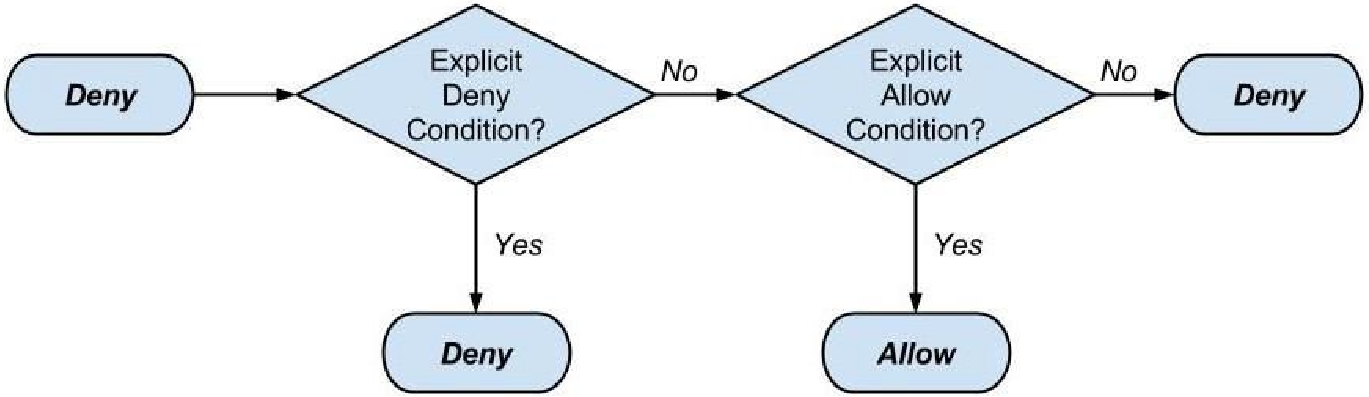
Conditions for Policy

Granular Access

Reduces Accidents

Works for AWS APIs / Services

Common / AWS specific conditions



Example Conditions

→ Condition for Instance Type / Date / Tag

```
"Condition": {
        "StringEquals": {
          "ec2:InstanceType": “nano“
               }
      },
```

```
"Condition": { "DateGreaterThan" : { "aws:CurrentTime" : "2013-12-15T12:00:00Z" } }
```

```
"Condition": { "StringEquals": { "ec2:ResourceTag/department": "dev" `} }
```

→ Allow users to access a "home directory" in Amazon S3

```
"Condition":{"StringLike":{"s3:prefix":["home/${aws:username}/*"]}}
```

→ Deny Access if IP not from specific Range

```
"Condition":{ "NotIpAddress":{ "aws:SourceIp":["192.0.2.0/24", "203.0.113.0/24"] } }
```

MFA

Extra layer of security

Virtual / Hardware

Additional key to be used with username / password

Account level as well for IAM users

https://aws.amazon.com/iam/details/mfa/

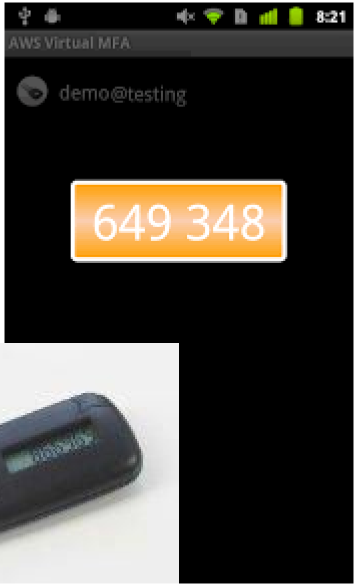
MFA

→ Integrated into
  » AWS Management Console
  » Billing pages on the AWS Portal
  » API protection!

→ Virtual MFA as well via OATH standard IETF RFC 6238
  » Android : AWS Virtual MFA; Google Authenticator
  » iPhone : Google Authenticator
  » Windows Phone : Authenticator
  » Blackberry : Google Authenticator

→ Hardware MFA with key Fob or Display Card
  » $12.99/$19.99

→ SMS MFA Device
  » Only for IAM users



IAM Roles

An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources.

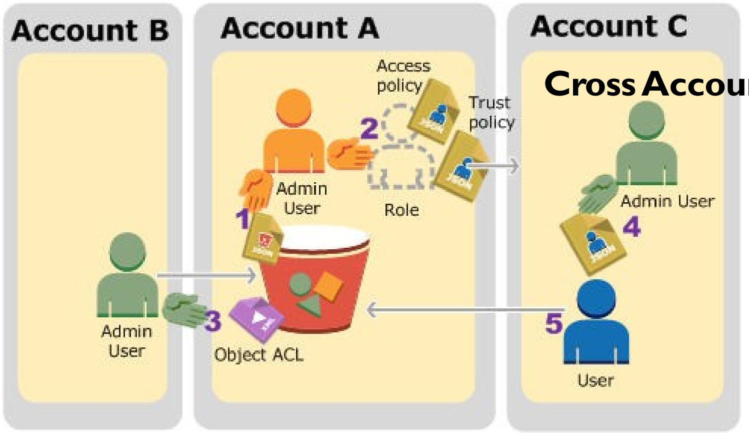
A role is essentially a set of permissions that grant access to actions and resources in AWS. These permissions are attached to the role, not to an IAM user or group.

Better security with AWS Keys

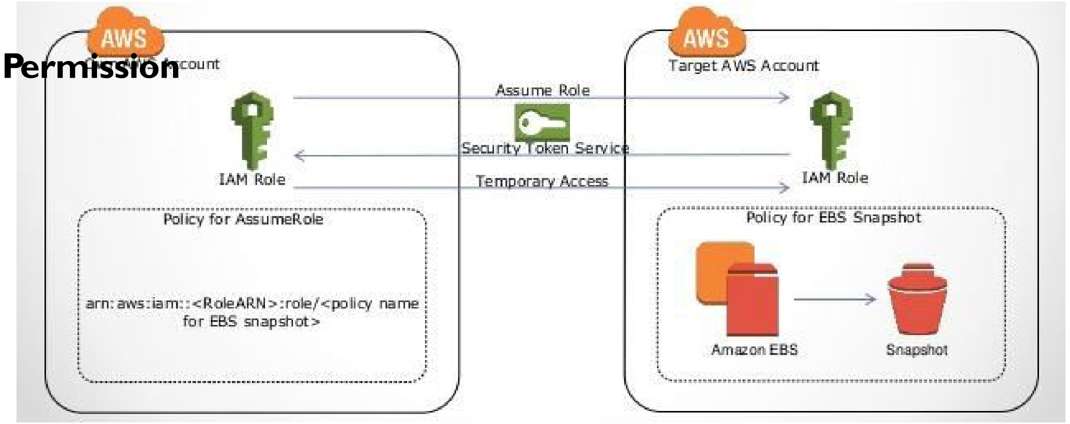
Automatic Key Rotation

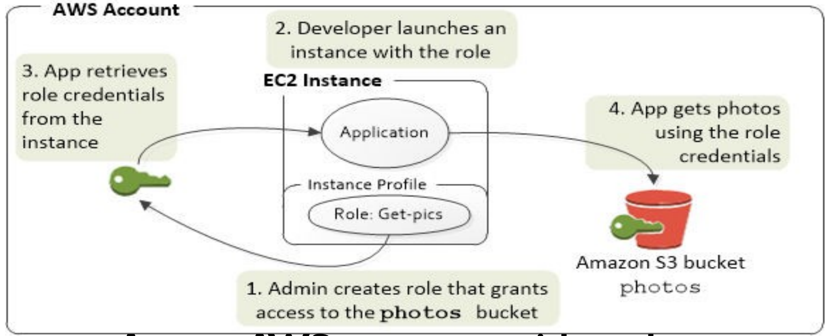
Delegate Access to Accounts

IAM Roles Use Cases

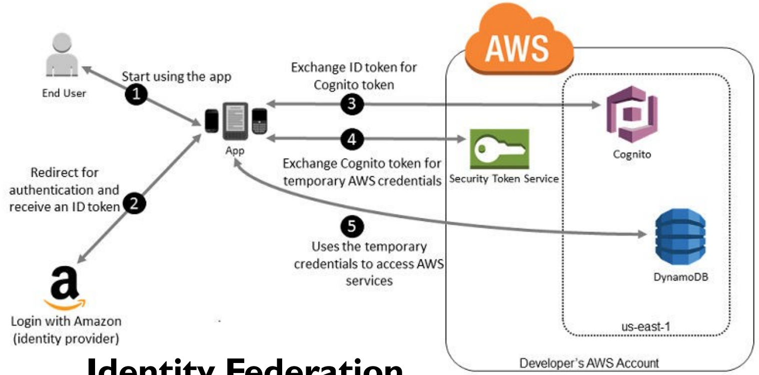


Cross Account Permission




AWS Account
2. Developer launches an instance with the role
3. App retrieves role credentials from the instance
EC2 Instance
Application
Instance Profile
Role: Get-pics
4. App gets photos using the role credentials
Amazon S3 bucket photos
1. Admin creates role that grants access to the photos bucket

Access AWS resources without keys


AWS
End User
Start using the app
1
App
Exchange ID token for Cognito token
3
4
Exchange Cognito token for temporary AWS credentials
Security Token Service
Cognito
Redirect for authentication and receive an ID token
2
5
Uses the temporary credentials to access AWS services
DynamoDB
Login with Amazon (identity provider)
us-east-1
Developer's AWS Account

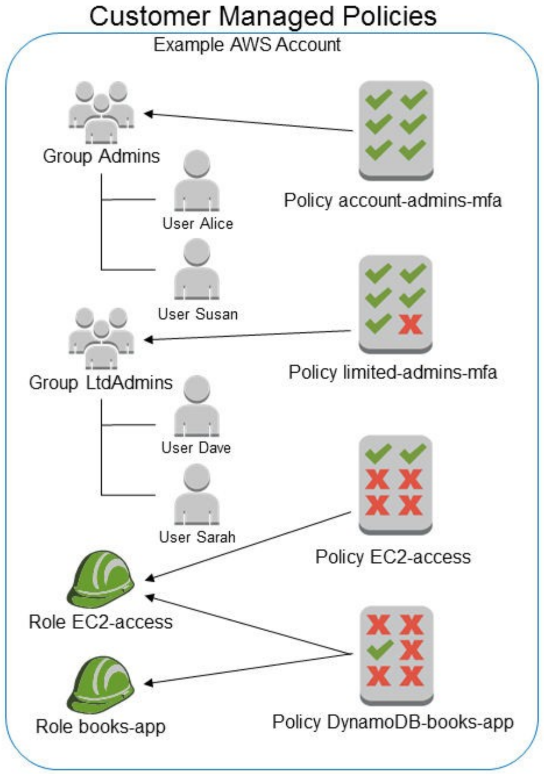
Identity Federation

Roles Users & Groups

IAM Users are account objects that allow an individual user to access AWS services with a set of credentials. You can assign various Policies to this individual user for AWS access

IAM Groups are objects that have permissions assigned to them via Policies allowing the members of the Group access to specific resources. You can use multiple users to same group and all users will have same permission.

IAM Roles are also objects created within IAM which have Policy permissions associated to them. They behave like a dummy Group where instead of being associated with Users as Groups are, Roles are assigned to instances at the time of launch.
Roles are similar to Groups in that they are objects created within IAM. Instead of providing permissions for Users, they provide permissions for instances,



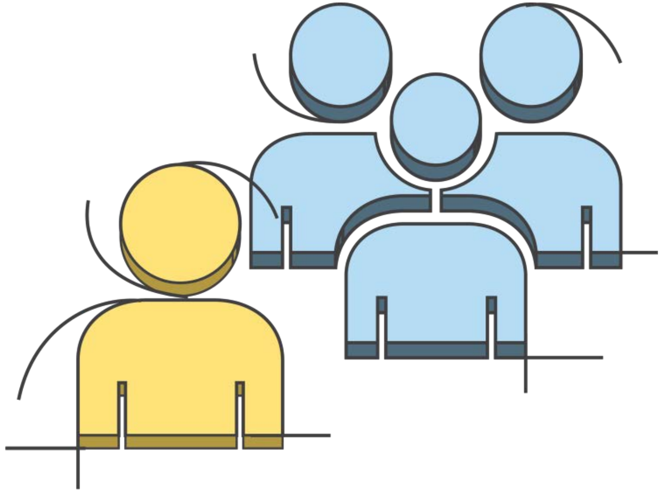
Root Account

The account used to register with AWS:

- It has potential of big threat
- Reduce/Remove use of Root to avoid any accidents
- Control usage granular way with IAM

→ Recommended Ways to use

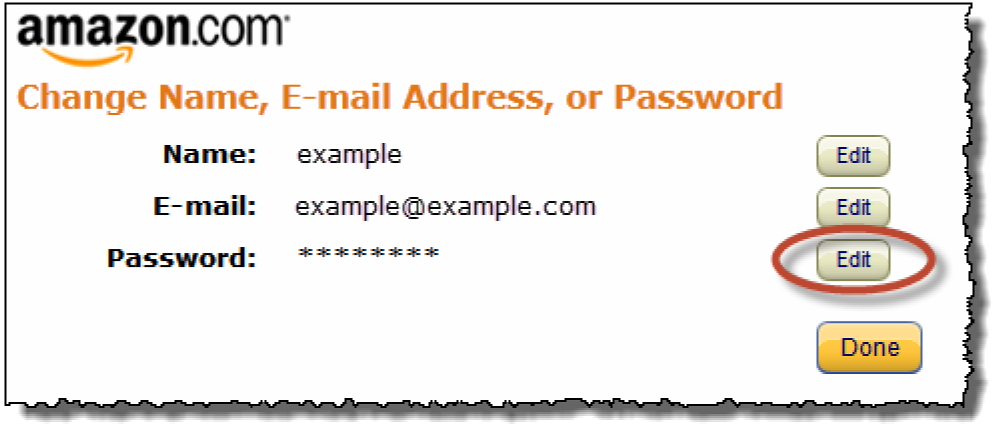
- » Security Credentials Page
  - → Delete Access Keys
  - → Activate a MFA Device
- » Ensure you have set a “strong” password



Manage Passwords

- → Always keep root account password with strong combination.
- → Enforce a strong Password Policy for IAM users.

- → How To steps:
  - » What is your company's password policy?
  - » You can configure:
    - → Minimum password length
    - → Require any combination of:
      - » One uppercase letter
      - » One lowercase letter
      - » One number
      - » One non-alphanumeric character



Change Name, E-mail Address, or Password

Name:	example	Edit
E-mail:	example@example.com	Edit
Password:	*********	Edit

  - » Grant IAM User Permission to rotate credentials
  - » Password change in IAM console
  - » IAM roles for EC2 automatically rotates credentials

Security Token Service (STS)

The AWS Security Token Service (STS) 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).

» Use AWS Security Token Service (AWS STS) to create and provide trusted users with temporary security credentials that can control access to your AWS resources
» They can be configured to last from a few minutes to several hours.
» Not stored with the user but are generated dynamically and provided to the user when requested

→ **Advantages**

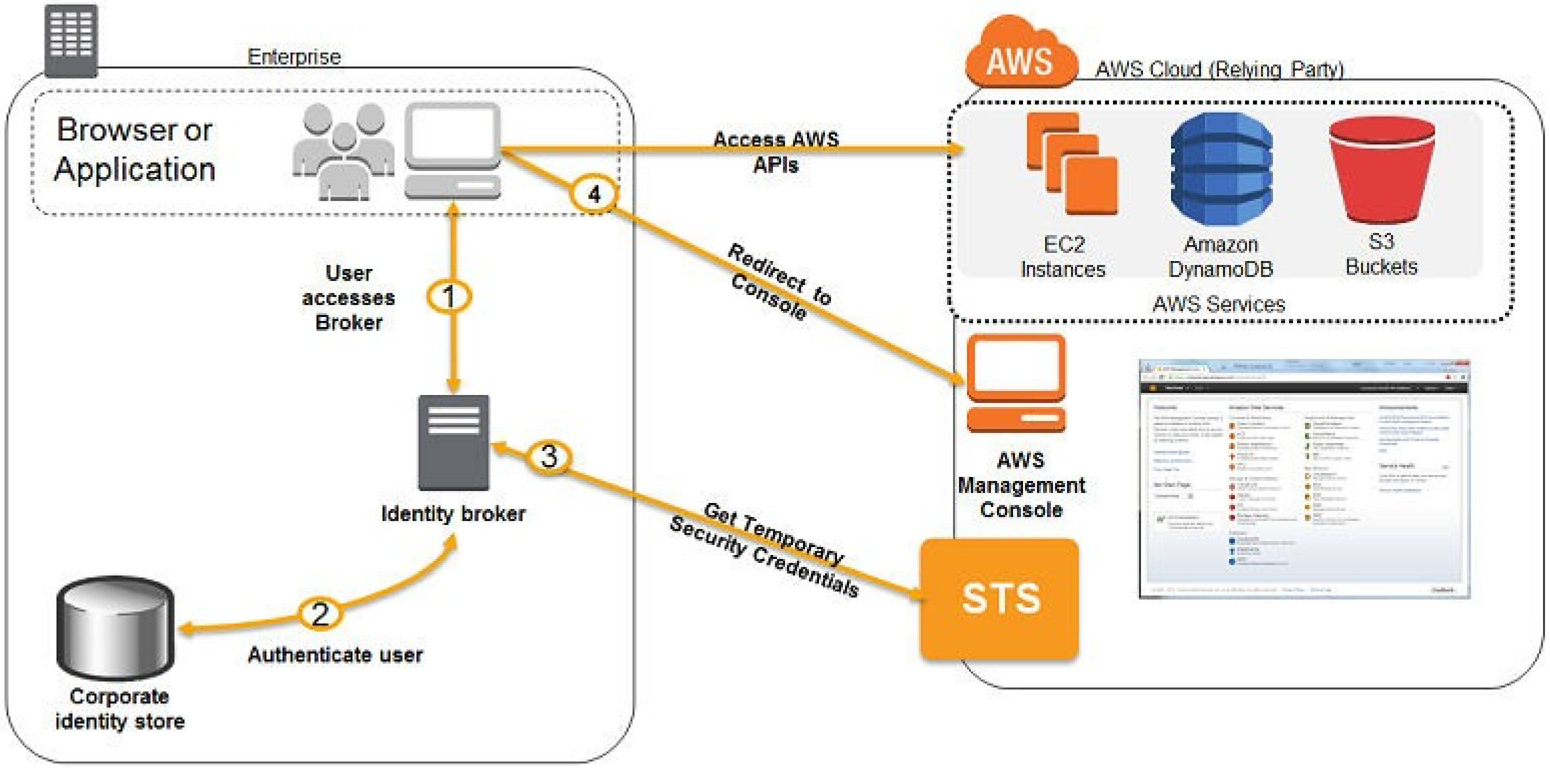
» No need to distribute or embed long-term AWS security credentials with an application.
» Provide access to AWS resource without creating an AWS identity for user. Temporary credentials plays the basis for roles and identity federation.
» Limited lifetime, so you do not have to rotate them or explicitly revoke them

→ **Use Cases**

» Roles for instances
» Identity Federation to AWS APIs
» Mobile and browser-based applications
» Consumer applications with unlimited users
» MFA-based API protection policies

http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html

Identity Federation (Contd.)



Identity Federation (Contd.)

1. A user logs in to the ADFS (Active Directory Federation Service) inside his organization domain.
2. The SSO/authentication mechanism, authenticates user against AD. User's browser receives a SAML assertion in the form of an authentication response from ADFS.
3. User' browser posts the SAML assertion to the AWS sign-in endpoint for SAML (STS) (https://signin.aws.amazon.com/saml).
4. Behind the scenes, sign-in uses the AssumeRoleWithSAML API to request temporary security credentials and then constructs a sign-in URL for the AWS Management Console.
5. User's browser receives the sign-in URL and is redirected to the console.

IAM – Challenges

→ Very powerful tool but if not used properly, it may create problems.

→ Easy to give up permission than controlling so don’t fall in to loop.

→ Writing the actual policies for enterprise, users, roles.

Demo

In the next video we will do hands on with AWS IAM

Thank You

Email us – support@intellipaat.com

Visit us - https://intellipaat.com