Amazon Web Services (AWS)

Analytics Tensor
Mahesh KC
mahesh.kc@analyticstensor.com
https://analyticstensor.com

AWS

Amazon Web Services (AWS) is a cloud computing services platform that provides computing, storage, content delivery, and over 160 cloud services based on a pay-as-you-go approach. According to Gartner-Magic-Quadrant, AWS is leader in cloud IaaS.

https://aws.amazon.com/what-is-aws/?nc1=f_cc

Cloud Computing

According to NIST(National Institute of Standards and Technology, U.S. Department of Commerce, Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Let's check it out.

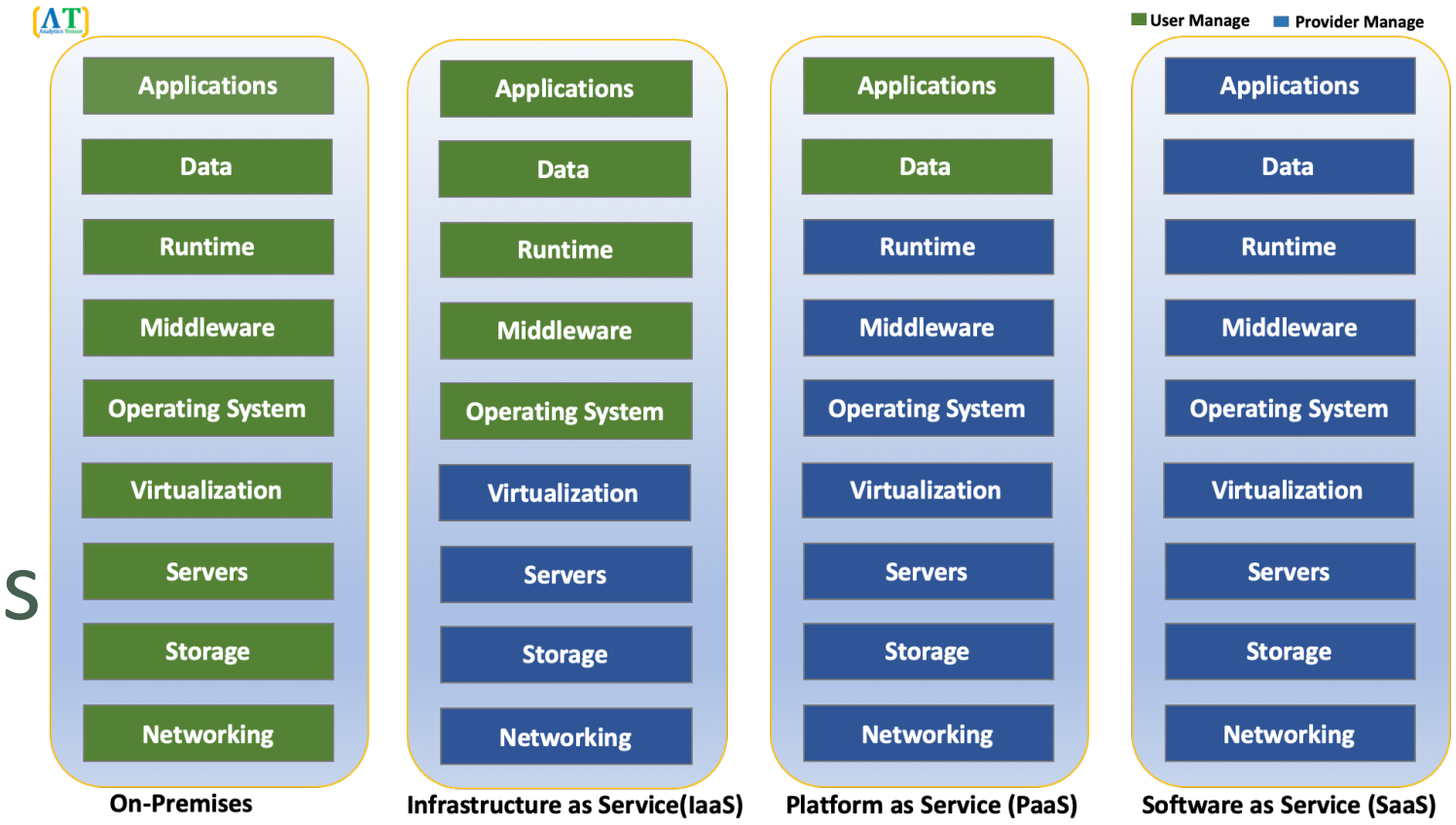
Characteristics:

- On-Demand: Get resource when needed. Don't have to call someone.
- Broad Network Access: Access to private and public networking from a wide range of devices such as PC, Macs, tables, phones etc.
- Resource Pooling: Able to serve multiple clients, customer or tenants with provisional and scalable services according to their demands.
- Rapid Elasticity: Capable to scale in any quantity at any time.
- Measured Service: Usage of resource such as storage, processing, bandwidth etc. should be measurable, controlled and reported by providing transparency to customer. Pricing as Pay as you go model. Not mandatory long-term contract.

Service Models

There various type of service models:

- Infrastructure as a Service (IaaS)
- Software as a Service (SaaS)
- Platform as a Service (PaaS)



Deployment Models

Private Cloud

Community Cloud

Public Cloud

Hybrid Cloud

Global Infrastructure

Regions: Region is a separate geographic area. It is a physical location around the world which clusters the data centers. As of today, April 02 2020, there are 22 Regions. https://aws.amazon.com/about-aws/global-infrastructure/?p=ngi&loc=1 and https://aws.amazon.com/about-aws/global-infrastructure/regions_az/?p=ngi&loc=2

Availability Zones (AZ's): Availability Zones are the collection of data centers in an AWS Region interconnected with high-bandwidth, low-latency networking, over fully redundant, and dedicated metro fiber. Every region has multiple, isolated, and physically separated AZ's. Each AZ has it own independent power, cooling, and physical security connected via redundant, ultra-low- latency networks to provide high-throughput, low-latency networking among them. AZ's provides high availability, fault tolerant and scalability for any application and databases. AZ's are separated within range of 60 miles (100 km) distance. As of today, April 02 2020, there are 70 Availability Zones.

Edge Locations: A site that CloudFront uses to cache copies of the content for faster delivery to users at any location. Edge Locations is used to serve the contents based on availability zone. We have some control over edge location. But we don't have direct access. It performs Amazon CloudFront to deliver the contents. Basically, it is delivering the content that is close to the server with lowest latency.

For e.g. If there is US business serving in Europe then customer don't need to access the site across whole Atlantic continent, It better have content in Europe. Similarly, If we customer in Asia, then they should download the content in Asia. So, Edge location cache the content and deliver quickly when access next time. Other server is Amazon Route 53, which Amazon Web Service global DNS service. When user does DNS query, they can get ip address or answer back as quickly possible. Edge Locations are not necessarily with in regions. They stand on their own. We have some control over edge location. But we don't have any direct access in Edge Location. So, Edge Locations serves servers AWS contents delivery network known, as AWS CloudFront and also server DNS services known as Amazon Route 53.

Scope of Services

Global

AWS IAM

Amazon CloudFront

Amazon Route53

Regional

Amazon DynamoDB

Amazon Simple Storage Service

Elastic Load Balancing

Amazon Virtual Private Cloud

Availability Zone

Amazon Elastic Block Store

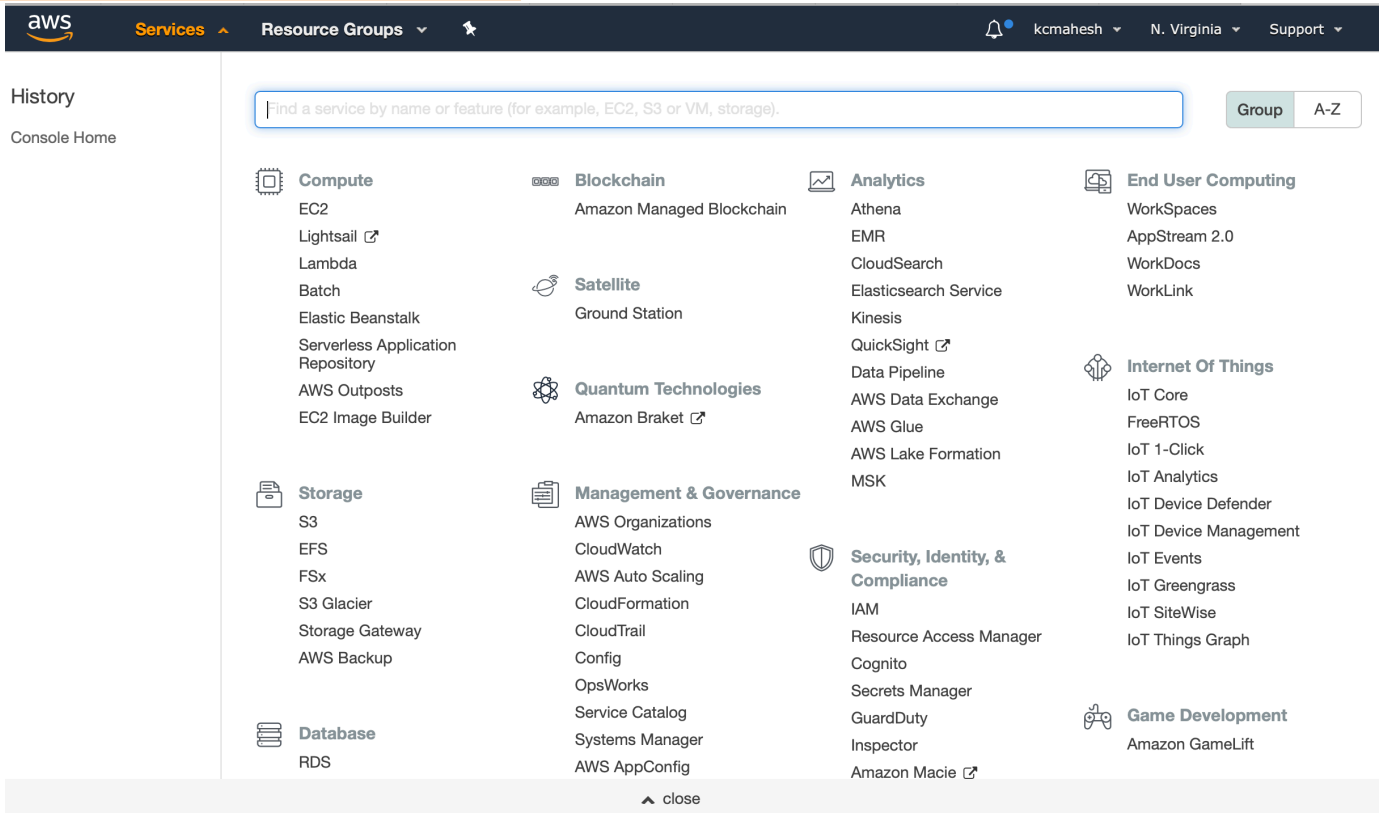
Amazon Elastic Compute Cloud

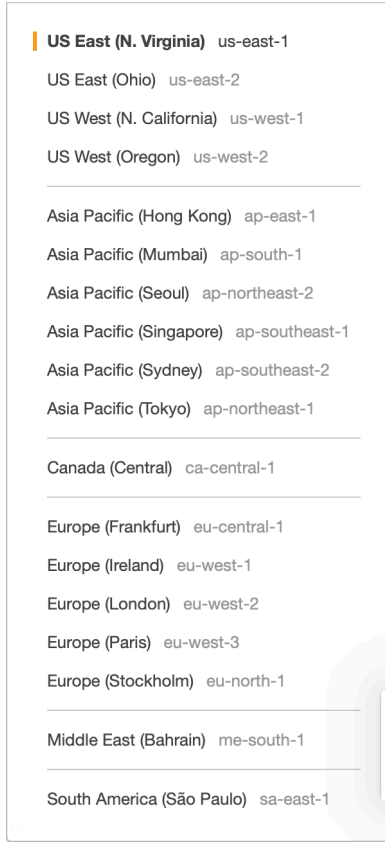
Subnets

When we start to use Amazon Web Services i.e. start VM, database. Certain services has particular scope, some services are global such as shown above. Those are Global because we don't have to choose the Region or Availability Zone. Let see for IAM, if we create the user then is global for all the region. Regional services as DynamoDB table or S3, if we choose the region then it uses all the availability zone with in the region, we just have to specify the Region. For EC2, if we want to create a volume in VM, then we have to choose the availability zone because those services are specific to Availability zone. So, services can be Global, Regional or Availability Zone.

Service Overview

http://console.aws.amazon.com Services: Services are divided into several categories. Region





Note: We'll not cover everything in this course.

Analytics Tensor

Identity and Access Management (IAM)

Authentication: Who the person is.

Authorization: What they are allowed to see and do.

Users: Allow to create users.

Groups: Allow to create groups.

Password Policy: Control password policy for setting strong password.

Multifactor Authentication: Enable multifactor authentication.

AWS IAM services is specifically for authenticating for AWS API, not for OS level, application or database level (except Dynamo DB) authentication. So, IAM services is used for authenticating, authorizing users or groups of user against Amazon Web Services API.

Demo

- Create User
- Create Group
- Provide services access to the group

Services-> IAM-> Users-> Add user
User name: analytics.tensor
Create Group: developer
Choose IAmazonEC2FullAccess. i.e provide full access to Amazon EC2.
Copy the password and login with newly generated user.
The user has only access to create EC2.
Let's check it out!

Access keys: The user should be able to login to console. When the user want to programmatically, access using command line tool or use SDK to develop locally, then the user need access key to authenticate against AWS API.

Let's create access key for the user. Go to IAM, choose user, choose security credentials, and click on create access key. **Note:** This is the only time we see the access key. We won't see it again so copy and paste. Other wise we have to recreate the user again. Copy and paste the access key. ***The best practice is to enable user to create their own secret key instead of admin creating the key.*** With access key, user should be able to do their local development using SDK or interact using Command Line.

For more info: https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

Permissions and Policies

- Permissions(authorization) granted via policies.
- Policies are written in JSON.
- Type of policy:
  - Managed Policy:
    - AWS Managed: Top level first class object apart from other resources. Managed and created by AWS Web Services.
    - Customer Managed: Customer creates and manages policy by them self.
  - Inline Policy: Put policy directly to user account than referencing to manage policy.
- Method of creating policy:
  - Generator: UI tools provided by AWS to choose services and action that create policy in JSON files.
  - Hand written policies: Create policy in JSON file ourselves.
- Evaluation logic:
  - Implicit deny: By default, all the requests are denied. Policy starts with implicit deny.
  - Explicit deny: Explicitly deny the actions. To create safely around the things.
  - Explicit allow: If it doesn't find explicit deny it will check for explicit allow.

  It will check for implicit deny, explicit deny and explicit allow in hierarchal order.

  Let's check out the flow chart of decision.

Amazon Resource Name (ARN)

ARN is a way of uniquely identifying a particular resources across Amazon resources. There are lot of customer and resources in Amazon. ARN helps to uniquely identify the resources. ARN has its own format shown below which is followed by colon(:).

Format Pattern
arn: partition:service:region:account-id:resourceType/resource
arn: partition:service:region:account-id:resourceType:resource

Partition: Partition is a group of AWS regions. It is high level organization of AWS. It is place where the resource is located. Supported partition are:

- aws: Standard Regions
- aws-cn: China Regions
- aws-us-gov: AWS GovCloud (US) Regions

The widely used partition is aws.
Services: The namespace that identifies the AWS product. This is the service of AWS product For e.g. S3 for Amazon S3 resources.
Region: Region, the service is operated in i.e. us-east-1, us-west-2.
Accound-id: AWS account id that owns the resource, without the hyphens.
Resource Type: Specify the resource such as s3 bucket, databases, EC2
Resource: Actual name of the resources. Some resource are separate by colon or slash. Always review AWS documentation to find right type.

e.g.
arn:aws:iam::123456789123:user/analyticstensor. # no region necessary for global services so it is blank
arn:aws:s3:::bucket1/* # no account id necessary for s3 buckets
arn:aws:rds:us-east-2:1234456789123:db:mysql-db

For more info: https://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html

Policy Example

```
{
   "Version": "2020-04-03",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:GetObject",
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::bucket1/*"
         ]
      }
   ]
}
```

List

Allow or Deny

List of Actions

List of resources

ARNS

Version mean when the policy was created.
Statement is a list which specifies effect and action.
Above example indicates allow user to get and put object in s3.
Resource is a list which specifies the arn. Above example, specifies arn for all object inside bucket name bucket1.

Permission and Policies

- Policies are just a collections of statements. There can be multiple policies for user or group of users.
- Statement specifies
  - Principal(resource-based policies)
  - Actions
    - EC2:RunInstances
    - S3:ListBucket
  - Resources
    - EC2 Instances
    - S3 Buckets (or objects)
  - Condition:
    - Time of day: Access on certain time of day
    - From Specific IP address: Specify access from given IP address
    - Resource contains particular tag
    - And more.

For more info: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html#policy-vars-intro
Policy Grammar: https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_grammar.html

Creating and Attaching Policies

We'll see how to create and attach policies.

- Go to AWS Console.
- Choose Policies.
- We can see the existing policies by filtering policies either aws managed or customer managed policies.
- Choose Create Policy.
- Choose S3 services.
- Choose Action: GetBucketLocation, Getojbect
- Choose All resources on Resources.
- Click Review Policy.
- Type tester_s3_policy as name of policy.
- Click Create Policy.
- Choose the policy we just created, Under Policy actions choose Attach.
- Select the user to attach the policy.
- Click Attach Policy.
- Check the policy, By choosing Policies under IAM, Filter the policy and choose Policy usage.
- Detach the policy. (Self Assignment)

Roles

- Roles is an alternate form of authentication. They provides temporary credentials.
- Way to authenticate and authorize users but without having to embed some long term credential such username, password etc. It will eliminate to downstream effect when some one changes the password that relies on multiple system.
- We can attach a role to services such as EC2 instance. The machine will now authenticate via that role without any credentials.
- Role is also used to authenticate third party account.

Demo on creating Roles with existing policy.

Do not:

- Embed access keys, user name and password in code.
- Embed in environment variables.
- Share with Third parties or other users.

EC2 Role Example

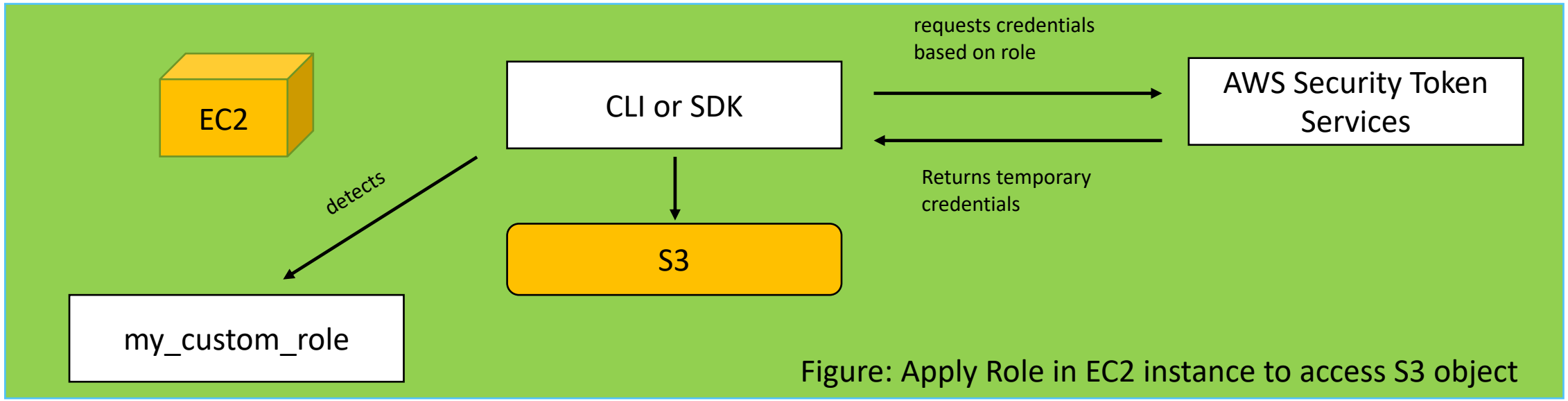


Figure: Apply Role in EC2 instance to access S3 object

Let's start EC2 Virtual Machine, we'll attach role to this machine. For e.g. Upload and download file from S3 to this machine. Using either CLI or SDK tools, it will automatically request temporary credential based on the role. AWS Security Token will those deliver those temporary credential. The CLI or SDK will apply those credential and use them to authenticate what ever services it is trying to reach. In this example, it will be S3. The CLI or SDK will also automatically renew those Credential when it expires. The temporary credential will default remain for 15 mins but we can control. Let's watch this video.

Federated Users

In organization, where there are less than 20 users we can manage users and resources easily. But in large organization having 100 or 1000's of users, it is very difficult to manage all the users. Large organization, they might have already managed users using LDAP or Active Directory. Federated users means taking existing user who we already know and giving them temporary access in AWS based on temporary credentials. So, using this will helps to create SSO(Single Sign-On) by single on corporate network and they have access based upon those credentials. We also can federate mobile application users, where user can directly access AWS services by-passing backend APIs/Proxies.
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml.html

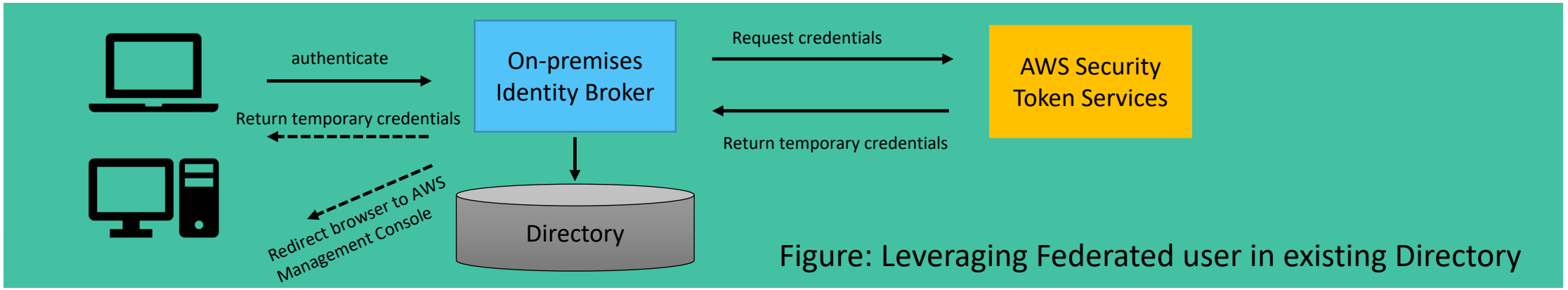


Figure: Leveraging Federated user in existing Directory

Resource Policies

The policies that we talked earlier is about granting user or group of users to access services within AWS based on the permission.

- Resource policies is a ways of applying permission directly to the AWS resources. For e.g. applying policies to:
  - S3 Bucket: Applying permission to S3 bucket for external users or anonymous users.
  - DynamoDB table: Apply permission to Dynamo DB to be publicly readable.
  - Simple Queue Services
  - And more
- Written in JSON
- Can provide cross-account resource sharing.
- Can allow anonymous use.
- It is similar to what we talked about policies earlier but it in not applied to users/group of user, it is applied to the actual resources itself.

Demo create bucket policy in S3.

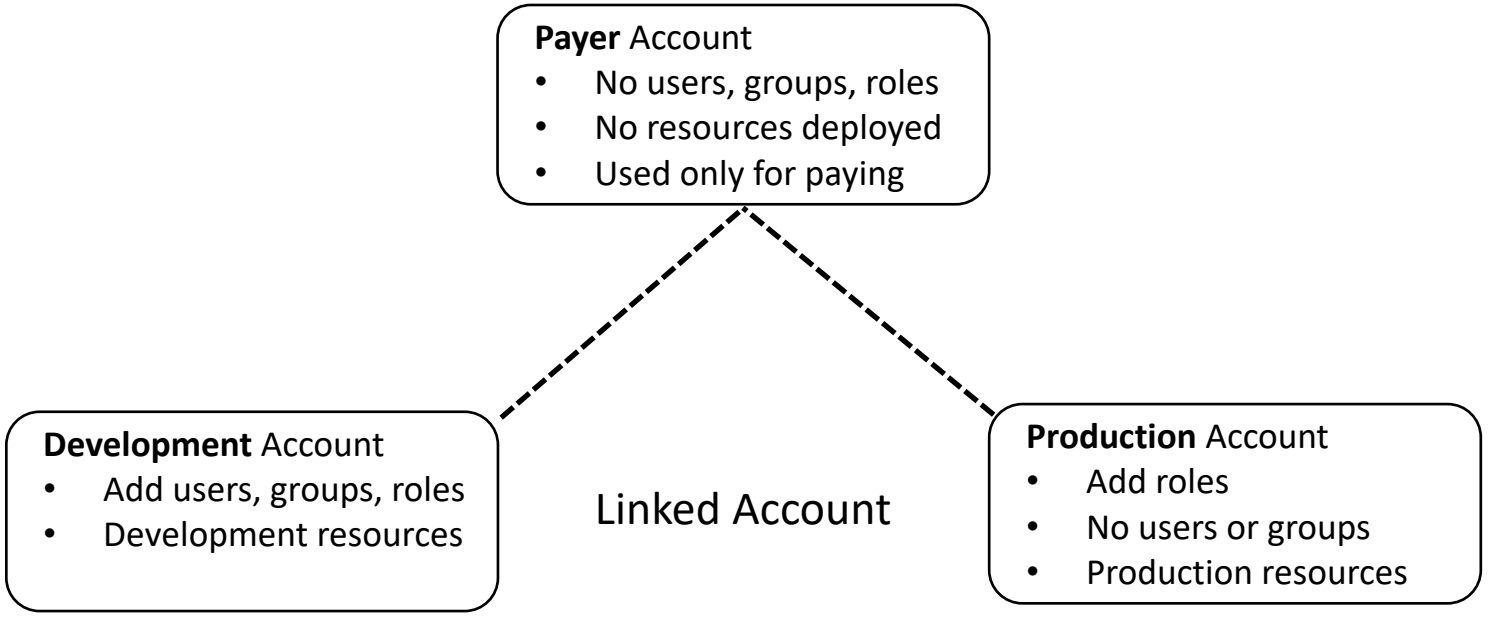
Create S3 bucket, select bucket, click permissions, click bucket policy, click policy generator, choose s3 bucket policy, effect as allow, principal as *, Actions: Getobject, arn as arn:aws:s3:::analyticstensor/* Click generate policy. Copy and paste the policy. Click Save.

Resource Policies (cont.)

This policy will let everyone access the object inside analyticstensor bucket.

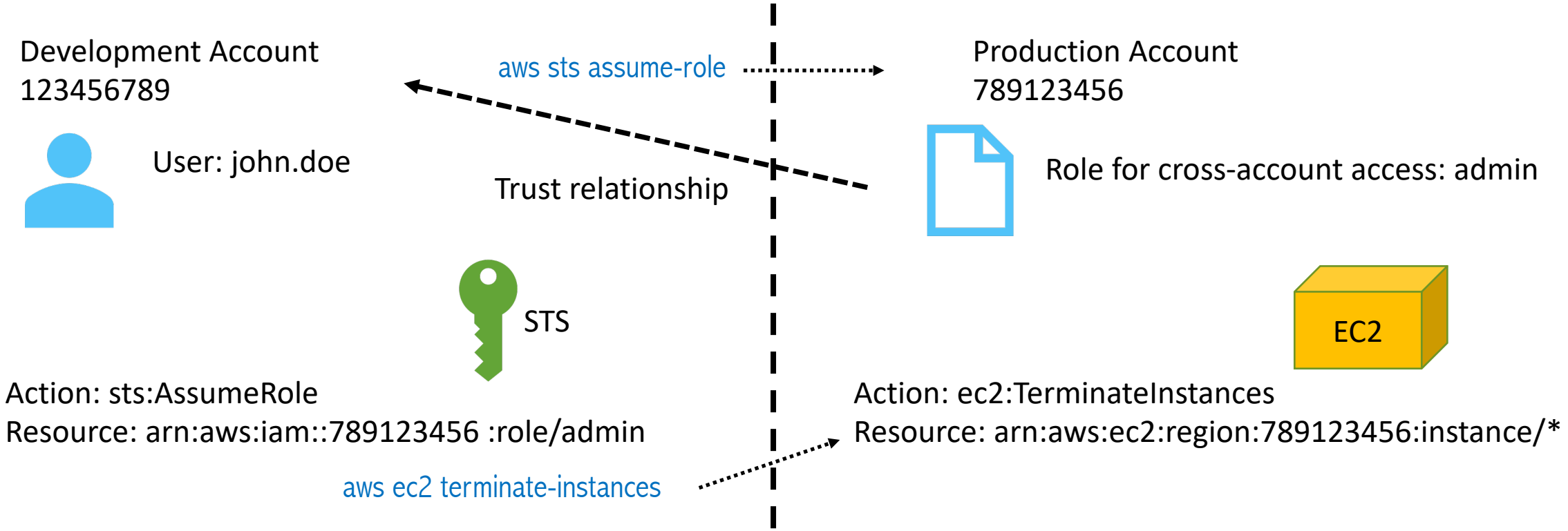
```
{
 "Id": "Policy1586199076464",
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Stmt1586199039804",
   "Action": [
    "s3:GetObject"
   ],
   "Effect": "Allow",
   "Resource": "arn:aws:s3:::analyticstensor/*",
   "Principal": "*"
  }
 ]
}
```

Using Roles for Cross Account Access



This use case shows leveraging multiple AWS account into one organization. This is an example but it can done multiple ways. We have payer, development and production account. The payer account is only for paying the bills. The other two account is linked account. We create user, groups, roles and manage resources. In development account, we have users, groups, roles. In production, account it has only roles but users and groups. Linked account can be any thing like: QA, UAT, Test etc. It can be split by department or business unit. For e.g. engineering, research, business etc.

Using Roles for Cross Account Access (cont.)



User (john.doe) in development account has assumerole in production account as admin to terminate instances. We are trusting development account. User in development account is trusted by production account.

Best Practices

- Leverage groups
- Grant least privilege
- Implement strong password policy
- Leverages roles for cross-account access
- Use deny statements for added security
- Don’t share credentials
- Don’t email, print, or publish to repos
- Leverage multiple accounts for isolations
- Always protect master account(root). Your email.
- Delete default access keys of master account.
- Read security white paper: https://aws.amazon.com/whitepapers/