IAM

======

* IAM allows you to manage users and their level of access to aws console.
* IAM is universal. It does not apply to regions at this time i.e not region based.
* The "root account" is simply the account created when first setup your AWS account. It has complete admin access by default.

IAM allows you to setup users, groups permissions.

=================================================================

Features of IAM:

-----------------

* centralised control of our aws account
* shared access to our aws account
* Granular permissions i.e i want people to access this service and i want people to access that service .
* Identity federation(including AD, facebook, linkedin etc)i.e you can use AD,facebook,linkedin with idetity and access management., your users can use the credentilas of AD to login into aws console.
  + Multifactor Authentication
* it also provides temporary access for users/devices and services when necessary.
* Allow you to set up your own password rotation policy.

===============================================================

IAM users sign-in link:

-----------------------

* It is the link that you can send to your user to login into you aws console.

============================================================

types of MFA device to assign:

--------------------------------

* Virtual MfA device: Authenticator app installed on your mobile device or computer.
* U2F security key: YubiKey or any other compliant U2f device
* other hardware MFA device: Gemalto token

============================================================

steps set up virtual MFA device:

-----------------------------------

1)Install a compatible app an your mobile device or computer

2)use your virtual MFA app and your device's camera to scan the QR code.

3)Type two consecutive MFA codes to configure your MFA.

==============================================================

AWS user access type:

----------------------

* Programmatic access
* AWS management console access

================================================================

Policy Documents:

-----------------

* policy is a document that you attach to users or groups or roles individually.
* policy script are made of JSON(java script object notation)
* policy is a key value pair
* we can add/remove policy.

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow"

"Action": [

"acm:Get"

]

"Resource":

}

]

}

======================================================================

Access key ID & secret Key

-------------------------

* Acess key ID & secret Key are not used to login into aws console, rather you can use this to access aws via the API and CLI.
* You only get to view the credentials once
* If a user loss his Secret key, we can regenerate the key in users->security credentials->create access key

=====================================================================

Role: allow one part of aws to do something with another part.

So IAM Role are a secure way to grant permissions to entities that you trust.

=========================================================================

Security Token Services(STS)

----------------------------

STS Grants users limited and temporary access to AWS resources.

Users can come from three sources:

----------------------------------

🡪Federation(typically Active Directory):

* Users security assertion markup language (SAML)
* Grants temporary access based off the users Active Directory credentials. Does not need to be a user in IAM.
* Single sign on allows users to log in to AWS console without assigning IAM credentials.

🡪Federation with mobile apps:

Use Facebook/Amazon/Google or other openID providers to log in.

🡪Cross Account Access:

Let’s users from one aws account access resources in another.

Key Terms of STS:

-----------------

Federation:

Combining or joining a list of users in one domain (such as IAM) with list of users in another domain (such as Active Directory, Facebook etc)

Identity Broker:

A service that allows you to take an identity from point A and join it to point B.

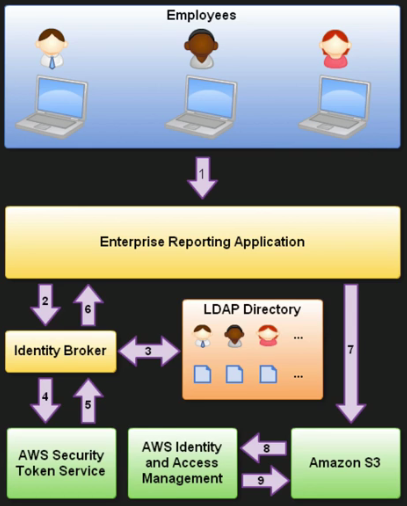
Identity store:

Services like Active Directory, facebook, google etc.

Identities:

A user of a service like facebook etc.

Employees login by using username and password, this is given to the identity broker. The identity broker evaluates that against the Active directory first, once it says that yes it the correct username and password, it then contacts Security token service. The security token service will give you 4 things: access key, secret access key, a token and duration that is then passed back to the actual application through identity broker. Then the actual application call the s3 to get objects from s3 and then request goes from s3 to IAM, IAM verifies if the token is correct and then the user is able to use the service.

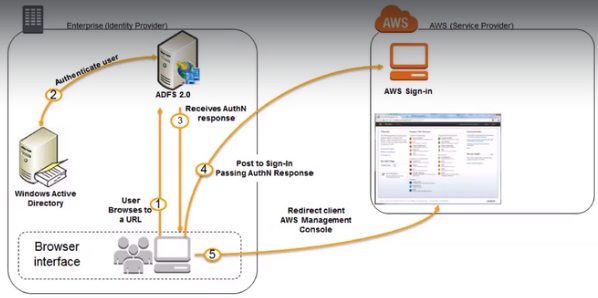


Employee enters their username and password.

The application calls an Identity broker. The Identity Broker captures the username and password.

Identity Broker uses the organization’s LDAP directory to validate the employee’s identity.

Then Identity broker calls the new GetFederationToken function using IAM credentials.



Can you authenticate without active directory?

Yes and it’s using SAML(Security Assertion Markup Language)

We would against active directory first, then assign the temporary security credentials.

Security Assertion Markup Language (SAML) is a standard for logging users into applications based on their sessions in another context. This single sign-on (SSO) login standard has significant advantages over logging in using a username/password:

* No need to type in credentials
* No need to remember and renew passwords
* No weak passwords

VPC

====

* Think of a VPC as logical datacenter in AWS.
* NACL are applicable at subnet level, security groups has to be assigned explicitly to the instance.
* By default our default vpc, will have default NACL which would allow all traffic, both inbound and outbound. When we create a private NACL everything is denied in both Inbound and outbound. In security groups all outbound traffic are allowed by default.
* **We can monitor Network traffic in our VPC by using VPC flow-logs**

Nat instance:

* When creating a NAT instance, disable source/destination check on the instance.
* Nat instance must be in public subnet.
* There must be a route out of the private subnet to the NAT instance, in order for this to work.
* The amount of traffic that Nat instances can support depends on the instance size. If you are bottlenecking, increase the instance size.
* You can create high availability using Autoscaling Group, multiple subnets in different AZs, and a script to automate failover.
* Nat instance is behind a security group.

Nat Gateway:

* Scale automatically up to 10gbps
* No need to patch
* Not associated with security groups
* Automatically assigned a public ip address.
* No need to disable source/destination checks
* More secure than a NAT instance.

===================================================================

* Launch instances into a subnet of your choosing.
* Assign custom IP address ranges in each subnet
* Configure route table between subnets
* Create internet gateway and attach it to our VPC
* Much better security control over your AWS resources
* Instance security groups
* Subnet network access control list
* All subnets in default VPC have a route out to the internet.
* Each Ec2 instance has both a public and private IP address in default VPC

VPC peering:

* VPC peering is simply a connection between two VPCs that enables you to route traffic between them using private IP addresses.
* You can create VPC peering connection between your own VPCs, or with a VPC in another AWS account within a single region.
* AWS uses the existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection, and does not rely on a separate piece of physical hardware. There is no single point of failure for communication or a bandwidth bottleneck when we peer VPC.
* VPC peering allow you to connect one VPC with another via a direct network route using private IP addresses i.e I can have instance on one subnet inside a different VPC that could communicate to a another instance in an another subnet inside another VPC and would do that over private ip address.
* Instances behave as if they were on the same private network.
* You can peer VPC’s with other aws accounts as well as with other VPCs in the same account.
* There is no transitive peering.
* We can’t create a VPC peering connection between VPCS that have matching or overlapping CIDR Blocks.

Direct connect:

---------------

AWS direct connect makes it easy to establish a dedicated network connection from our premises to aws.

Using aws direct connect, you can establish private connectivity between aws and our datacenter, office or colocation environment, which in many case can reduce our network costs, increase bandwidth throughput, and provide a more consistent network experience than internet-based connections.

Advantage: Reduce costs when using large volumes of traffic, Increase reliability and increase bandwidth.

you can very well connect company network with aws by establishing a VPN connection between your company’s network and Amazon VPC.