IAM (Identity and access management)

First, IAM is not an identity store/authorization system for your applications. The permissions that you assign are permissions to manipulate AWS infrastructure, not permissions within your application.

AWS Directory Service

Amazon Cognito - for identity management for mobile applications

IAM service – can be managed using AWS console, CLI, REST API/SDK

Principals

A principal is an IAM entity that is allowed to interact with AWS resources. Below 3 types

Root User

When you first create an AWS account, you begin with only a single sign-in principal that has complete access to all AWS Cloud services and resources in the account. This principal is called the root user

Similar to linux root user

Not recommended to use for administrative purpose

using the root user only to create your first IAM user and then securely locking away the root user credentials

IAM Users

Users are persistent identities set up through the IAM service to represent individual people or applications

IAM users can be created by principals with IAM administrative privileges at any time

no expiration period; they are permanent entities that exist until an IAM administrator takes an action to delete them

Roles/Temporary Security Tokens

Roles are used to grant specific privileges to specific actors for a set duration of time. These actors can be authenticated by AWS or some trusted external system.

When one of these actors assumes a role, AWS provides the actor with a temporary security token from the *AWS Security Token Service (STS)* that the actor can use to access AWS Cloud services.

Token has a lifetime – 15 minutes to 35 hours

Roles and temporary security tokens enable a number of use cases:

**Amazon EC2 Roles**—Granting permissions to applications running on an Amazon EC2 instance.

Using IAM roles for Amazon EC2 removes the need to store AWS credentials  in a configuration file.

             When the application running on the instance uses the Application Programming Interface (API) to access the Amazon S3 bucket, it assumes the role assigned to the instance and obtains a temporary token that it sends to the API.

**Cross-Account Access**—Granting permissions to users from other AWS accounts, whether you control those accounts or not.

**Federation—**Granting permissions to users authenticated by a trusted external system.

*IAM Identity Providers* provide the ability to federate these outside identities with IAM and assign privileges to those users authenticated outside of IAM.

2 different types of identity providers

For federating internal identities, such as Active Directory or LDAP,

IAM supports integration via Security Assertion Markup Language 2.0 (SAML).

web-based identities such as Facebook, Google, or Login with Amazon

IAM supports integration via OpenID connect (OIDC). allows IAM to grant privileges to users authenticated with some of the major web-based IdP

In each case, federation works by returning a temporary token associated with a role to the IdP for the authenticated identity to use for calls to the AWS API. The actual role returned is determined via information received from the IdP, either attributes of the user in the on-premises identity store or the user name and authenticating service of the web identity store.

**Authentication**

There are three ways that IAM authenticates a principal

User Name/Password—When a principal represents a human interacting with the console, the human will provide a user name/password pair to verify their identity. IAM allows you to create a password policy enforcing password complexity and expiration.

Access Key—An access key is a combination of an access key ID (20 characters) and an access secret key (40 characters). When a program is manipulating the AWS infrastructure via the API, it will use these values to sign the underlying REST calls to the services. The AWS SDKs and tools handle all the intricacies of signing the REST calls, so using an access key will almost always be a matter of providing the values to the SDK or tool.

Access Key/Session Token—When a process operates under an assumed role, the temporary security token provides an access key for authentication. In addition to the access key (remember that it consists of two parts), the token also includes a session token. Calls to AWS must include both the two-part access key and the session token to authenticate

**Authorization**

The process of specifying exactly what actions a principal can and cannot perform is called authorization

How authorization is handled ? –

* Create a policy
* Define privileges
* Associate policy with principal

**Policies**

JSON document that fully defines a set of permissions to access and manipulate AWS resources.

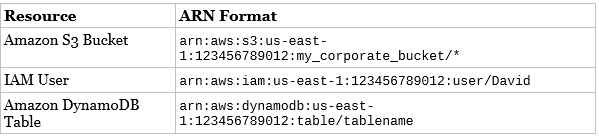
Each permission will have:

Effect—A single word: Allow or Deny.

Service – example S3

Resource - specifies the specific AWS infrastructure. Wild cards are allowed

"arn:aws:service:region:account-id:[resourcetype:]resource"



Action – Read, write , onlyGET actions , etc

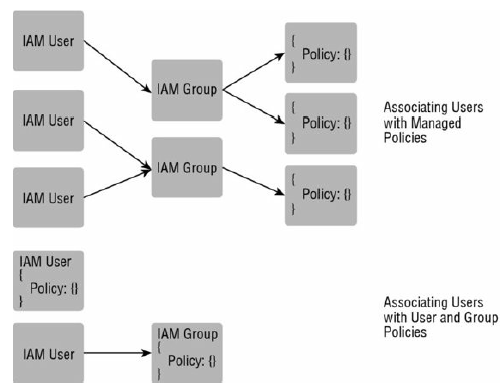
Condition – allow only from a specific IP only, during specific time period

**Associating policies with Principal**

***User Policy***— These policies exist only in the context of the user to which they are attached

***Group Policy -*** These policies exist only in the context of the group to which they are attached.

***Managed Policies***— exist independently of any individual user or a group



**Multi-Factor Authentication (MFA)**

User with MFA enabled needs second method of authentication beyond just a password or access key. One-Time Password (OTP)

**Resolving Multiple Permissions**

It is important to know how conflicts between these permissions are resolved:

1. Initially the request is denied by default.

2. All the appropriate policies are evaluated; if there is an explicit “deny” found in any policy, the request is denied and evaluation stops.

3. If no explicit “deny” is found and an explicit “allow” is found in any policy, the request is allowed.

4. If there are no explicit “allow” or “deny” permissions found, then the default “deny” is maintained and the request is denied.

The only exception to this rule is if an AssumeRole call includes a role and a policy, **the policy cannot expand the privileges of the role** (for example, the policy cannot override any permission that is denied by default in the role).

**Manage federated users**

Users that are present in external identity management solution like AD (Should support SAML 2.0)

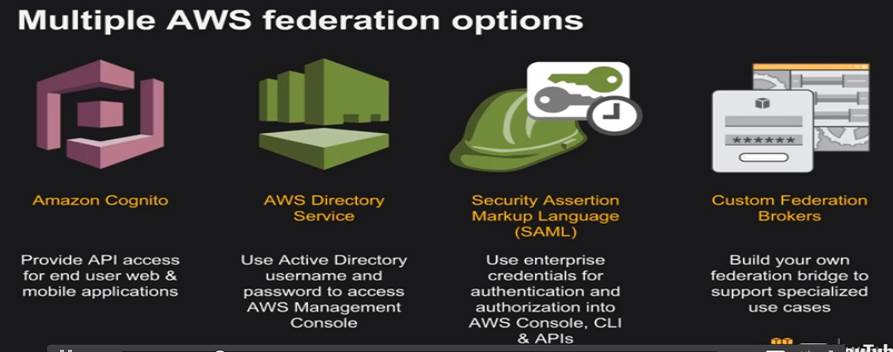
SAML

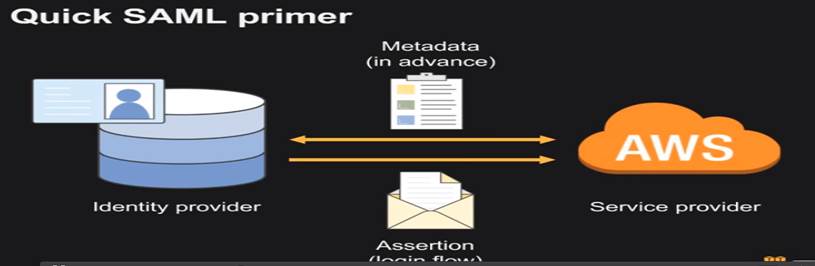
Security Assertion Markup Language 2.0 (SAML) is an open standard for exchanging identity and security information with applications and service providers.

Applications and service providers that support SAML enable you to sign in using your corporate directory credentials, such as your user name and password from Microsoft Active Directory.

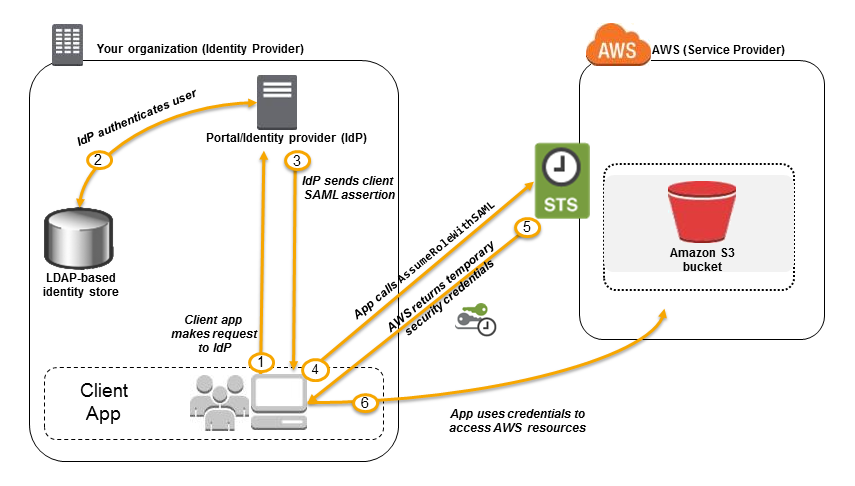
With SAML, you can use single sign-on (SSO) to sign in to all of your SAML-enabled applications by using a single set of credentials.

By enabling SAML authentication, you also can manage access to your applications centrally. SAML-enabled applications delegate authentication requests to your corporate directory. When users are removed from your directory, they are no longer able to sign in.





<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_saml.html>



**OpenID connect**

It is an open standard for authentication that is supported by a number of login providers

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol

**Amazon cognito**

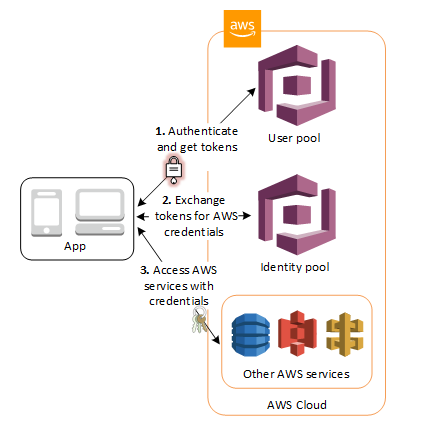
Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, or Google.

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services. You can use identity pools and user pools separately or together.

**An Amazon Cognito user pool and identity pool used together**

See the diagram for a common Amazon Cognito scenario. Here the goal is to authenticate your user, and then grant your user access to another AWS service.

1. In the first step your app user signs in through a user pool and receives user pool tokens after a successful authentication.
2. Next, your app exchanges the user pool tokens for AWS credentials through an identity pool.
3. Finally, your app user can then use those AWS credentials to access other AWS services such as Amazon S3 or DynamoDB.



Thanks & Regards,

Shantaram Vernekar

**From:** Shantaram Vernekar   
**Sent:** Sunday, July 29, 2018 11:16 PM  
**To:** Shantaram Vernekar <shantaram\_vernekar@persistent.com>; 'shantaram.vernekar@citi.com' <shantaram.vernekar@citi.com>  
**Subject:** AWS - IAM

**Reinvent vedio**

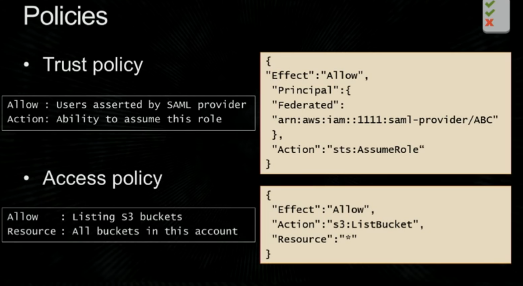
Federation - [**https://www.youtube.com/watch?v=debJ3o5w0MA**](https://www.youtube.com/watch?v=debJ3o5w0MA)

**D**elegation -[**https://www.youtube.com/watch?v=0zJuULHFS6A**](https://www.youtube.com/watch?v=0zJuULHFS6A)

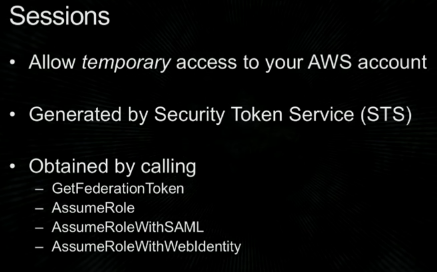
**Blog -** [**http://jayendrapatil.com/tag/iam-role/**](http://jayendrapatil.com/tag/iam-role/)

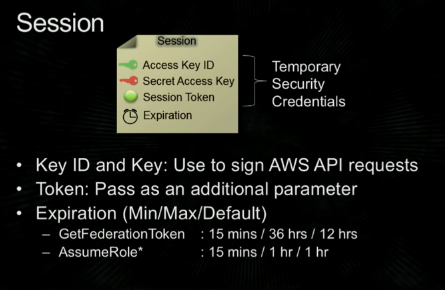
**Roles**

* Trust Policy
* Access Policy
  + What operations can be performed on a given resource

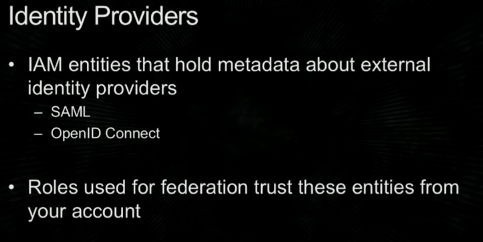


**Sessions**

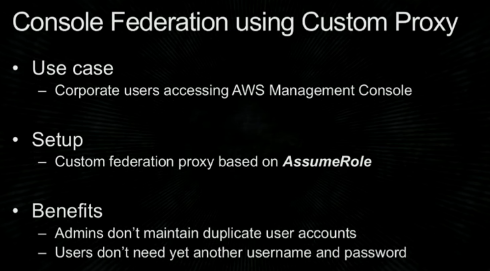




**Identity provider**



**SCENARIO 1: Corporate identity using custom federation – AWS Console access**



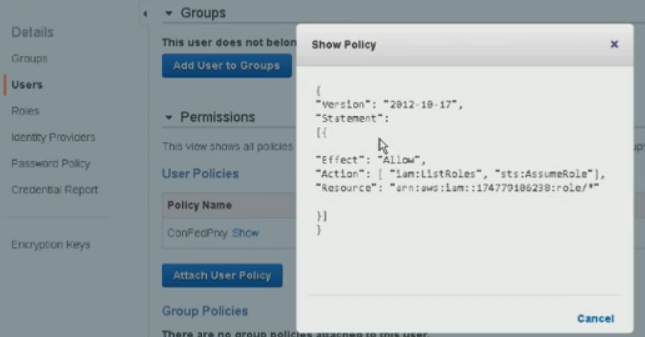
**Pre-requisites**

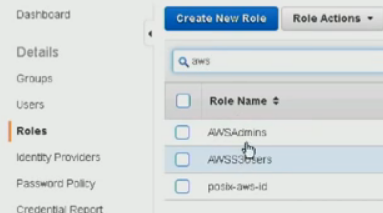
AWS – A user(not matching with AD, kind of functional ID – common for all users..) having permissions to listroles and can make a call to STS (AssumeRole)

* Role (Admins and USer)
  + Admins – trust own account and has all permissions
  + User – trust own account and has only S3 permissions

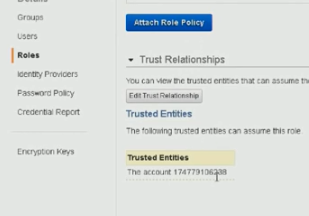
Corporate side

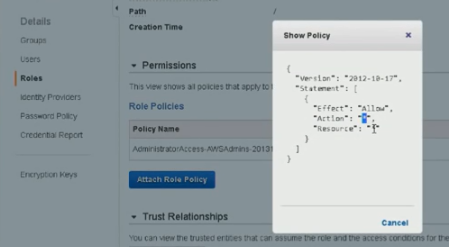
* In AD a user and two groups (Group names matching)

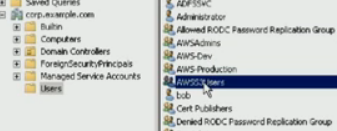


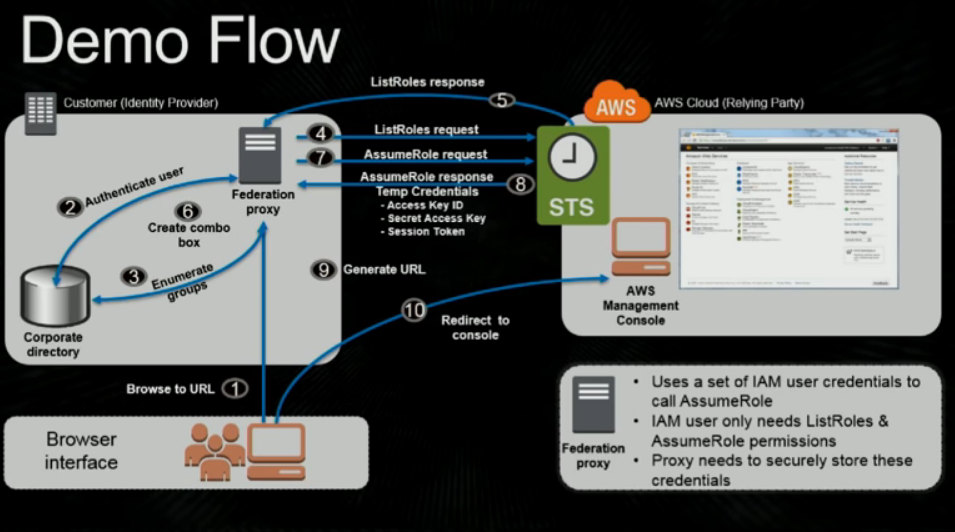


Trust account

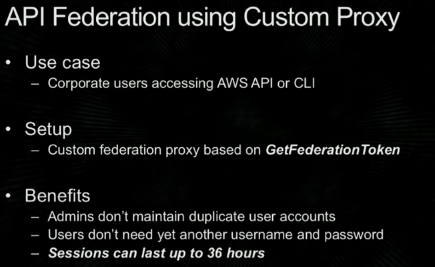






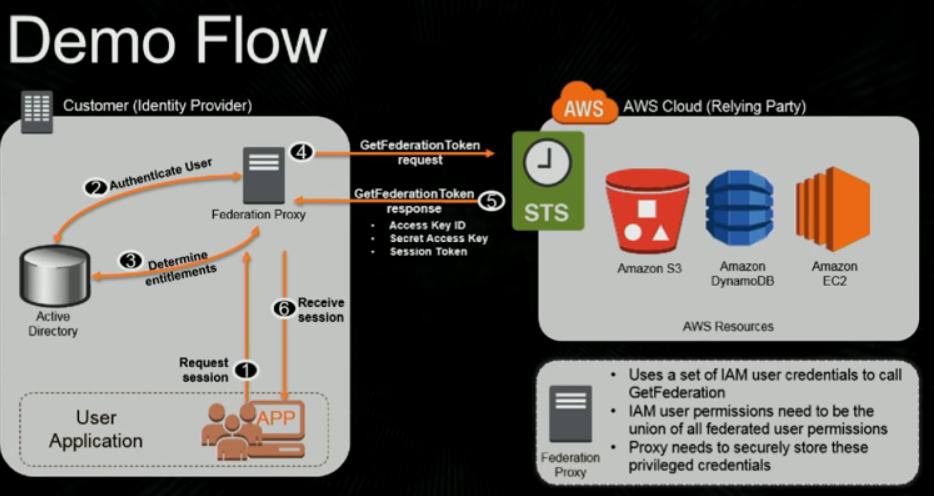


**SCENARIO 1: Corporate identity using custom federation – API access**



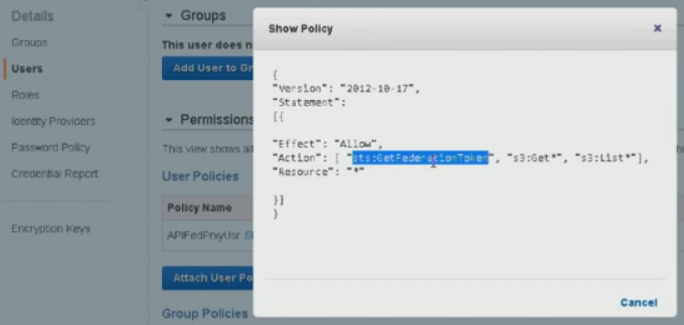
A command line user trying to access files under a S3 bucket under his home directory

* Command line makes a request to proxy
* Authenticates user
* Get entitlements (Access permissions) – What permissions are allowed for this user example : FileAccess read only, no delete..
  + Can be retrieved using Windows group membership or AD memberships..
* Make a request to STS using GetFederatioToken (Include allowed permissions on S3 for this under)
* STS will return session token () – this token allows to perform actions on S3 that were requested (last step)
* Using that token/session, a command line user can make requests.
  + Can only perform allowed actions.. Any other actions are not allowed..



**Pre-requisites**

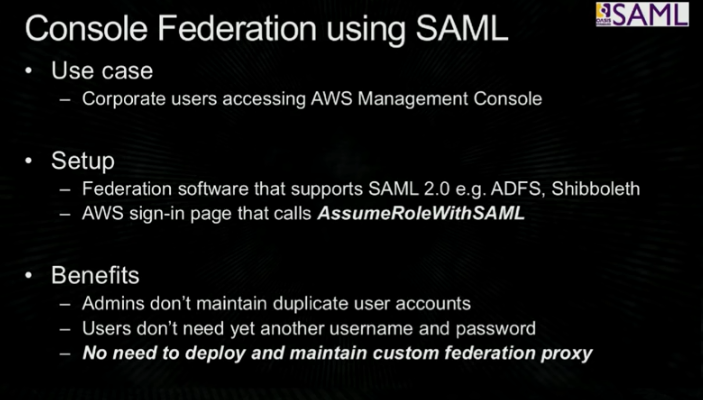
AWS – a user having below permissions (No roles)





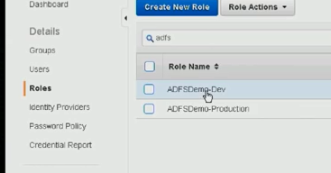
#####

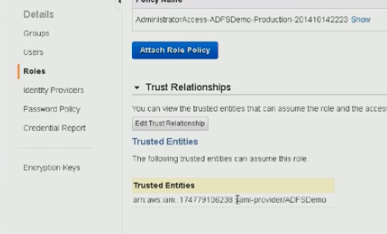
Use case 3

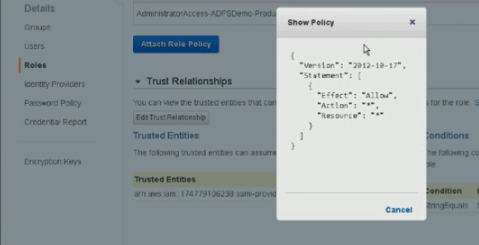


AWS :

* Create identity provider 🡪 provide meta data
* Roles - 2 roles

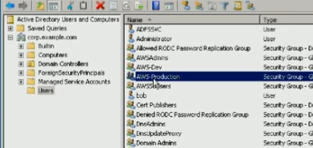




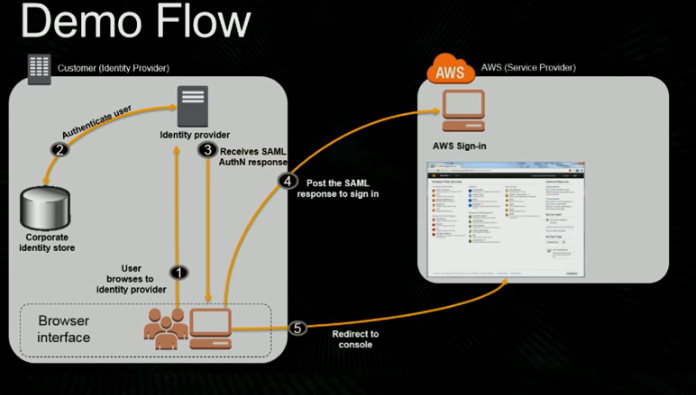


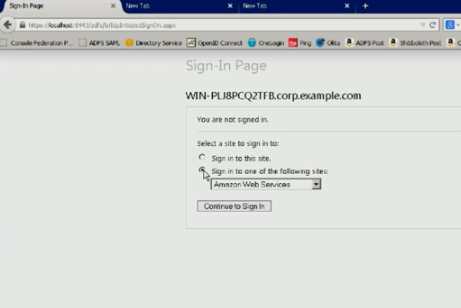
Client side:

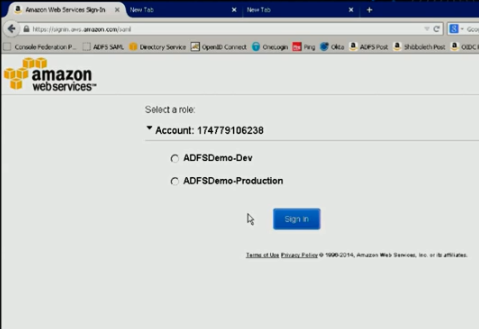
Two Groups

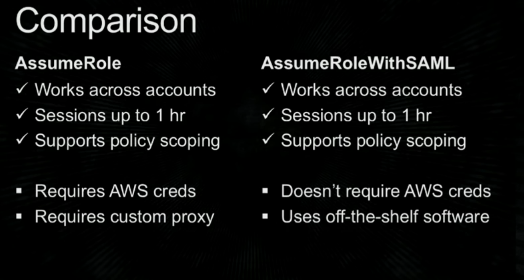


AWS sign on calls STS 🡪 assume role with SAML





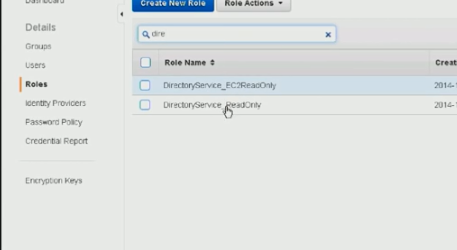




##

Option 4:

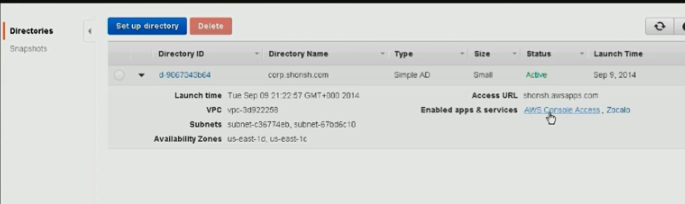




Trust the directory service..

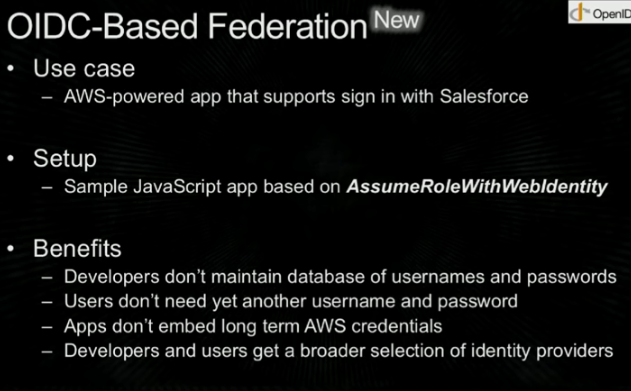


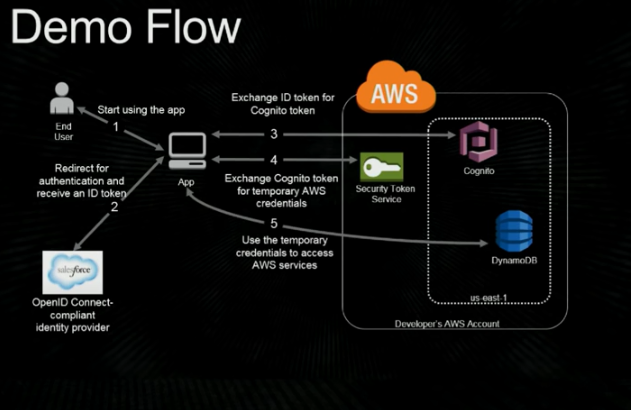
Simple directory service..

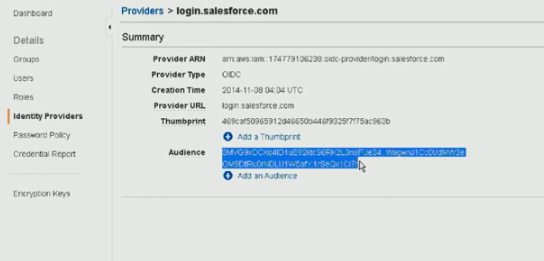


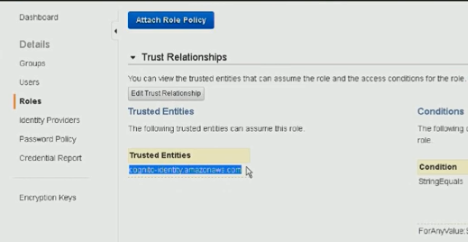
**Web identities..**

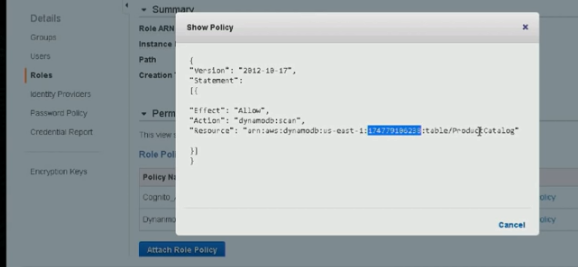


















Identity management service from AWS

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps

Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, or Google.

two main components of Amazon Cognito are

* user pools - provide sign-up and sign-in options for your app users

 user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

* identity pools - enable you to grant your users access to other AWS services

Support following identity providers that you can use to authenticate users for identity pools:

* Amazon Cognito user pools
* Social sign-in with Facebook, Google, and Login with Amazon
* OpenID Connect (OIDC) providers
* SAML identity providers
* Developer authenticated identities

See the diagram for a common Amazon Cognito scenario. Here the goal is to authenticate your user, and then grant your user access to another AWS service.

1. In the first step your app user signs in through a user pool and receives user pool tokens after a successful authentication.
2. Next, your app exchanges the user pool tokens for AWS credentials through an identity pool.
3. Finally, your app user can then use those AWS credentials to access other AWS services such as Amazon S3 or DynamoDB.

