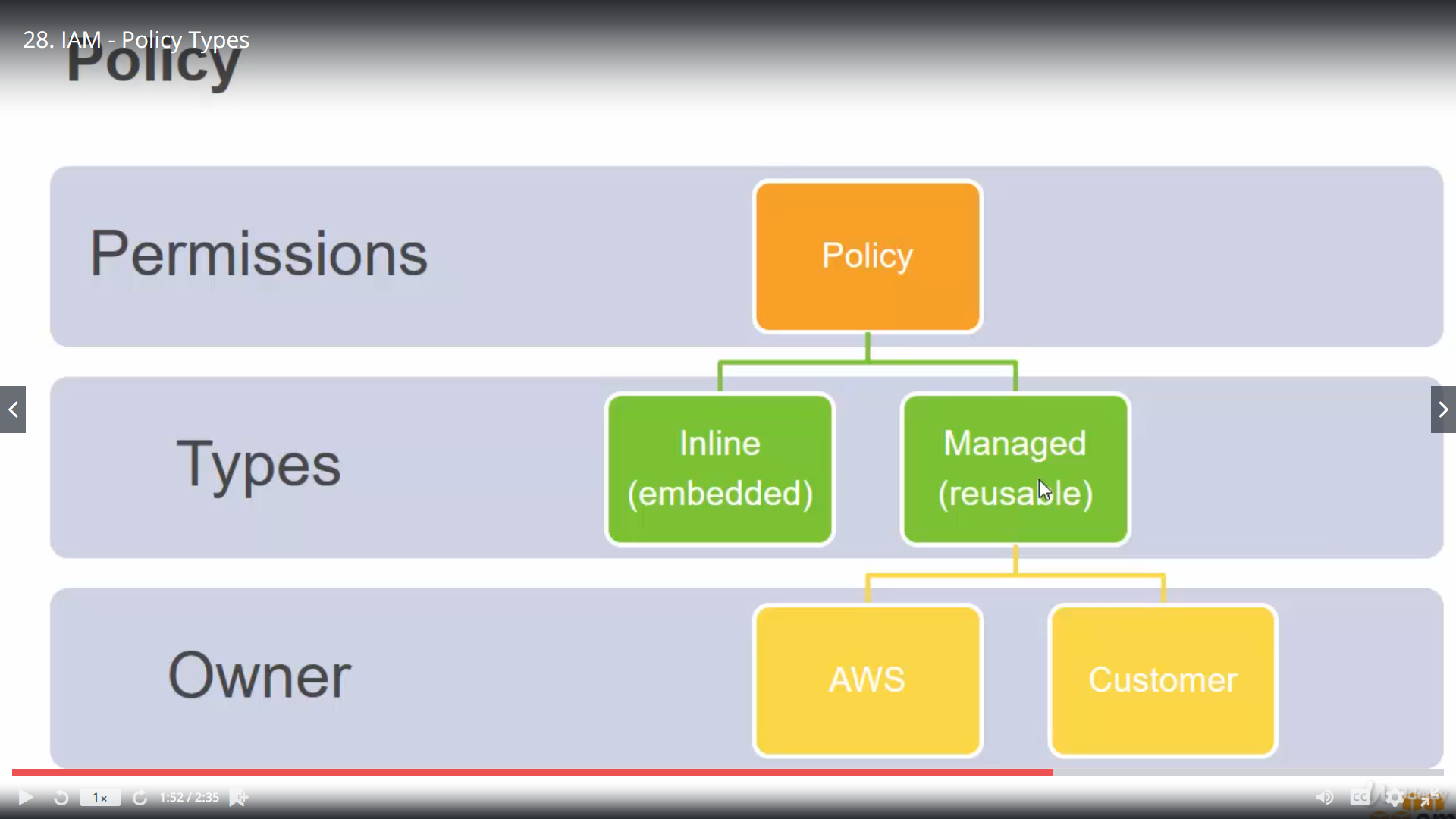
**Identity access management**

* Users : End users
* Groups : A collection of users under one set of permission.
* Roles : You create roles and then assign them to aws resources.
* Policies : A document that defines one or more permission. We can assign policy to each of the components mentioned above.



**What is inline policy and Managed reusable policy ?**

You can create a inline policy for a particular user, if the user is deleted related inline policy will be automatically deleted. For managed reusable policy can be attached with other users as well. They are independent of user.

**Is version maintained for policy ?**AWS manages 5 versions of a policy automatically. At any time we can roll back the the previous version of the policy.

**Can we modify aws existing policy ?**

No, but you can copy and modify your own version.

**What is multifactor user assignment ?**

This is to secure the login access. To activate virtual MFA device we need to install aws mfa compatible application on the user’s smartphone or laptop. Connect using QR code.

**What is user access ?**

By default we have only one access that is root access, which has all the access in aws account i.e. our aws user name. Apart from that if we want to give access to users or users group we create different users.

**Different access type :**

Programmatic access : Enables an access key id and secret access key for the aws api, cli, sdk or other development tools.

AWS management console access : Enables a password that allows users to sign in aws management console.

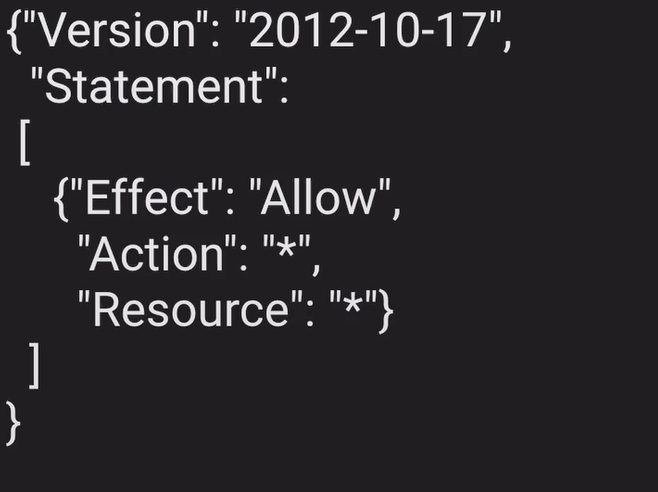
**What is access keyId and accesskey ?**

Each and every programmatic user can have access key id and access key pair to programmatically connect with aws instance. this is not password. If we remove this, we need to regenerate a new access key id and key.

**What is user group policy and roles ?**

We can assign one or more user group and attach some policy like admin, finance. Each group can have multiple users, who have same policy. We can also attach separate policy for the user as well. Roles are different. It is categorized by different AWS service type like EC2, S3, IOT. Here also we can attach a policy. Suppose we create an EC2 role with s3admin access policy then EC2 service can access S3 resources of our account on behalf of the user. EC2 can’t access Lambda services , for that we need assign one more permission for the role.

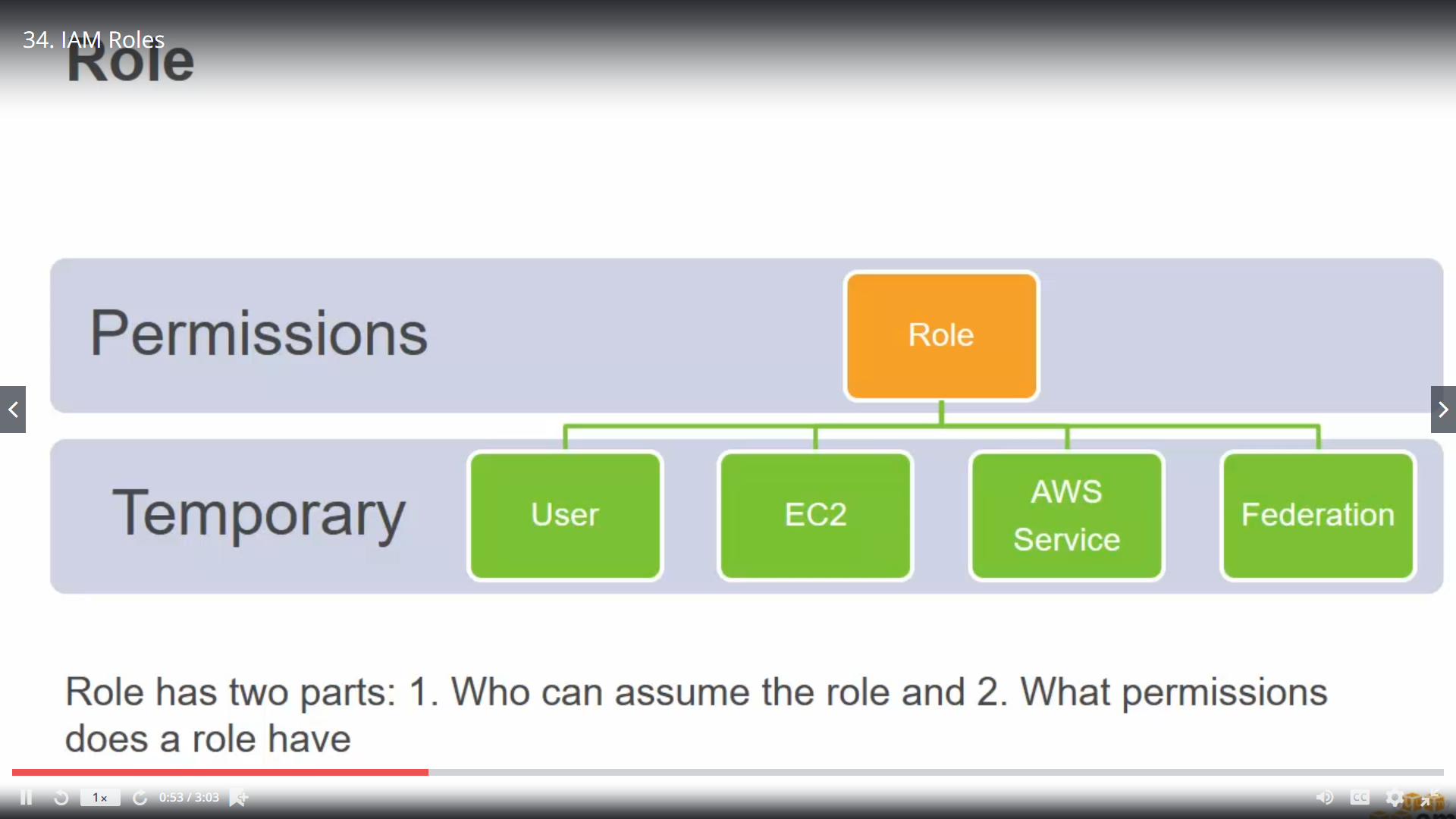
**How policy documents look like ?**



**What is the relation of IAM with regions ?**

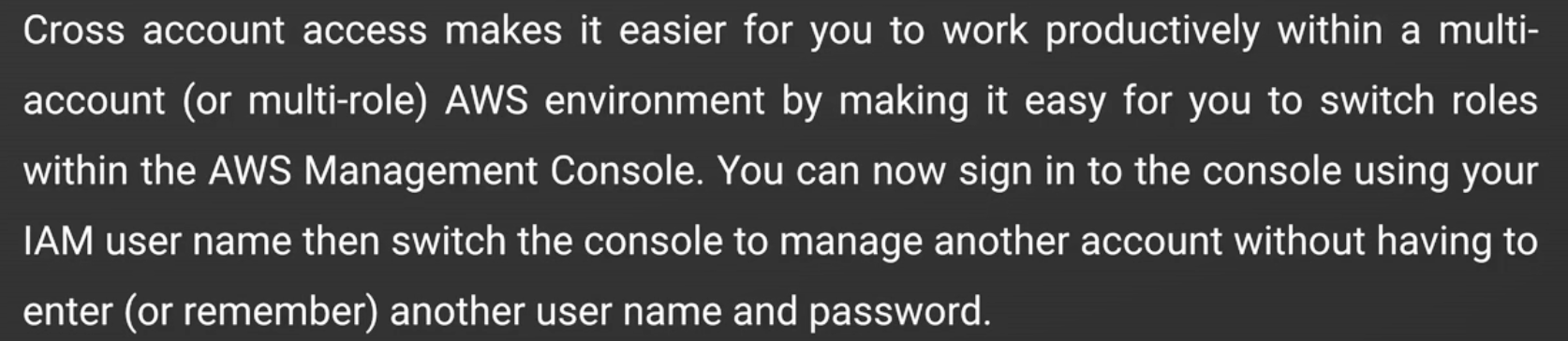
In aws for other services we need to select a region like Mumbai, uk. IAM is not region specific. For each region we canot have all the aws services. But IAM is available to all the regions.

**What is Role ?**



Role is a mechanism or service with which you can grant temporary privileges to end user or a service that allow them to access your resources. Rather than create so many users in your account you can create other users in different account, give them a IAM role to access aws resources in your account. Similarly if you have EC2 instances and lambda service instead of managing credentials in individual application level you can assign a role to the instance, so each application deployed in that ec2 instance can access an aws service. Federation is when users are logged in with google, facebook or any other openid connect. You can map the temporary user credentials with a role to access aws services.

**What is cross account access ?**



**Steps :**

1. Identify account numbers for both production and Development.
2. Create an user group Development from Dev account
3. Create an user in the group.
4. Log in to Production account, create your own policy
5. Create a policy named say ‘red-write-bucket-policy’

{

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

"Statement": [

{

"Effect": "Allow",

"Action": "s3:ListAllMyBuckets",

"Resource": "arn:aws:s3:::\*"

},

{

"Effect": "Allow",

"Action": [

"s3:ListBucket",

"s3:GetBucketLocation"

],

"Resource": "arn:aws:s3:::productionapp"

},

{

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:PutObject",

"s3:DeleteObject"

],

"Resource": "arn:aws:s3:::productionapp/\*"

}

]

}

1. Replace the name of the bucket with prodcutionapp mentioned in the policy document.
2. Create new role say “MyDeveloperAccess” from Role for cross account access option
3. Enter the account id for developer account.
4. Attach the newly created policy to the role.
5. Logout from production account and login to developer’s account.
6. Developer’s group in this account has already a policy attached. Attach a new inline policy.

{

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

"Statement": {

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/MyDevelopersAccess"

}

}

1. Replace the production account id and MyDeveloperaAccess with the role you created in production account.
2. Login as user in developer’s account, from dropdown at the top right and select switch role.
3. Put production account id and Mydeveloperaccess , which is role in production.

**What is trusted and access policy ?**

Access policy means it allows some certain actions to the user for the resource.

As mentioned above

{

"Effect": "Allow",

"Action": "s3:ListAllMyBuckets",

"Resource": "arn:aws:s3:::\*"

}

Trusted policy defines who are the trusted entities, it can be another aws account or an federated user. Mentioned above this is the trust policy

1. policy.

{

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

"Statement": {

"Effect": "Allow",

"Action": "sts:AssumeRole",

"Resource": "arn:aws:iam::PRODUCTION-ACCOUNT-ID:role/MyDevelopersAccess"

}

}

Which defines only this user can access the resource which has the access policy named MyDevelopersAccess. We can define in this policy like any user asserted by identity provider and connect aws with saml .

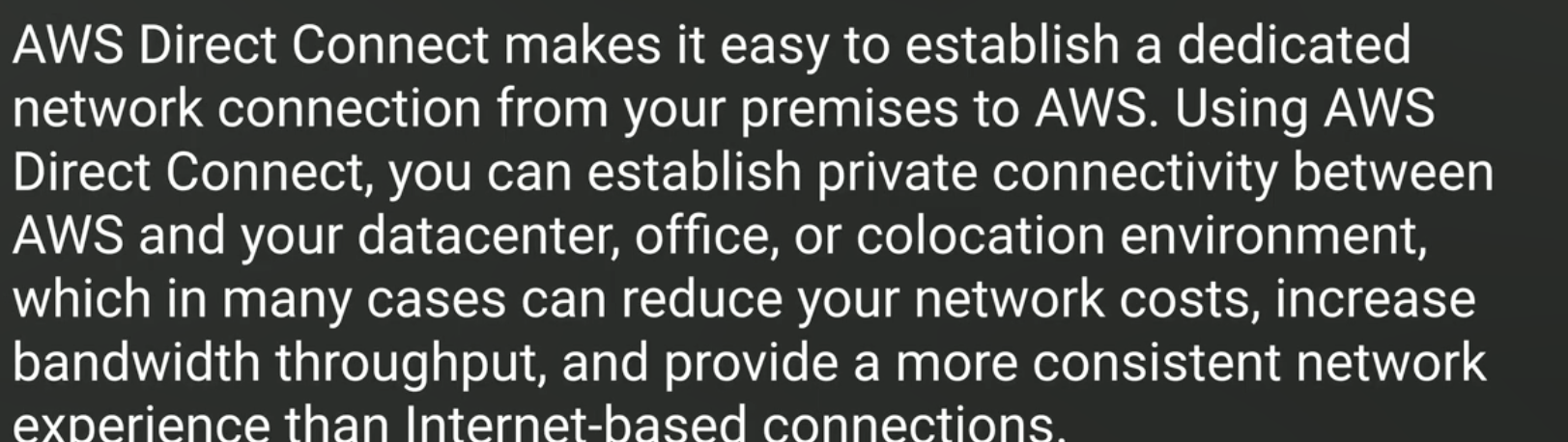
arn:aws :: 111:saml-provider/ABC

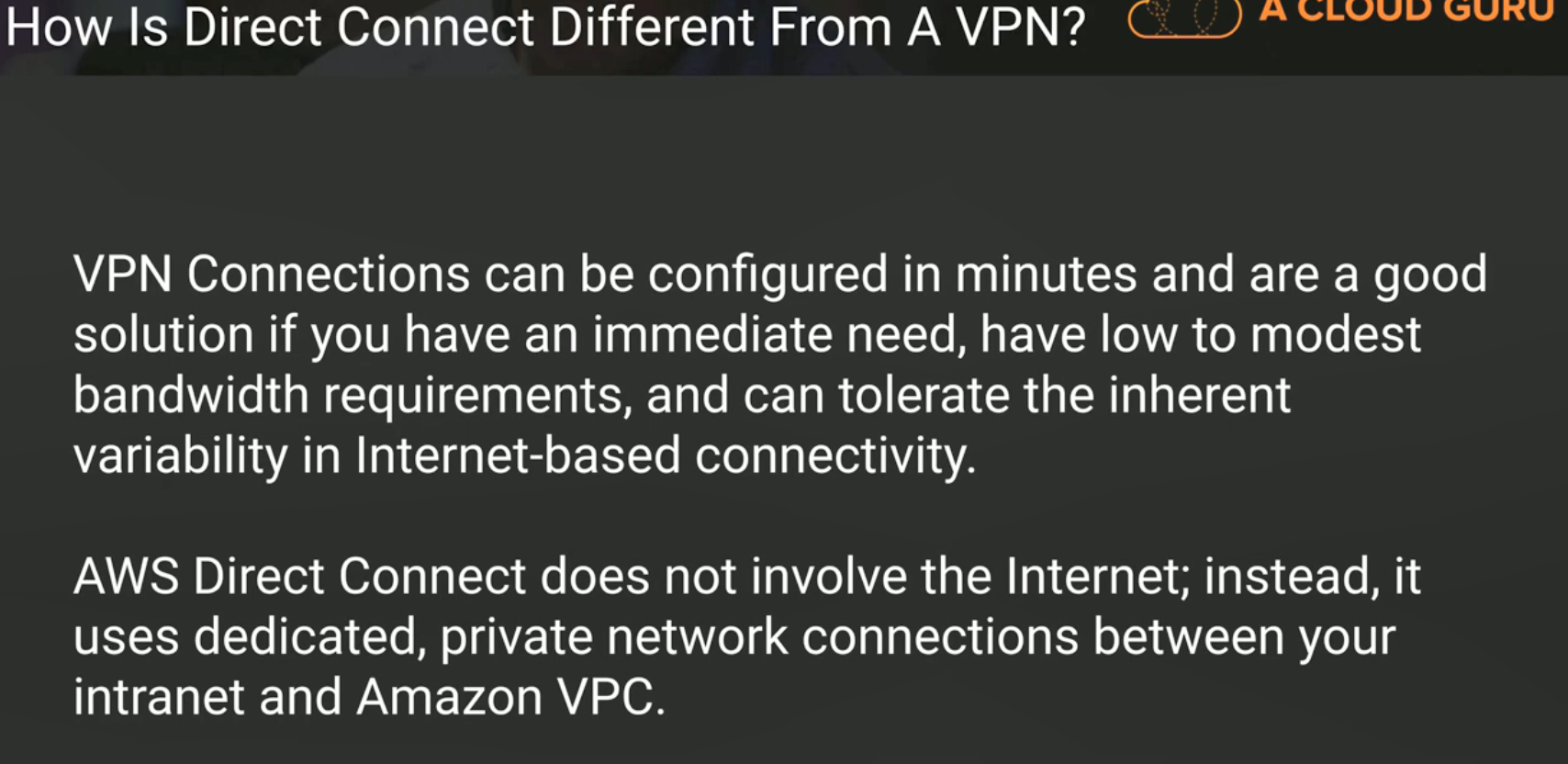
**Difference between Delegation and federation ?**

Cross account access is delegation and Using corporate or social directory is federation.

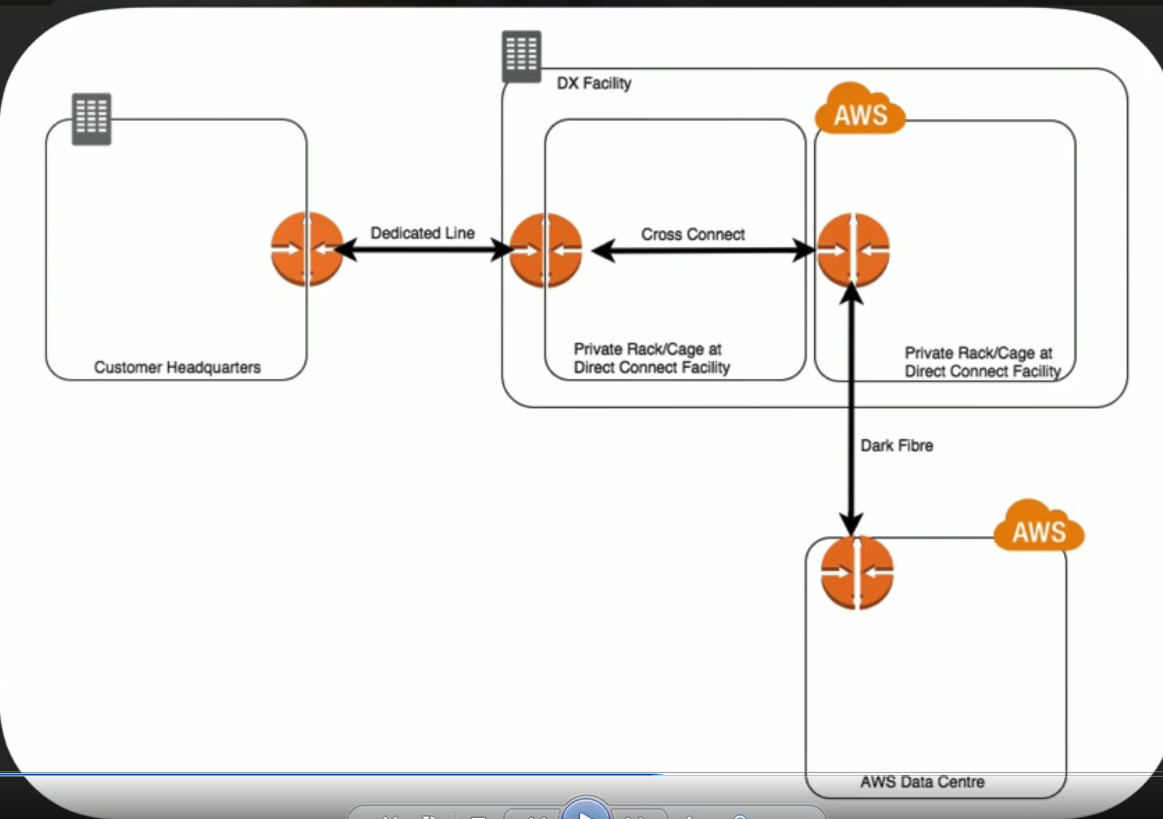
Both try to access a temporary access to particular aws service through STS.

**What is direct connect ?**

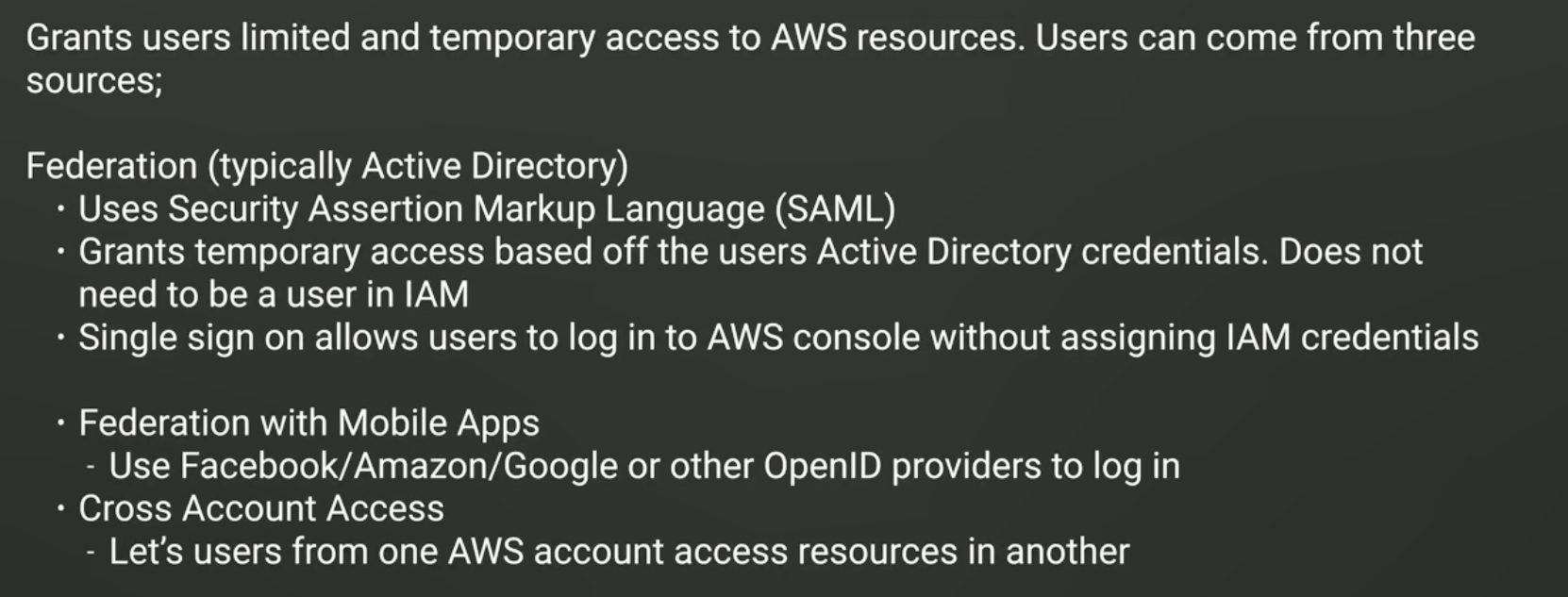




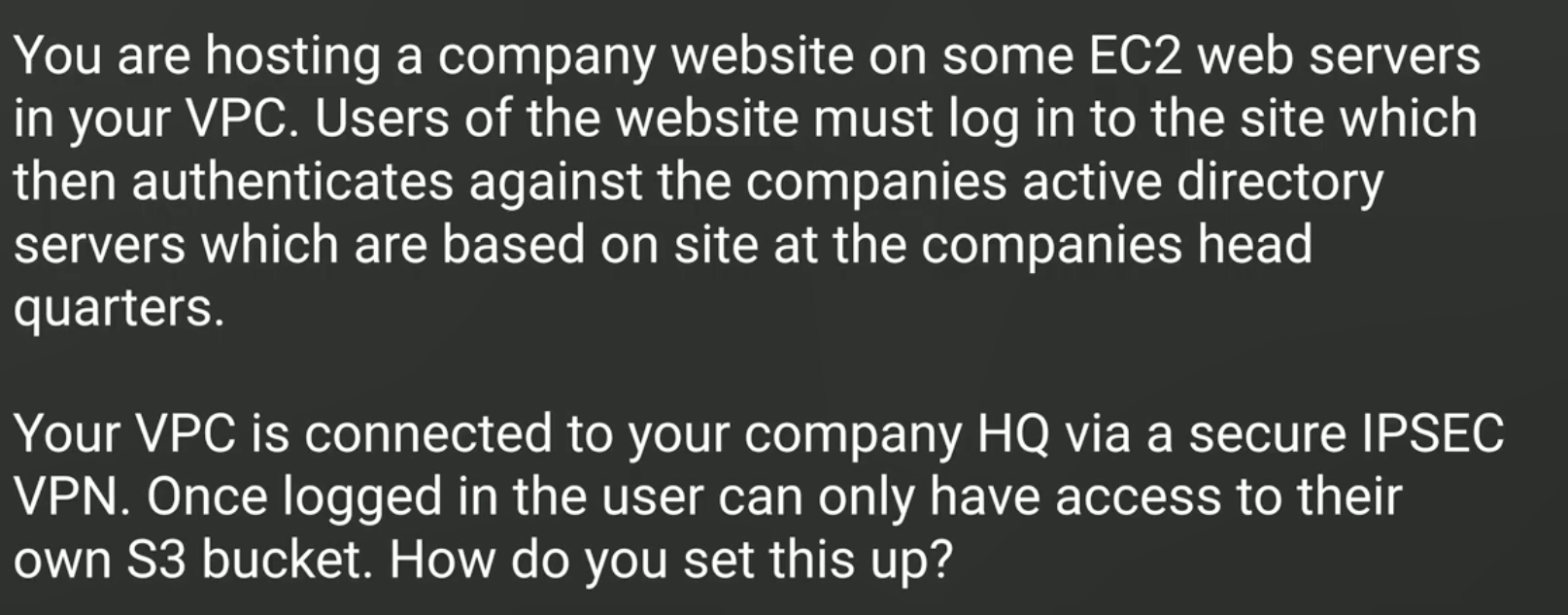
**How does it work**

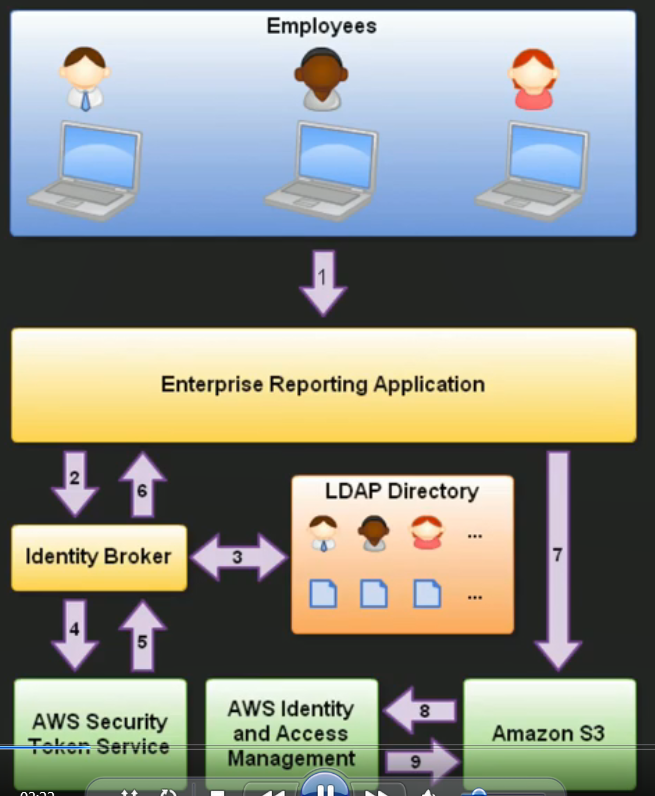


**AWS STS**

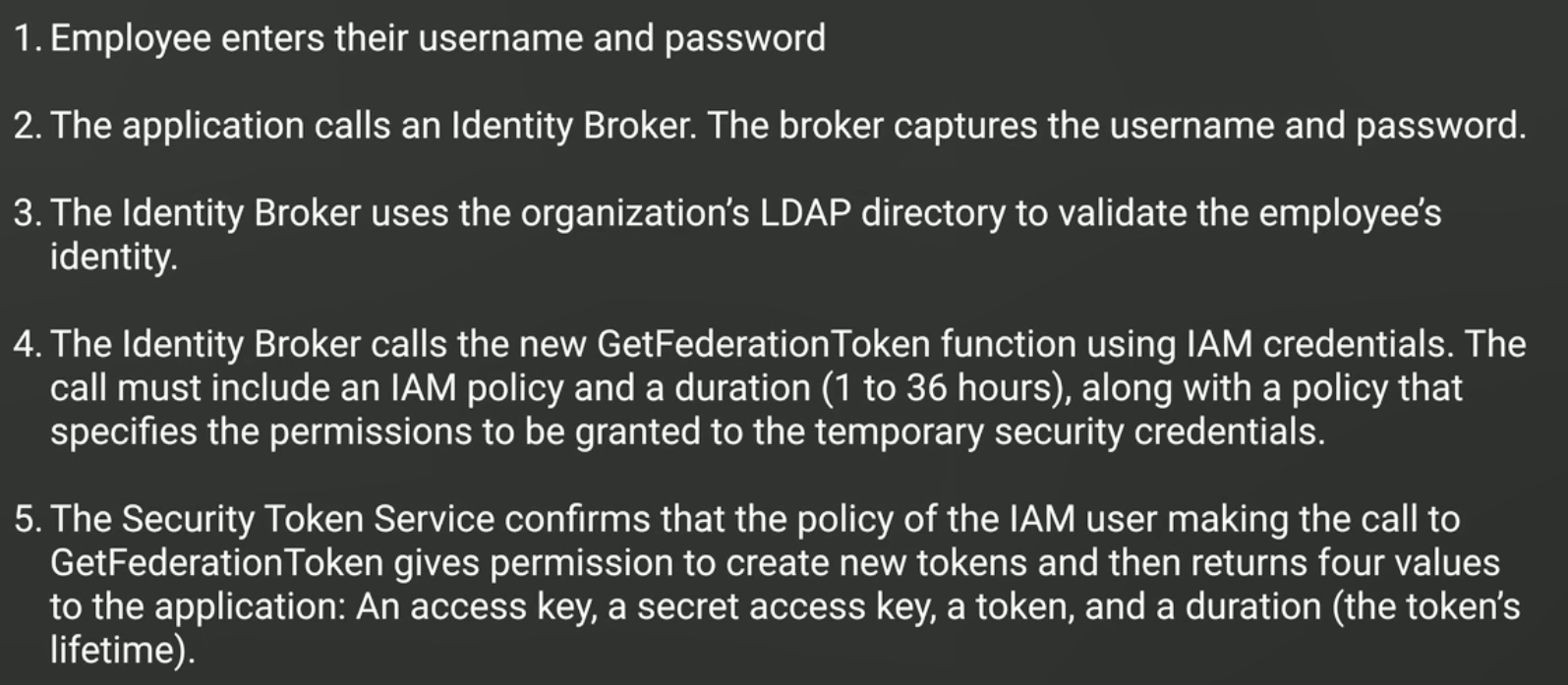


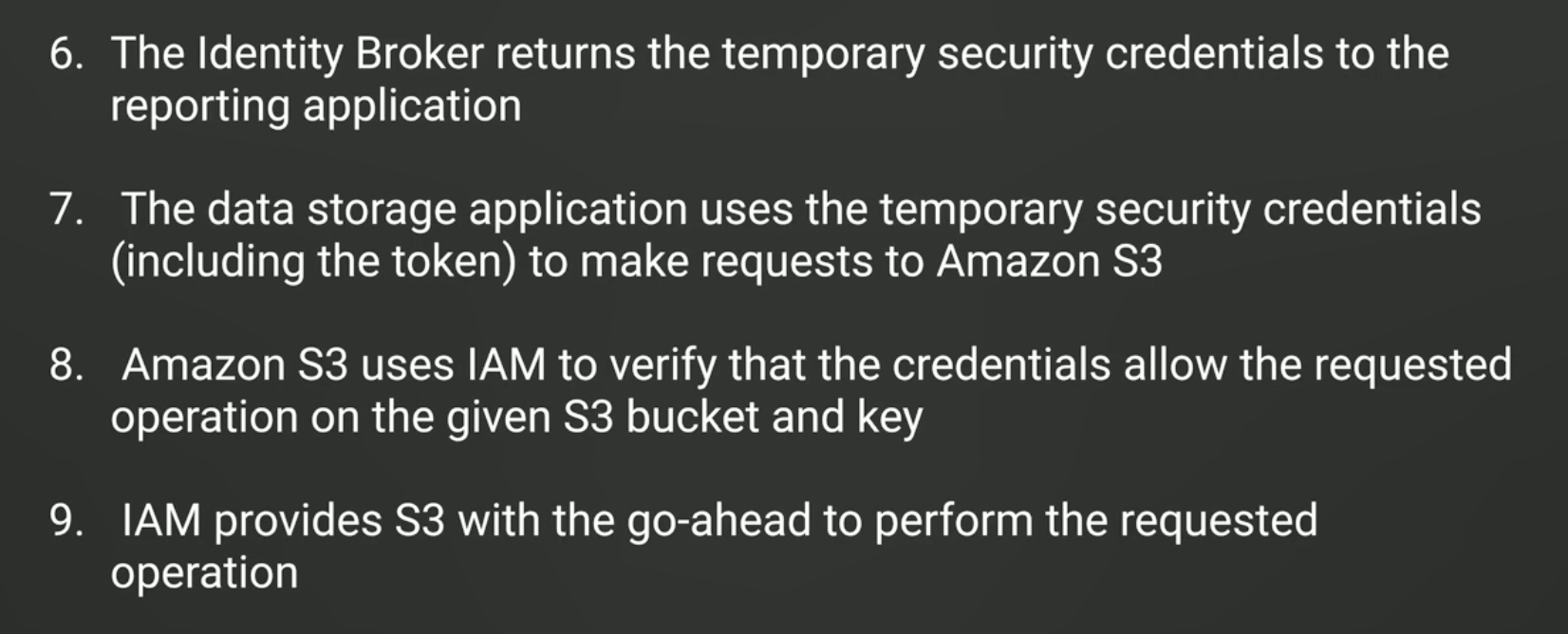
**Sample question :**





Steps to solve :





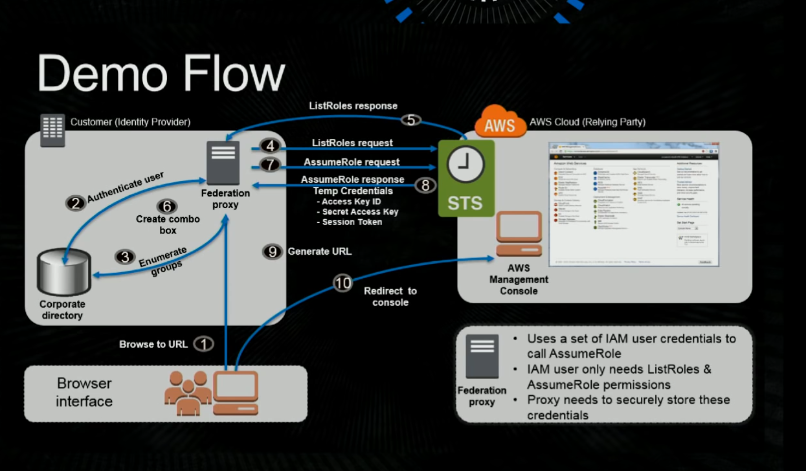
**What is session ?**

By providing IAM trust policy to Identity broker got a session from STS which is nothing but

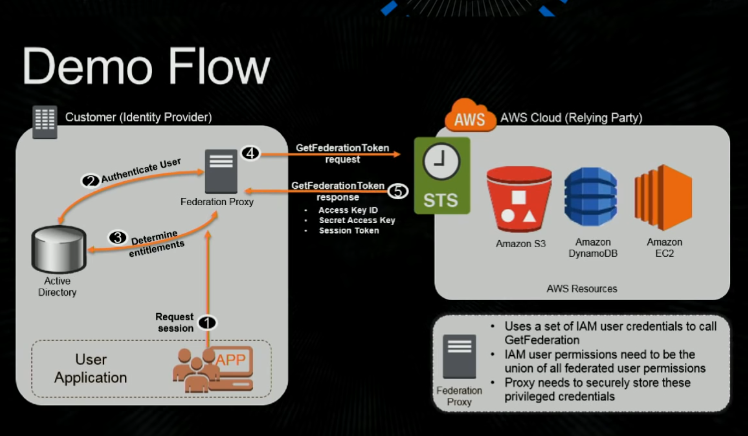
access key, secret access key, expiration date and a session token.

**AssumeRole action from federated user through federation proxy**

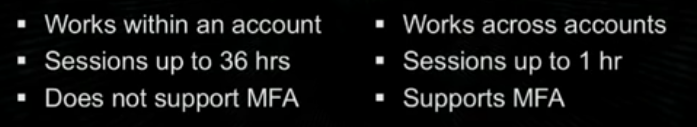
For aws console



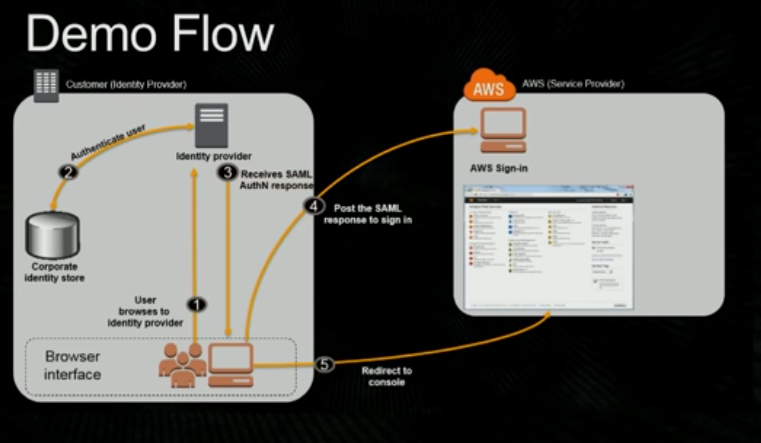
for api access using getfederationToken



**Difference between getfederationToken and AssumeRole**



**AssumeRoleWithSAML**

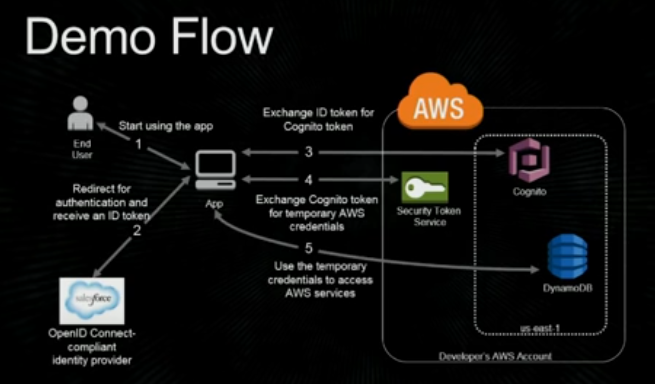


**What are the different api to federate a user**

**Comparing your API options**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AWS STS API** | **Who can call** | **Credential lifetime (min | max | default)** | **MFA support¹** | **Session policy support²** | **Restrictions on resulting temporary credentials** |
| [AssumeRole](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html) | IAM user or IAM role with existing temporary security credentials | 15 m | Maximum session duration setting³ | 1 hr | Yes | Yes | Cannot call GetFederationToken or GetSessionToken. |
| [AssumeRoleWithSAML](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRoleWithSAML.html) | Any user; caller must pass a SAML authentication response that indicates authentication from a known identity provider | 15 m | Maximum session duration setting³ | 1 hr | No | Yes | Cannot call GetFederationToken or GetSessionToken. |
| [AssumeRoleWithWebIdentity](https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRoleWithWebIdentity.html) | Any user; caller must pass a web identity token that indicates authentication from a known identity provider | 15 m | Maximum session duration setting³ | 1 hr | No | Yes | Cannot call GetFederationToken or GetSessionToken. |
| [GetFederationToken](https://docs.aws.amazon.com/STS/latest/APIReference/API_GetFederationToken.html) | IAM user or AWS account root user | IAM user: 15 m | 36 hr | 12 hr  Root user: 15 m | 1 hr | 1 hr | No | Yes | Cannot call IAM API operations directly.⁴  Cannot call AWS STS API operations except GetCallerIdentity.  SSO to console is allowed.⁵ |
| [GetSessionToken](https://docs.aws.amazon.com/STS/latest/APIReference/API_GetSessionToken.html) | IAM user or AWS account root user | IAM user: 15 m | 36 hr | 12 hr  Root user: 15 m | 1 hr | 1 hr | Yes | No | Cannot call IAM API operations unless MFA information is included with the request.  Cannot call AWS STS API operations except AssumeRole or GetCallerIdentity.  SSO to console is not allowed.⁶ |

**Use of aws cognito and AssumeRoleWithWebIdentity**



Cognito seats over openid identity provider which is responsible to call AssumeRoleWithWebIndentity to sts using webidentity secure token. It needs the client id provided by facebook developers or salesforce when you registrar your application in those IDP.

**If you want to use an SSL protocol but do not want to terminate the connection on your load balancer, you can use a \_\_\_\_\_\_\_\_\_\_ protocol for connection from the client to your load balancer.**

Ans TCP

Explanation : Answer: (D) If you want to use an SSL protocol but do not want to terminate the connection on your load balancer, you can use a TCP protocol for connection from the client to your load balancer. Use the SSL protocol for connection from the load balancer to your back-end application, and install certificates on all the back-end instances handling requests.