Introduction

SAML 2.0 Security Assertion Markup Language.

Open standard used by many Identity providers (eg. MS ADFS)

SAML 2.0 Federation allows you to **Indirectly** use on-premises identities with AWS (Console & CLI)

Enterprise Identity Provider and **SAML 2.0 Compatible**

Existing Identity management team to allow AWS Access

This uses IAM Roles and AWS Temporary Credentials (12 hours validity)

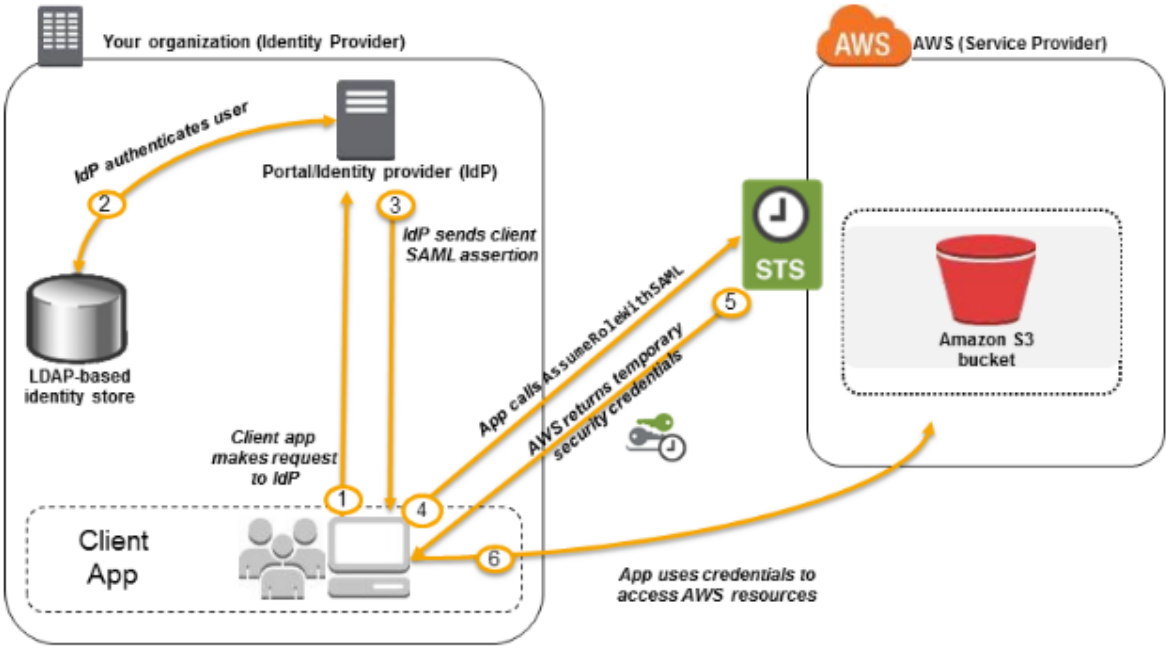
NOTE: Google, Facebook, Twitter, or Web or anything that does not support SAML support, we should not use SAML Identity Federation

You can't use External Identity Providers. It needs to be SAML based Identity providers.

Application and AWS Environment setups:

a. From client Application:
    i. SAML trust is completed by configuring the Organization's IdP with information about AWS and the role(s) that you want the federated users to use. This is referred to as configuring **relying party trust between your IdP and AWS**
b. From AWS Account Site:
    i. Create a SAML provider entity in AWS using the SAML metadata document provided by the Organizations IdP to establish a "**trust**" between your AWS account and the IdP
    ii. SAML metadata document includes the issuer name, a creation date, an expiration date, and keys that AWS can use to validate authentication responses (assertions) from your organization.
    iii. Create an IAM role and define the following two policies:
        1. Trust Policy: Trust policy with the SAML provider as the principal, which establishes a trust relationship between the organization and AWS
        2. Permission Policy: Permission policy establishes what users from the organization are allowed to do in AWS

Main steps of workflow:



Step 1: Application calls the sign-in interface for the Organization IdP to login
Step 2: IdP authenticates the user and generates a SAML authentication response which includes assertions that identify the user and include attributes about the user
Step 3: Application then makes an unsigned call to the STS service with the **AssumeRoleWithSAML** action to request temporary security credentials.
Step 4: Application passes

1. the ARN of the SAML provider,
2. the ARN of the role to assume,
3. the SAML assertion about the current user returned by IdP and
4. the time for which the credentials should be valid.
5. an optional IAM Policy parameter can be provided to further restrict the permissions to the user

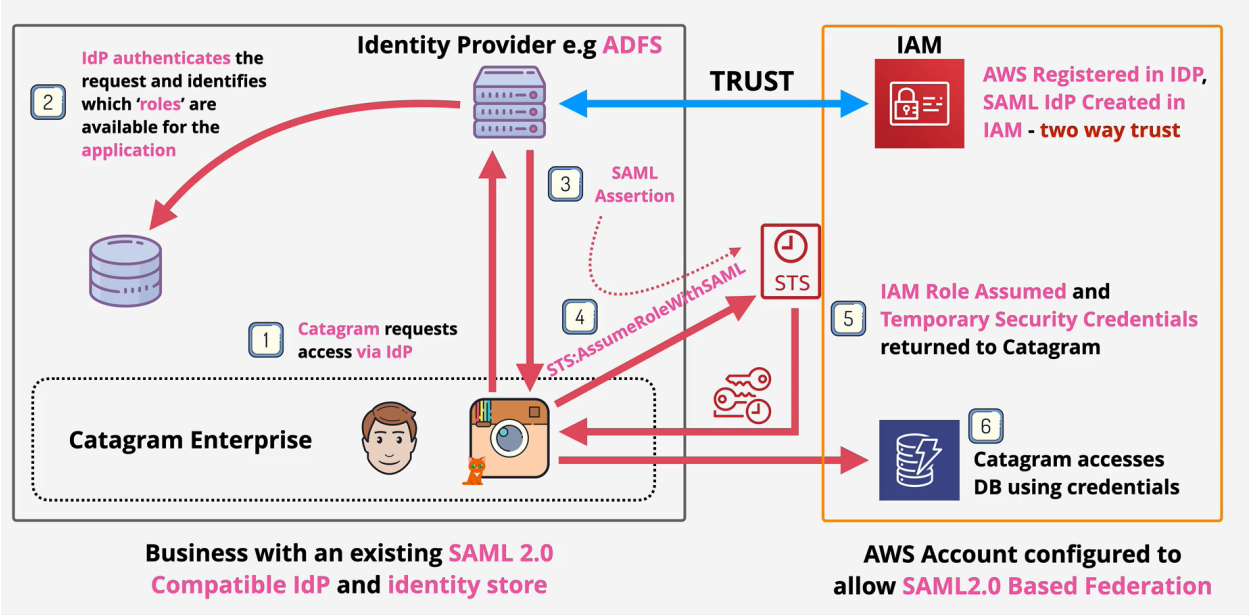
Step 5: AWS verifies that the SAML assertion is trusted and valid and if so, returns

1. temporary security credentials (access key, secret access key, session token, expiry time) to the application that have the permissions for the role named in the request.
2. STS response also includes metadata about the user from the IdP, such as the unique user ID that the IdP associates with the user.

Step 6: Using the Temporary credentials, the application makes signed requests to AWS to access the services

Step 7: Application can cache the temporary security credentials and refresh them before their expiry accordingly. Temporary credentials, by default, are good for an hour.

Example:



An External User Tried to Access AWS Console