IAM Role – Identity Providers and Federation

# Graphical user interface, text, application Description automatically generated

# A picture containing diagram Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Diagram

Description automatically generated

# IAM Role – Identity Providers and Federation

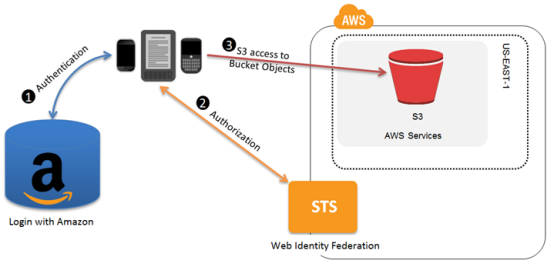
* Identity Provider can be used to grant external user identities permissions to AWS resources without having to be created within your AWS account.
* External user identities can be authenticated either through the organization’s authentication system or through a well-know identity provider such as login with Amazon, Google etc.
* Identity providers help keep the AWS account secure without having the need to distribute or embed long term in the application
* To use an IdP, you create an IAM identity provider entity to establish a trust relationship between your AWS account and the IdP.
* **IAM supports IdPs that are compatible with OpenID Connect (OIDC) or SAML 2.0 (Security Assertion Markup Language 2.0)**

Graphical user interface, text, application

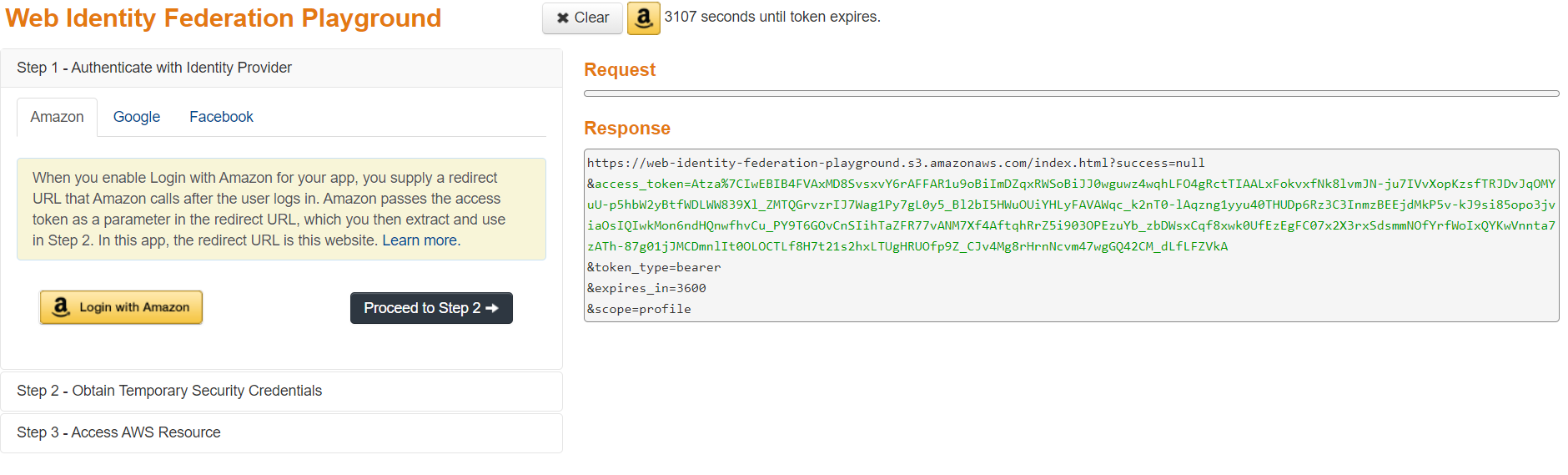
Description automatically generated

## Web Identity Federation

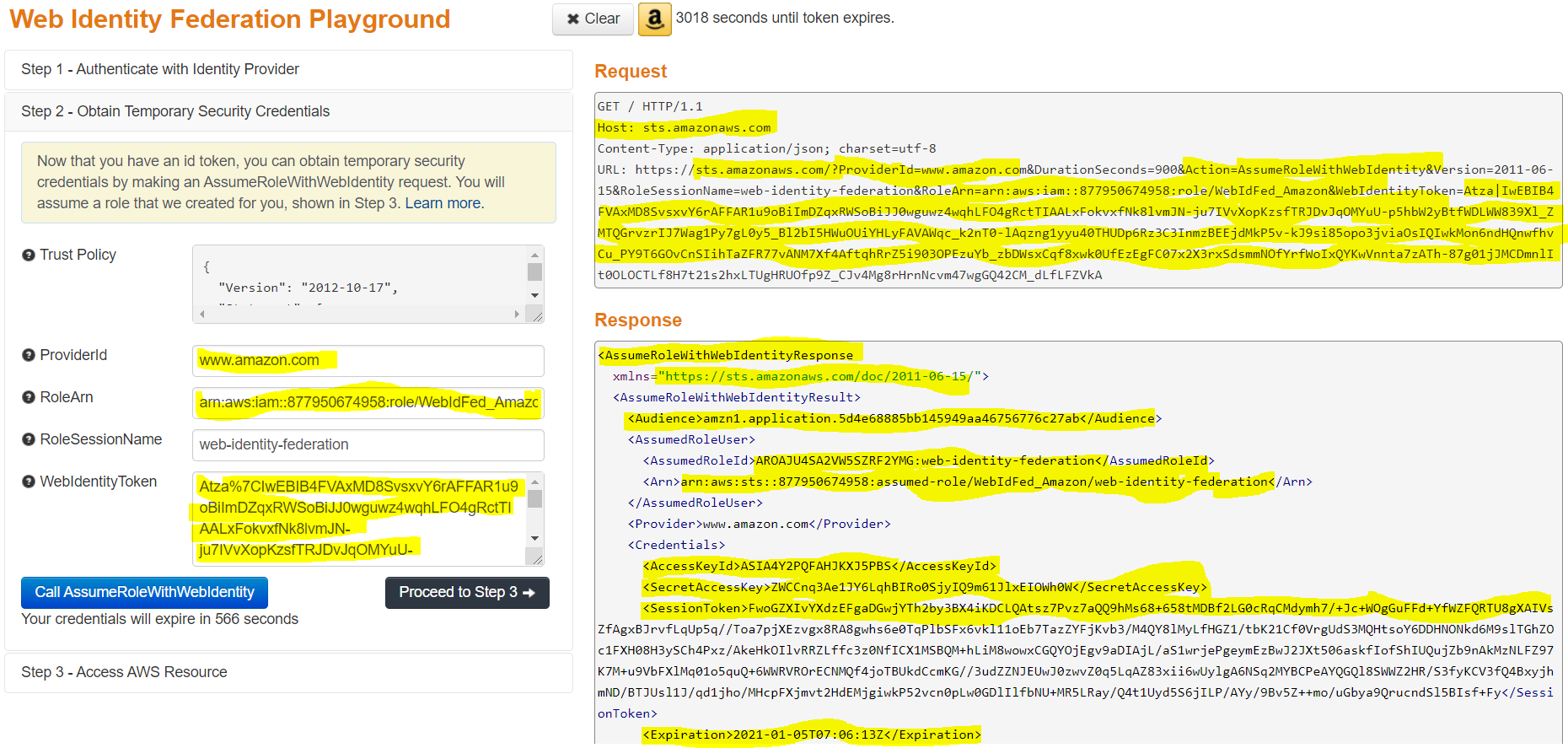
### Complete Process Flow



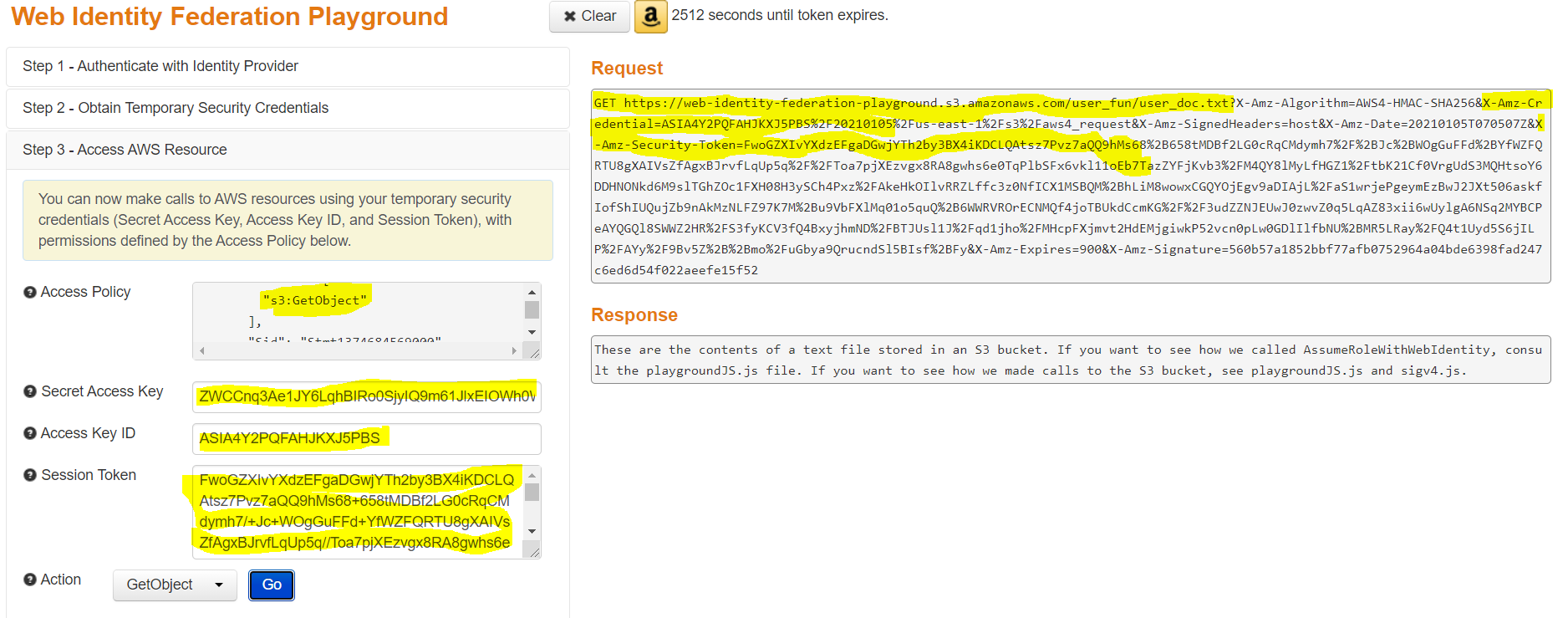
1. Mobile or Web Application needs to be configured with the IdP which gives each application a unique ID or client ID (also called audience)
2. Create an Identity Provider entity for OIDC compatible IdP in IAM.
3. Create IAM role and define the
   1. Trust policy – specify the IdP (like Amazon) as the Principal (the trusted entity), and include a Condition that matches the IdP assigned app ID
   2. Permission policy – specify the permissions the application can assume
4. Application calls the sign-in interface for the IdP to login
5. IdP authenticates the user and returns an authentication token (OAuth access token or OIDC ID token) with information about the user to the application



1. Application then makes an unsigned call to the STS service with the ***AssumeRoleWithWebIdentity***action to request temporary security credentials.
2. Application passes the IdP’s authentication token along with the Amazon Resource Name (ARN) for the IAM role created for that IdP.
3. AWS verifies that the token is trusted and valid and if so, returns temporary security credentials (access key, secret access key, session token, expiry time) to the application that have the permissions for the role that you name in the request.



1. STS response also includes metadata about the user from the IdP, such as the unique user ID that the IdP associates with the user.
2. Using the Temporary credentials, the application makes signed requests to AWS
3. User ID information from the identity provider can distinguish users in the app for e.g., objects can be put into S3 folders that include the user ID as prefixes or suffixes. This lets you create access control policies that lock the folder so only the user with that ID can access it.
4. Application can cache the temporary security credentials and refresh them before their expiry accordingly. Temporary credentials, by default, are good for an hour.

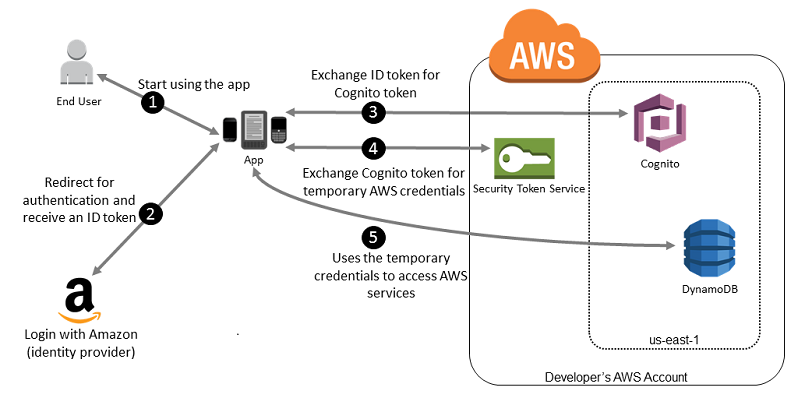


[Interactive Website](https://web-identity-federation-playground.s3.amazonaws.com/index.html) provides a very good way to understand the flow

## Mobile or Web Identity Federation with Cognito

### Amazon Cognito

* Use Amazon Cognito as the identity broker for almost all web identity federation scenarios
* Amazon Cognito is easy to use and provides additional capabilities like anonymous (unauthenticated) access
* Amazon Cognito also helps synchronizing user data across devices and providers



Timeline

Description automatically generated with medium confidence

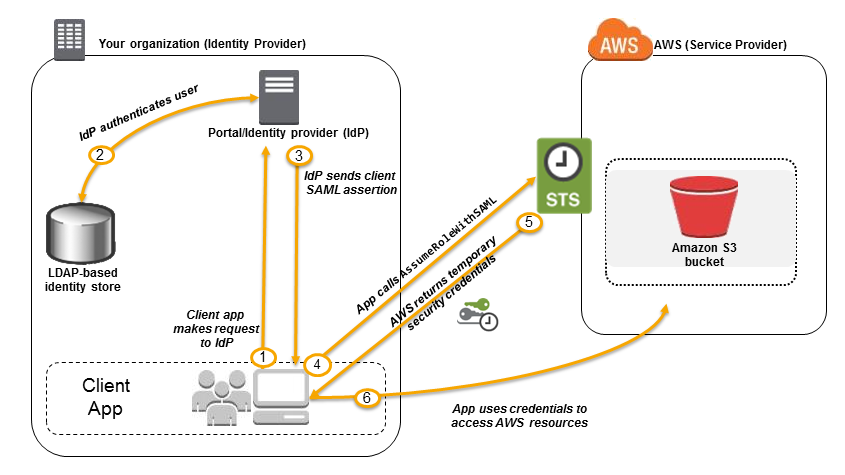
Timeline

Description automatically generated

## SAML 2.0-based Federation

* AWS supports identity federation with SAML 2.0 (Security Assertion Markup Language 2.0), an open standard that many identity providers (IdPs) use.
* SAML 2.0 based federation feature enables federated single sign-on (SSO),  so users can log into the AWS Management Console or call the AWS APIs without having to create an IAM user for everyone in the organization
* By using SAML, the process of configuring federation with AWS can be simplified by using the IdP’s service instead of writing custom identity proxy code.
* **This is useful in organizations that have integrated their identity systems (such as Windows Active Directory or OpenLDAP) with software that can produce SAML assertions to provide information about user identity and permissions (such as Active Directory Federation Services or Shibboleth)**

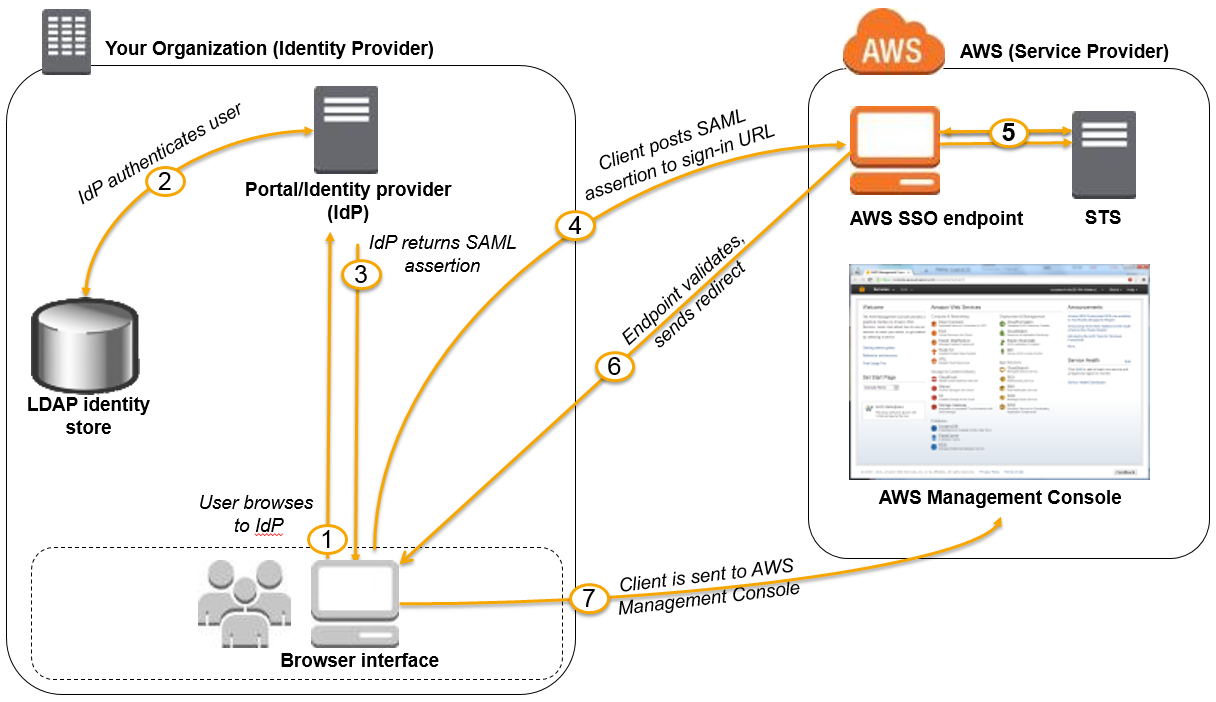
### Complete Process Flow



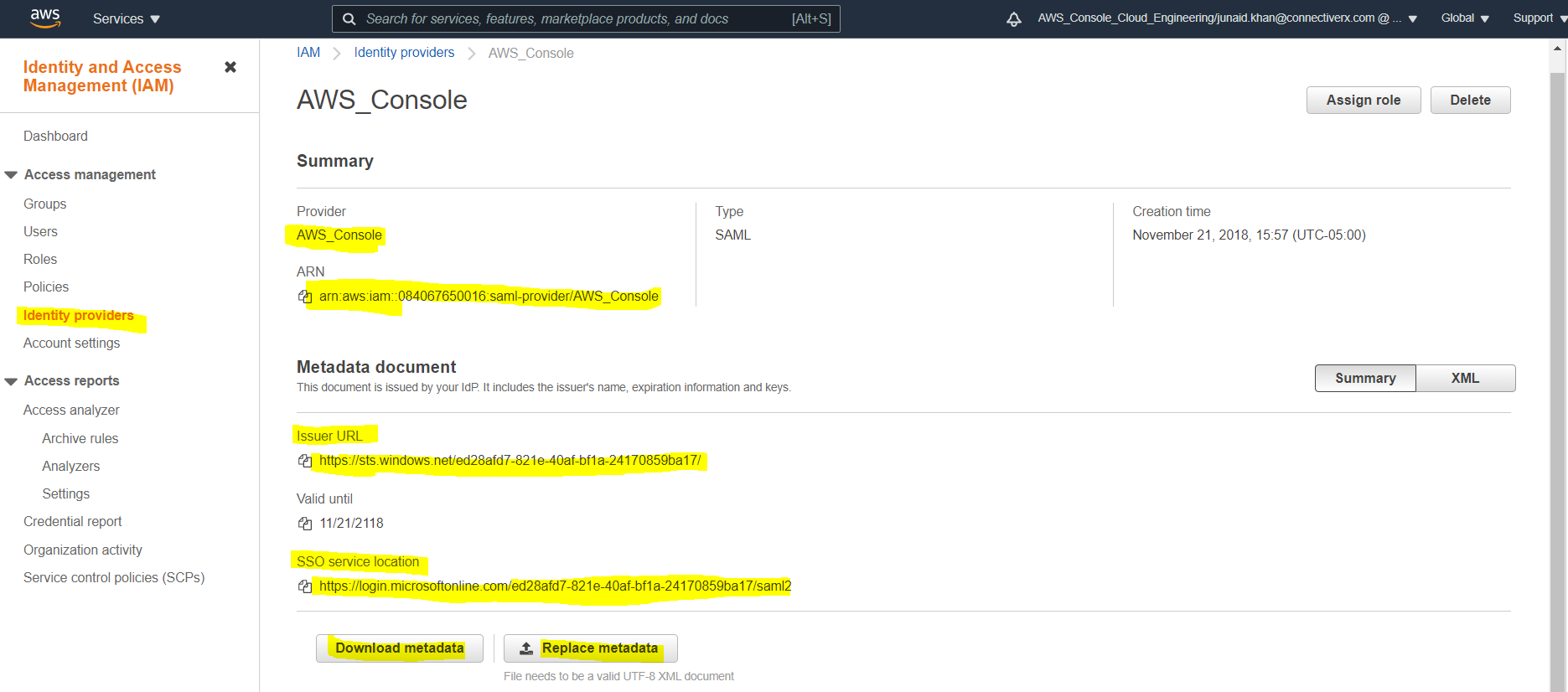
1. Create a SAML provider entity in AWS using the SAML metadata document provided by the Organizations IdP to establish a “trust” between your AWS account and the IdP
2. SAML metadata document includes the issuer name, a creation date, an expiration date, and keys that AWS can use to validate authentication responses (assertions) from your organization.
3. Create IAM roles which defines
   1. Trust policy with the SAML provider as the principal, which establishes a trust relationship between the organization and AWS
   2. Permission policy establishes what users from the organization are allowed to do in AWS
4. 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
5. Application calls the sign-in interface for the Organization IdP to login
6. IdP authenticates the user and generates a SAML authentication response which includes assertions that identify the user and include attributes about the user
7. Application then makes an unsigned call to the STS service with the ***AssumeRoleWithSAML***action to request temporary security credentials.
8. Application passes the ARN of the SAML provider, the ARN of the role to assume, the SAML assertion about the current user returned by IdP and the time for which the credentials should be valid. An optional IAM Policy parameter can be provided to further restrict the permissions to the user
9. AWS verifies that the SAML assertion is trusted and valid and if so, returns 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.
10. STS response also includes metadata about the user from the IdP, such as the unique user ID that the IdP associates with the user.
11. Using the Temporary credentials, the application makes signed requests to AWS to access the services
12. Application can cache the temporary security credentials and refresh them before their expiry accordingly. Temporary credentials, by default, are good for an hour.

SAML 2.0 based federation can also be used to grant access to the federated users to the AWS Management console. This requires the use of the AWS SSO endpoint instead of directly calling the *AssumeRoleWithSAML*API. The endpoint calls the API for the user and returns a URL that automatically redirects the user’s browser to the AWS Management Console.

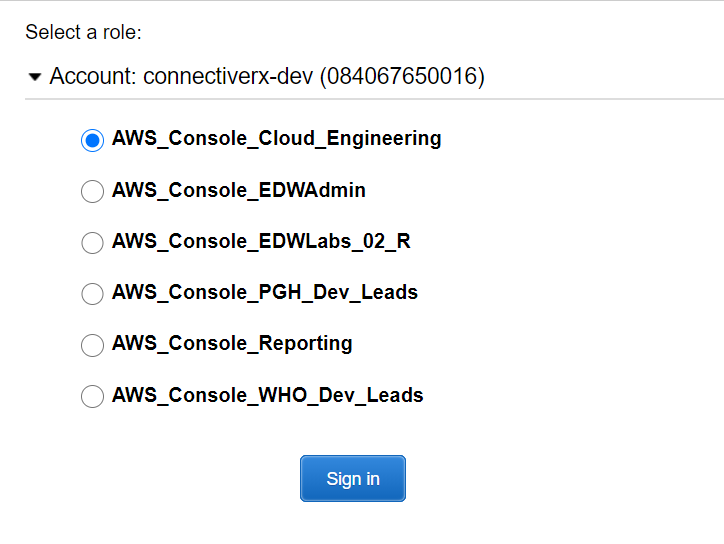
#### **Complete Process Flow**



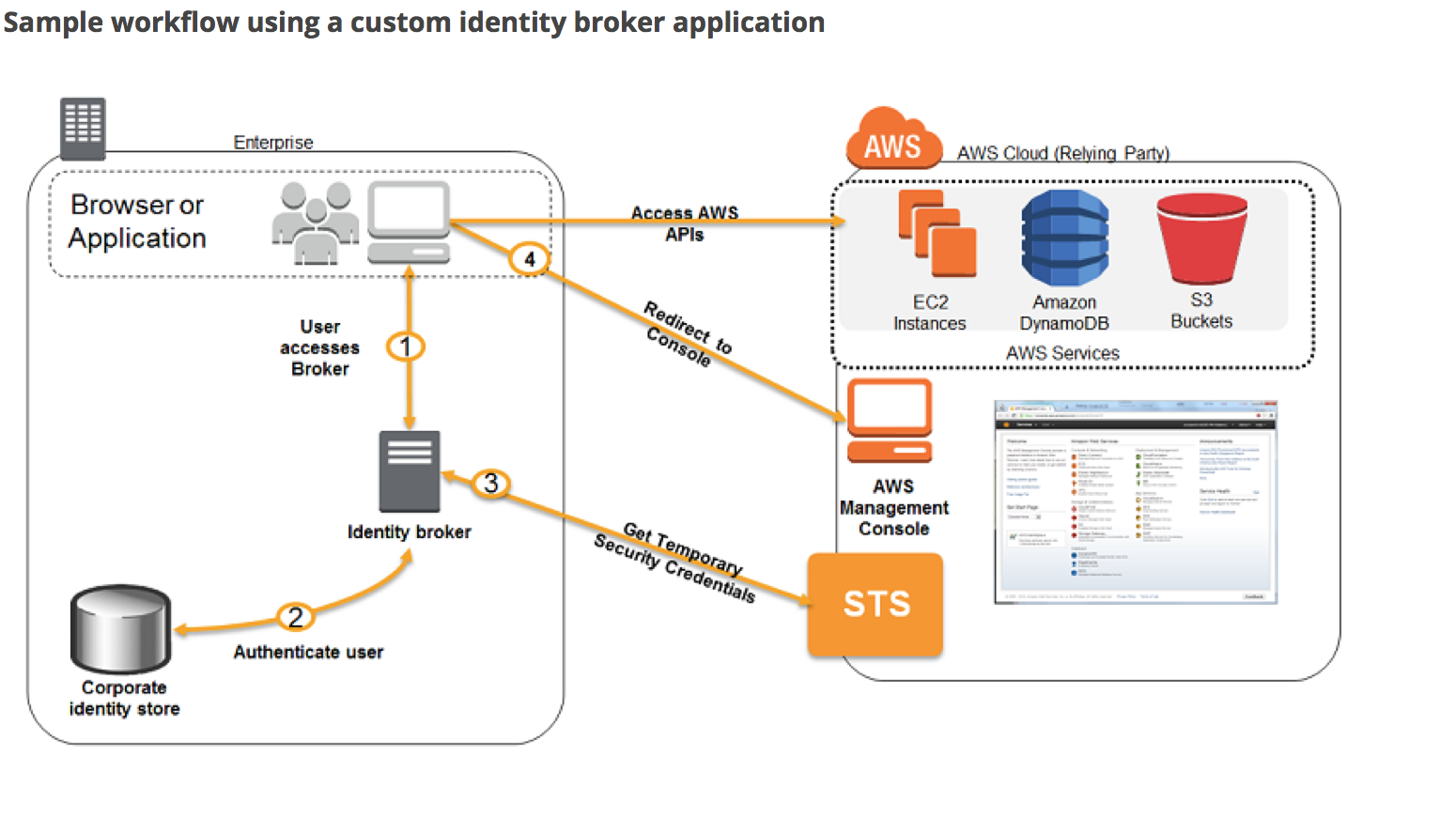
1. User browses to the organization’s portal and selects the option to go to the AWS Management Console.
2. Portal performs the function of the identity provider (IdP) that handles the exchange of trust between the organization and AWS.
3. Portal verifies the user’s identity in the organization.
4. Portal generates a SAML authentication response that includes assertions that identify the user and include attributes about the user.
5. Portal sends this response to the client browser.
6. Client browser is redirected to the AWS SSO endpoint and posts the SAML assertion.
7. AWS SSO endpoint handles the call for the *AssumeRoleWithSAML*API action on the user’s behalf and requests temporary security credentials from STS and creates a console sign-in URL that uses those credentials.



1. AWS sends the sign-in URL back to the client as a redirect.
2. Client browser is redirected to the AWS Management Console. If the SAML authentication response includes attributes that map to multiple IAM roles, the user is first prompted to select the role to use for access to the console.



### Custom Identity broker Federation



* If the Organization doesn’t support SAML compatible IdP, a Custom Identity Broker can be used to provide the access
* Custom Identity Broker should perform the following steps
  + Verify that the user is authenticated by the local identity system.
  + Call the AWS Security Token Service (AWS STS) ***AssumeRole***(recommended) or **GetFederationToken** (by default, has a expiration period of 36 hours) APIs to obtain temporary security credentials for the user.
  + Temporary credentials limit the permissions a user has to the AWS resource
  + Call an AWS federation endpoint and supply the temporary security credentials to get a sign-in token.
  + Construct a URL for the console that includes the token.
  + URL that the federation endpoint provides is valid for 15 minutes after it is created.
  + Give the URL to the user or invoke the URL on the user’s behalf.

## AWS Certification Exam Practice Questions

1. A photo-sharing service stores pictures in Amazon Simple Storage Service (S3) and allows application sign-in using an OpenID Connect-compatible identity provider. Which AWS Security Token Service approach to temporary access should you use for the Amazon S3 operations?
   1. SAML-based Identity Federation
   2. Cross-Account Access
   3. AWS IAM users
   4. **Web Identity Federation**
2. Which technique can be used to integrate AWS IAM (Identity and Access Management) with an on-premise LDAP (Lightweight Directory Access Protocol) directory service?
   1. Use an IAM policy that references the LDAP account identifiers and the AWS credentials.
   2. Use SAML (Security Assertion Markup Language) to enable single sign-on between AWS and LDAP
   3. **Use AWS Security Token Service from an identity broker to issue short-lived AWS credentials**. (Refer [Link](https://aws.amazon.com/blogs/aws/aws-identity-and-access-management-now-with-identity-federation/))
   4. Use IAM roles to automatically rotate the IAM credentials when LDAP credentials are updated.
   5. Use the LDAP credentials to restrict a group of users from launching specific EC2 instance types.
3. You are designing a photo sharing mobile app the application will store all pictures in a single Amazon S3 bucket. Users will upload pictures from their mobile device directly to Amazon S3 and will be able to view and download their own pictures directly from Amazon S3. You want to configure security to handle potentially millions of users in the most secure manner possible. What should your server-side application do when a new user registers on the photo-sharing mobile application? **[PROFESSIONAL]**
   1. Create a set of long-term credentials using AWS Security Token Service with appropriate permissions Store these credentials in the mobile app and use them to access Amazon S3.
   2. **Record the user’s Information in Amazon RDS and create a role in IAM with appropriate permissions. When the user uses their mobile app create temporary credentials using the AWS Security Token Service ‘AssumeRole’ function. Store these credentials in the mobile app’s memory and use them to access Amazon S3. Generate new credentials the next time the user runs the mobile app.**
   3. Record the user’s Information in Amazon DynamoDB. When the user uses their mobile app create temporary credentials using AWS Security Token Service with appropriate permissions. Store these credentials in the mobile app’s memory and use them to access Amazon S3 Generate new credentials the next time the user runs the mobile app.
   4. Create IAM user. Assign appropriate permissions to the IAM user Generate an access key and secret key for the IAM user, store them in the mobile app and use these credentials to access Amazon S3.
   5. Create an IAM user. Update the bucket policy with appropriate permissions for the IAM user Generate an access Key and secret Key for the IAM user, store them In the mobile app and use these credentials to access Amazon S3.
4. Your company has recently extended its datacenter into a VPC on AWS to add burst computing capacity as needed Members of your Network Operations Center need to be able to go to the AWS Management Console and administer Amazon EC2 instances as necessary. You don’t want to create new IAM users for each NOC member and make those users sign in again to the AWS Management Console. Which option below will meet the needs for your NOC members? **[PROFESSIONAL]**
   1. Use OAuth 2.0 to retrieve temporary AWS security credentials to enable your NOC members to sign in to the AWS Management Console.
   2. Use Web Identity Federation to retrieve AWS temporary security credentials to enable your NOC members to sign in to the AWS Management Console.
   3. **Use your on-premises SAML 2.O-compliant identity provider (IDP) to grant the NOC members federated access to the AWS Management Console via the AWS single sign-on (SSO) endpoint.**
   4. Use your on-premises SAML 2.0-compliant identity provider (IDP) to retrieve temporary security credentials to enable NOC members to sign in to the AWS Management Console
5. A corporate web application is deployed within an Amazon Virtual Private Cloud (VPC) and is connected to the corporate data center via an iPsec VPN. The application must authenticate against the on-premises LDAP server. After authentication, each logged-in user can only access an Amazon Simple Storage Space (S3) keyspace specific to that user. Which two approaches can satisfy these objectives? (Choose 2 answers) **[PROFESSIONAL]**
   1. Develop an identity broker that authenticates against IAM security Token service to assume a IAM role in order to get temporary AWS security credentials. The application calls the identity broker to get AWS temporary security credentials with access to the appropriate S3 bucket. (Needs to authenticate against LDAP and not IAM)
   2. **The application authenticates against LDAP and retrieves the name of an IAM role associated with the user. The application then calls the IAM Security Token Service to assume that IAM role. The application can use the temporary credentials to access the appropriate S3 bucket.**(Authenticates with LDAP and calls the AssumeRole)
   3. **Develop an identity broker that authenticates against LDAP and then calls IAM Security Token Service to get IAM federated user credentials The application calls the identity broker to get IAM federated user credentials with access to the appropriate S3 bucket.**(Custom Identity broker implementation, with authentication with LDAP and using federated token)
   4. The application authenticates against LDAP the application then calls the AWS identity and Access Management (IAM) Security Token service to log in to IAM using the LDAP credentials the application can use the IAM temporary credentials to access the appropriate S3 bucket. (Can’t login to IAM using LDAP credentials)
   5. The application authenticates against IAM Security Token Service using the LDAP credentials the application uses those temporary AWS security credentials to access the appropriate S3 bucket. (Need to authenticate with LDAP)
6. Company B is launching a new game app for mobile devices. Users will log into the game using their existing social media account to streamline data capture. Company B would like to directly save player data and scoring information from the mobile app to a DynamoDB table named Score Data When a user saves their game the progress data will be stored to the Game state S3 bucket. what is the best approach for storing data to DynamoDB and S3? **[PROFESSIONAL]**
   1. Use an EC2 Instance that is launched with an EC2 role providing access to the Score Data DynamoDB table and the GameState S3 bucket that communicates with the mobile app via web services.
   2. **Use temporary security credentials that assume a role providing access to the Score Data DynamoDB table and the Game State S3 bucket using web identity federation**
   3. Use Login with Amazon allowing users to sign in with an Amazon account providing the mobile app with access to the Score Data DynamoDB table and the Game State S3 bucket.
   4. Use an IAM user with access credentials assigned a role providing access to the Score Data DynamoDB table and the Game State S3 bucket for distribution with the mobile app.
7. A user has created a mobile application which makes calls to DynamoDB to fetch certain data. The application is using the DynamoDB SDK and root account access/secret access key to connect to DynamoDB from mobile. Which of the below mentioned statements is true with respect to the best practice for security in this scenario?
   1. User should create a separate IAM user for each mobile application and provide DynamoDB access with it
   2. User should create an IAM role with DynamoDB and EC2 access. Attach the role with EC2 and route all calls from the mobile through EC2
   3. **The application should use an IAM role with web identity federation which validates calls to DynamoDB with identity providers, such as Google, Amazon, and Facebook**
   4. Create an IAM Role with DynamoDB access and attach it with the mobile application
8. You are managing the AWS account of a big organization. The organization has more than 1000+ employees and they want to provide access to the various services to most of the employees. Which of the below mentioned options is the best possible solution in this case?
   1. The user should create a separate IAM user for each employee and provide access to them as per the policy
   2. The user should create an IAM role and attach STS with the role. The user should attach that role to the EC2 instance and setup AWS authentication on that server
   3. The user should create IAM groups as per the organization’s departments and add each user to the group for better access control
   4. **Attach an IAM role with the organization’s authentication service to authorize each user for various AWS services**
9. Your fortune 500 company has under taken a TCO analysis evaluating the use of Amazon S3 versus acquiring more hardware The outcome was that all employees would be granted access to use Amazon S3 for storage of their personal documents. Which of the following will you need to consider so you can set up a solution that incorporates single sign-on from your corporate AD or LDAP directory and restricts access for each user to a designated user folder in a bucket? (Choose 3 Answers) **[PROFESSIONAL]**
   1. **Setting up a federation proxy or identity provider**
   2. **Using AWS Security Token Service to generate temporary tokens**
   3. Tagging each folder in the bucket
   4. **Configuring IAM role**
   5. Setting up a matching IAM user for every user in your corporate directory that needs access to a folder in the bucket
10. An AWS customer is deploying a web application that is composed of a front-end running on Amazon EC2 and of confidential data that is stored on Amazon S3. The customer security policy that all access operations to this sensitive data must be authenticated and authorized by a centralized access management system that is operated by a separate security team. In addition, the web application team that owns and administers the EC2 web front-end instances is prohibited from having any ability to access the data that circumvents this centralized access management system. Which of the following configurations will support these requirements? **[PROFESSIONAL]**
    1. Encrypt the data on Amazon S3 using a CloudHSM that is operated by the separate security team. Configure the web application to integrate with the CloudHSM for decrypting approved data access operations for trusted end-users. (S3 doesn’t integrate directly with CloudHSM, also there is no centralized access management system control)
    2. **Configure the web application to authenticate end-users against the centralized access management system. Have the web application provision trusted users STS tokens entitling the download of approved data directly from Amazon S3**(Controlled access and admins cannot access the data as it needs authentication)
    3. Have the separate security team create and IAM role that is entitled to access the data on Amazon S3. Have the web application team provision their instances with this role while denying their IAM users access to the data on Amazon S3 (Web team would have access to the data)
    4. Configure the web application to authenticate end-users against the centralized access management system using SAML. Have the end-users authenticate to IAM using their SAML token and download the approved data directly from S3. (not the way SAML auth works and not sure if the centralized access management system is SAML complaint)
11. What is web identity federation?
    1. Use of an identity provider like Google or Facebook to become an AWS IAM User.
    2. **Use of an identity provider like Google or Facebook to exchange for temporary AWS security credentials.**
    3. Use of AWS IAM User tokens to log in as a Google or Facebook user.
    4. Use of AWS STS Tokens to log in as a Google or Facebook user.
12. Games-R-Us is launching a new game app for mobile devices. Users will log into the game using their existing Facebook account and the game will record player data and scoring information directly to a DynamoDB table. What is the most secure approach for signing requests to the DynamoDB API?
    1. Create an IAM user with access credentials that are distributed with the mobile app to sign the requests
    2. Distribute the AWS root account access credentials with the mobile app to sign the requests
    3. **Request temporary security credentials using web identity federation to sign the requests**
    4. Establish cross account access between the mobile app and the DynamoDB table to sign the requests
13. You are building a mobile app for consumers to post cat pictures online. You will be storing the images in AWS S3. You want to run the system very cheaply and simply. Which one of these options allows you to build a photo sharing application without needing to worry about scaling expensive uploads processes, authentication/authorization and so forth?
    1. **Build the application out using AWS Cognito and web identity federation to allow users to log in using Facebook or Google Accounts. Once they are logged in, the secret token passed to that user is used to directly access resources on AWS, like AWS S3.**(Amazon Cognito is a superset of the functionality provided by web identity federation. Refer [link](https://blogs.aws.amazon.com/security/post/Tx3SYCORF5EKRC0/How-Does-Amazon-Cognito-Relate-to-Existing-Web-Identity-Federation))
    2. Use JWT or SAML compliant systems to build authorization policies. Users log in with a username and password, and are given a token they can use indefinitely to make calls against the photo infrastructure.
    3. Use AWS API Gateway with a constantly rotating API Key to allow access from the client-side. Construct a custom build of the SDK and include S3 access in it.
    4. Create an AWS oAuth Service Domain ad grant public signup and access to the domain. During setup, add at least one major social media site as a trusted Identity Provider for users.
14. The Marketing Director in your company asked you to create a mobile app that lets users post sightings of good deeds known as random acts of kindness in 80-character summaries. You decided to write the application in JavaScript so that it would run on the broadest range of phones, browsers, and tablets. Your application should provide access to Amazon DynamoDB to store the good deed summaries. Initial testing of a prototype shows that there aren’t large spikes in usage. Which option provides the most cost-effective and scalable architecture for this application? **[PROFESSIONAL]**
    1. Provide the JavaScript client with temporary credentials from the Security Token Service using a Token Vending Machine (TVM) on an EC2 instance to provide signed credentials mapped to an Amazon Identity and Access Management (IAM) user allowing DynamoDB puts and S3 gets. You serve your mobile application out of an S3 bucket enabled as a web site. Your client updates DynamoDB. (Single EC2 instance not a scalable architecture)
    2. **Register the application with a Web Identity Provider like Amazon, Google, or Facebook, create an IAM role for that provider, and set up permissions for the IAM role to allow S3 gets and DynamoDB puts. You serve your mobile application out of an S3 bucket enabled as a web site. Your client updates DynamoDB.**(Can work with JavaScript SDK, is scalable and cost effective)
    3. Provide the JavaScript client with temporary credentials from the Security Token Service using a Token Vending Machine (TVM) to provide signed credentials mapped to an IAM user allowing DynamoDB puts. You serve your mobile application out of Apache EC2 instances that are load-balanced and autoscaled. Your EC2 instances are configured with an IAM role that allows DynamoDB puts. Your server updates DynamoDB. (Is Scalable but Not cost effective)
    4. Register the JavaScript application with a Web Identity Provider like Amazon, Google, or Facebook, create an IAM role for that provider, and set up permissions for the IAM role to allow DynamoDB puts. You serve your mobile application out of Apache EC2 instances that are load-balanced and autoscaled. Your EC2 instances are configured with an IAM role that allows DynamoDB puts. Your server updates DynamoDB. (Is Scalable but Not cost effective)