

## **Part 3 - Authentication and API Gateway**

We used the API endpoint using HTTPS to get the list of dragons.  
Anyone with that API endpoint will be able to get that same data  
which is probably fine for getting the list of dragons.

However, for adding a dragon via POST/dragons,  
it would probably be a good idea to not make this public at all.

We should **authenticate** and **authorize** the call.

API Gateway supports this.

API Gateway protects the backend.

If someone that isn't authorized to access your application makes a request, your backend, which is behind API Gateway, will never see that request.



What kind of authentication and authorization API Gateway supports?

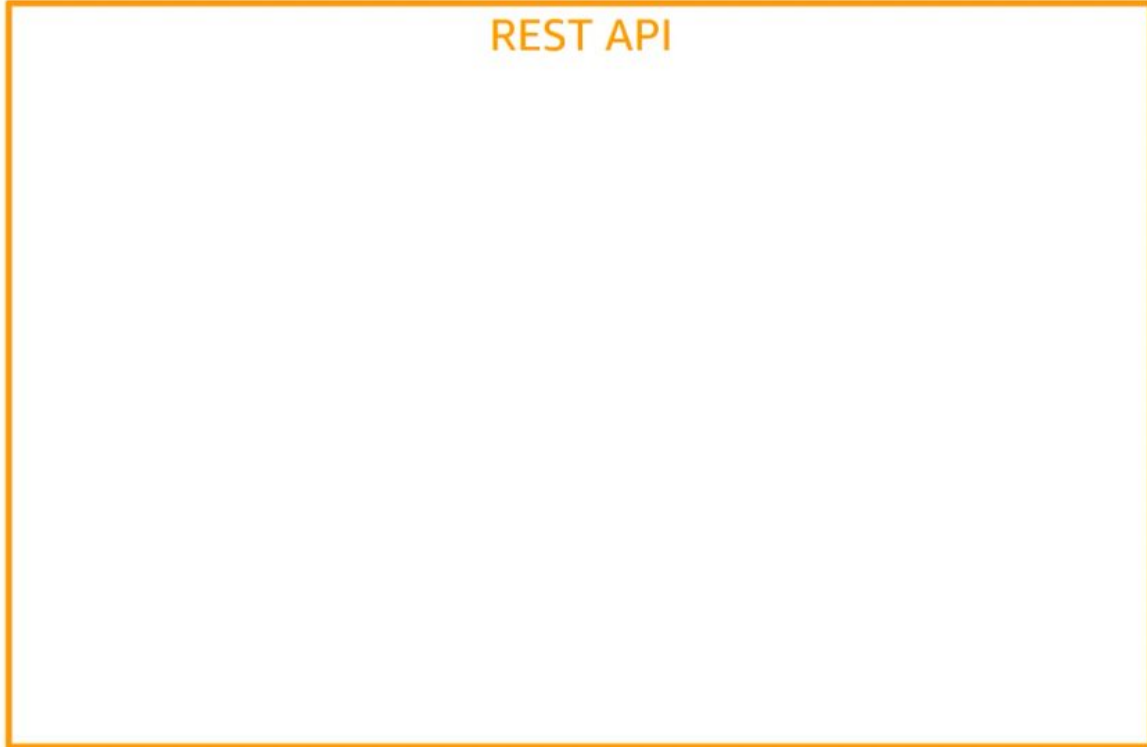
It depends on the type of API you use.



Amazon API  
Gateway



REST API



HTTP API



For the REST API type,  
the mechanism for API Gateway to do authentication and authorization are  
**Identity and Access Management,**  
**API Gateway Lambda authorizers,** and  
**Amazon Cognito user pools.**



Amazon API  
Gateway

## REST API

### AuthN and AuthZ



AWS Identity and  
Access Management



AWS Lambda



Amazon Cognito

## HTTP API

There are a few other mechanisms  
related to **access control** around this type of API:

- API endpoint resource policy: resource policies applied at an API Gateway endpoint
- VPC endpoint policy: endpoint policies for interface VPC endpoint
- CORS: cross-origin resource sharing
- AWS WAF: a web application firewall
- Client-side SSL certificate
- Usage plans API keys



Amazon API  
Gateway

## REST API

### AuthN and AuthZ



AWS Identity and  
Access Management



AWS Lambda



Amazon Cognito

### Access control



API endpoint  
resource policy



AWS WAF



VPC endpoint  
policy



Client-side  
SSL certificate

CORS



Usage plan  
API key

## HTTP API

We will discuss each of these authentication and authorization mechanisms as well as all of those access controls later in this lecture.

For HTTP APIs, the only way to do authentication and authorization is by using **JSON Web Tokens** or **JWT**, that are part of the **OpenID Connect** and **OAuth 2.0** protocols.

What this means?



For HTTP APIs, the only way to do authentication and authorization is by using **JSON Web Tokens** or **JWT**, that are part of the **OpenID Connect** and **OAuth 2.0** protocols.

This means that you can integrate with any third-party identity providers.



Amazon API Gateway

## REST API

### AuthN and AuthZ



AWS Identity and Access Management



AWS Lambda



Amazon Cognito

### Access control



API endpoint resource policy



AWS WAF



VPC endpoint policy



Client-side SSL certificate

CORS



Usage plan API key

## HTTP API

### AuthN and AuthZ



OpenID Connect



JWT



OAuth2.0

Because **Amazon Cognito User Pools** uses JWT as well,  
it means that it's also a supported method of authentication.



Amazon API Gateway

## REST API

### AuthN and AuthZ



AWS Identity and Access Management



AWS Lambda



Amazon Cognito

### Access control



API endpoint resource policy



AWS WAF



VPC endpoint policy



Client-side SSL certificate

CORS



Usage plan API key

## HTTP API

### AuthN and AuthZ



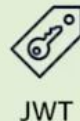
OpenID Connect



OAuth2.0



Amazon Cognito



JWT

To be able to further restrict certain routes to privileged users,  
you can use **authorization scopes**.

For example, you may have GET/dragons  
that you want anyone *authenticated* to be able to hit.

but for a POST/dragon,  
you may only want to allow this action to *specific individuals*.

This is where the **'scope'** parameter in the access token,  
returned by an OAuth 2.0 identity provider, comes into play.

# API Gateway Access Controls



Amazon API Gateway

## REST API

### AuthN and AuthZ



AWS Identity and Access Management



AWS Lambda



Amazon Cognito

### Access control



API endpoint resource policy



AWS WAF



VPC endpoint policy



Client-side SSL certificate

CORS



Usage plan API key

## HTTP API

### AuthN and AuthZ



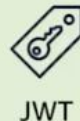
OpenID Connect



OAuth2.0



Amazon Cognito



JWT



You can use access controls on top of IAM, AWS Lambda authorizers, and Amazon Cognito user pools.

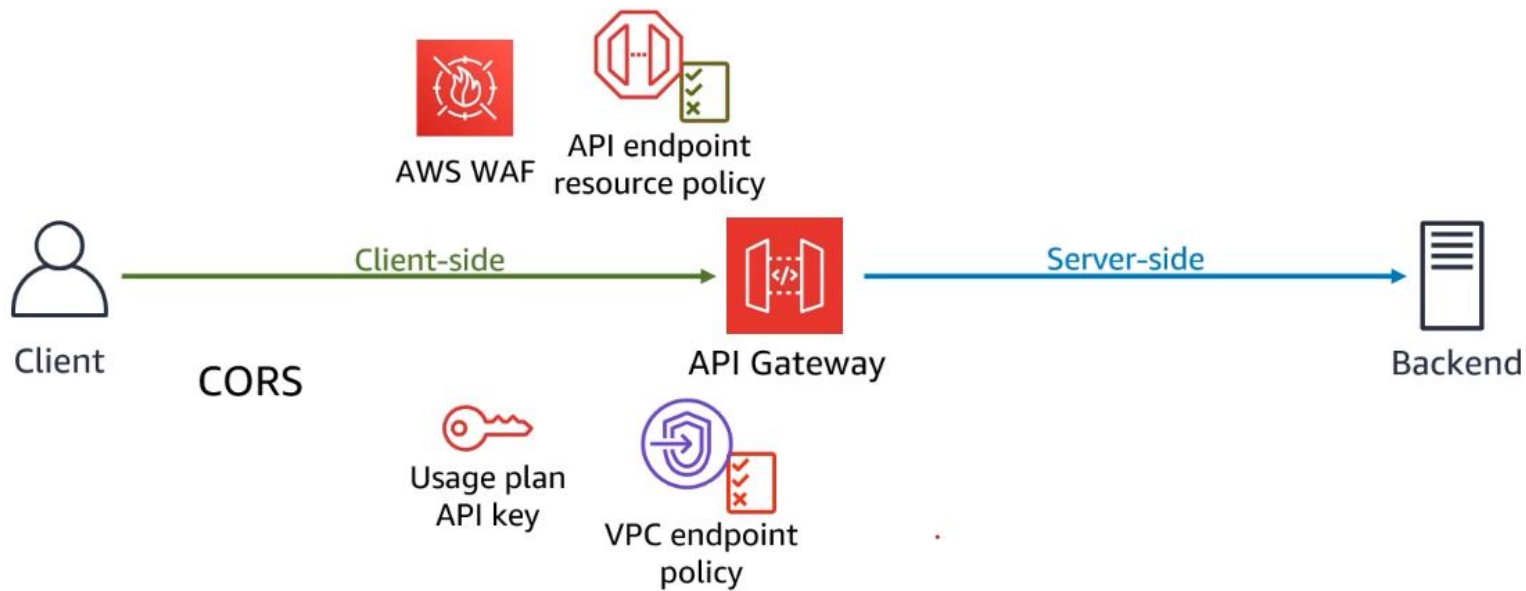
This means that you don't have to choose between Access Control and AuthN/AuthZ.

In fact, you can even combine multiple access controls as well.

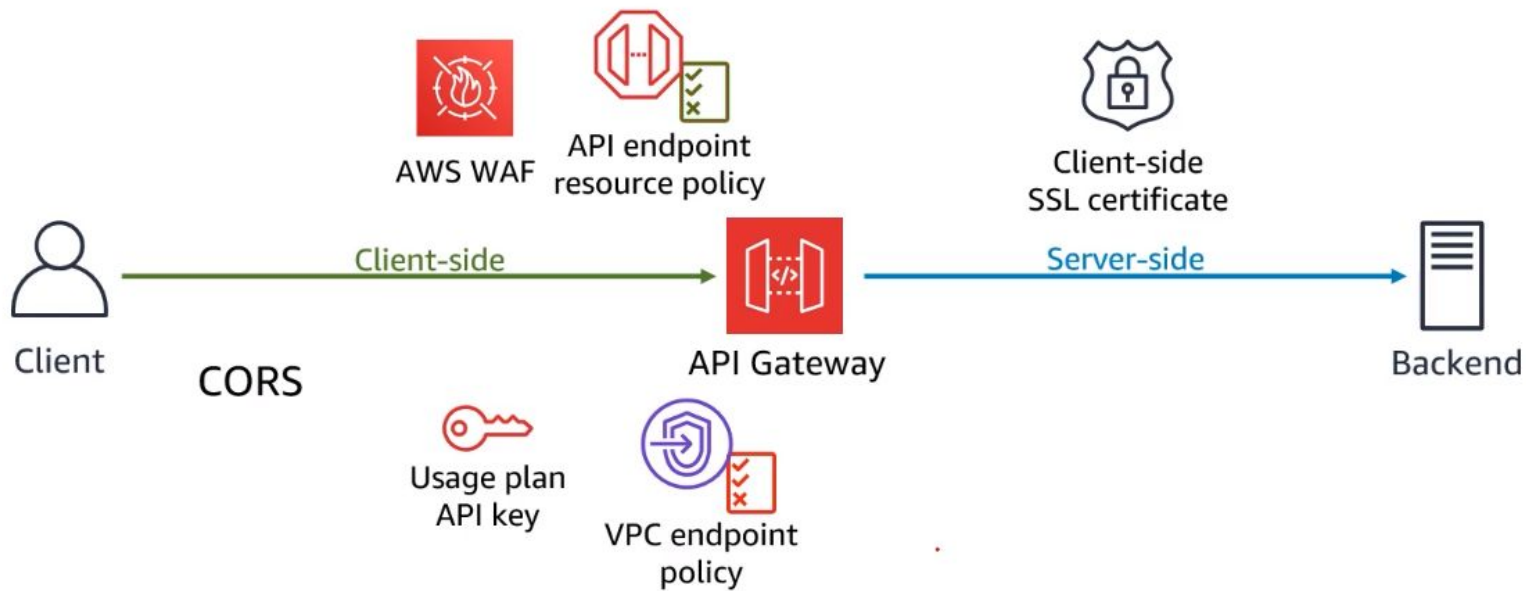
# 1 - Client-side SSL Certificate

All access control mechanisms (except one)  
are client-side where the request from the client comes in.

It is to prevent or allow the access in.



On the server-side, the only mechanism is  
the **Client-side SSL certificate**.



On the server-side, the only mechanism is  
the **Client-side SSL certificate**.

This is a mechanism to authenticate API Gateway itself to your backend.

**2 - CORS**



CORS:

"cross-origin resource sharing"

is a browser security feature that restricts cross-origin HTTP requests.



Let's say that the website of your application is [www.example.com](http://www.example.com)



and your API endpoint is dragons.api.com

```
← → C www.example.com

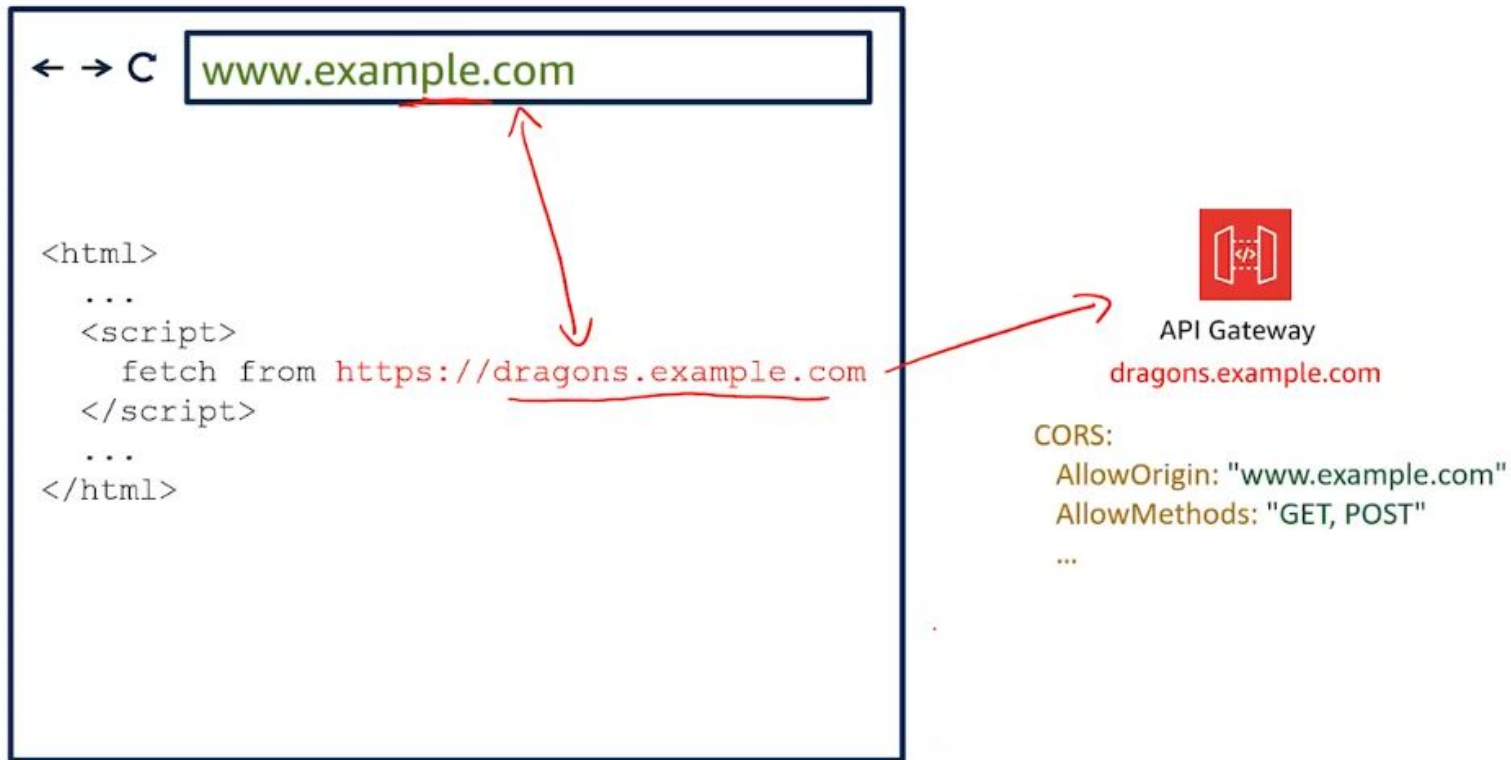
<html>
  ...
  <script>
    fetch from https://dragons.example.com
  </script>
  ...
</html>
```



Then if you have some Javascript that is executing as part of the clients browser, and is going to dragons.example.com to fetch some data from your API, that mean that the script will make a cross-origin HTTP request to dragons.example.com.



What cross-origin means here is that the origin or domain in the URL is different than the origin or domain inside of the script.



only the clients coming from `www.example.com`, as their origin, are all allowed to query for your `dragons.example.com` API.

In most cases, where the browser of a client makes a request to API Gateway, you will need to configure CORS.

The origin of a webpage is determined by its protocol, domain name, and port.

For example, the following URL has  
protocol http, domain name www.example.com, and port 80.

<http://www.example.com/index.html>



- <http://wikipedia.org/a/> and <http://wikipedia.org/b/> have the same origin
- <http://wikipedia.org> and <http://www.wikipedia.org> **do not** have the same origin
- <http://wikipedia.org> and <https://wikipedia.org> **do not** have the same origin
- <http://wikipedia.org:81> and <http://wikipedia.org:82> **do not** have the same origin
- [www.example.com](http://www.example.com) and [dragon.example.com](http://dragon.example.com) ?

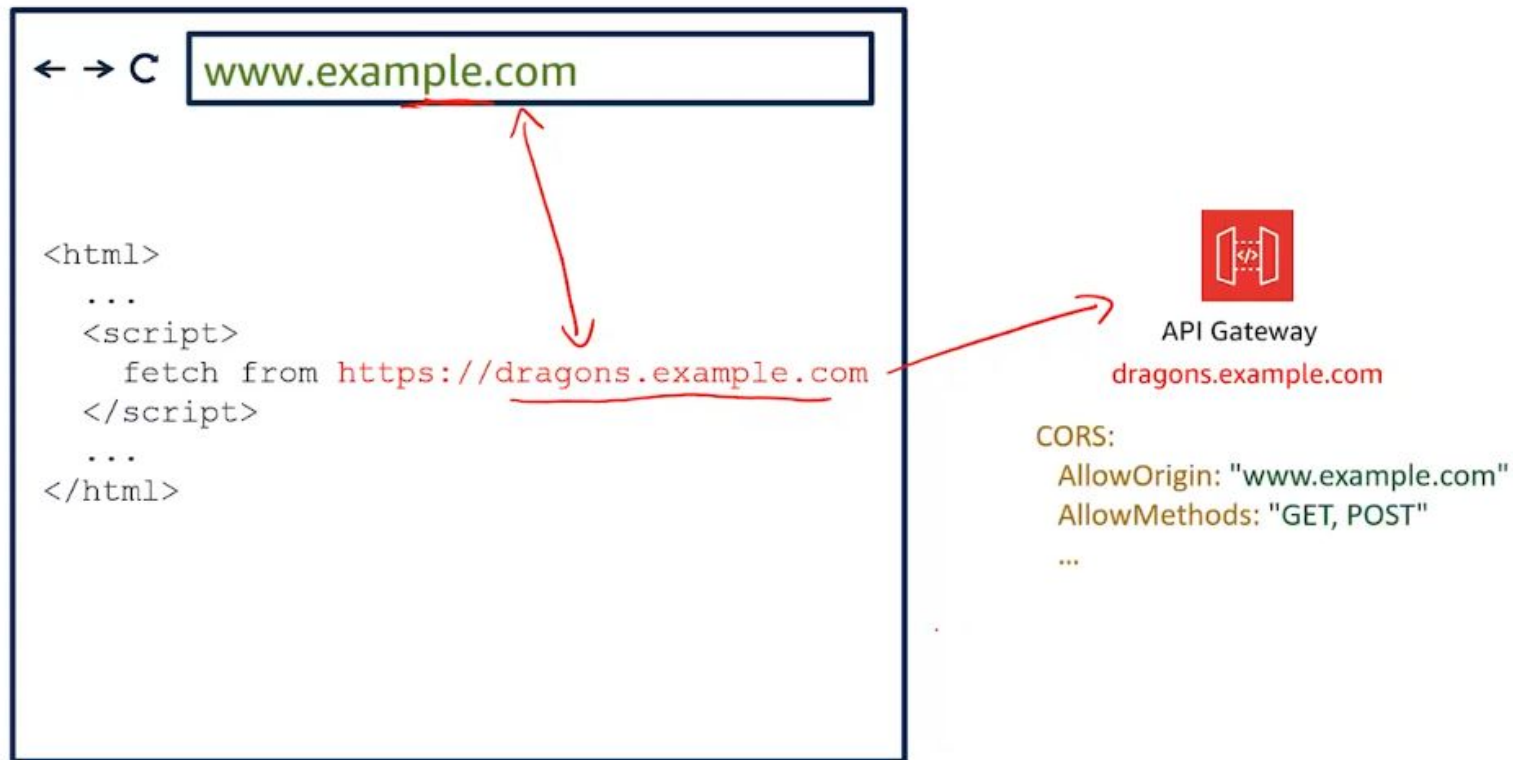
- `http://wikipedia.org/a/` and `http://wikipedia.org/b/` have the same origin
- `http://wikipedia.org` and `http://www.wikipedia.org` **do not** have the same origin
- `http://wikipedia.org` and `https://wikipedia.org` **do not** have the same origin
- `http://wikipedia.org:81` and `http://wikipedia.org:82` **do not** have the same origin
- `www.example.com` and `dragon.example.com` **do not** have the same origin.

If you configure Cross-Origin Resource Sharing (CORS) on your server, you can allow resources on your server to be accessed by web pages from different origins.

**CORS** is enforced by the **browser** and not the server.

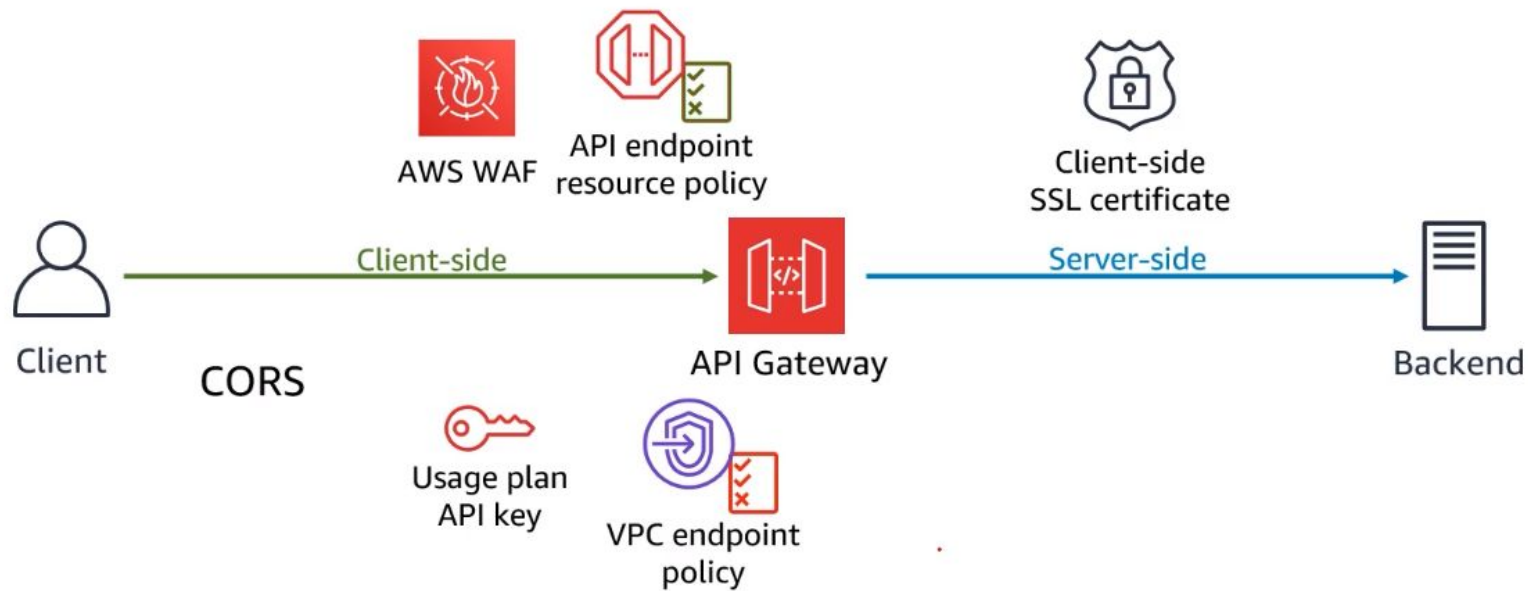
When a web page makes a request to a different domain, the browser sends an HTTP request with an **Origin header** indicating the source domain.

The server can then respond with appropriate **CORS headers** to indicate whether or not the request is allowed.



recap

## 3 - AWS WAF



Recap

**AWS WAF** is a web application firewall  
that can be used to protect your API from common web exploits.



You can use the managed rules for WAF to address issues like **OWASP top 10 security risks** which are regularly updated as new issues emerge.

2017

2021

A01:2017-Injection

A02:2017-Broken Authentication

A03:2017-Sensitive Data Exposure

A04:2017-XML External Entities (XXE)

A05:2017-Broken Access Control

A06:2017-Security Misconfiguration

A07:2017-Cross-Site Scripting (XSS)

A08:2017-Insecure Deserialization

A09:2017-Using Components with Known Vulnerabilities

A10:2017-Insufficient Logging & Monitoring

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

(New) A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

(New) A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures\*

(New) A10:2021-Server-Side Request Forgery (SSRF)\*

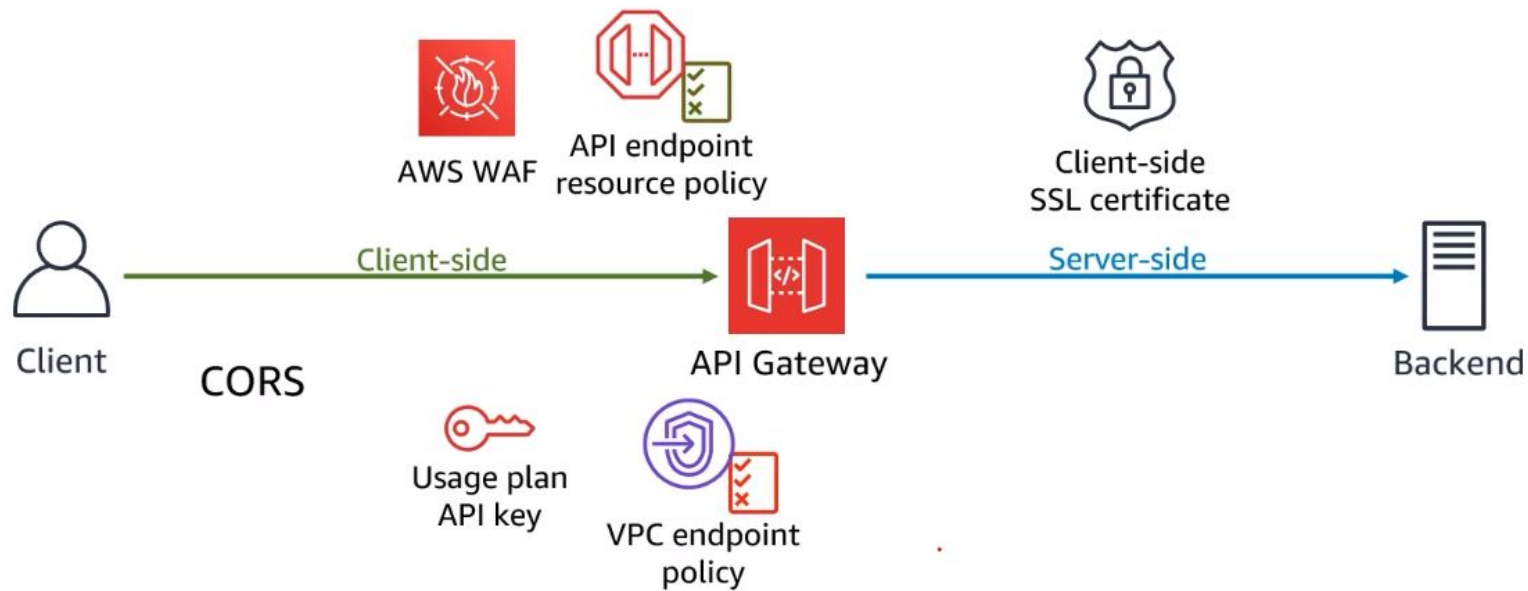
\* From the Survey

## OWASP Top Ten

It can also be configured to filter traffic based on **rules** that you define.

For example, you can filter any part of the web request, such as IP addresses, HTTP headers, HTTP body, or URI strings.

## 4 - Usage Plan API Key



Recap

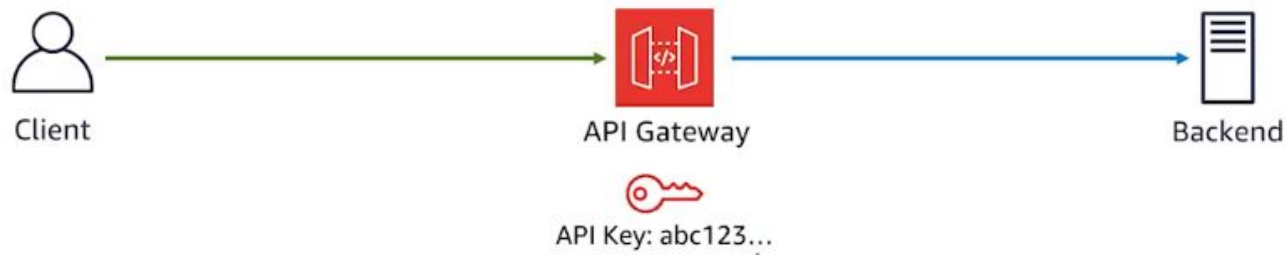
**Usage Plans** is related to limiting the access to authorize clients.

To use it, you first need to create an **API key** in API Gateway.

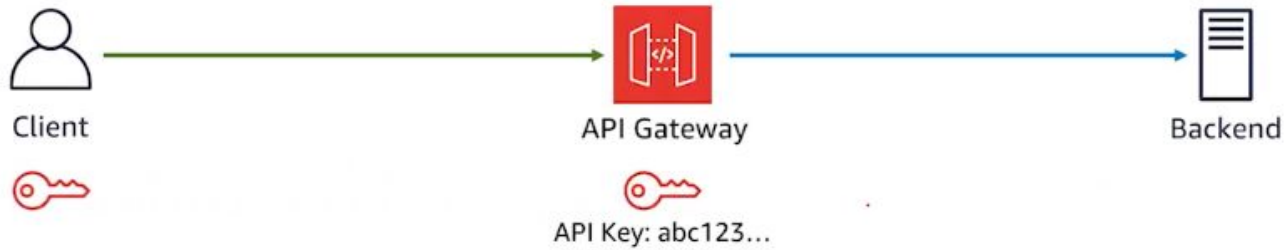


Let's say you have a API Gateway

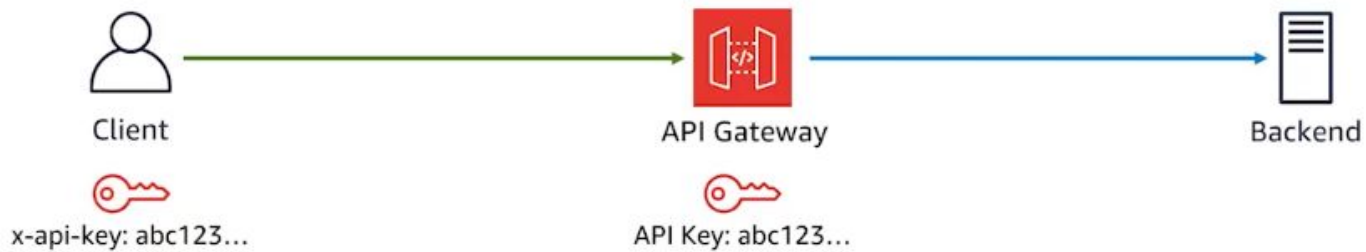




To use Usage Plans, you first need to create an API key in API Gateway.



You then must send this key to a client. This key is a string made up of alphanumeric characters.



When a client wants to send a request, it inserts this API key as the value of the HTTP header `x-api-key`.

Not a good idea for authentication.

HTTPS request header is in clear text.

This mechanism is if you want to apply some **throttling** or quotas to your REST API per client.

For example, a quota of 100 requests per month and a throttling of 10 requests per second.



Client



x-api-key: abc123...



API Gateway



API Key: abc123...

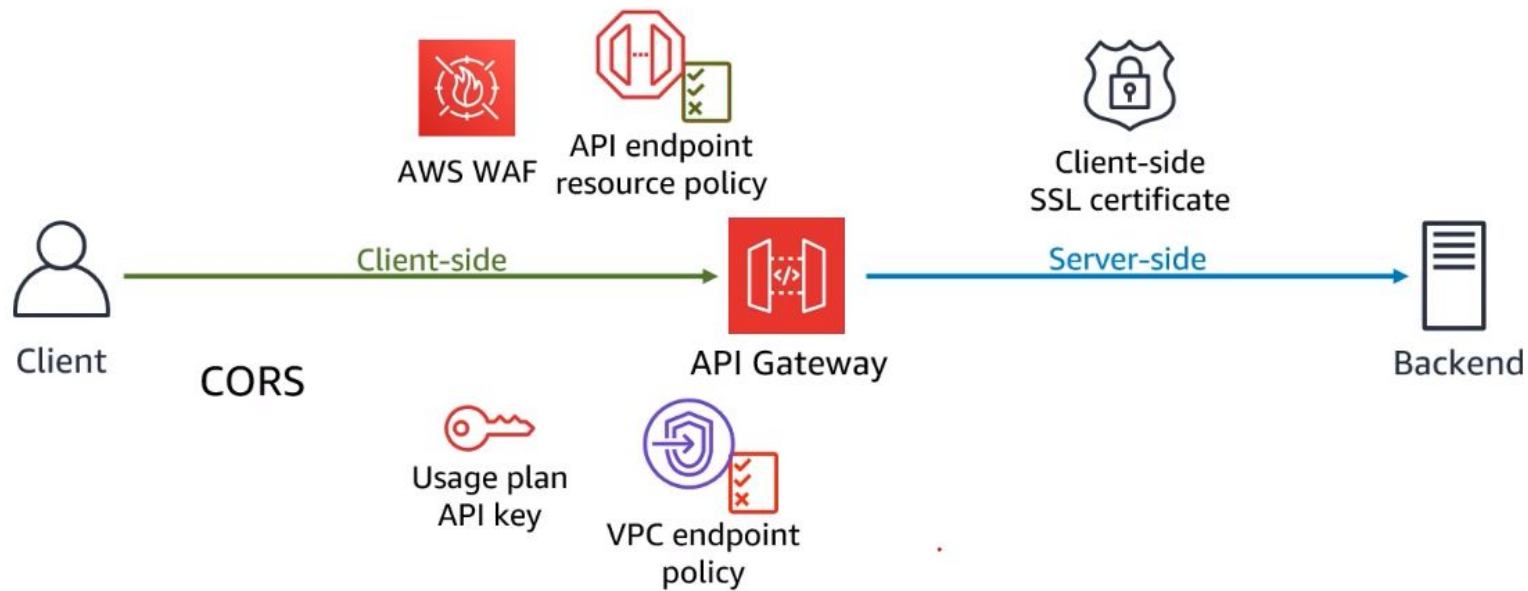


Backend

Usage Plan:

- Quota: 100 requests/month
- Throttling: 10 requests/second

## 5 - API Endpoint Resource Policy



Recap



A few AWS services have **Resource Policy** capability,  
like API and S3.

It allows you to create **resource-based policies**  
to allow or deny access to your APIs and methods, using **IAM conditions elements**.

**IAM conditions elements** includes users from a specific AWS account,  
a specific source IP address range,  
or a specific Virtual Private Cloud or VPC endpoint.

Let's see an example

Amazon API Gateway

APIs > dragons (nl18eym14c) > Resources > / (ac2lnz384m)

APIs

Custom Domain Names

VPC Links

API: dragons

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Dashboard

Settings

Resources

Actions

/

/dragons

GET

POST

/ Methods

No methods defined for the resource.

Feedback English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Priva

Let's look at an example in the API Gateway console on the dragons API.



APIs

Custom Domain Names

VPC Links

API: dragons

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Dashboard

Settings

## Resource Policy

Configure access control to this private API using a Resource Policy. Access can be controlled by IAM condition elements, including conc Source VPC, VPC Endpoints (Private API), and/or IP range. If the Principal in the policy is set to \*, other authorization types can be used a policy. If the Principal is set to AWS, then authorization will fail for all resources not secured with AWS\_IAM auth, including unsecured res

```
1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Principal": "*",
7        "Action": "execute-api:Invoke",
8        "Resource": "arn:aws:execute-api:us-east-1:XXXXXXXXXXXX:nl18eym14c/test/GET/dragons"
9      },
10     {
11       "Effect": "Allow",
12       "Principal": "*",
13       "Action": "execute-api:Invoke",
14       "Resource": "arn:aws:execute-api:us-east-1:XXXXXXXXXXXX:nl18eym14c/test/POST/dragons",
15       "Condition": {
16         "IpAddress": {
17           "aws:SourceIp": "3.3.3.3/32"
18         }
19       }
20     }
21   ]
22 }
```

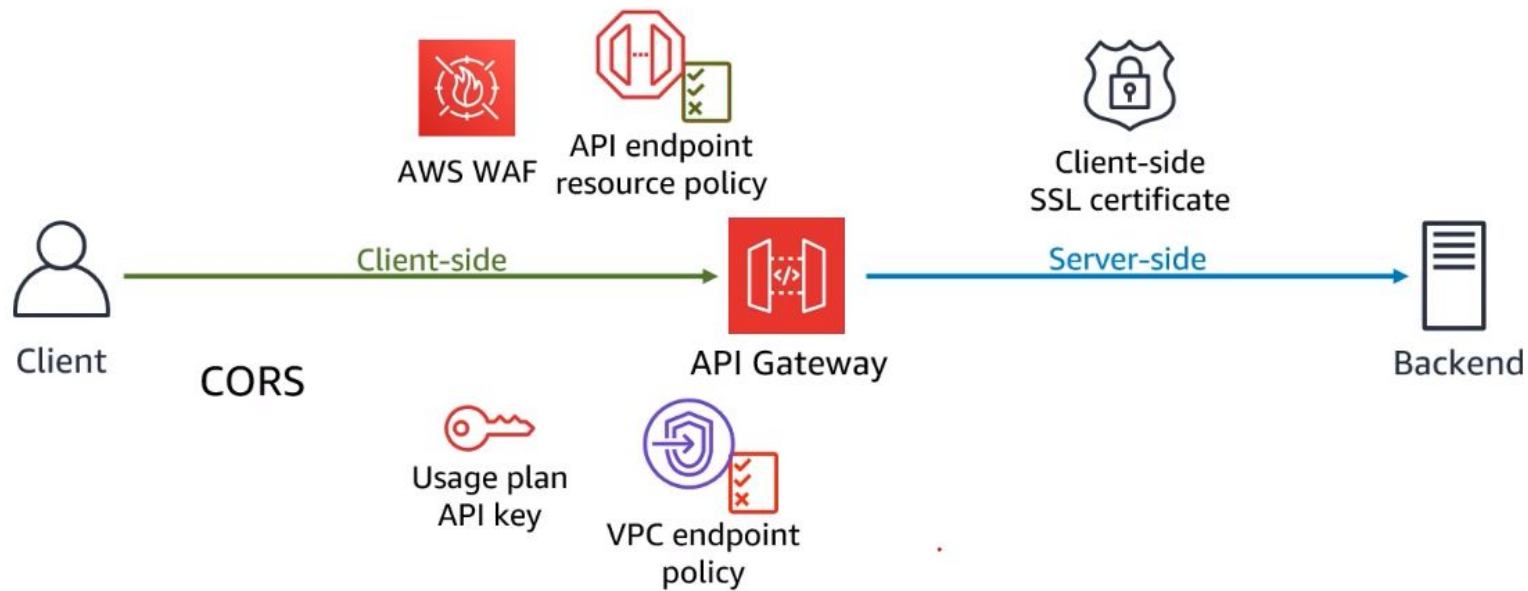
allowing everyone to invoke the GET/dragons.

allowing everyone to invoke POST/dragons but only if you come from this IP address.

The resource policy access control mechanism is great for cross AWS account access and applying more restrictions to the other authorizers.

It's also great for limiting access only to certain ranges of IP addresses.

## 6 - VPC Endpoint Policy



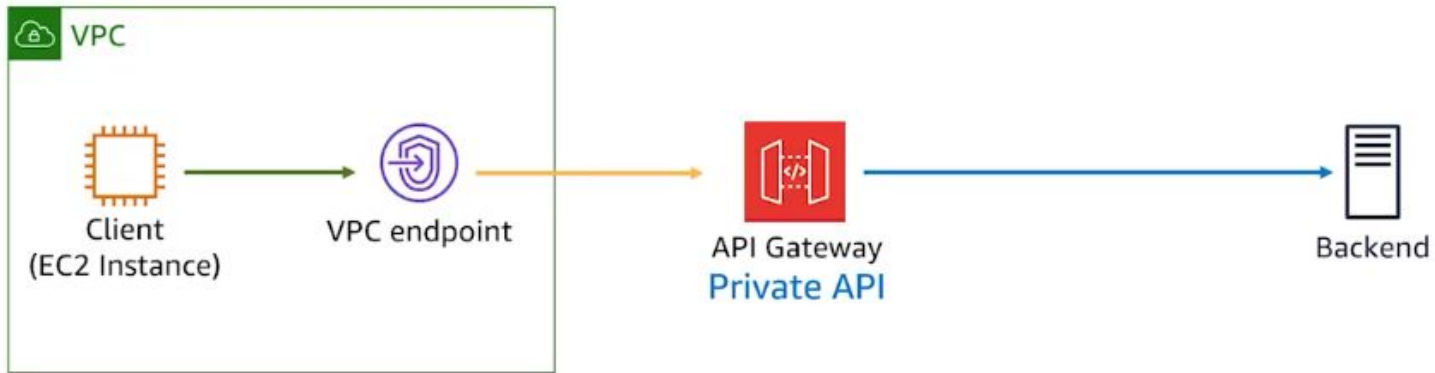
Recap



You can create a **private API** that is only reachable from inside of your Virtual Private Cloud or VPC.

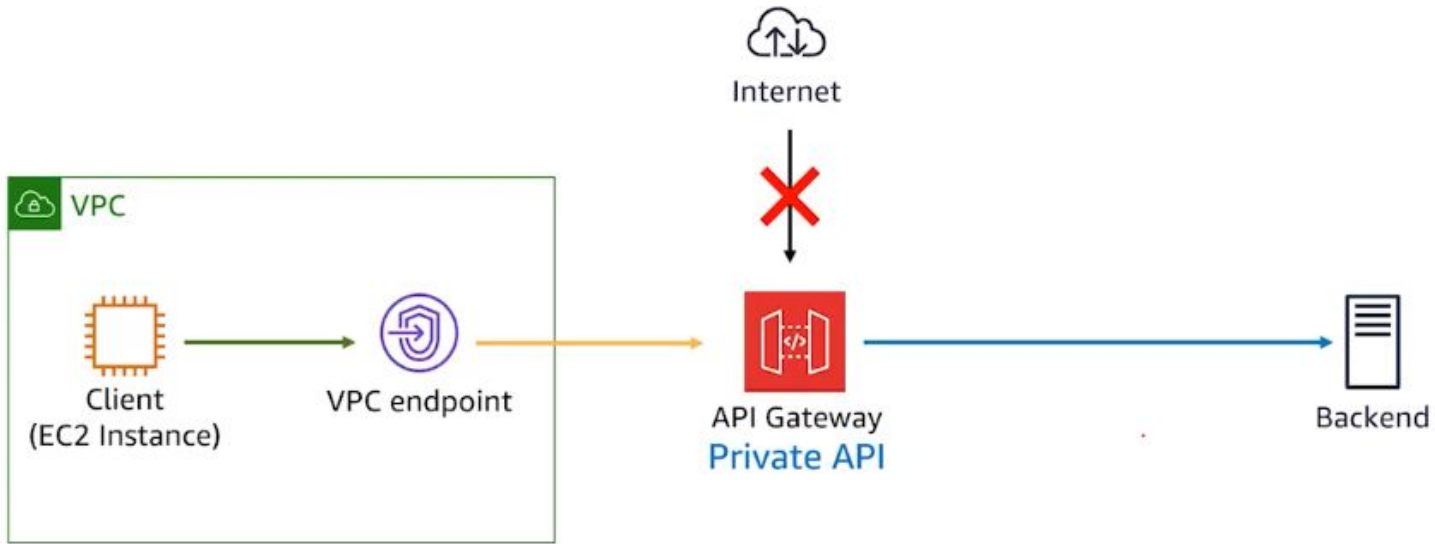


Let say you have a private API

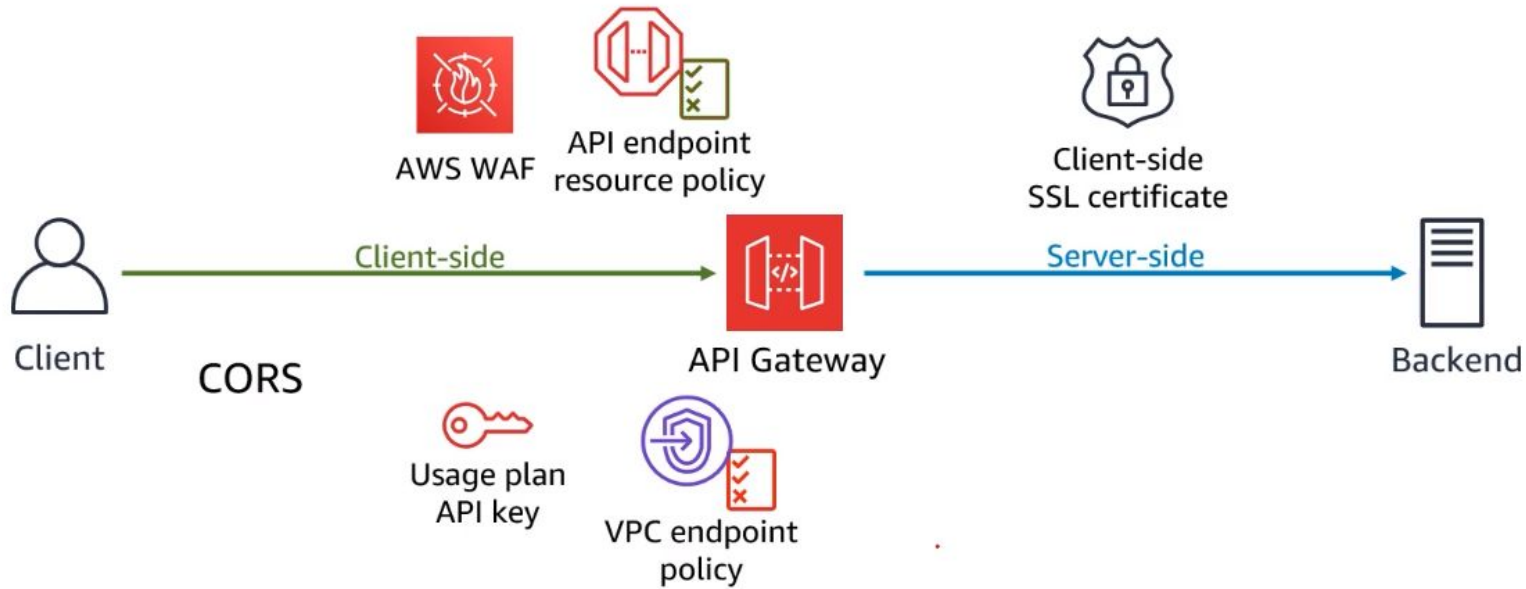


If you want to access this private API from an EC2 instance inside the VPC, you must use a VPC interface endpoint.

Interface endpoints are powered by **AWS PrivateLink** technology that enables you to privately access AWS services by using private IP addresses.



Keeps the traffic entirely within AWS and not exposing your API to the internet at all.



We covered six different mechanism for controlling and managing access to a REST API in API Gateway.

# Authentication and Authorization

**Authentication:** process of using supporting evidence to corroborate an asserted identity

**Identification** (recognition): establish identity from available information (without assertion)

**Authorization:** determining if a request should be granted based on an entity



## User Authentication

- something you know
  - e.g., password, pin code
- something you have
  - e.g., hardware token, bank card
- something you are
  - i.e., biometrics
  - e.g., fingerprint, iris



Amazon API Gateway

## REST API

### AuthN and AuthZ



AWS Identity and Access Management



AWS Lambda



Amazon Cognito

### Access control



API endpoint resource policy



AWS WAF



VPC endpoint policy



Client-side SSL certificate

CORS



Usage plan API key

## HTTP API

### AuthN and AuthZ



OpenID Connect



OAuth2.0



Amazon Cognito



JWT

aws Services Resource Groups buildingmodernapps @ 3022-1... N. Virginia Support

Amazon API Gateway APIs > dragons (nl18eym14c) > Resources > / (ac2lnz384m) Show all hints ?

APIs Resources Actions / Methods

Custom Domain Names

VPC Links

API: dragons

- Resources
- Stages
- Authorizers
- Gateway Responses
- Models
- Resource Policy
- Documentation
- Dashboard
- Settings

/

- /dragons
  - GET
  - POST

No methods defined for the resource.

Feedback English (US) © 2008–2020 Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

In the "Resources" section, choose the specific execution method you wish to associate with an authorization.

This means you can pick an authorization type,  
that's different per method of execution.

For example, you could keep GET/dragons, without authorization for public.

Then add authorizations for POST/dragons.

aws Services Resource Groups buildingmodernapps @ 3022-1... N. Virginia Support

Amazon API Gateway APIs > dragons (nl18eym14c) > Resources > /dragons (o5zw4d) > POST Show all hints ?

APIs Resources Actions

Custom Domain Names / /dragons GET POST

VPC Links

API: dragons Resources Stages Authorizers Gateway Responses Models Resource Policy Documentation Dashboard Settings

### /dragons - POST - Method Execution

```
graph LR; Client[Client] --> MR[Method Request]; MR --> IR[Integration Request]; IR --> ME[Mock Endpoint]; ME --> IRes[Integration Response]; IRes --> MRes[Method Response]; MRes --> Client;
```

**Method Request**  
Auth: AWS\_IAM  
ARN: arn:aws:execute-api:us-east-1: :nl18eym14c/\*/P

**Integration Request**  
Type: MOCK

**Method Response**  
HTTP Status: 200  
Models: application/json => Empty

**Integration Response**  
HTTP status pattern: -  
Output passthrough: Yes

TEST ⚡ Client Mock Endpoint

Feedback English (US) © 2008–2020 Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Since authorization is the first step, that API Gateway shall process when receiving a request, the authorizer option is in the Method Request.

The screenshot shows the AWS API Gateway console interface. The breadcrumb navigation at the top indicates the path: **APIs** > **dragons (nl18eym14c)** > **Resources** > **/dragons (o5zw4d)** > **POST**. The left-hand navigation pane shows the API **dragons** and its **Resources** section, with **POST** selected under the **/dragons** resource. The main content area is titled **Method Execution /dragons - POST - Method Request**. Below the title, there is a description: "Provide information about this method's authorization settings and the parameters it can receive." The **Settings** section contains three configuration items: **Authorization** (set to **AWS\_IAM**), **Request Validator** (set to **NONE**), and **API Key Required** (set to **false**). A dropdown menu is open over the **Authorization** field, showing the following options: **AWS\_IAM** (selected), **NONE**, and **API Key Required**. Below the settings, there are expandable sections for **URL Query String Parameters**, **HTTP Request Headers**, **Request Body**, and **SDK Settings**. The footer of the console shows the year **© 2008 - 2020 Amazon Web Services, Inc. or its affiliates. All rights reserved.**

You can select the appropriate authorization.  
In this case, only AWS IAM is available.

In this case, only **AWS IAM** is available,  
as we haven't created any **Lambda**, or **Cognito User Pool** authorizers yet.

**AWS IAM** adds the same authentication and authorization on top of your API, that is available in front of any AWS service.

This is done via an **access key** and a **secret key**, using the **Signature Version 4 protocol**.



**AWS IAM** is typically used for **server to server** communication.

For example, if your code was running in a container in Elastic Container Service or, on an EC2 instance, and it was making a call to your API, then you should authenticate and authorize it via IAM.

The same way, as if your code was to make a call, to a service of AWS like S3.

AWS IAM is for server to server communication.

Your code running in AWS,  
speaking to API Gateway,  
and communicating with your server backend.

To use the other two authorizers,  
you will need to configure the service itself first.

For example, you will configure a new Cognito User Pool first,  
then, you will configure an authorizer in API Gateway.

[APIs](#)[Custom Domain Names](#)[VPC Links](#)**API: dragons**[Resources](#)[Stages](#)**[Authorizers](#)**[Gateway Responses](#)[Models](#)[Resource Policy](#)[Documentation](#)[Dashboard](#)[Settings](#)

## Authorizers

Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.

[+ Create New Authorizer](#)

Click on Create New Authorizer

aws Services Resource Groups buildingmodernapps @ 3022-1... N. Virginia Support

Amazon API Gateway APIs > dragons (nl18eym14c) > Authorizers Show all hints ?

APIs  
Custom Domain Names  
VPC Links

API: **dragons**  
Resources  
Stages  
**Authorizers**  
Gateway Responses  
Models  
Resource Policy  
Documentation  
Dashboard  
Settings

## Authorizers

Authorizers enable you to control access to your APIs using Amazon Cognito User Pools or a Lambda function.

[+ Create New Authorizer](#)

### Create Authorizer

**Name \***

**Type \* ⓘ**

Lambda  Cognito

**Lambda Function \* ⓘ**

us-east-1

**Lambda Invoke Role ⓘ**

**Lambda Event Payload \* ⓘ**

Token  Request

Feedback English (US) © 2008–2020 Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Here you can see the other two authorizers: Lambda and Cognito.

We will go in details on how Cognito works in the next lecture.

You will use it for a web or a mobile application, to authenticate to API Gateway.

Servers making calls to API Gateway using IAM as the authorizer.

Web and mobile applications use Cognito User Pool to authorize.

What could we possibly need now?

Customization and Flexibility.



Lambda provides the greatest flexibility.

But it comes with a price:  
you have to code it.

What Lambda authorizer allows you to do,  
is call a piece of code that is executed whenever a request is made.

This means, you can integrate with any authentication tools that you want,  
as you are responsible for coding the authorizer.

We will discuss Lambda next lecture.

Let's see how this authorizer actually works.



Client



API Gateway



Backend

a client that wants to go through API Gateway, to access a backend.



AWS Lambda  
authorization code



Client



API Gateway



Backend

Assume we have created a authorization code using AWS Lambda.



AWS Lambda  
authorization code



Client

Request with params

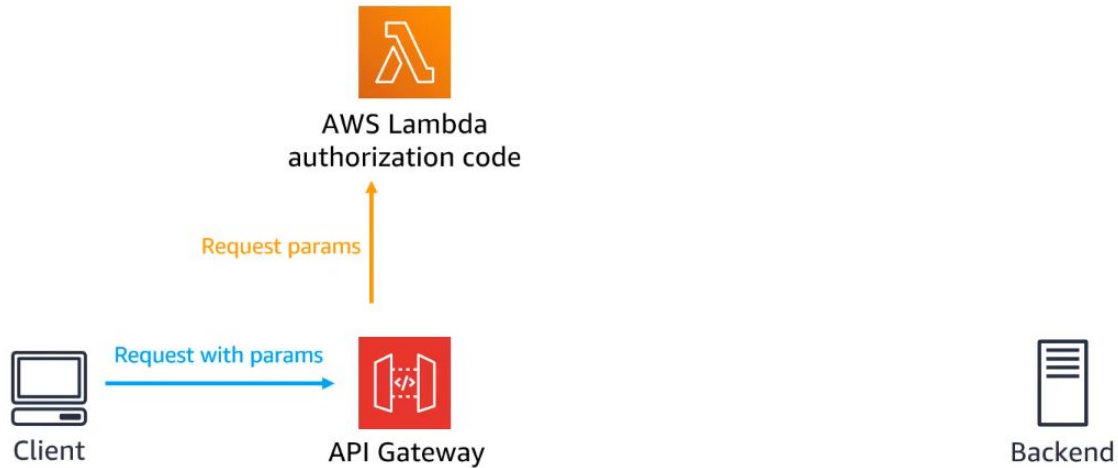


API Gateway



Backend

First, the client will send a request with some parameters. Those parameters could be a token, username password, or anything that is required to authenticate them.

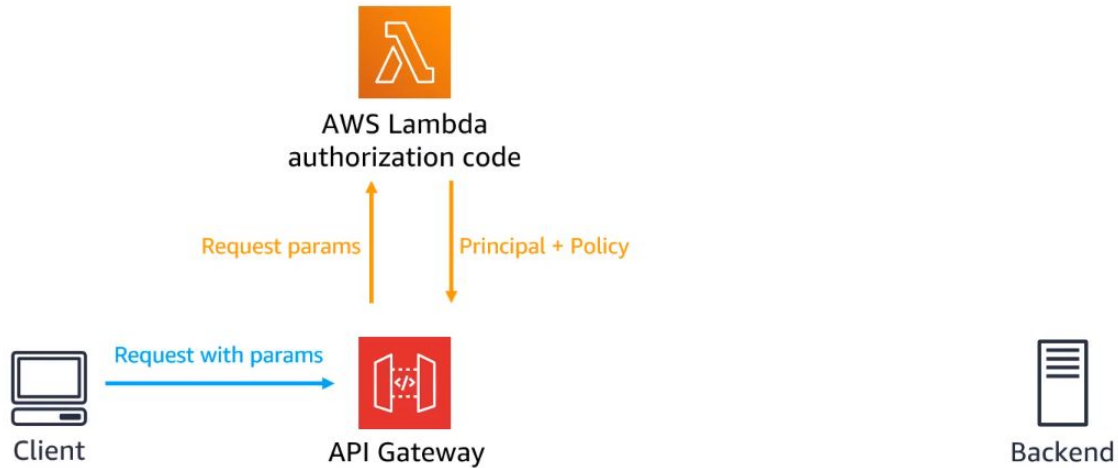


These requests parameters will then be sent the authorization code in Lambda.



That code can do whatever it wants, using those authentication parameters.

Maybe it could call an OAuth 2.0, or SAML identity provider,  
or maybe it will call LDAP or Active Directory.

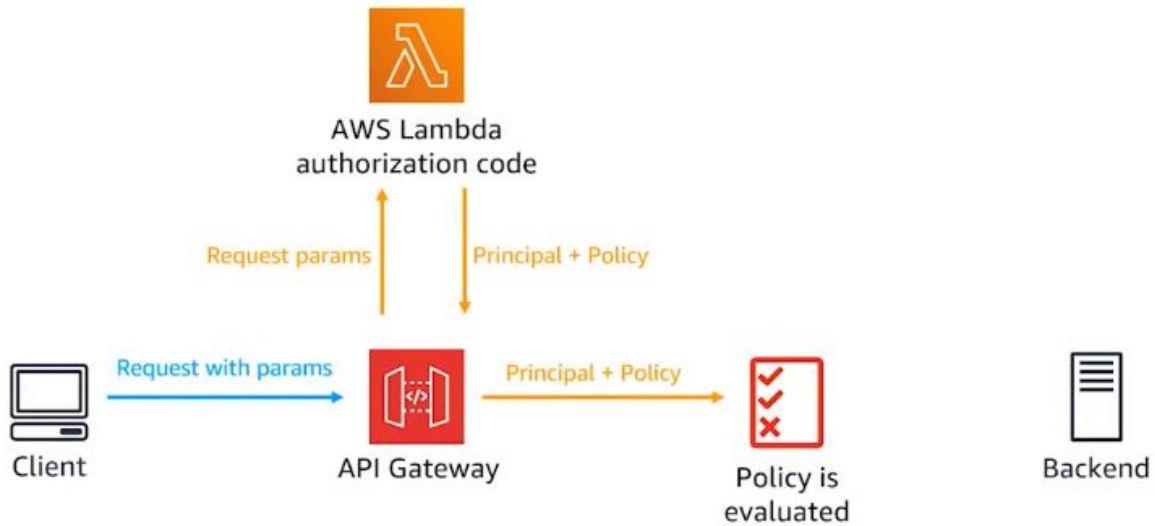


Then, that code needs to return a principal and a policy.

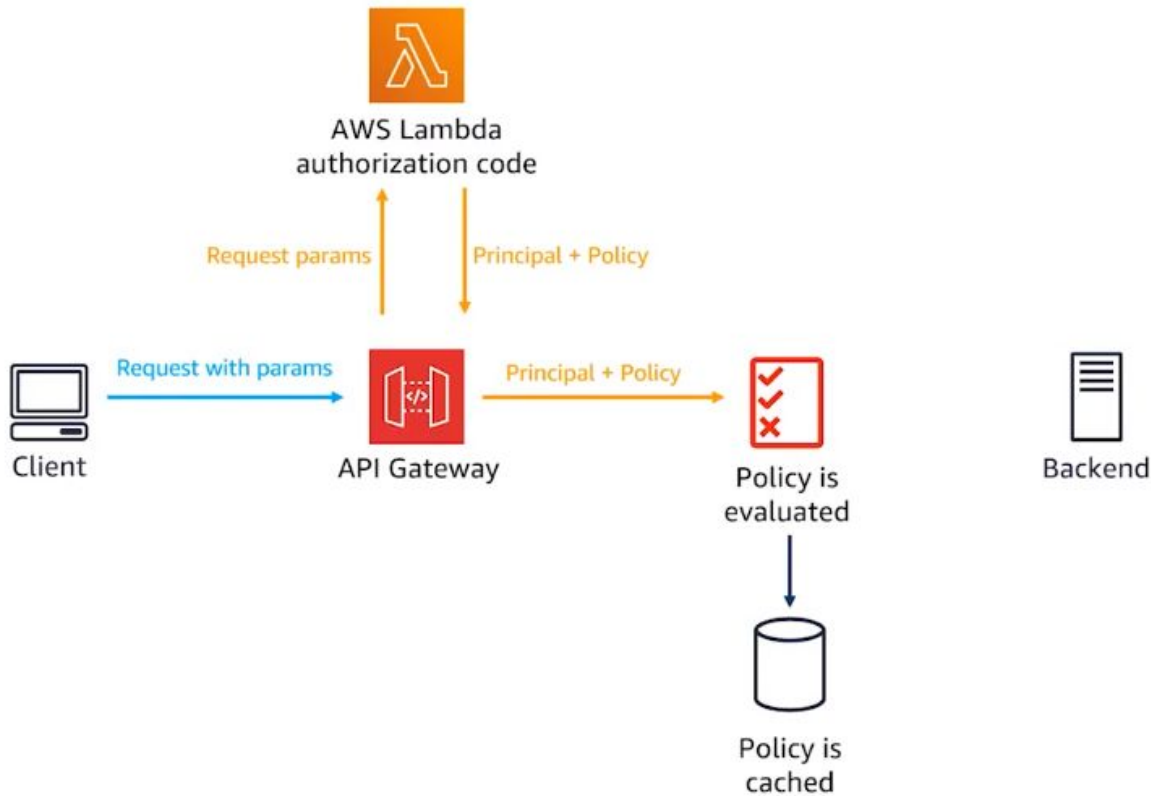
The **principal** is the definition of this user.

Maybe it's a username, or user ID, or anything that identifies this client uniquely.

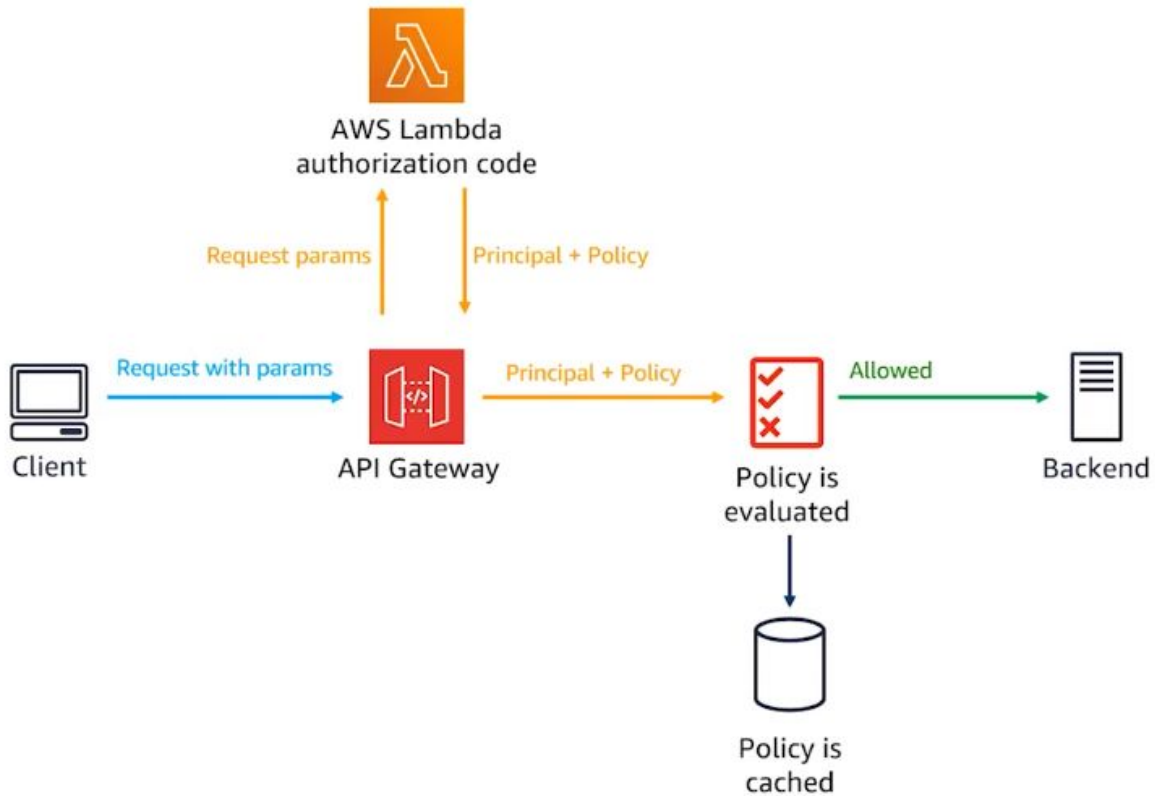
The policy, is like an IAM policy it defines what the user is allowed to do, and not.



This policy is going to be evaluated.



and it will be cached, so that the subsequent request, don't need to go through the authorization code again.



IAM,

Lambda,

and Cognito,

are your three available options for  
authentication and authorization for REST APIs.

**Amazon Cognito**



**Amazon Cognito** is a **identity management** service provided by AWS.

Cognito provides features like user sign-up, sign-in, and access control, making it suitable for web and mobile app development.

It supports various authentication methods including username/password, social identity providers, and multifactor authentication.

Amazon Cognito service is made up of two services:  
**User Pool** and **Federated Identities** (or identity pool).

Let's start with **Cognito User Pools**.

This service is a user directory,  
which you can also refer to as an identity provider.

It allows you to sign up users with email or phone number validation,  
sign them in, help them reset their password,  
and even ask them to do multi-factor authentication.

User pool also has an option for a hosted user interface.

Instead of having to create your own web page for asking your users for their username and password, AWS already created web pages for this workflow, which you can customize and AWS will host it for you.



Cognito User pool



Client

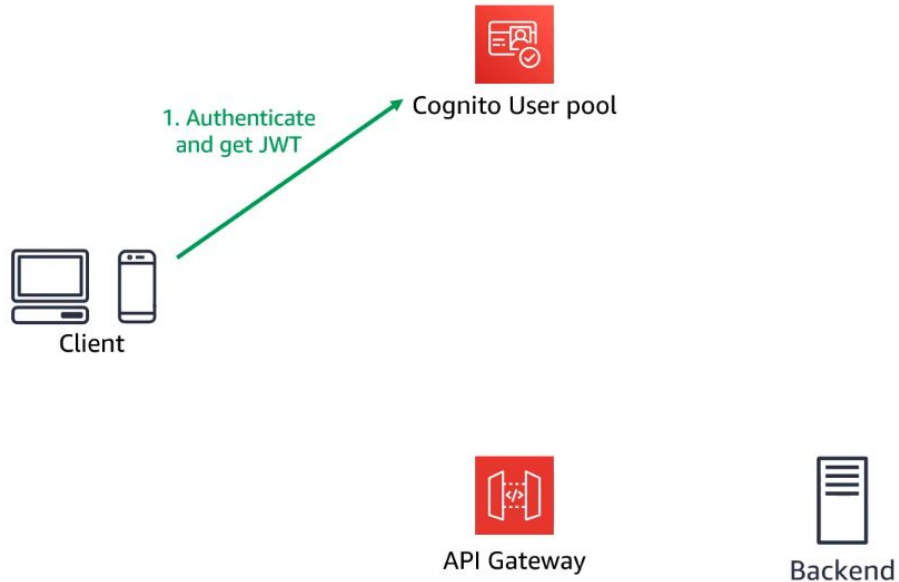


API Gateway

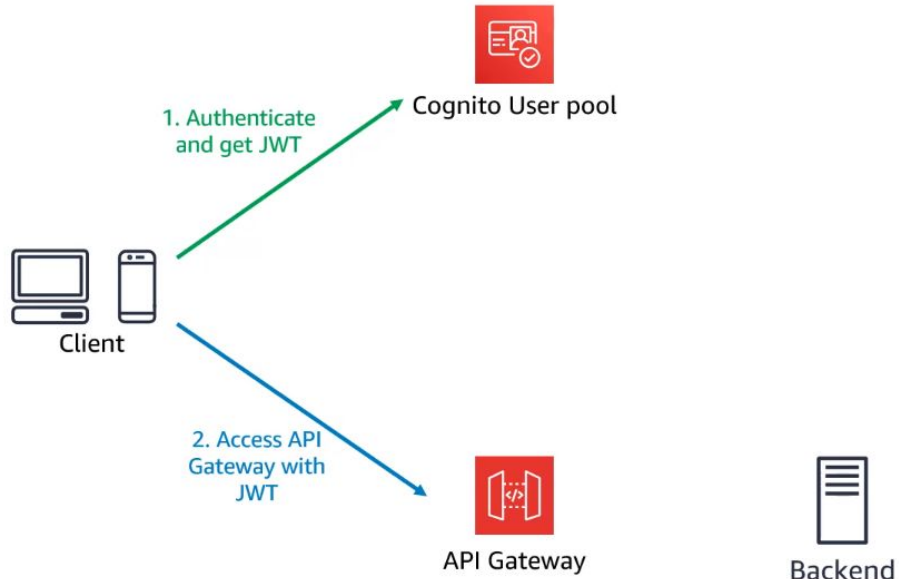


Backend

The first step is that the client needs to authenticate to Amazon Cognito user pool.

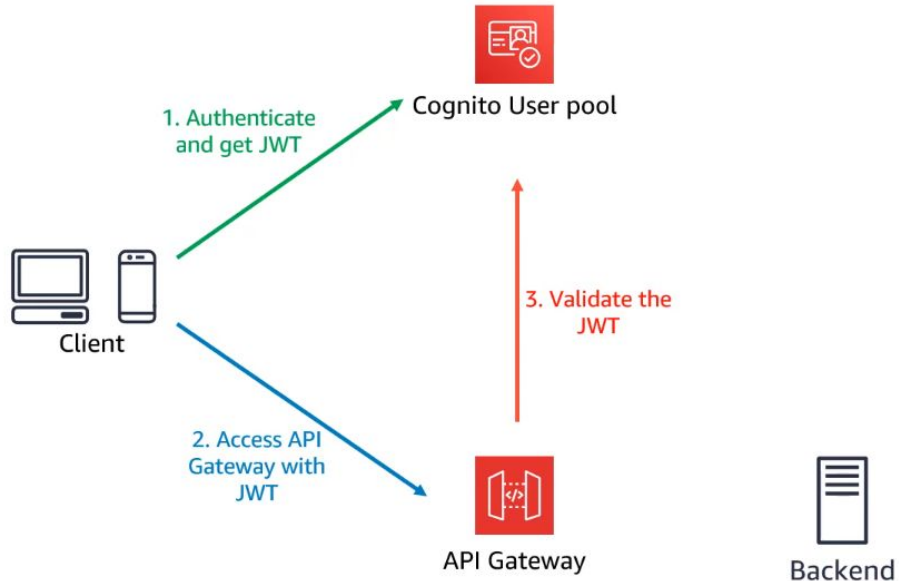


Once authenticated, user pool, we'll send back an OAuth 2.0 protocol JSON Web Token or JWT.

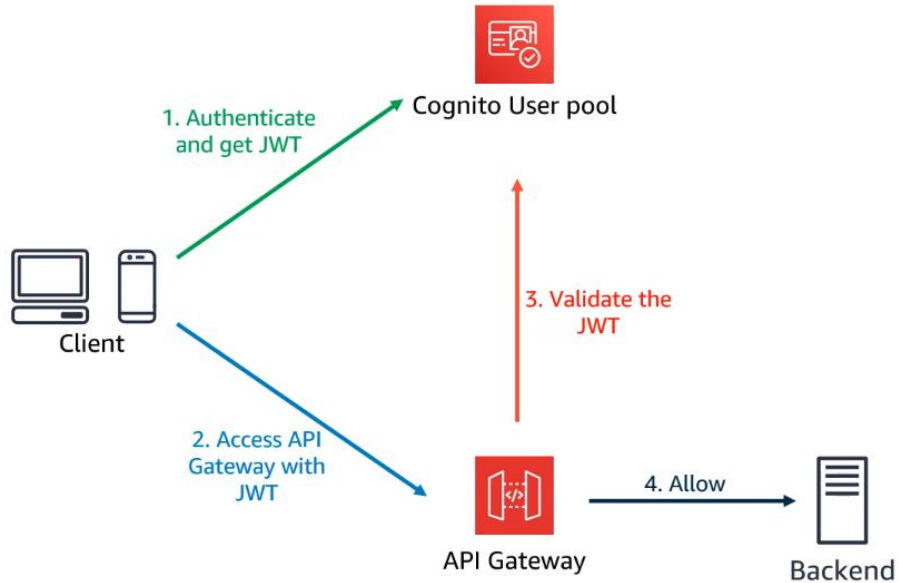


The client can then send a request to API Gateway using that JWT token.





API Gateway will validate that token with the user pool.



and if everything checks out, it will allow the request to the backend.

Another great feature of user pool is that you can add triggers to specify your own code.

For example, you could run your own code, at the time that someone signs up to add this user to a backend database during that time.

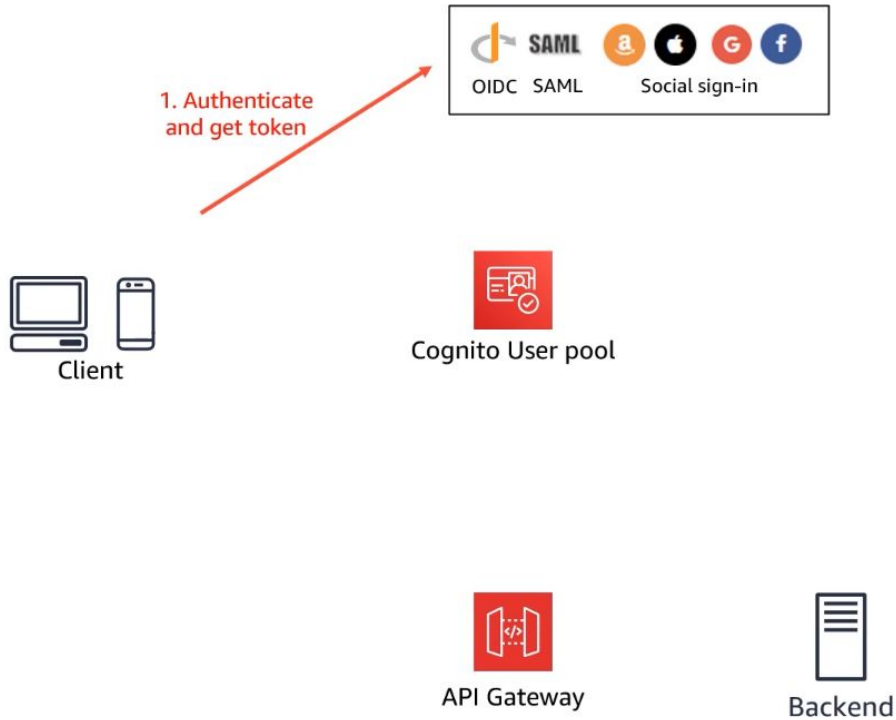
User pool can also do **federation** within user pools.

Instead of authenticating the user on user pool itself,  
and hosting the user name and password,  
you can have them authenticate via a third party.

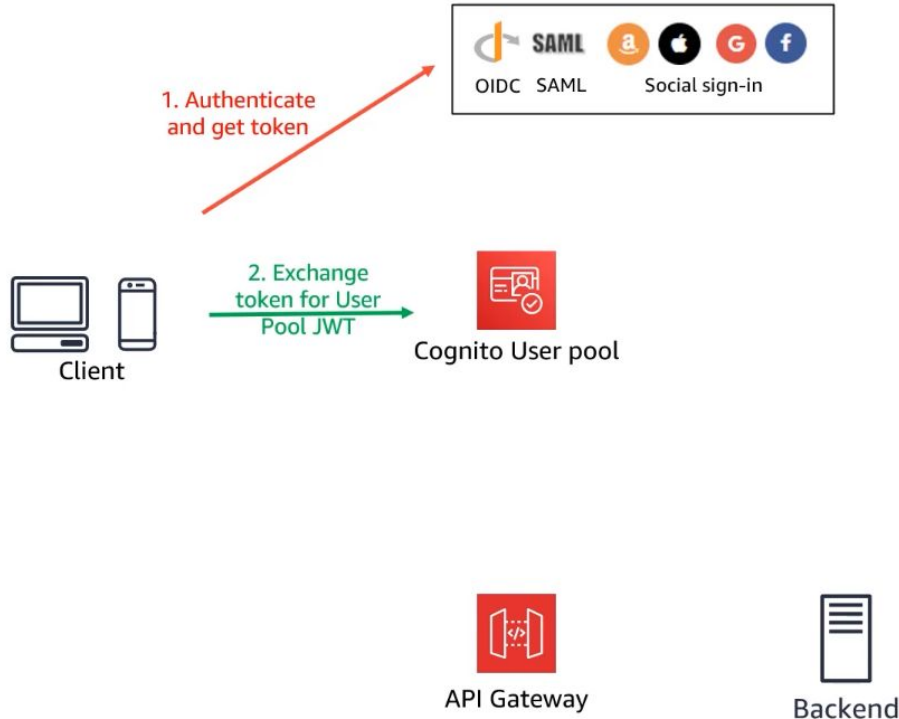
This way, the user doesn't have to create and remember yet another set of credentials.



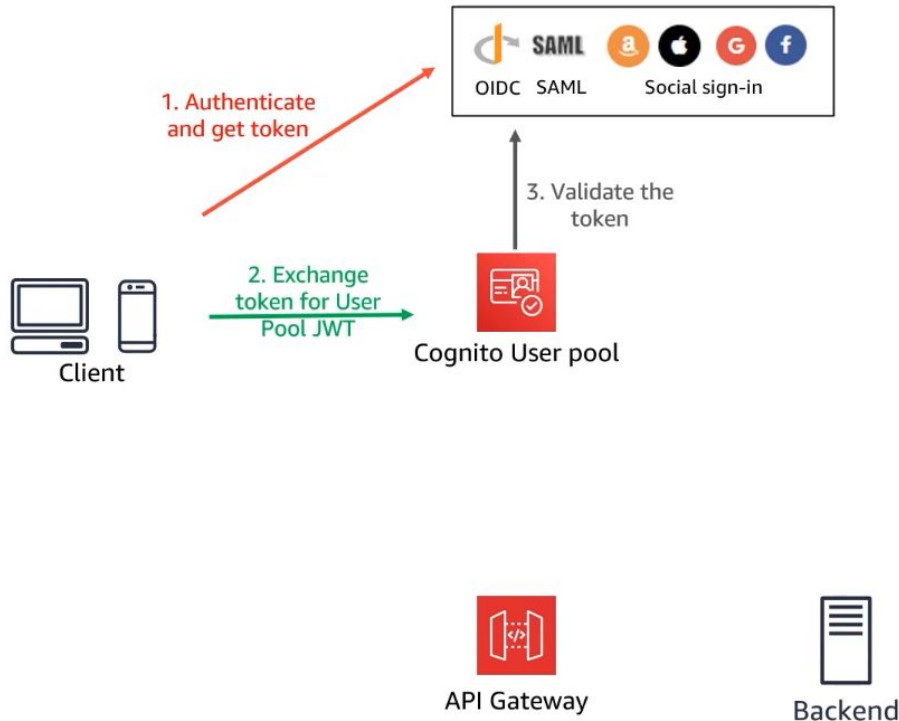
The first step is for the client to authenticate via OpenID Connect, SAML, or some social sign-in providers



After the user sign in, the client will receive a token.

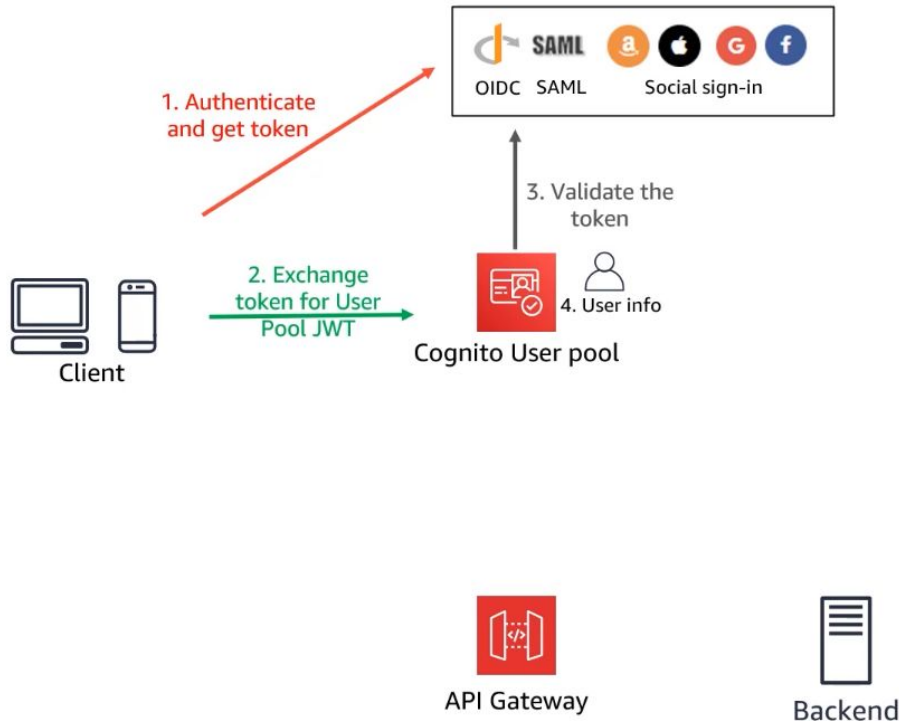


client sends a request to Cognito user pool, to exchange the token received from one of those third party and get a JWT token in return.

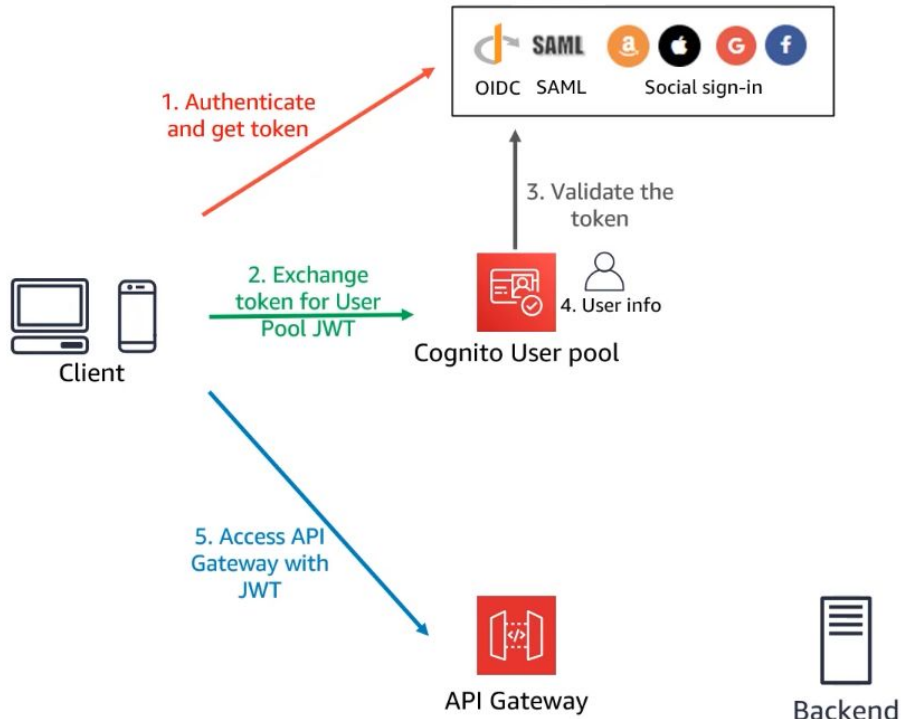


Before providing the JWT token, Cognito will validate the token with the provider.

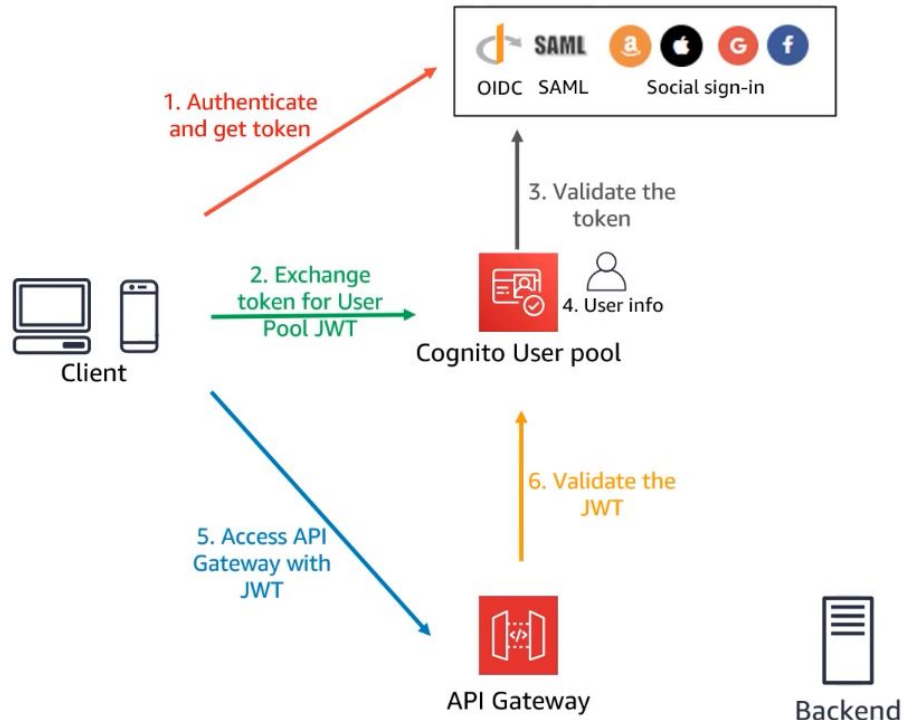




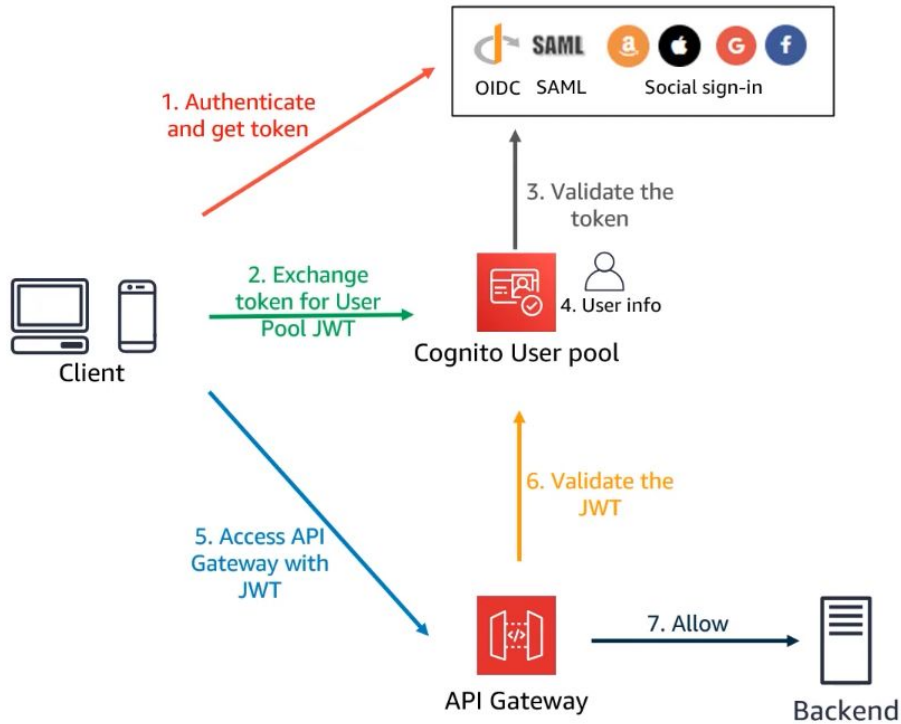
Then it will create a copy of the information it just received. A new copy of this user info. And it will finally return the JWT token to the client.



The client can then use that JWT to authenticated and authorized to API Gateway.



API Gateway which will again validate the JWT token with user pool.



And finally allow (or not) access to the backend.

why not just directly integrate your backend with the social providers?

Why have Cognito user pool in the middle of all of this?

why not just directly integrate your backend with the social providers?

Why have Cognito user pool in the middle of all of this?

Authorization.

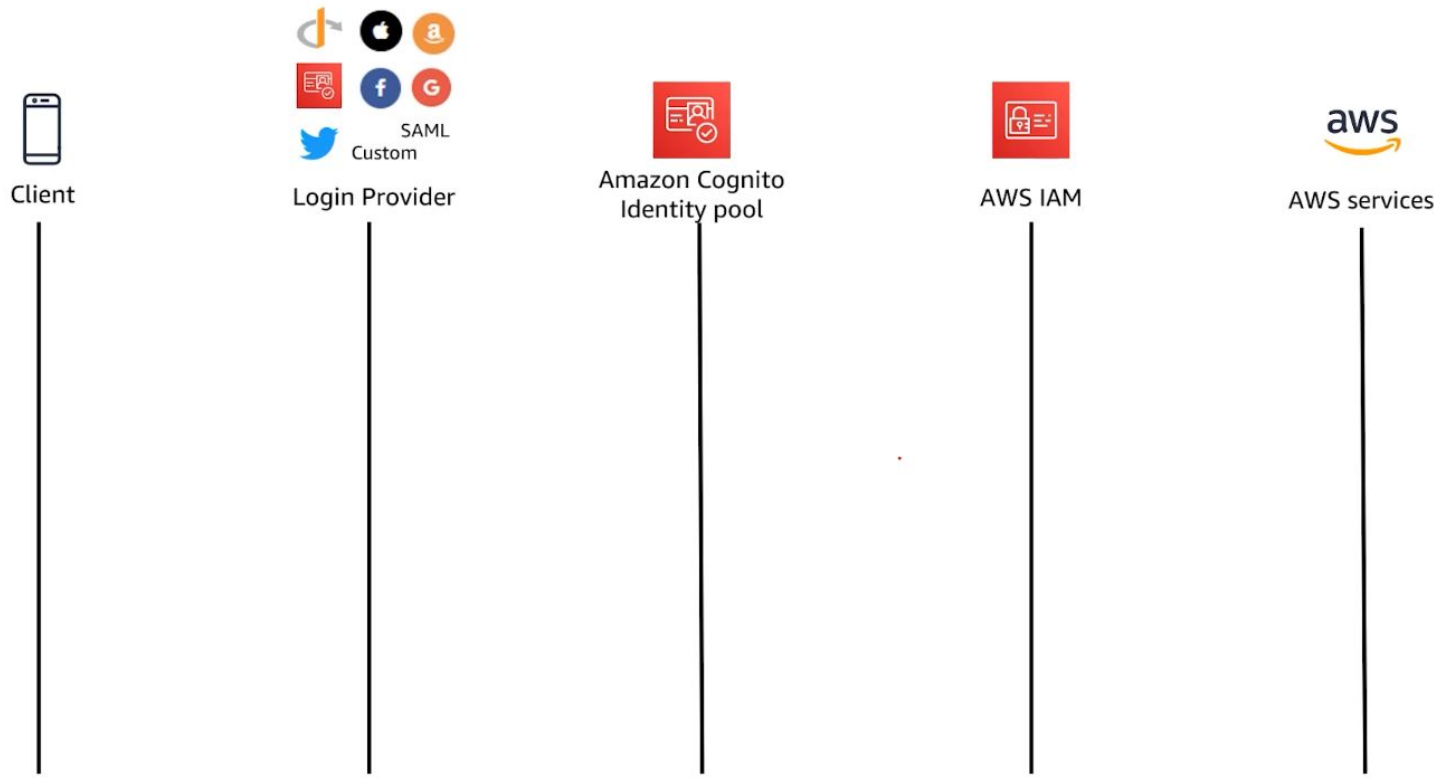
Cognito is for the time that you wanted to give certain users different permissions.

You can assign the user copy  
with some permissions or authorization scopes within Cognito user pool.

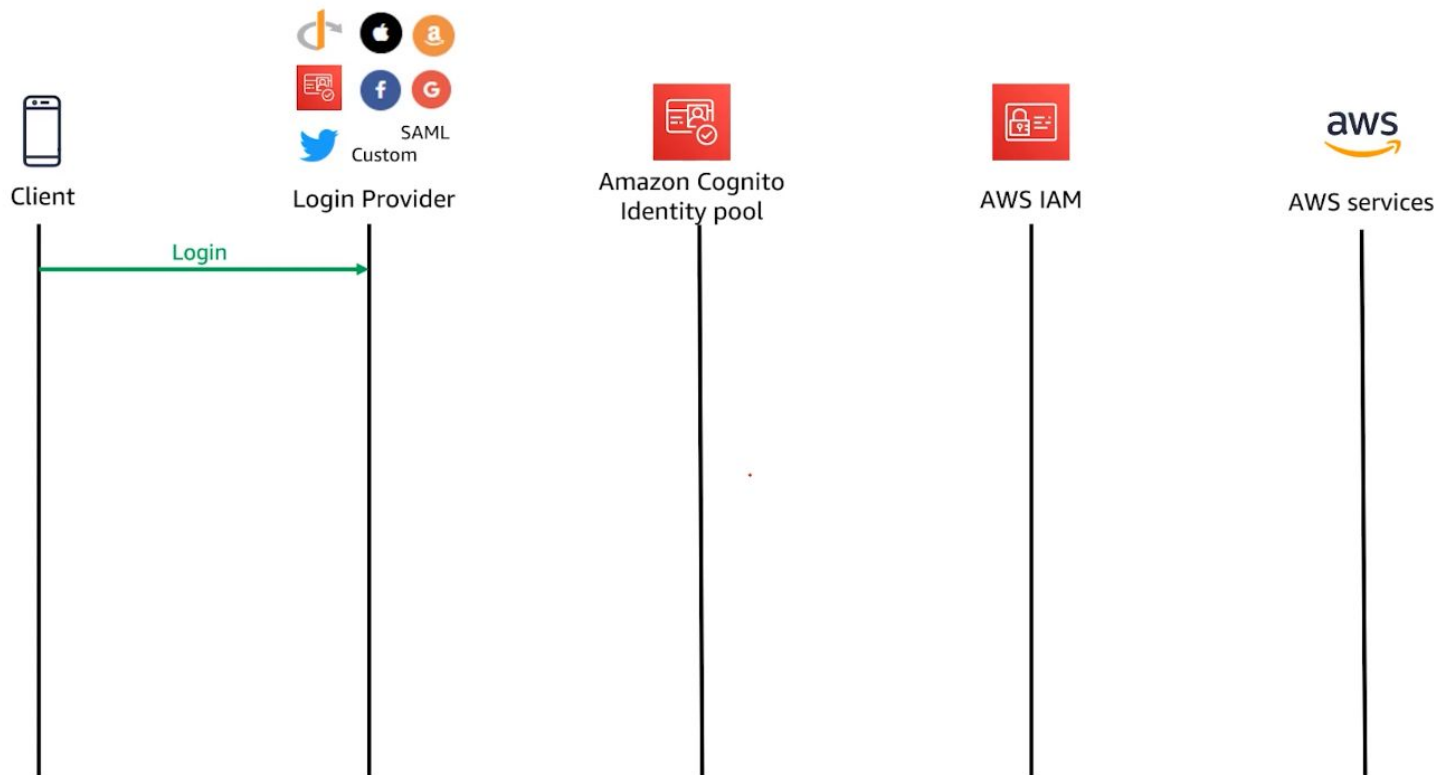
Let's talk about the second service of Amazon Cognito  
**Federated Identities**

This is a service that's used to assign IAM roles to users who authenticate through a separate identity provider.

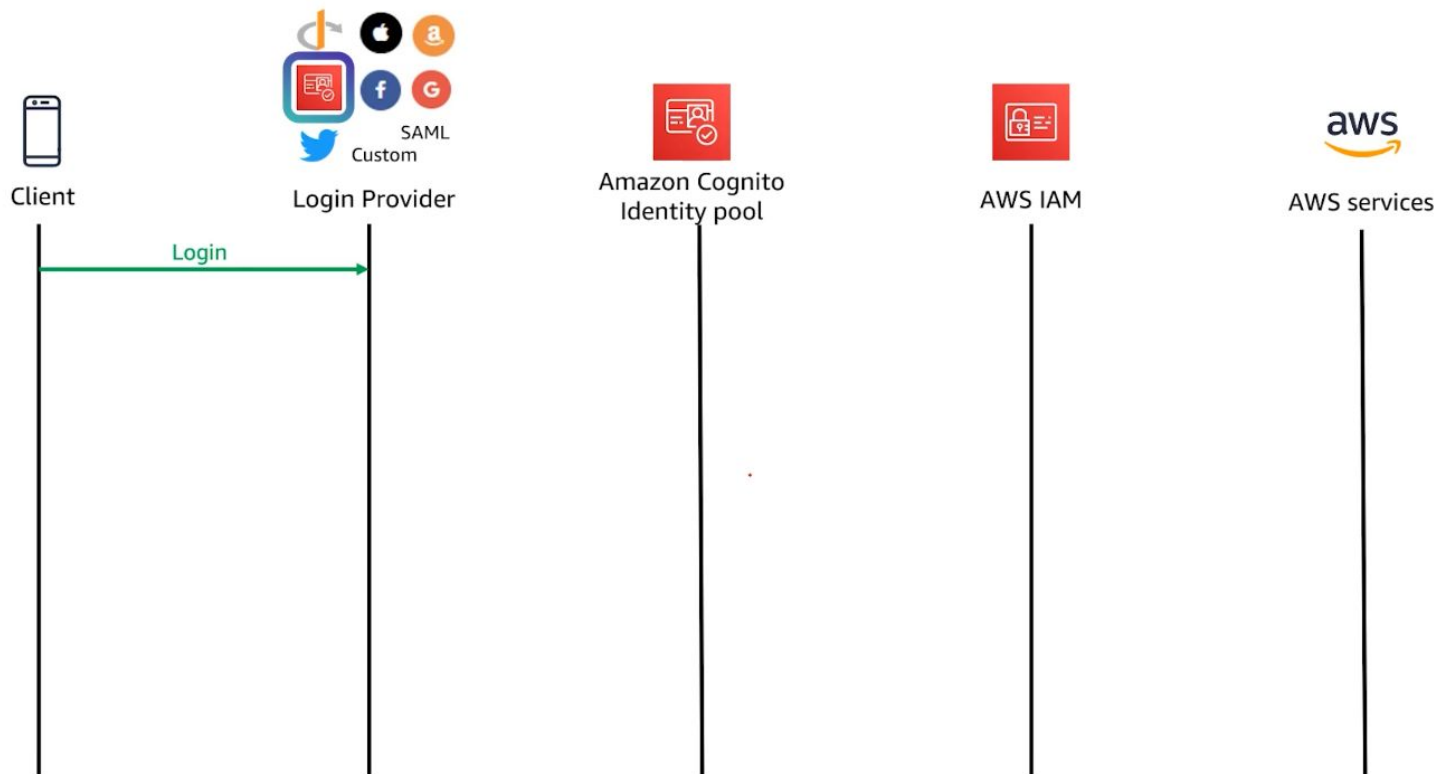




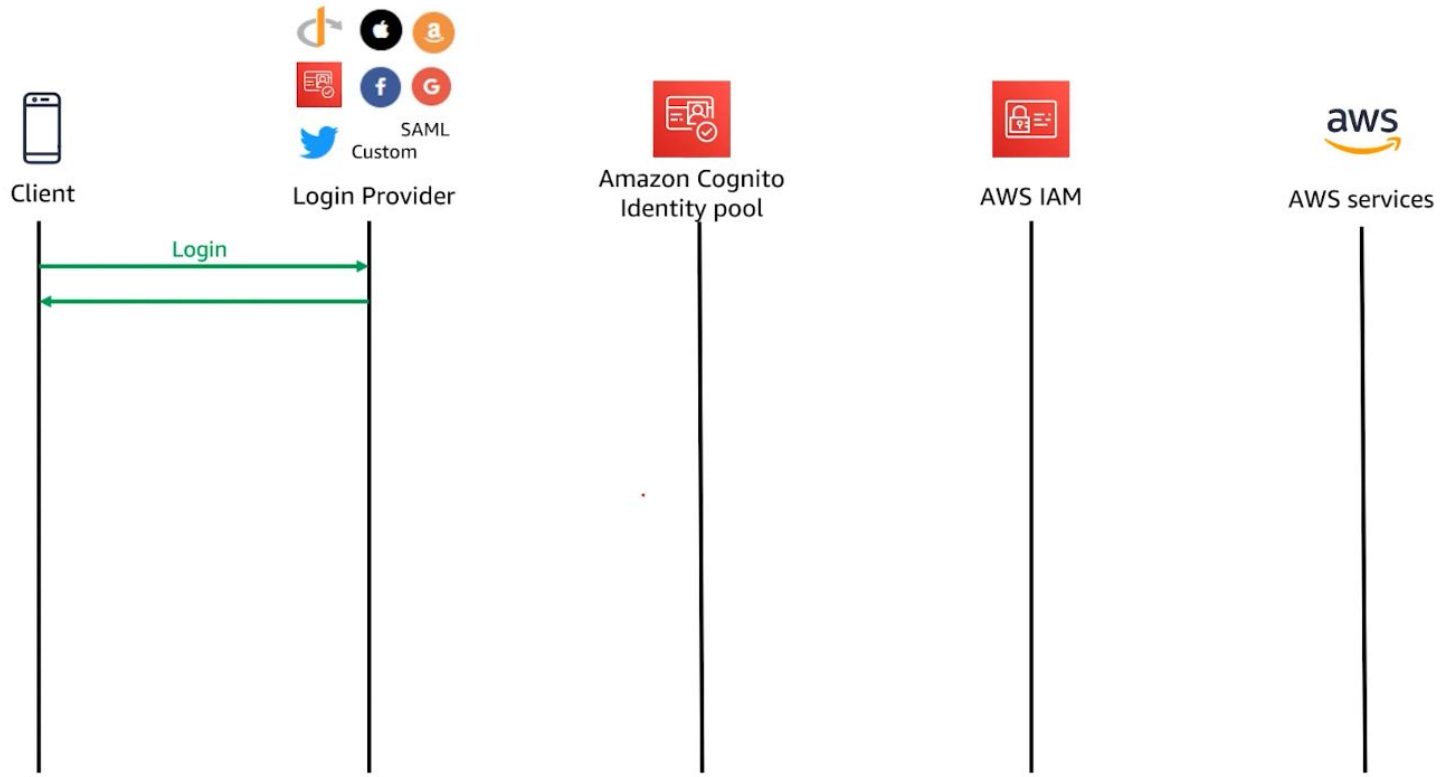
Let's look at a workflow of Federated Identities.



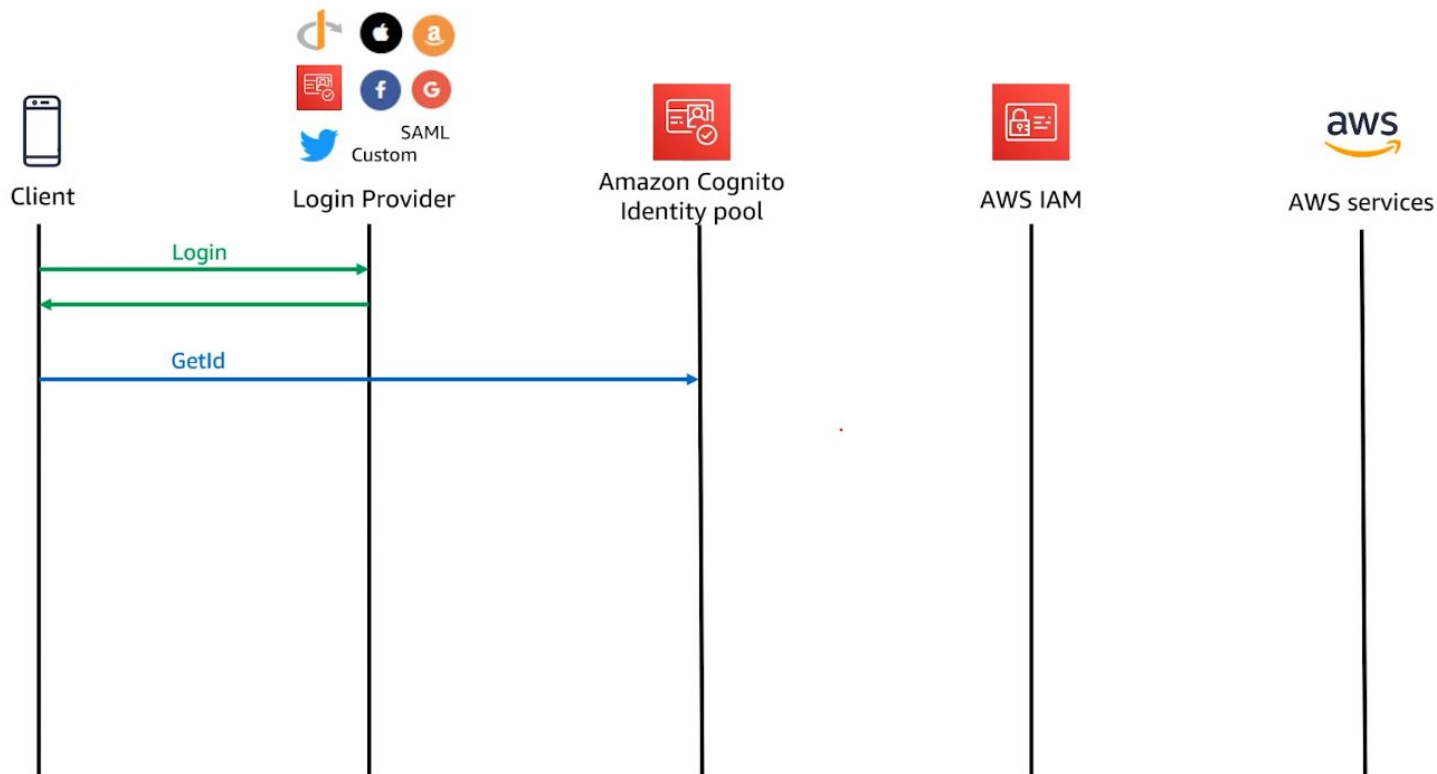
First, the client will log in to a login provider.



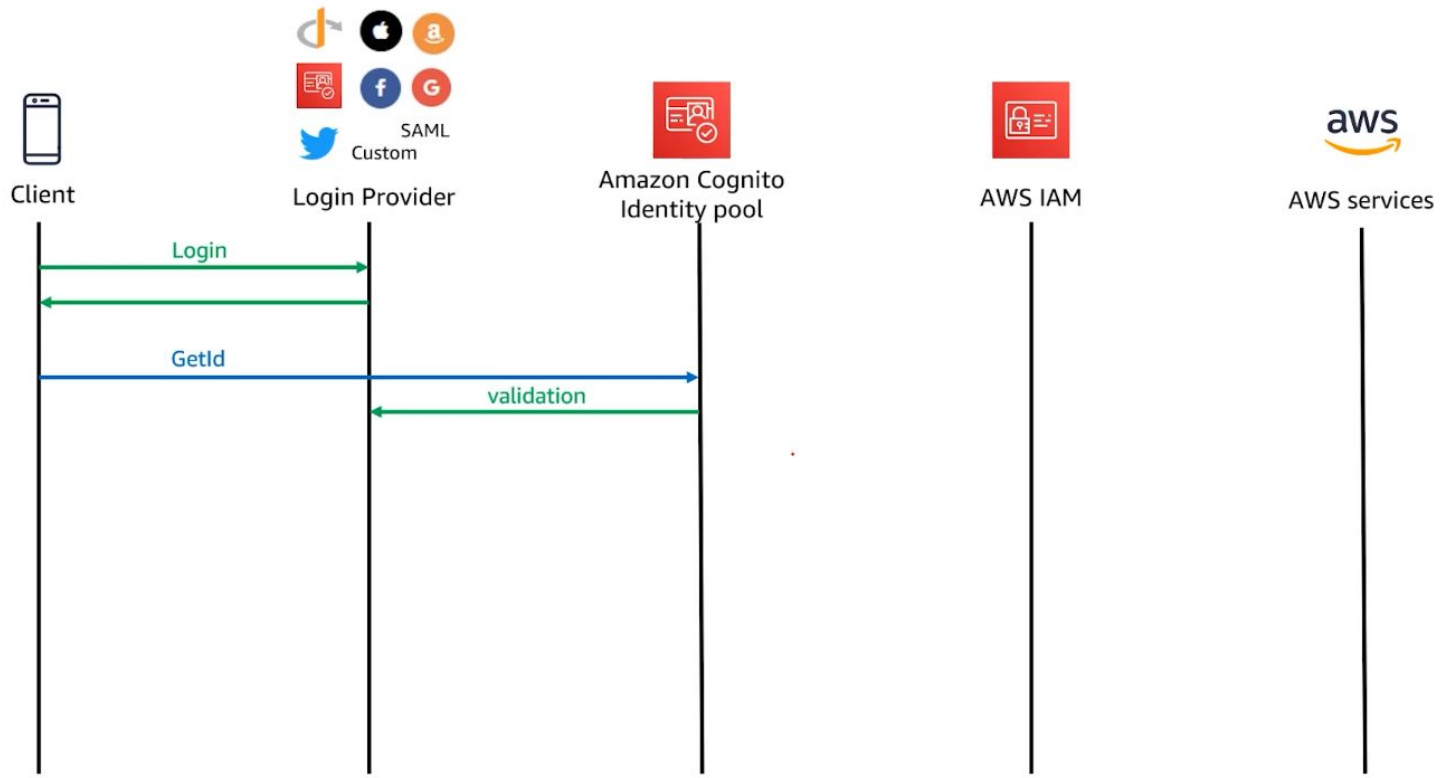
Cognito user pool and Federated Identities?! We'll come back to that later.



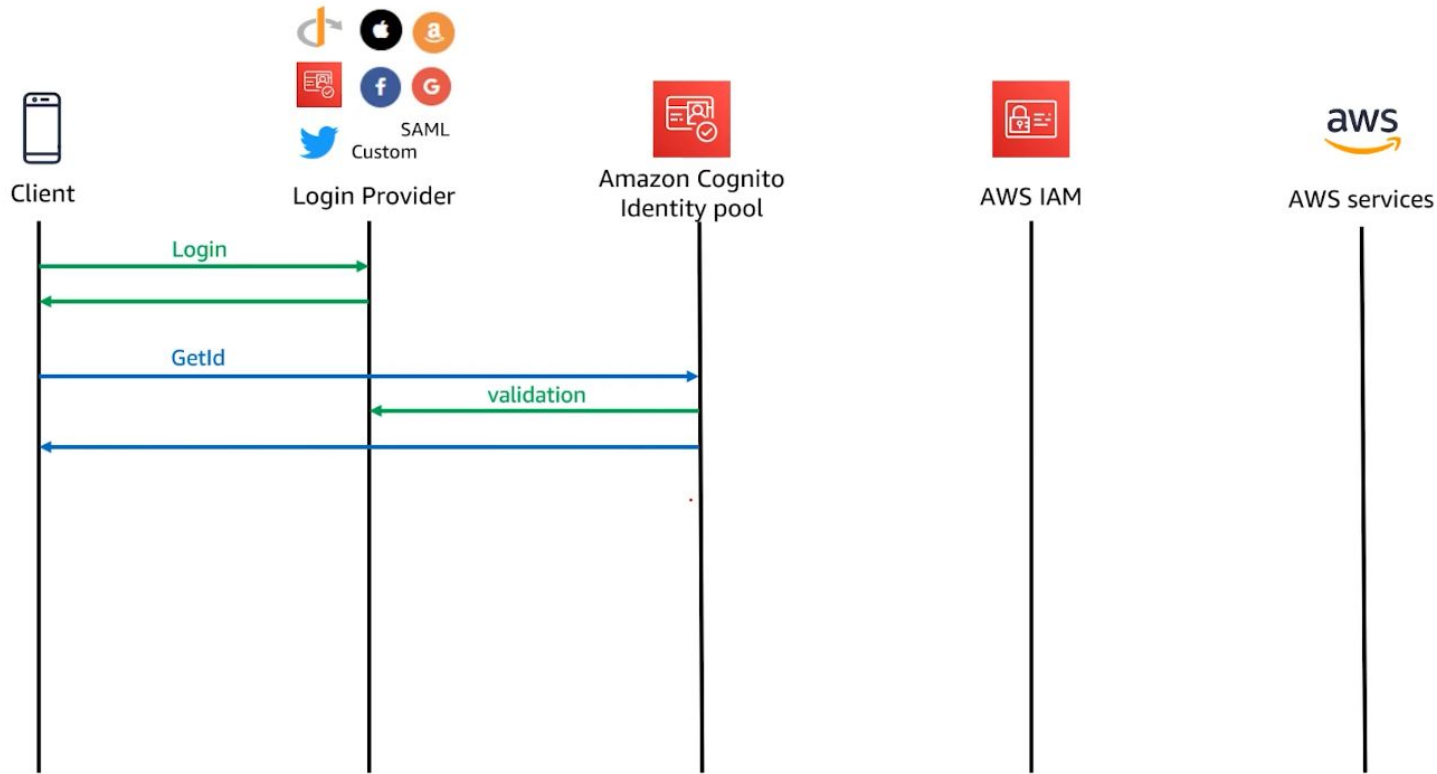
Then once you're done with that login, the response (jwt token) will be sent back towards the client.



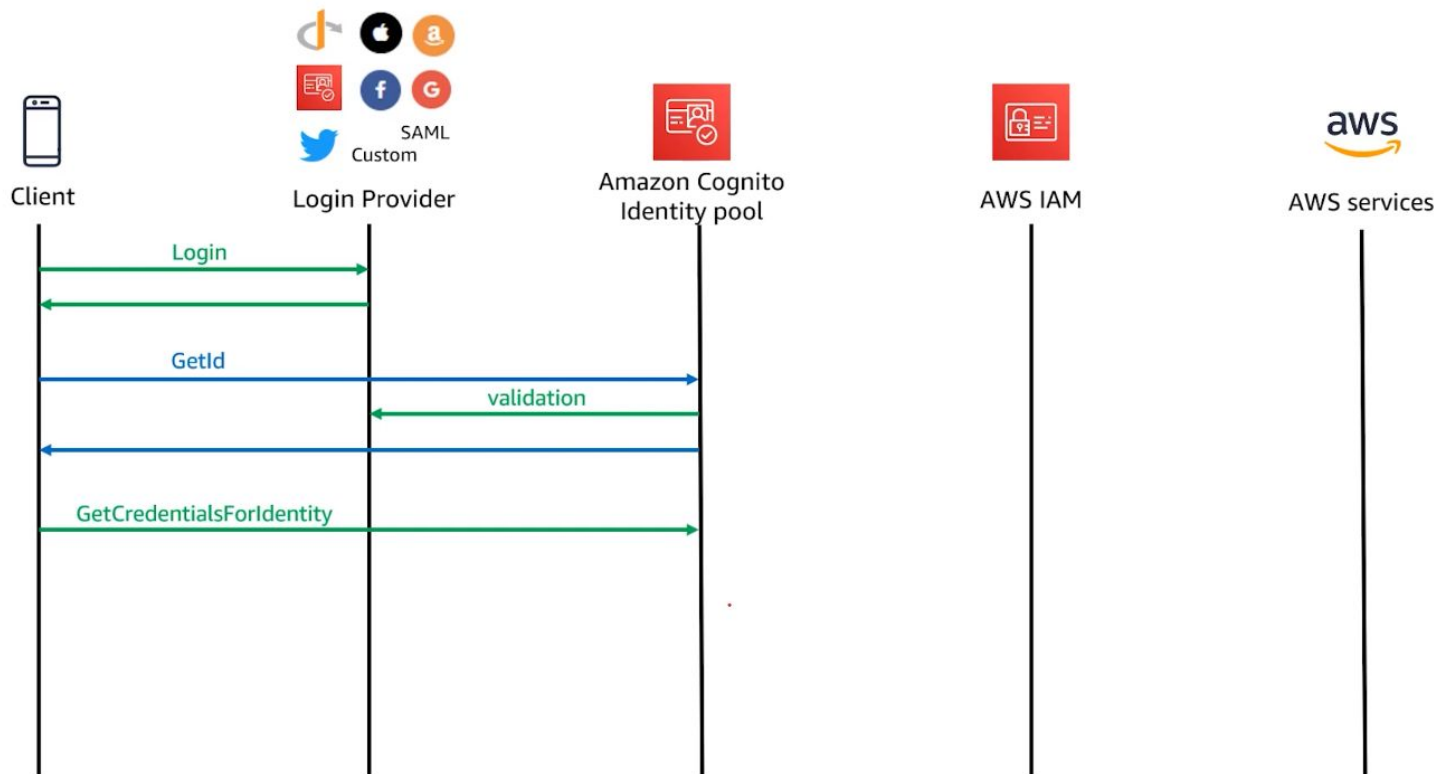
Then the client will send the API call, GetID, to the identity pool, using the token it received from the login provider.



That token will be validated with the login provider and identity pool Will let the login provider and the client know about it.

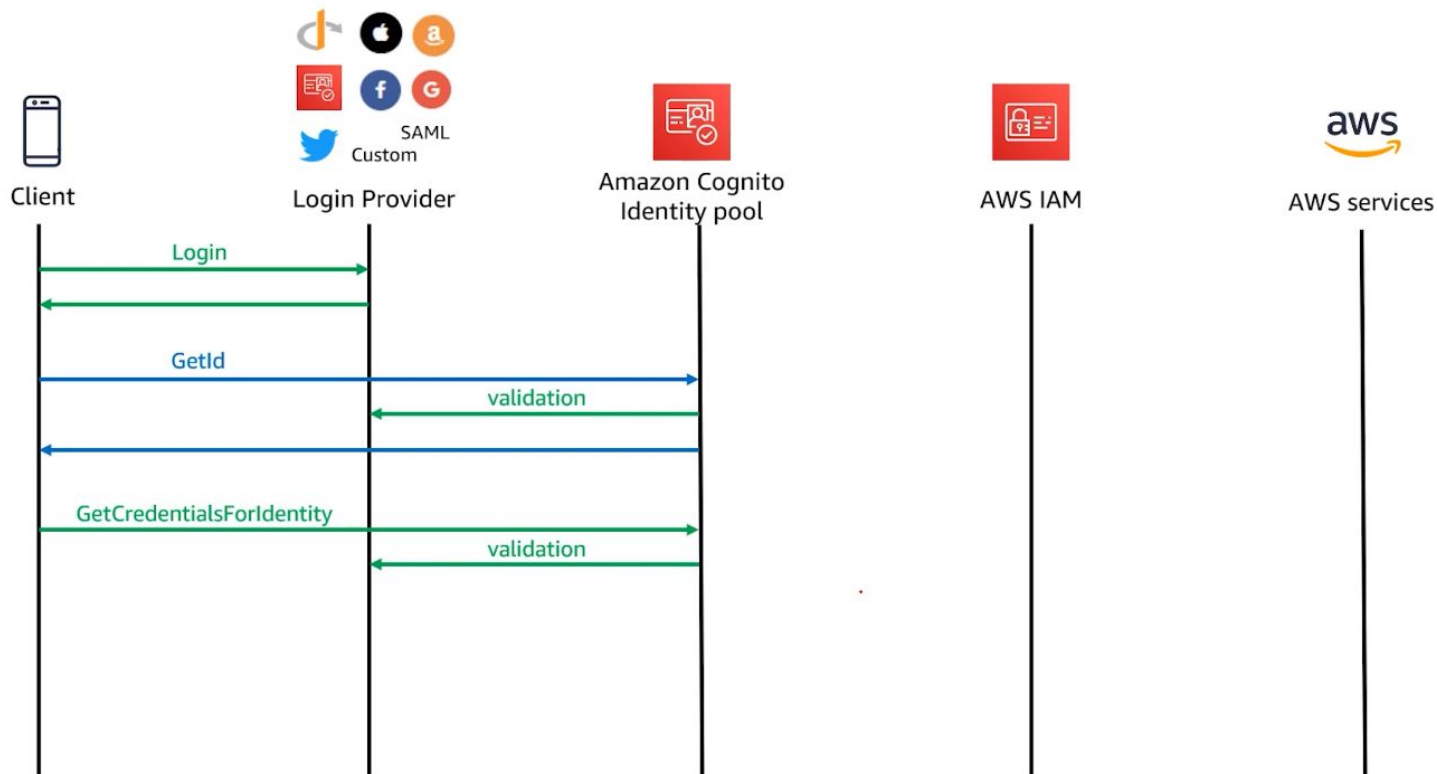


That token will be validated with the login provider and identity pool Will let the login provider and the client know about it.

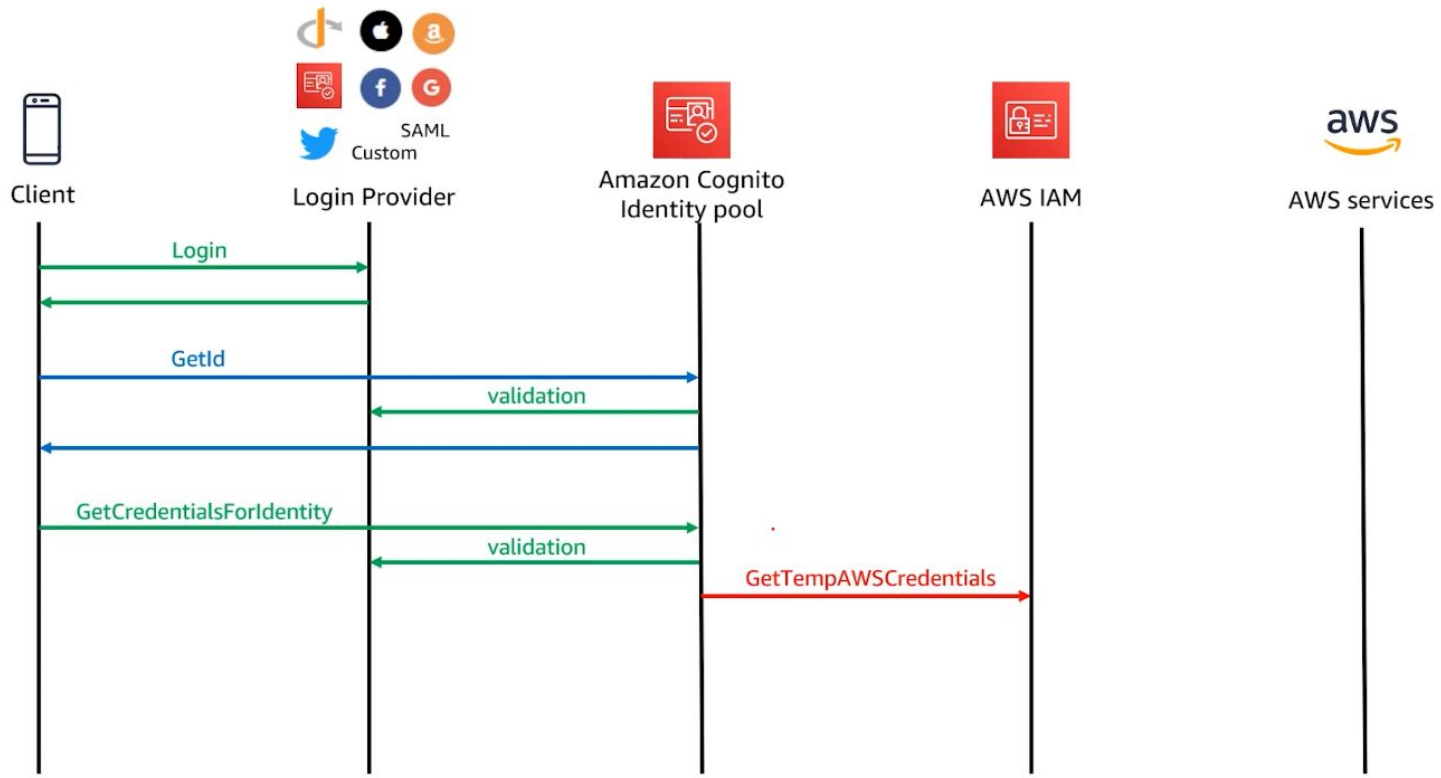


Next, the client will use the API call, `GetCredentialsForIdentity` request, for specifying what IAM role it would like to become.

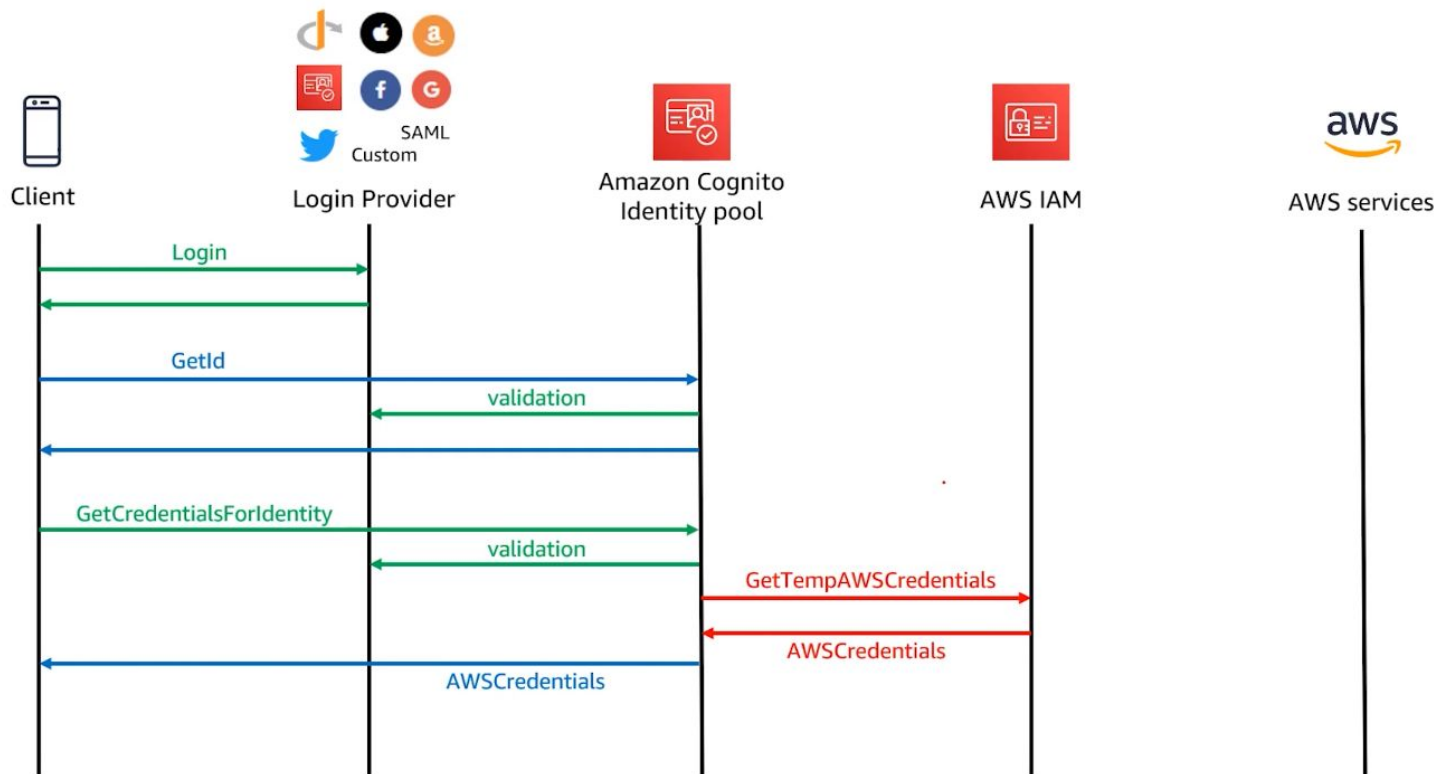




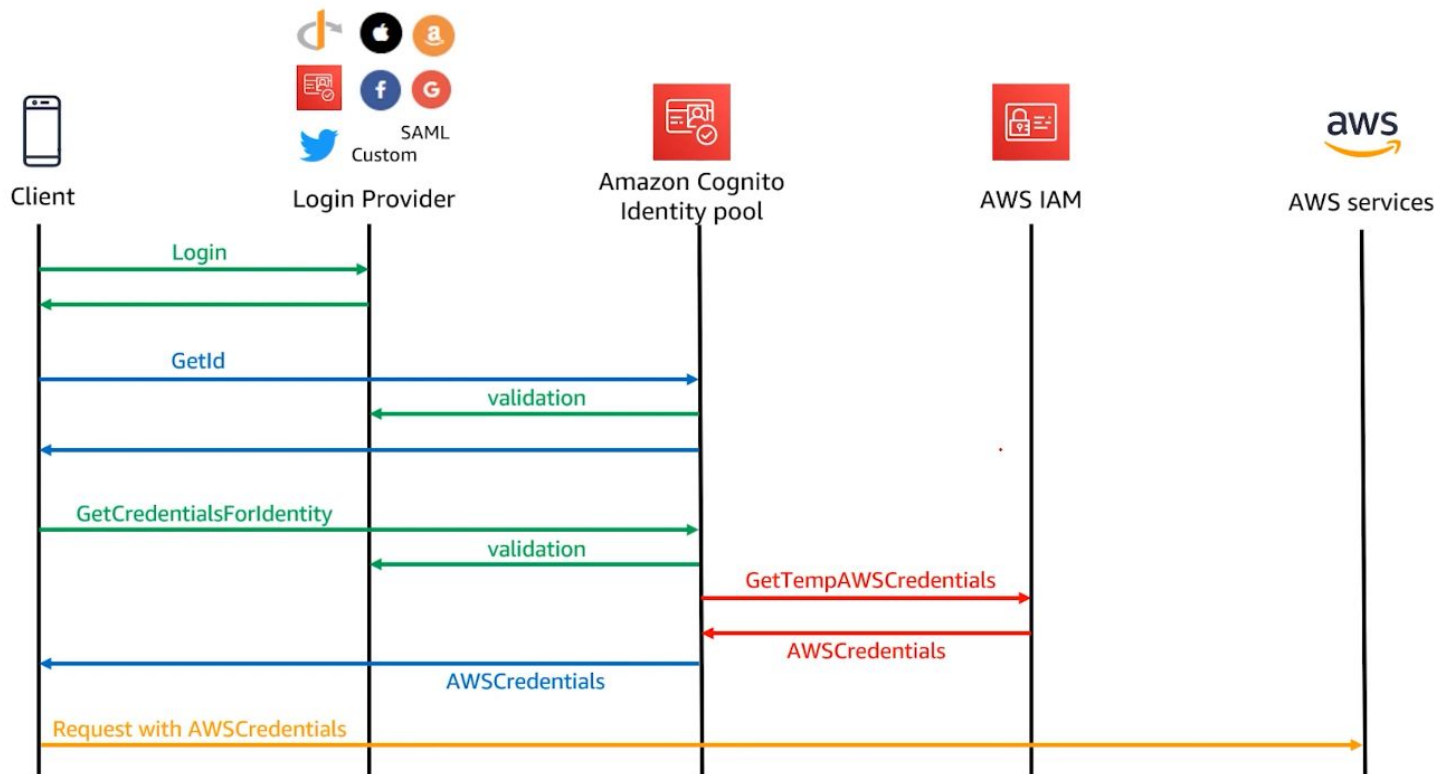
That will be validated against the login provider



Cognito identity pool, will request AWS temporary credentials from AWS IAM.



Credentials will then be returned to Cognito, and back to the client.



Finally, the client can use those AWS credentials to access any AWS services.

Up until now, we directly authorize Cognito user pools,  
in API Gateway via JSON Web Tokens.

And now, with Cognito Federated Identities,  
we can actually authenticate with one of those login providers,  
which includes Cognito user pool!

That means there's two ways to authenticate to API Gateway!

Let go through the workflow of these two services that are working together.



Client

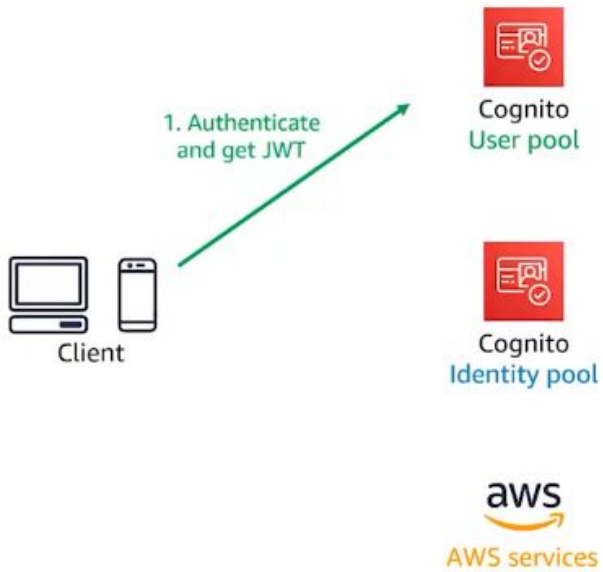


Cognito  
User pool



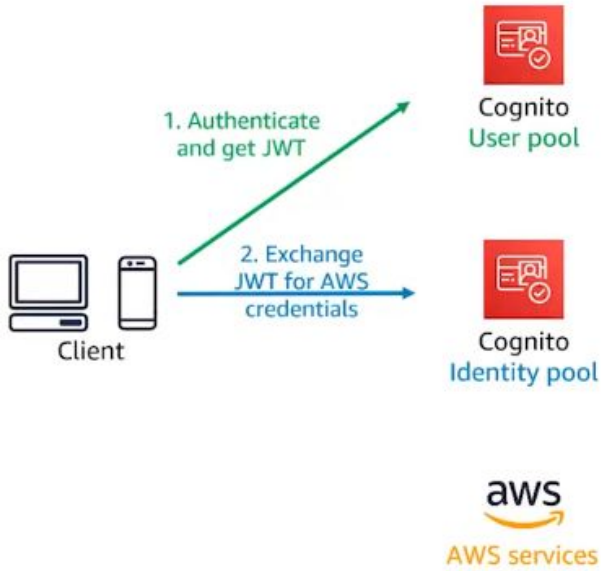
Cognito  
Identity pool

aws  
AWS services

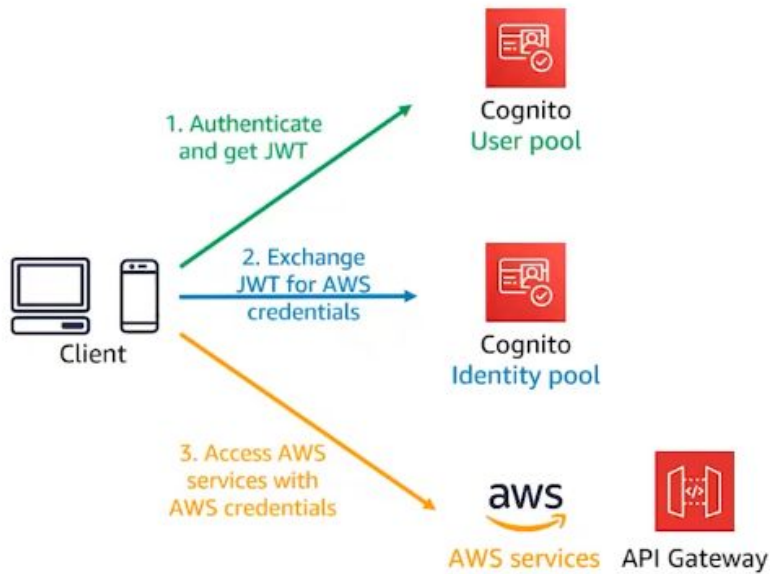


First, the client will authenticate and get a JWT token from Cognito user pool.





Then it will exchange that JWT token for AWS credentials, via Cognito identity pool.



Now, it can use those credentials to access any AWS services, including API Gateway.

Which one should I choose?

authorizing directly via Cognito user pool or using Federated Identities (identity pool)?

If you only need to communicate with API Gateway  
and you want your backend to have more information about the user,  
then integrating with API Gateway directly with **user pool** is a good idea.

If you need to communicate with almost any other AWS services directly, then you will need to use the **Federated Identities**.

The last feature of Federated Identities we want to discuss is  
the **unauthenticated identity**.

You can give a default IAM role,  
for anyone that **doesn't** have credentials to authenticate.

For example, this could be to provide a very small amount of access to your backend, while the user isn't authenticated yet, or just doesn't want to.

This is something that **Federated Identities** provides via **unauthenticated identities**.

It's a great way to provide temporary credentials to users,  
even if they don't want to share their identities with you.



**Use Amazon Cognito to Sign In and Call API Gateway**

Let's go over a demonstration of how to configure API Gateway to authenticate users via an Amazon Cognito User Pool.



A client (a browser) wants to interact with the Dragons API

Our goal is to add authentication and authorization  
to the GET/dragons API.



The first step is create an S3 bucket that we'll be hosting the website.



- Website:
- index.html
  - callback.html



API Gateway

It's a very basic website with only two files: index.html and callback.html



1



S3

- Website:
- index.html
  - callback.html

2



Cognito User Pool



API Gateway

The second step is to create a Cognito User Pool.



1



S3

Website:

- index.html
- callback.html

2



Cognito User Pool

App client

- Callback URL: callback.html
- Hosted UI



API Gateway

Define the client application specifying that when the user is authenticated, redirect that user back to the callback HTML page in S3.





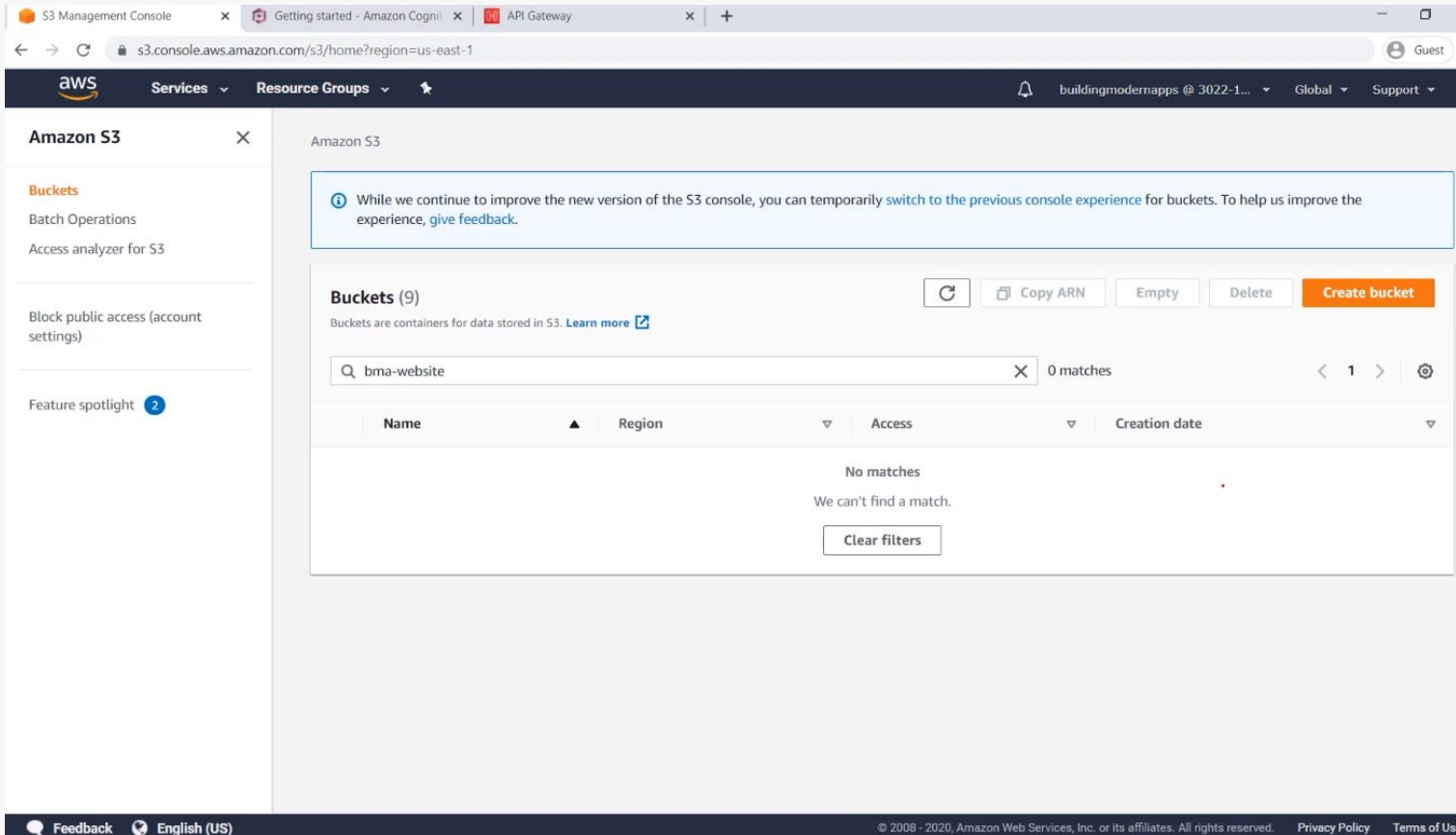
also configure it with the hosted UI to sign up and sign in the user.



third, configure the dragons API and API Gateway to add it onto an authorizer pointing to this Cognito User Pool, and then I will add this authorizer to the GET/dragons API.

Let's go to do that.

First we need to create our website in S3.



Here we are in the S3 console to create a brand new S3 bucket.

Amazon S3 > Create bucket

## Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

### General configuration

Bucket name

bma-website

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

Region

US East (N. Virginia) us-east-1

### Bucket settings for Block Public Access

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

**Block all public access**

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

**Block public access to buckets and objects granted through new access control lists (ACLs)**

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

select a name

S3 Management Console | Getting started - Amazon Cognito | API Gateway | s3.console.aws.amazon.com/s3/bucket/create?region=us-east-1

aws Services Resource Groups buildingmodernapps @ 3022-1... Global Support

Bucket name  
bma-website  
Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

Region  
US East (N. Virginia) us-east-1

### Bucket settings for Block Public Access

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

- Block all public access**  
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
- Block public access to buckets and objects granted through new access control lists (ACLs)**  
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**  
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**  
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**  
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use


it's fine for a website to be fully public

S3 Management Console | Getting started - Amazon Cognito | API Gateway | s3.console.aws.amazon.com/s3/bucket/create?region=us-east-1

Services | Resource Groups | buildingmodernapps @ 3022-1... | Global | Support


Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through new access control lists (ACLs)**  
S3 will block public access permissions applied to newly-added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through any access control lists (ACLs)**  
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through new public bucket or access point policies**  
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through any public bucket or access point policies**  
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

 **Turning off block all public access might result in this bucket and the objects within becoming public**  
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

► **Advanced settings**

 After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel **Create bucket**

Feedback | English (US) | © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. | Privacy Policy | Terms of Use

acknowledge and create the bucket.



Now, It's time to add our website with  
two files, right index.html, callback.html.

But we need to modify one of this to add some information about  
our Cognito User Pool later.

So we only upload callback file now.

S3 Management Console | Getting started - Amazon Cognito | API Gateway | s3.console.aws.amazon.com/s3/home?region=us-east-1

Services | Resource Groups | buildingmodernapps @ 3022-1... | Global | Support

**Amazon S3** ✕

**Buckets**

Batch Operations

Access analyzer for S3

Block public access (account settings)

Feature spotlight 2

**Successfully created bucket "bma-website"**  
To upload files and folders, or to configure additional bucket settings choose [View details](#).

Amazon S3

While we continue to improve the new version of the S3 console, you can temporarily [switch to the previous console experience](#) for buckets. To help us improve the experience, [give feedback](#).

**Buckets (10)** ↻ Copy ARN Empty Delete Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

Q bma-website ✕ 1 match < 1 > ⚙️

Name	Region	Access	Creation date
<input type="radio"/> <a href="#">bma-website</a>	US East (N. Virginia) us-east-1	<a href="#">Objects can be public</a>	July 16, 2020, 01:55 (UTC-04:00)

Waiting for us-east-1.console.aws.amazon.com... | © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

click on the bucket name



Amazon S3 &gt; bma-website

## bma-website

Overview

Properties

Permissions

Management

Access points



Upload

+ Create folder

Download

Actions

US East (N. Virginia)



This bucket is empty. Upload new objects to get started.



### Upload an object

Buckets are globally unique containers for everything that you store in Amazon S3.

[Learn more](#)

### Set object properties

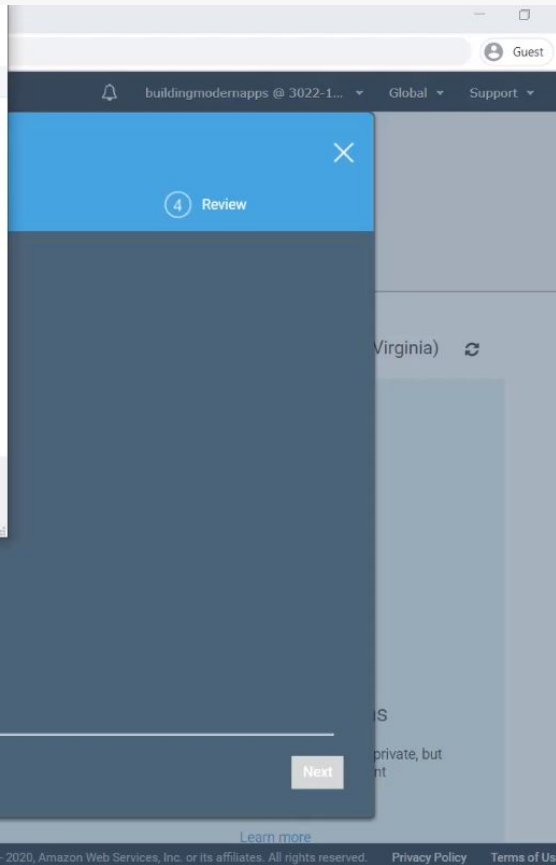
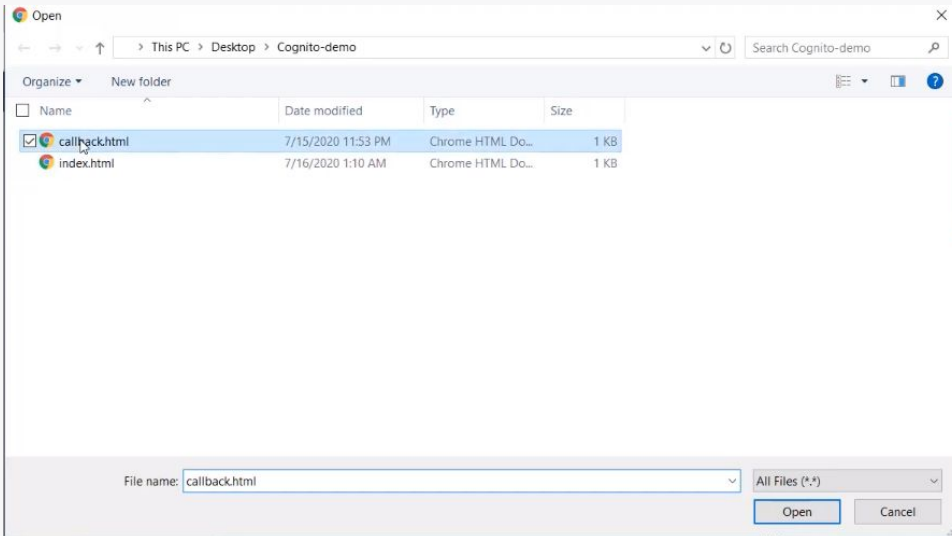
After you create a bucket, you can upload your objects (for example, your photo or video files).

[Learn more](#)

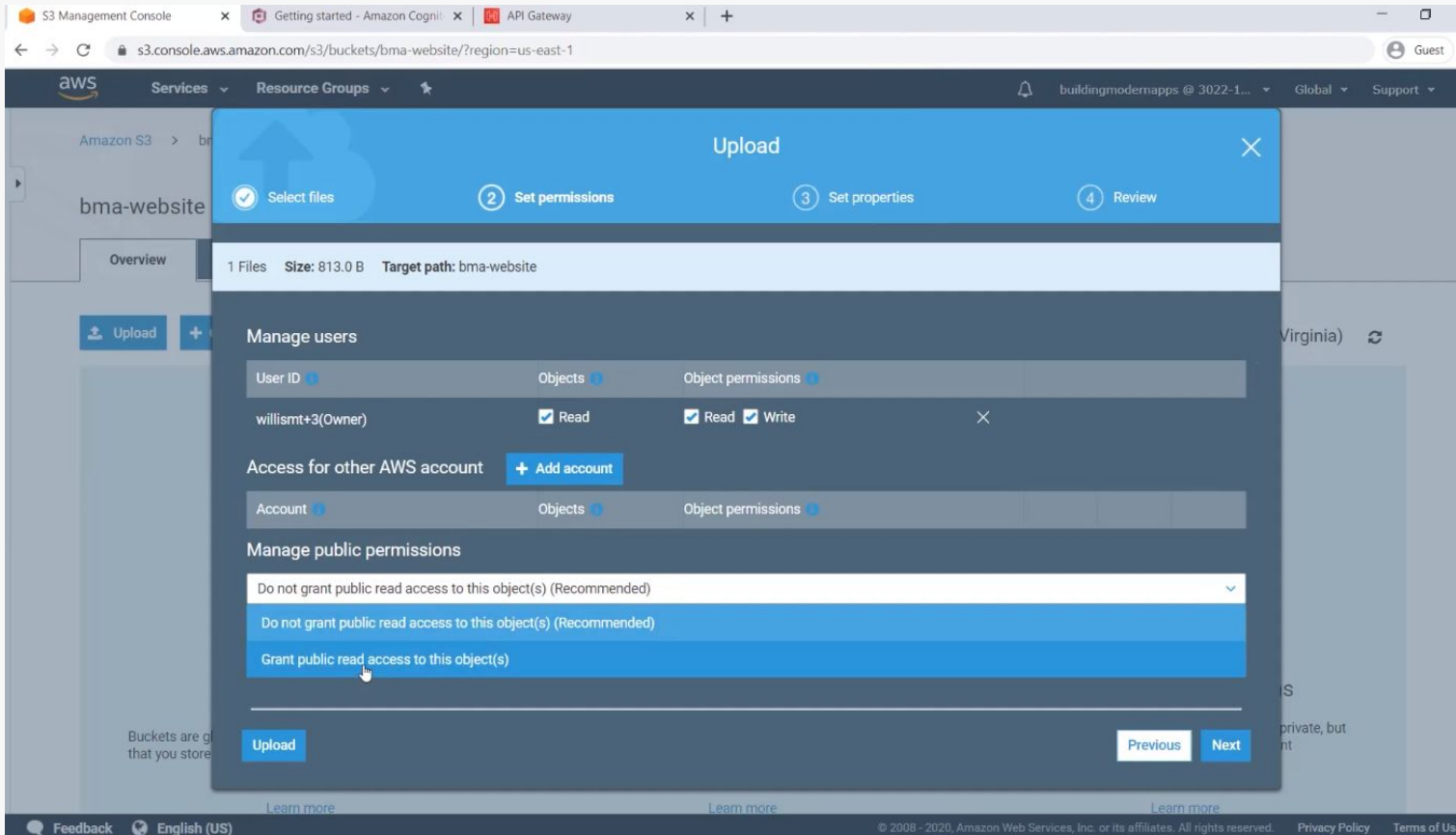
### Set object permissions

By default, the permissions on an object are private, but you can set up access control policies to grant permissions to others.

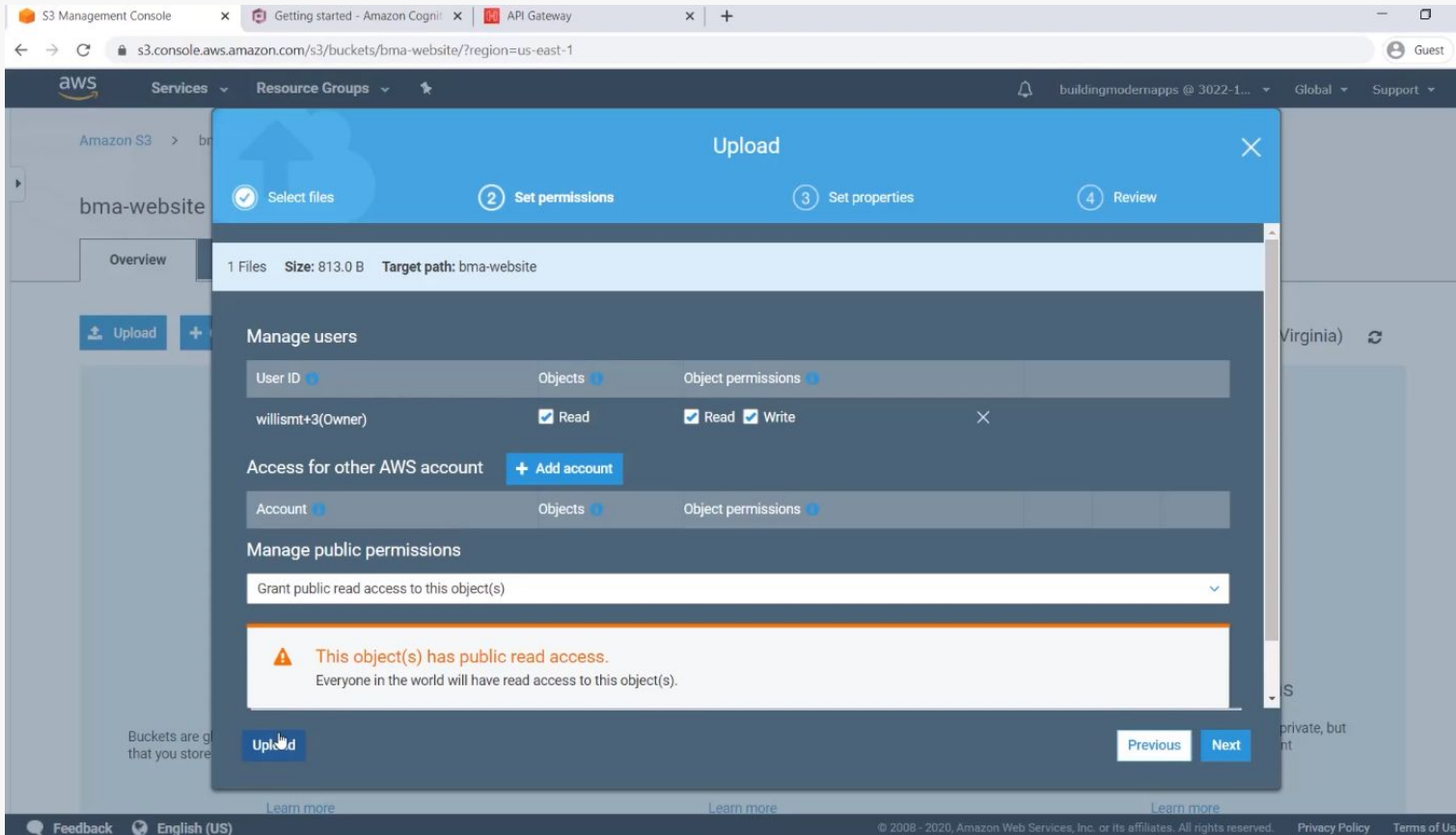
[Learn more](#)



Files?



select grant public read access to this object



click on the upload button.

Step number two is to configure our user pool.



# Amazon Cognito

Amazon Cognito offers user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools provide AWS credentials to grant your users access to other AWS services.

[Manage User Pools](#)

[Manage Identity Pools](#)



## Add Sign-up and Sign-in

With Cognito User Pools, you can easily and securely add sign-up and sign-in functionality to your mobile and web apps with a fully-managed service that scales to support hundreds of millions of users.



## Grant your users access to AWS services

With Cognito Identity Pools, your app can get temporary credentials to access AWS services for anonymous guest users or for users who have signed in.

Cognito console





# Amazon Cognito

Amazon Cognito offers user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools provide AWS credentials to grant your users access to other AWS services.

[Manage User Pools](#)

[Manage Identity Pools](#)



## Add Sign-up and Sign-in

With Cognito User Pools, you can easily and securely add sign-up and sign-in functionality to your mobile and web apps with a fully-managed service that scales to support hundreds of millions of users.



## Grant your users access to AWS services

With Cognito Identity Pools, your app can get temporary credentials to access AWS services for anonymous guest users or for users who have signed in.

which service? in this case, we use the user pool



## Your User Pools

[Create a user pool](#)

You have no user pools. [Click here to create a user pool.](#)

- Name**
- Attributes
- Policies
- MFA and verifications
- Message customizations
- Tags
- Devices
- App clients
- Triggers
- Review

### What do you want to name your user pool?

Give your user pool a descriptive name so you can easily identify it in the future.

Pool name

### How do you want to create your user pool?

**Review defaults**  
Start by reviewing the defaults and then customize as desired

**Step through settings**  
Step through each setting to make your choices

remember the name of your cognito user pool for later use

S3 Management Console x User Pools - Amazon Cognito x API Gateway x +

console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/new/details?\_k=5v2u5d Guest

aws Services Resource Groups

User Pools Federated Identities

# Create a user pool

Cancel

- Name
- Attributes
- Policies
- MFA and verifications
- Message customizations
- Tags
- Devices
- App clients
- Triggers
- Review

<b>Pool name</b>	demo
<b>Required attributes</b>	email
<b>Alias attributes</b>	<a href="#">Choose alias attributes...</a>
<b>Username attributes</b>	<a href="#">Choose username attributes...</a>
<b>Enable case insensitivity?</b>	Yes
<b>Custom attributes</b>	<a href="#">Choose custom attributes...</a>
<b>Minimum password length</b>	8
<b>Password policy</b>	uppercase letters, lowercase letters, special characters, numbers
<b>User sign ups allowed?</b>	Users can sign themselves up
<b>FROM email address</b>	Default
<b>Email Delivery through Amazon SES</b>	Yes
<b>MFA</b>	<a href="#">Enable MFA...</a>
<b>Verifications</b>	Email
<b>Tags</b>	<a href="#">Choose tags for your user pool</a>

You can see in this page what those default features are.

Devices  
App clients  
Triggers  
**Review**

**Username attributes** [Choose username attributes...](#)

**Enable case insensitivity?** Yes

**Custom attributes** [Choose custom attributes...](#)

**Minimum password length** 8

**Password policy** uppercase letters, lowercase letters, special characters, numbers

**User sign ups allowed?** Users can sign themselves up

**FROM email address** Default

**Email Delivery through Amazon SES** Yes

**MFA** [Enable MFA...](#)

**Verifications** Email

**Tags** [Choose tags for your user pool](#)

**App clients** [Add app client...](#)

**Triggers** [Add triggers...](#)

Create pool

The next step is to configure the domain name of our hosted UI to do the authentication and authorization for us.

### What domain would you like to use?

- General settings
  - Users and groups
  - Attributes
  - Policies
  - MFA and verifications
  - Advanced security
  - Message customizations
  - Tags
  - Devices
  - App clients
  - Triggers
  - Analytics
- App integration
  - App client settings
  - Domain name**
  - UI customization
  - Resource servers
- Federation
  - Identity providers
  - Attribute mapping

Type a domain prefix to use for the sign-up and sign-in pages that are hosted by Amazon Cognito. The prefix must be unique across the selected AWS Region. Domain names can only contain lower-case letters, numbers, and hyphens. [Learn more about domain prefixes.](#)

#### Amazon Cognito domain

Prefixed domain names can only contain lower-case letters, numbers, and hyphens. [Learn more about domain prefixes.](#)

#### Domain prefix

https://  .auth.us-east-1.amazoncognito.com

#### Your own domain

This domain name needs to have an associated certificate in [AWS Certificate Manager \(ACM\)](#). You also need the ability to add an alias record to the domain's hosted zone after it's associated with this user pool. [Learn more about using your own domain.](#)

[Go to summary](#)

[Customize UI](#)

Click on the Domain name here.

We can either configure and use a Cognito domain name  
or our own domain name.



General settings

- Users and groups
- Attributes
- Policies
- MFA and verifications
- Advanced security
- Message customizations
- Tags
- Devices
- App clients
- Triggers
- Analytics
- App integration
- App client settings
- Domain name**
- UI customization
- Resource servers
- Federation
- Identity providers
- Attribute mapping

### What domain would you like to use?

Type a domain prefix to use for the sign-up and sign-in pages that are hosted by Amazon Cognito. The prefix must be unique across the selected AWS Region. Domain names can only contain lower-case letters, numbers, and hyphens. [Learn more about domain prefixes.](#)

This domain is available.

#### Amazon Cognito domain

Prefixed domain names can only contain lower-case letters, numbers, and hyphens. [Learn more about domain prefixes.](#)

##### Domain prefix

https://  .auth.us-east-1.amazoncognito.com

#### Your own domain

This domain name needs to have an associated certificate in [AWS Certificate Manager \(ACM\)](#). You also need the ability to add an alias record to the domain's hosted zone after it's associated with this user pool. [Learn more about using your own domain.](#)

[Go to summary](#)

[Customize UI](#)

Next we need to define an **App Client**.

The user pool contains your users, their information  
(username, password, email, phone number, etc.)

Now this user pool doesn't exactly relate to one application or one API Gateway.

It can be a one to many applications.

However, each one of those applications,  
they need own settings for doing the authentication flow.

That's why we have app clients or application clients.

These are the definition of each one of these application that is going to use Cognito User Pool as their database of users.



## demo

## General settings

Users and groups

Attributes

Policies

MFA and verifications

Advanced security

Message customizations

Tags

Devices

App clients

Triggers

Analytics

## App integration

App client settings

Domain name

UI customization

Resource servers

## Federation

Identity providers

Attribute mapping

## Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

[Add an app client](#)[Return to pool details](#)

- General settings
- Users and groups
- Attributes
- Policies
- MFA and verifications
- Advanced security
- Message customizations
- Tags
- Devices
- App clients**
- Triggers
- Analytics
- App integration
  - App client settings
  - Domain name
  - UI customization
  - Resource servers
- Federation
  - Identity providers
  - Attribute mapping

## Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

**App client name**

**Refresh token expiration (days)**

Generate client secret

Auth Flows Configuration

Enable username password auth for admin APIs for authentication (ALLOW\_ADMIN\_USER\_PASSWORD\_AUTH) [Learn more.](#)

Enable lambda trigger based custom authentication (ALLOW\_CUSTOM\_AUTH) [Learn more.](#)

Enable username password based authentication (ALLOW\_USER\_PASSWORD\_AUTH) [Learn more.](#)

Enable SRP (secure remote password) protocol based authentication (ALLOW\_USER\_SRP\_AUTH) [Learn more.](#)


Enable refresh token based authentication (ALLOW\_REFRESH\_TOKEN\_AUTH) [Learn more.](#)

Prevent User Existence Errors [Learn more.](#)

Legacy

Enabled (Recommended)

[Set attribute read and write permissions](#)



This refresh token expiration is how long a user can stay logged in.

- General settings
- Users and groups
- Attributes
- Policies
- MFA and verifications
- Advanced security
- Message customizations
- Tags
- Devices
- App clients**
- Triggers
- Analytics
- App integration
  - App client settings
  - Domain name
  - UI customization
  - Resource servers
- Federation
  - Identity providers
  - Attribute mapping

## Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

**App client name**

**Refresh token expiration (days)**

Generate client secret

Auth Flows Configuration

Enable username password auth for admin APIs for authentication (ALLOW\_ADMIN\_USER\_PASSWORD\_AUTH) [Learn more.](#)

Enable lambda trigger based custom authentication (ALLOW\_CUSTOM\_AUTH) [Learn more.](#)

Enable username password based authentication (ALLOW\_USER\_PASSWORD\_AUTH) [Learn more.](#)

Enable SRP (secure remote password) protocol based authentication (ALLOW\_USER\_SRP\_AUTH) [Learn more.](#)


Enable refresh token based authentication (ALLOW\_REFRESH\_TOKEN\_AUTH) [Learn more.](#)

Prevent User Existence Errors [Learn more.](#)

Legacy

Enabled (Recommended)

[Set attribute read and write permissions](#)



remove this generate client secret checkbox. This is more for a server to server communication.

Users and groups

Attributes

Policies

MFA and verifications

Advanced security

Message customizations

Tags

Devices

App clients

Triggers

Analytics

App integration

App client settings

Domain name

UI customization

Resource servers

Federation

Identity providers

Attribute mapping

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

**App client name**

bma-website

**Refresh token expiration (days)**

30

 Generate client secret

## Auth Flows Configuration

 Enable username password auth for admin APIs for authentication (ALLOW\_ADMIN\_USER\_PASSWORD\_AUTH) [Learn more.](#) Enable lambda trigger based custom authentication (ALLOW\_CUSTOM\_AUTH) [Learn more.](#) Enable username password based authentication (ALLOW\_USER\_PASSWORD\_AUTH) [Learn more.](#) Enable SRP (secure remote password) protocol based authentication (ALLOW\_USER\_SRP\_AUTH) [Learn more.](#) Enable refresh token based authentication (ALLOW\_REFRESH\_TOKEN\_AUTH) [Learn more.](#)Prevent User Existence Errors [Learn more.](#) Legacy Enabled (Recommended)[Set attribute read and write permissions](#)

Cancel

Create app client

[Return to pool details](#)

The screenshot shows the AWS IAM console interface. The browser tabs include 'S3 Management Console', 'User Pools - Amazon Cognito', and 'API Gateway'. The URL is 'console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1\_w2Aolw5Oy/app-integration-app-settings?\_k=prbojb'. The page title is 'demo' and the breadcrumb is 'User Pools | Federated Identities'. The left sidebar shows navigation options: 'General settings' (Users and groups, Attributes, Policies, MFA and verifications, Advanced security, Message customizations, Tags, Devices, App clients, Triggers, Analytics), 'App integration' (App client settings, Domain name, UI customization, Resource servers), and 'Federation' (Identity providers, Attribute mapping). The main content area is titled 'What identity providers and OAuth 2.0 settings should be used for your app clients?' and contains a sub-section for 'App client bma-website' with ID '40j9fatn4ci9ua94d6doef6gqf'. The settings include: 'Enabled Identity Providers' with a 'Select all' checkbox and 'Cognito User Pool' checkbox; 'Sign in and sign out URLs' with a text input field; 'Callback URL(s)' with a text input field; 'Sign out URL(s)' with a text input field; and 'OAuth 2.0' settings with 'Allowed OAuth Flows' (Authorization code grant, Implicit grant, Client credentials) and 'Allowed OAuth Scopes'.

The next step is to change some configurations in the app client settings that goes back to this app client that we just created.



- General settings
  - Users and groups
  - Attributes
  - Policies
  - MFA and verifications
  - Advanced security
  - Message customizations
  - Tags
  - Devices
  - App clients
  - Triggers
  - Analytics
- App integration
  - App client settings**
  - Domain name
  - UI customization
  - Resource servers
- Federation
  - Identity providers
  - Attribute mapping

## What identity providers and OAuth 2.0 settings should be used for your app clients?

Each of your app clients can use different identity providers and OAuth 2.0 settings. You must enable at least one identity provider for each app client. [Learn more about identity providers.](#)

### App client **bma-website**

ID 40j9fatn4ci9ua94d6doef6gqf

**Enabled Identity Providers**  Select all

- Cognito User Pool

---

**Sign in and sign out URLs**

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

**Callback URL(s)**

**Sign out URL(s)**

---

**OAuth 2.0**

Select the OAuth flows and scopes enabled for this app. [Learn more about flows and scopes.](#)

**Allowed OAuth Flows**

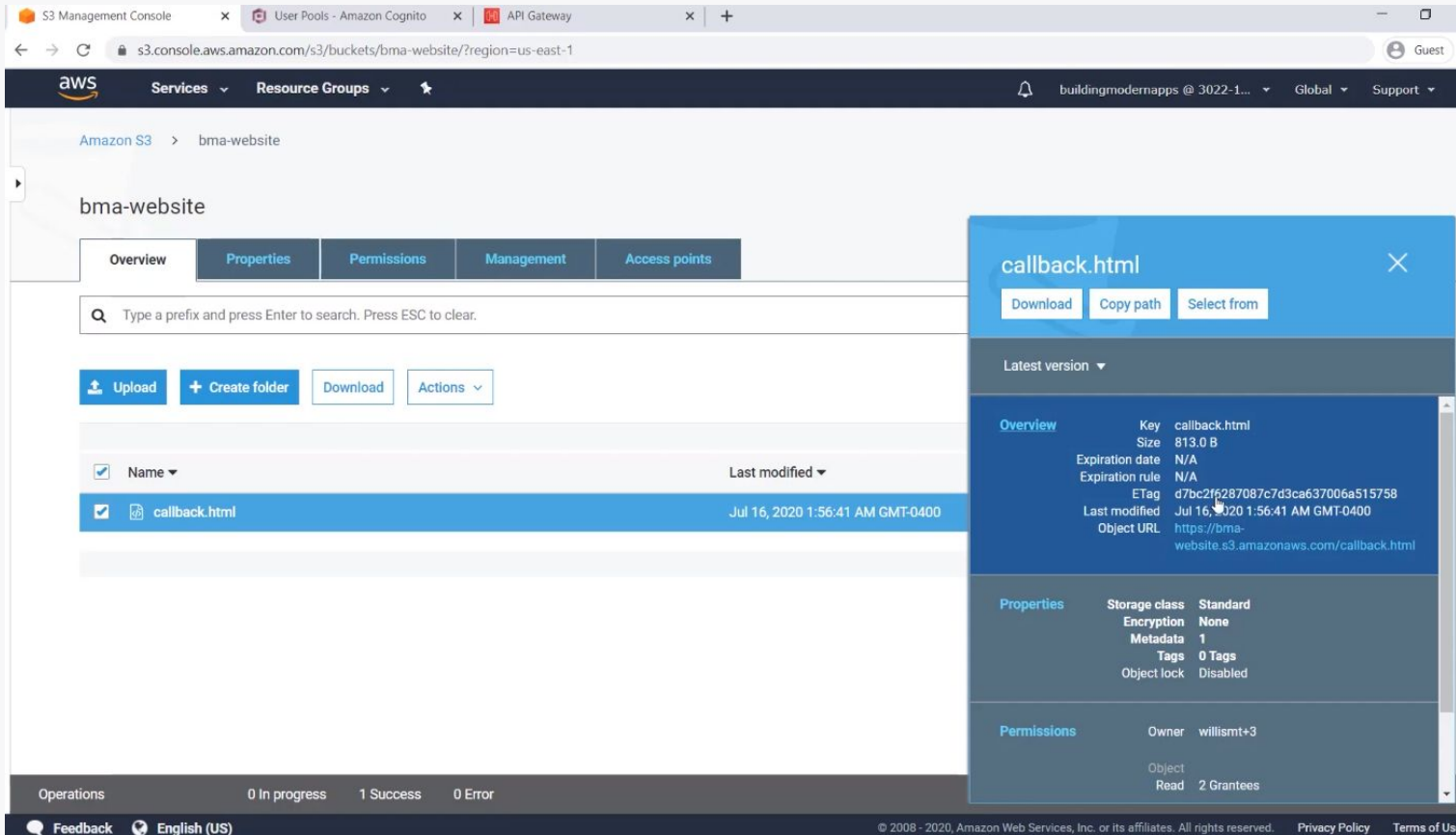
Authorization code grant  Implicit grant  Client credentials

**Allowed OAuth Scopes**

Enable Cognito as identity provider for this application.

The screenshot shows the AWS IAM console interface. The browser tabs include 'S3 Management Console', 'User Pools - Amazon Cognito', and 'API Gateway'. The URL is 'console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1\_w2Aolw5Oy/app-integration-app-settings?\_k=prbojb'. The page title is 'demo' and the breadcrumb is 'User Pools | Federated Identities'. The left sidebar shows navigation options: General settings, Users and groups, Attributes, Policies, MFA and verifications, Advanced security, Message customizations, Tags, Devices, App clients, Triggers, Analytics, App integration (highlighted), App client settings (highlighted), Domain name, UI customization, Resource servers, Federation, Identity providers, and Attribute mapping. The main content area is titled 'What identity providers and OAuth 2.0 settings should be used for your app clients?' and contains a sub-section for 'App client bma-website' with ID '40j9fatn4ci9ua94d6doef6gqf'. Under 'Enabled Identity Providers', 'Cognito User Pool' is checked. The 'Sign in and sign out URLs' section has empty input fields for 'Callback URL(s)' and 'Sign out URL(s)'. The 'OAuth 2.0' section has 'Allowed OAuth Flows' with 'Authorization code grant', 'Implicit grant', and 'Client credentials' all unchecked. A blue arrow icon is in the bottom right corner of the settings panel.

Then we need a callback URL so when our hosted UI will be done with authentication, send a client back to our website with the authentication token. So this is the URL of my callback file in S3.



in S3 console, click on the callback file, and copy its object URL name



## General settings

Users and groups

Attributes

Policies

MFA and verifications

Advanced security

Message customizations

Tags

Devices

App clients

Triggers

Analytics

## App integration

App client settings

Domain name

UI customization

Resource servers

## Federation

Identity providers

Attribute mapping

## What identity providers and OAuth 2.0 settings should be used for your app clients?

Each of your app clients can use different identity providers and OAuth 2.0 settings. You must enable at least one identity provider for each app client. [Learn more about identity providers.](#)

### App client bma-website

ID 40j9fatn4ci9ua94d6doef6gqf

#### Enabled Identity Providers

 Select all Cognito User Pool

#### Sign in and sign out URLs

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

##### Callback URL(s)

##### Sign out URL(s)

#### OAuth 2.0

Select the OAuth flows and scopes enabled for this app. [Learn more about flows and scopes.](#)

##### Allowed OAuth Flows

 Authorization code grant  Implicit grant  Client credentials

##### Allowed OAuth Scopes



S3 Management Console x User Pools - Amazon Cognito x API Gateway x +

console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1\_w2Aolw5Oy/app-integration-app-settings?\_k=prbojb Guest

Each of your app clients can use different identity providers and OAuth 2.0 settings. You must enable at least one identity provider for each app client. [Learn more about identity providers.](#)

### App client bma-website

ID 40j9fatn4ci9ua94d6doef6gqf

**Enabled Identity Providers**  Select all

Cognito User Pool

---

**Sign in and sign out URLs**

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

**Callback URL(s)**

**Sign out URL(s)**

---

**OAuth 2.0**

Select the OAuth flows and scopes enabled for this app. [Learn more about flows and scopes.](#)

**Allowed OAuth Flows**

Authorization code grant  Implicit grant  Client credentials

**Allowed OAuth Scopes**

phone  email  openid  aws.cognito.signin.user.admin  profile

**Hosted UI**

enable implicit grant here. This means that the JWT token will be returned back to the client and not be hidden or be using some kind of back channel for this.

S3 Management Console x User Pools - Amazon Cognito x API Gateway x +

console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1\_w2Aolw5Oy/app-integration-app-settings?\_k=prbojb Guest

Attributes  
Policies  
MFA and verifications  
Advanced security  
Message customizations  
Tags  
Devices  
App clients  
Triggers  
Analytics  
App integration  
App client settings  
Domain name  
UI customization  
Resource servers  
Federation  
Identity providers  
Attribute mapping

### App client bma-website

ID 40j9fatn4ci9ua94d6doef6gqf

**Enabled Identity Providers**  Select all

Cognito User Pool

**Sign in and sign out URLs**

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

**Callback URL(s)**

**Sign out URL(s)**

**OAuth 2.0**

Select the OAuth flows and scopes enabled for this app. [Learn more about flows and scopes.](#)

**Allowed OAuth Flows**


Authorization code grant  Implicit grant  Client credentials

**Allowed OAuth Scopes**

phone  email  openid  aws.cognito.signin.user.admin  profile

**Hosted UI**

The hosted UI is a web application that you can use to manage your user pool. It is built with the Amazon Cognito SDK and is hosted on Amazon S3. You can customize the look and feel of the hosted UI to match your application. For more information, see [Hosted UI](#).



what scopes we would like to give as part of the JWT token. This is going to be part of that token itself. So in case the backend will like to use them, select them.

S3 Management Console x User Pools - Amazon Cognito x API Gateway x +

console.aws.amazon.com/cognito/users/?region=us-east-1#/pool/us-east-1\_w2Aolw5Oy/app-integration-app-settings?\_k=prbojb Guest

- App clients
- Triggers
- Analytics
- App integration
  - App client settings**
  - Domain name
  - UI customization
  - Resource servers
- Federation
  - Identity providers
  - Attribute mapping

### Sign in and sign out URLs

Enter your callback URLs below that you will include in your sign in and sign out requests. Each field can contain multiple URLs by entering a comma after each URL.

**Callback URL(s)**

**Sign out URL(s)**

---

### OAuth 2.0

Select the OAuth flows and scopes enabled for this app. [Learn more about flows and scopes.](#)

**Allowed OAuth Flows**

Authorization code grant  Implicit grant  Client credentials


**Allowed OAuth Scopes**

phone  email  openid  aws.cognito.signin.user.admin  profile


---

### Hosted UI

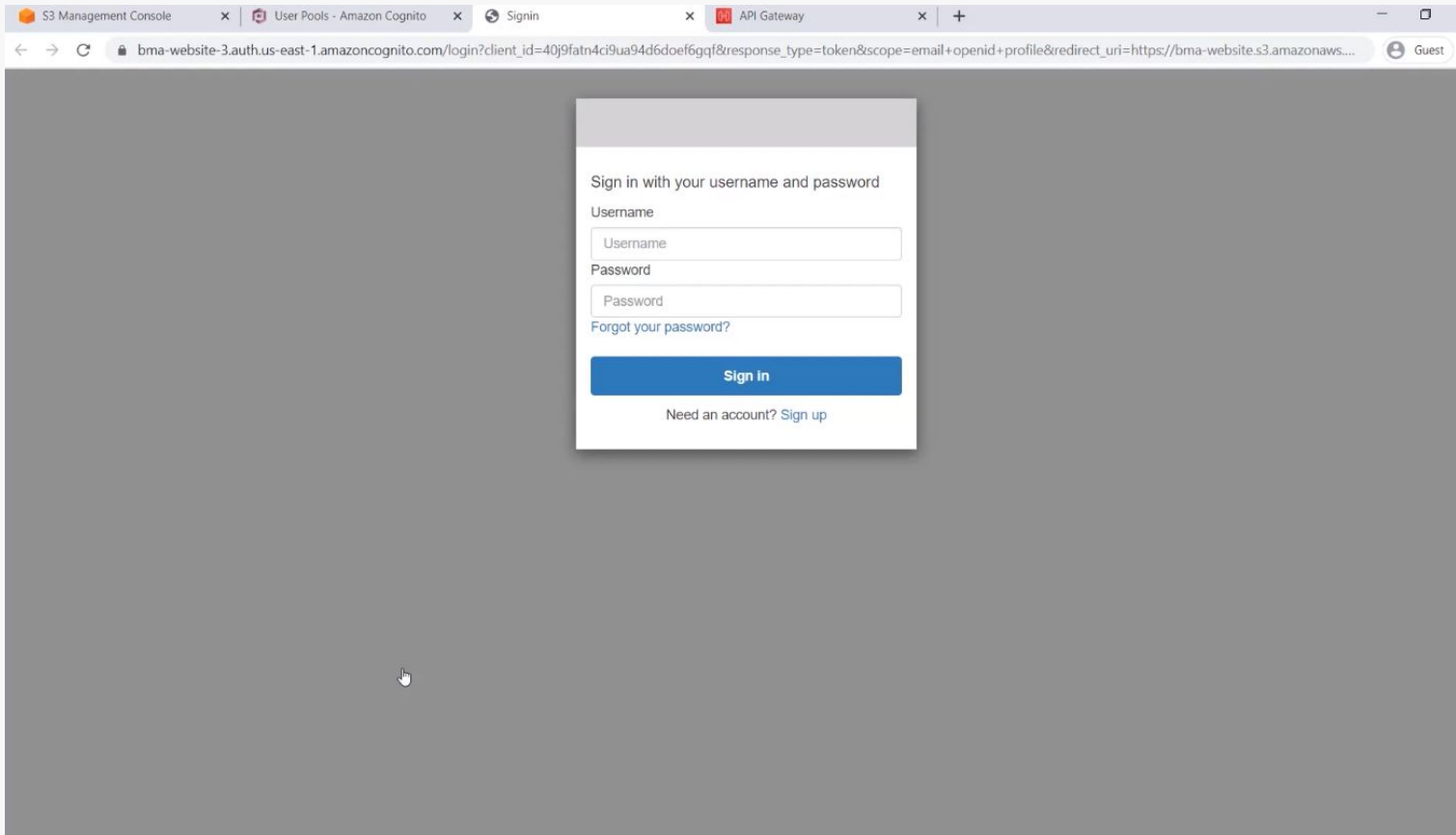
The hosted UI provides an OAuth 2.0 authorization server with built-in webpages that can be used to sign up and sign in users using the domain you created. [Learn more about the hosted UI](#)

**Launch Hosted UI** 

[Go to summary](#) [Choose domain name](#)



You can test UI here



This allows users to sign in and sign up (won't need to code).



If it does not look pretty according to your standards,  
remember that you can modify all of that.

Now that we have this login page,  
we can use it in our index.html file.

Because that's where we want to point our users to  
when the user click on the sign in button.

```
File Edit Selection View Go Run Terminal Help index.html - Visual Studio Code
index.html x callback.html
C:\Users> jotdi > Desktop > Cognito-demo > <> index.html > html > body > a.button
1 <html>
2 <head>
3 <title>Demo Cognito</title>
4 <style>
5 .button {
6 background-color: #1c87c9;
7 border: none;
8 color: white;
9 padding: 20px 34px;
10 text-align: center;
11 text-decoration: none;
12 display: inline-block;
13 font-size: 20px;
14 margin: 4px 2px;
15 cursor: pointer;
16 }
17 </style>
18 </head>
19 <body>
20 <a href="#" class="button">Sign In</a>
21 </body>
22 </html>
```

This is the index.html.

```
File Edit Selection View Go Run Terminal Help index.html - Visual Studio Code
<> index.html x <> callback.html
C:\Users> jotdi > Desktop > Cognito-demo > <> index.html > html > body > a.button
1 <html>
2 <head>
3 <title>Demo Cognito</title>
4 <style>
5 .button {
6 background-color: #1c87c9;
7 border: none;
8 color: white;
9 padding: 20px 34px;
10 text-align: center;
11 text-decoration: none;
12 display: inline-block;
13 font-size: 20px;
14 margin: 4px 2px;
15 cursor: pointer;
16 }
17 </style>
18 </head>
19 <body>
20 <a href="https://bma-website-3.auth.us-east-1.amazoncognito.com/login?client_id=40j9fatn4ci9ua94d6doef6gqf&response_type=token&scope=email+openid+profile&redirect_uri=https://bma-website.s3.amazonaws.com/callback.html" class="button">Sign In</a>
21 </body>
22 </html>
```

Link the button to the login page url.

Now we can upload this index.html file back into that S3.



Amazon S3 &gt; bma-website

## bma-website

Overview

Properties

Permissions

Management

Access points

Type a prefix and press Enter to search. Press ESC to clear.



Upload



Create folder

Download

Actions

US East (N. Virginia)



Viewing 1 to 1

<input type="checkbox"/>	Name	Last modified	Size	Storage class
<input type="checkbox"/>	callback.html	Jul 16, 2020 1:56:41 AM GMT-0400	813.0 B	Standard

Viewing 1 to 1

Operations

0 In progress

1 Success

0 Error



Feedback



English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

### Upload

1 Files | Size: 676.0 B | Target path: bma-website

1 Select files | 2 Set permissions | 3 Set properties | 4 Review

#### Manage users

User ID	Objects	Object permissions	
willismt+3(Owner)	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write	X

Access for other AWS account [+ Add account](#)

Account	Objects	Object permissions	
---------	---------	--------------------	--

#### Manage public permissions

Grant public read access to this object(s)

**⚠ This object(s) has public read access.**  
Everyone in the world will have read access to this object(s).

[Upload](#) [Previous](#) [Next](#)

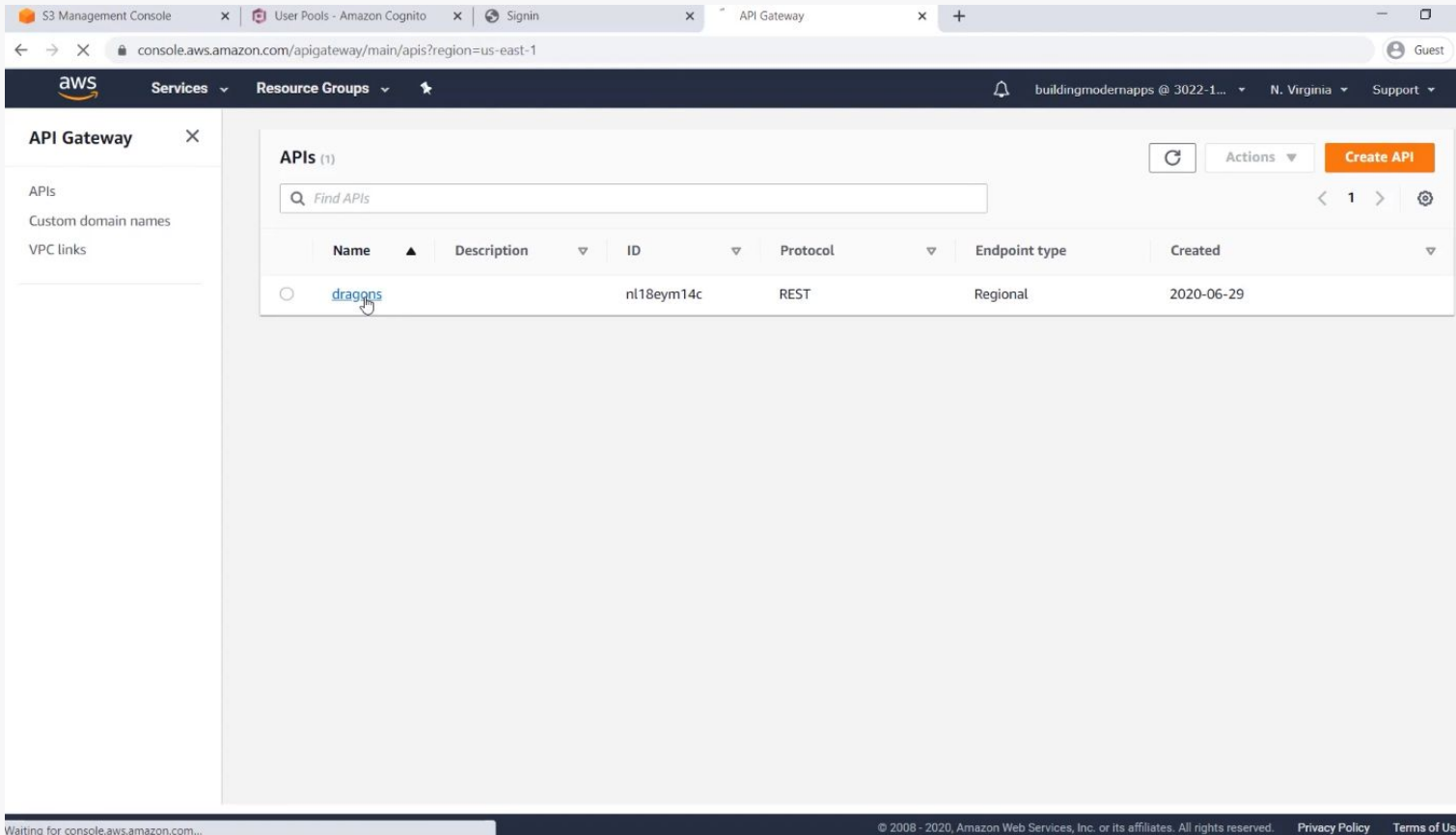
Now everything with S3 and Cognito is done.



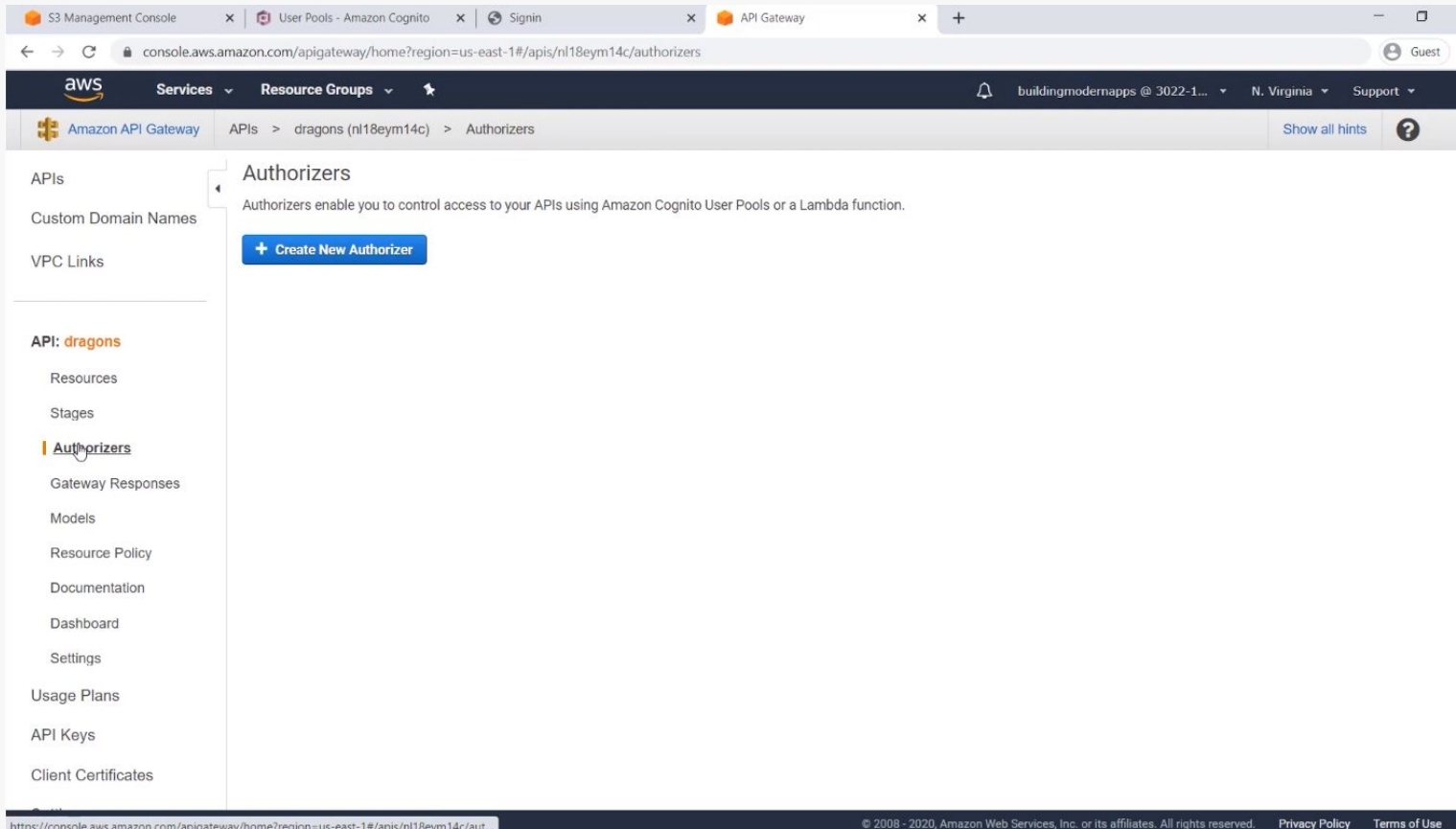


recap

The last step is to configure an authorizer  
in API Gateway and modify Get/dragons



API Gateway console



# Create New Authorizer

The screenshot shows the AWS API Gateway console interface. The breadcrumb navigation indicates the path: Amazon API Gateway > APIs > dragons (nl18eym14c) > Authorizers. The main heading is 'Authorizers', with a sub-heading explaining that they enable access control using Amazon Cognito User Pools or Lambda functions. A blue button labeled '+ Create New Authorizer' is visible. Below this is a 'Create Authorizer' form with the following fields:

- Name \***: A text input field containing 'demo'.
- Type \***: Radio buttons for 'Lambda' and 'Cognito', with 'Cognito' selected.
- Cognito User Pool \***: A dropdown menu showing 'us-east-1' and a text input field containing 'demo'.
- Token Source \***: An empty text input field.
- Token Validation \***: An empty text input field.

At the bottom of the form are 'Create' and 'Cancel' buttons. The left sidebar contains navigation links for various API Gateway features, with 'Authorizers' highlighted. The footer includes 'Feedback', 'English (US)', and copyright information for Amazon Web Services, Inc. (2008-2020).

Add the name of our Cognito User Pool. In our example, it was demo.

The screenshot shows the AWS API Gateway console interface. The breadcrumb navigation indicates the path: Amazon API Gateway > APIs > dragons (nl18eym14c) > Authorizers. The main heading is 'Authorizers', with a sub-heading explaining that they control access to APIs using Amazon Cognito User Pools or a Lambda function. A blue button labeled '+ Create New Authorizer' is visible. Below this is a 'Create Authorizer' form with the following fields:

- Name \***: A text input field containing 'demo'.
- Type \***: Two radio buttons, 'Lambda' (unselected) and 'Cognito' (selected).
- Cognito User Pool \***: A dropdown menu showing 'us-east-1' and 'demo'.
- Token Source \***: A text input field containing 'Authorization'.
- Token Validation**: An empty text input field.

At the bottom of the form are 'Create' and 'Cancel' buttons. The left sidebar contains a navigation menu with 'API: dragons' selected, and other options like Resources, Stages, Authorizers, Gateway Responses, Models, Resource Policy, Documentation, Dashboard, Settings, Usage Plans, API Keys, and Client Certificates. The footer includes 'Feedback', 'English (US)', and copyright information for Amazon Web Services, Inc. (2008-2020).

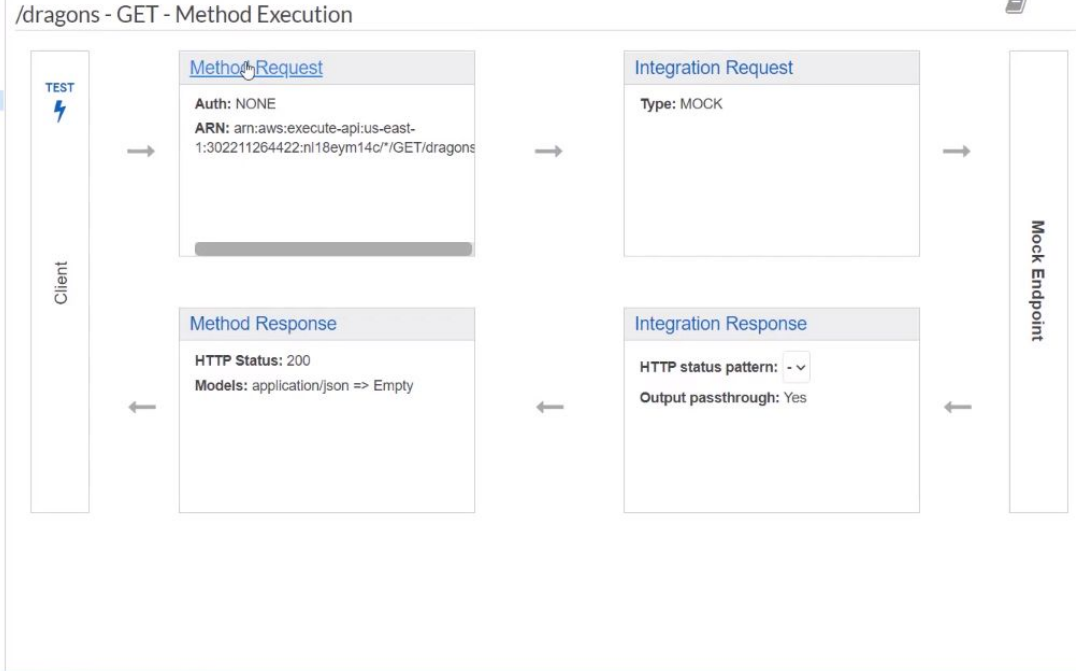
Now we need to tell API Gateway what will be the name of the HTTP header that will contain the JWT token that will be passed to it. For this application, use the authorization header.

The next step is to modify our resources that we have for this API.

- APIs
- Custom Domain Names
- VPCLinks
- API: dragons
  - Resources
  - Stages
  - Authorizers
  - Gateway Responses
  - Models
  - Resource Policy
  - Documentation
  - Dashboard
  - Settings
- Usage Plans
- API Keys
- Client Certificates

Resources Actions

- /
- /dragons
  - GET
  - POST





- APIs
- Custom Domain Names
- VPC Links

- Resources Actions
- /
  - dragons
    - GET
    - POST

- API: dragons
- Resources
    - Stages
    - Authorizers
    - Gateway Responses
    - Models
    - Resource Policy
    - Documentation
    - Dashboard
    - Settings
  - Usage Plans
  - API Keys
  - Client Certificates

### Method Execution /dragons - GET - Method Request

Provide information about this method's authorization settings and the parameters it can receive.

#### Settings

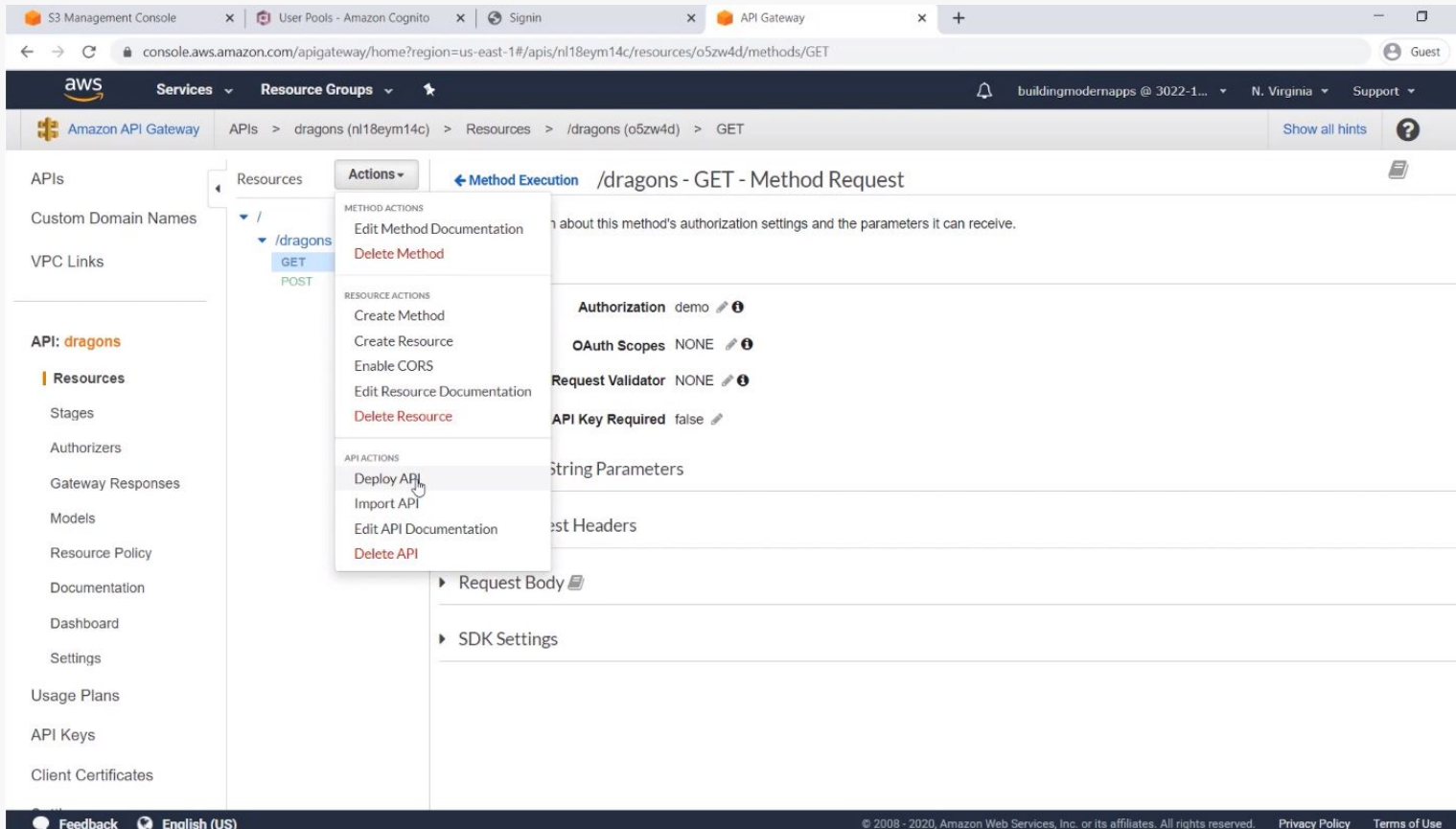
Authorization	NONE
Request Validator	NONE
API Key Required	AWS_IAM
	Cognito user pool authorizers
	demo

▶ URL Query String Parameters

▶ HTTP Request Headers

▶ Request Body

▶ SDK Settings



## Deploy API

Done!

It's time to test it.

First, let show that via using Postman without authentication,  
it's not going to allow us in.



Open Launchpad

Learn how to debug requests and perform manual testing [Start](#) X

GET https://nl18eym14c.execute-apl... + ... No Environment

Untitled Request Comments


GET https://nl18eym14c.execute-api.us-east-1.amazonaws.com/test Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies


Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Response



Hit Send to get a response

 Learn how to debug requests and perform manual testing [Start](#) X

GET https://nl18eym14c.execute-apl... + ... No Environment

Untitled Request Comments

GET https://nl18eym14c.execute-api.us-east-1.amazonaws.com/test/dragons Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...
Key	Value	Description	Bulk

Body Cookies Headers (7) Test Results Status: 401 Unauthorized Time: 193 ms Size: 299 B Save Response

```
1 {  
2   "message": "Unauthorized"  
3 }
```

Status 401 (unauthorized)

i.e. I'm not authorized to access to this API.

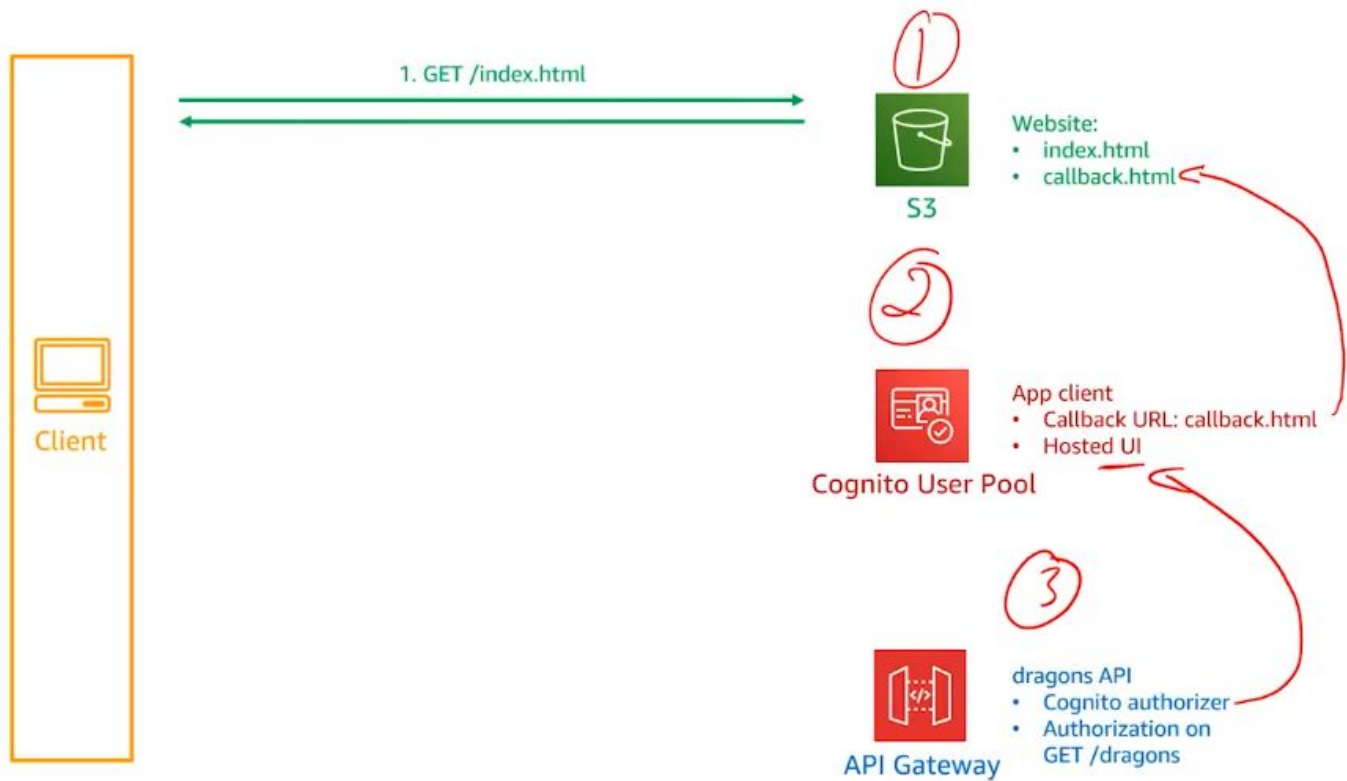
Perfect!

That's exactly what we wanted.

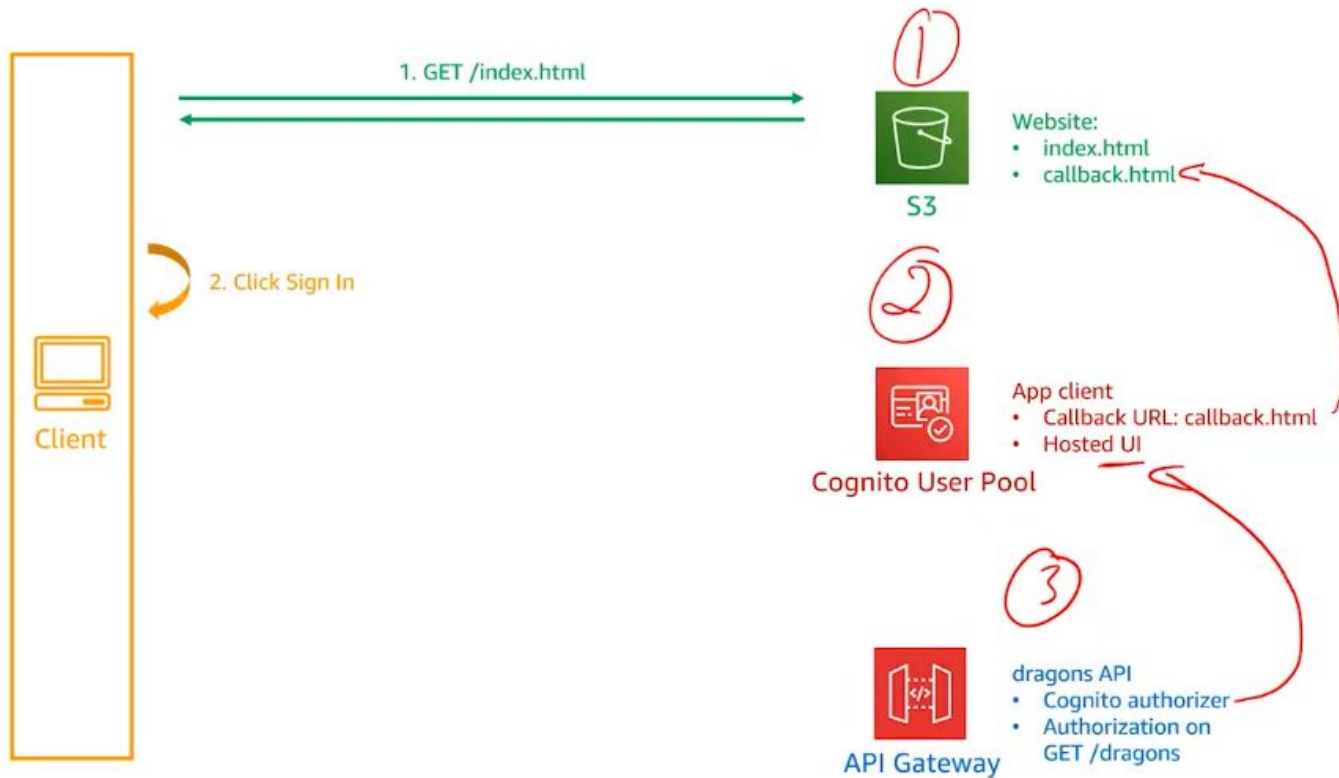


Let go through the flow of authentication and authorization  
all of these tools that we have been using so far.

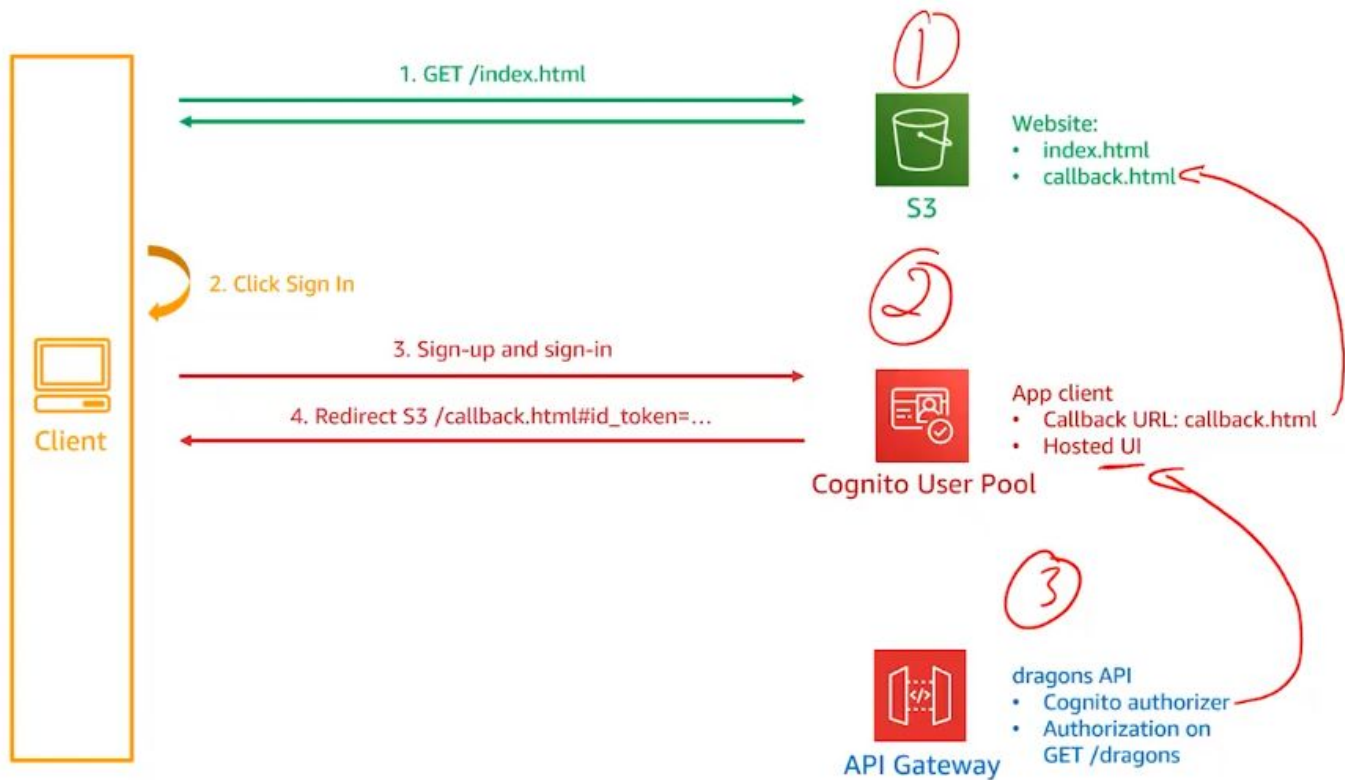




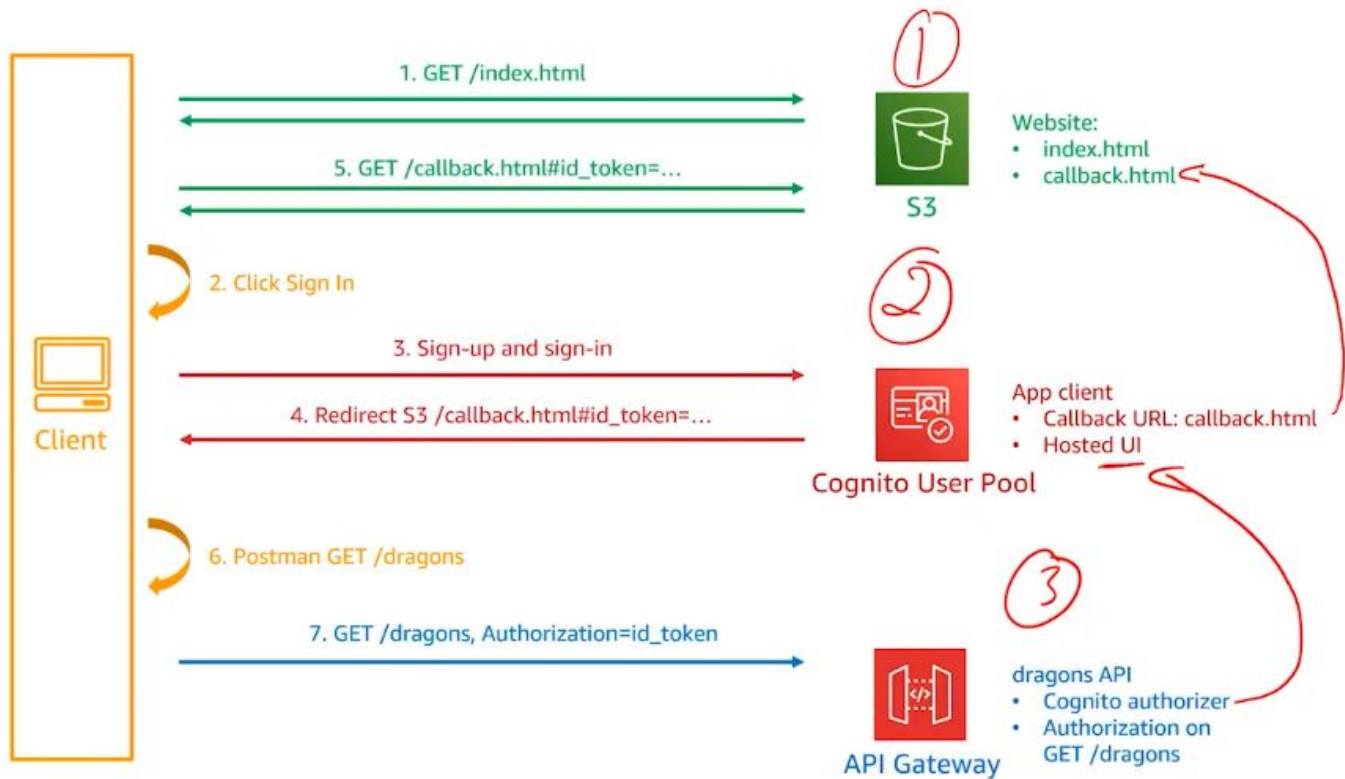
First, the client will send a GET requests for the index page inside of S3.



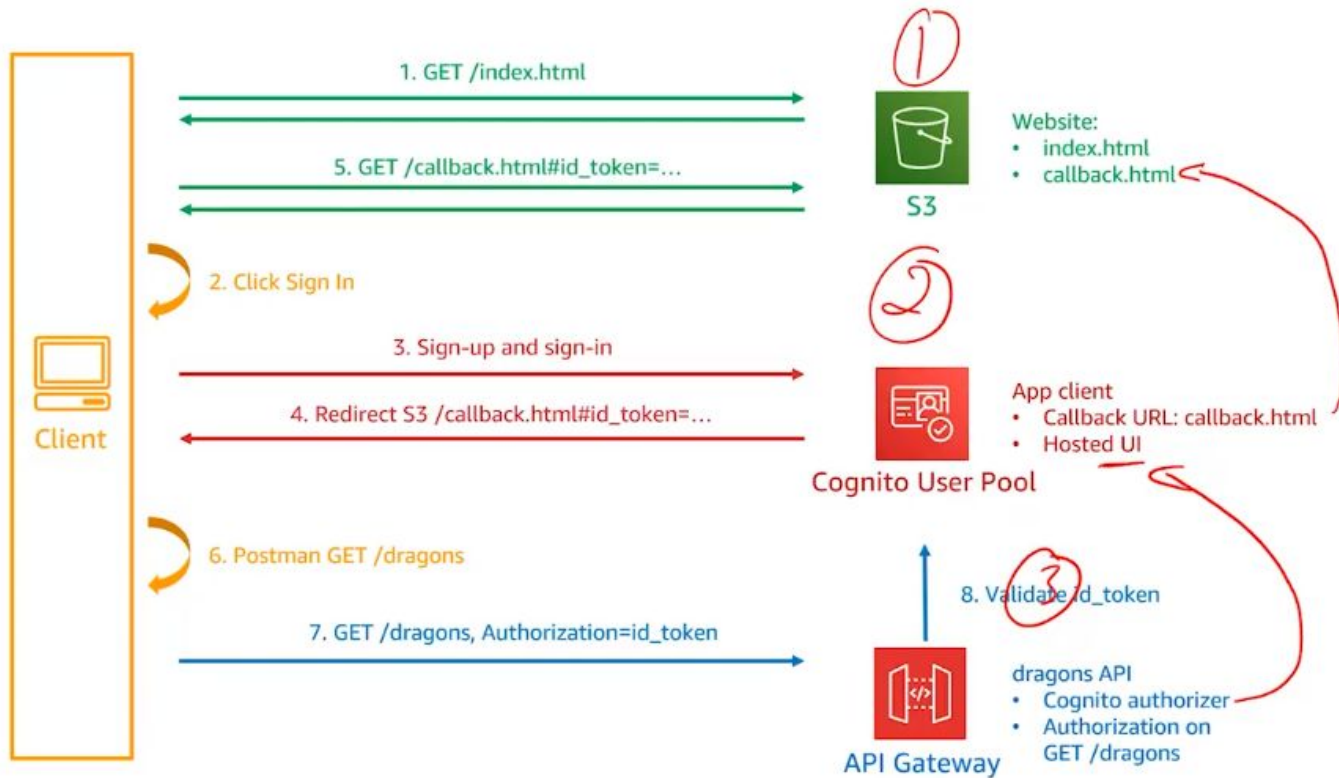
Then the client will click the sign in link inside of that page that they received,



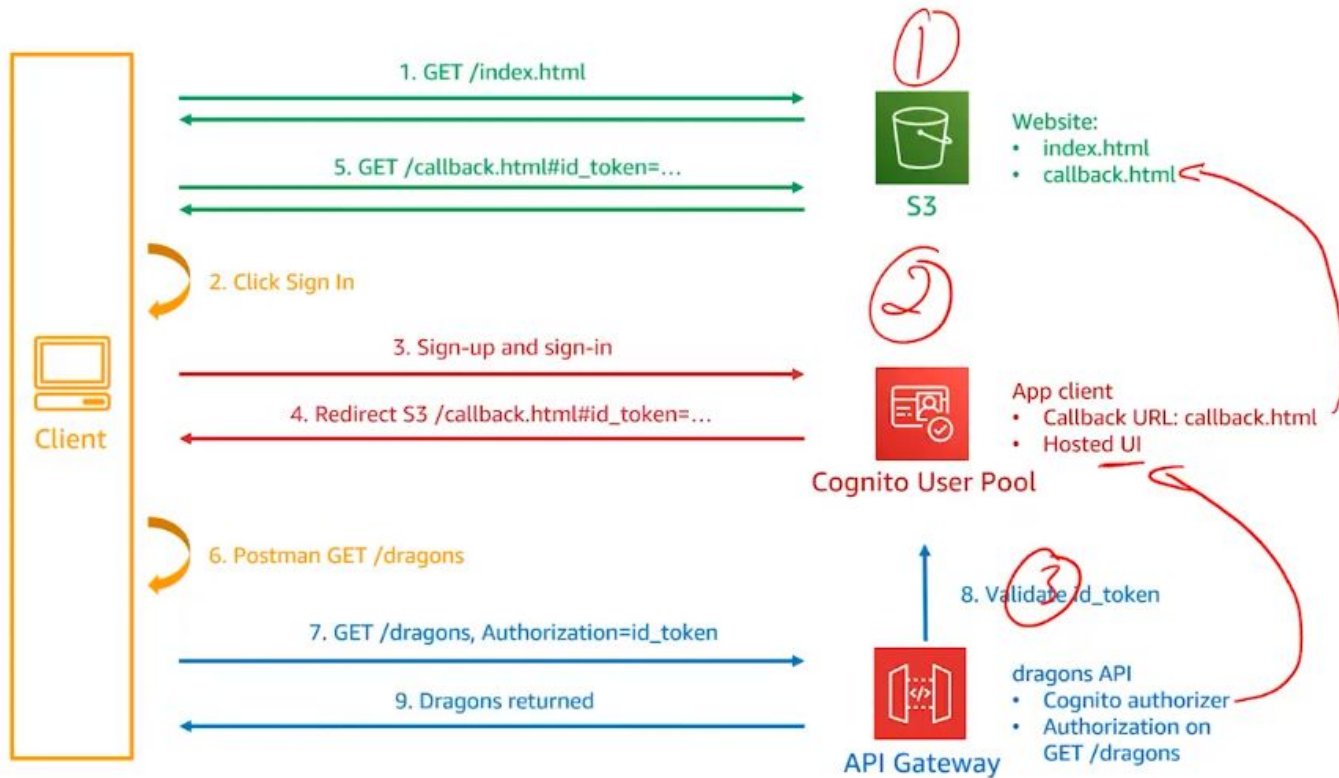
When the client will sign in (or after sign up), the hosted UI will send a redirect back to the client, pointing to the S3 callback.html page with the JWT token inside of it.



Then the client will take that token and use Postman to execute a GET/dragon, passing the authorization, and then that will be sent into API Gateway.

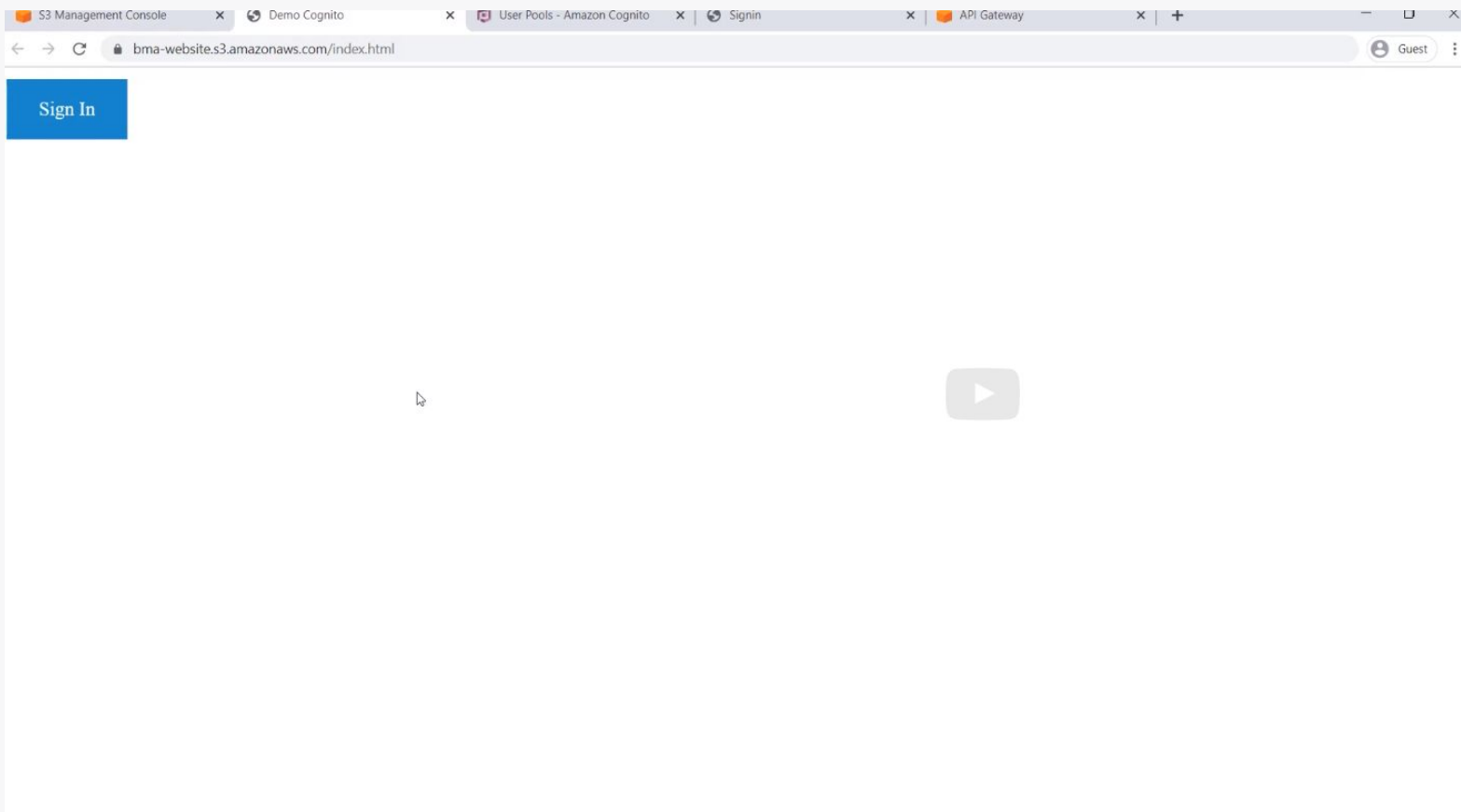


Then API Gateway will validate that token with Cognito User Pool.



if that's successful, it will return the dragons.





Get the index url from S3.

Sign in with your username and password

Username

Password

[Forgot your password?](#)

[Sign in](#)

Need an account? [Sign up](#)

Sign in with your username and password

Username

Password

[Forgot your password?](#)

**Sign in**

Need an account? [Sign up](#)

Sign up with a new account

**Username**

**Email**

**Password**

[Sign up](#)

Already have an account? [Sign in](#)

We have sent a code by email to j\*\*\*@a\*\*\*.com.  
Enter it below to confirm your account.

Verification Code

**Confirm Account**

Didn't receive a code? [Resend it](#)



We have sent a code by email to j\*\*\*@a\*\*\*.com.  
Enter it below to confirm your account.

Verification Code

**Confirm Account**

Your verification code - Message (HTML)

File Message Insert Options Format Text Review Help Tell me what you want to do

no-reply@verificationemail.com | Dion, Jonathan | 12:57 AM

**Your verification code**

Your confirmation code is 739366



```
File Edit Selection View Go Run Terminal Help callback.html - Visual Studio Code
<> index.html <> callback.html X
C:\Users> jotdi > Desktop > Cognito-demo > <> callback.html > html > body
2 <head>
3 <title>Demo Cognito</title>
4 <style>
5     span {
6         display: block;
7         word-wrap: break-word;
8         width: 500px;
9         white-space: normal;
10        padding: 10px;
11        color: grey;
12        background-color: white;
13        border: black 2px solid;
14    }
15 </style>
16 </head>
17
18 <body>
19     <div class="add-info">
20         The token to authenticate is: <span id="token"></span>
21     </div>
22
23     <script>
24         (function init() {
25             var id_token_str = window.location.hash.split("&access_token=")[0];
26             var authorization_str = id_token_str.replace("#id_token=", "");
27             document.getElementById("token").innerHTML = authorization_str;
28         })();
29     </script>
30 </body>
31 </html>
```

callback.html



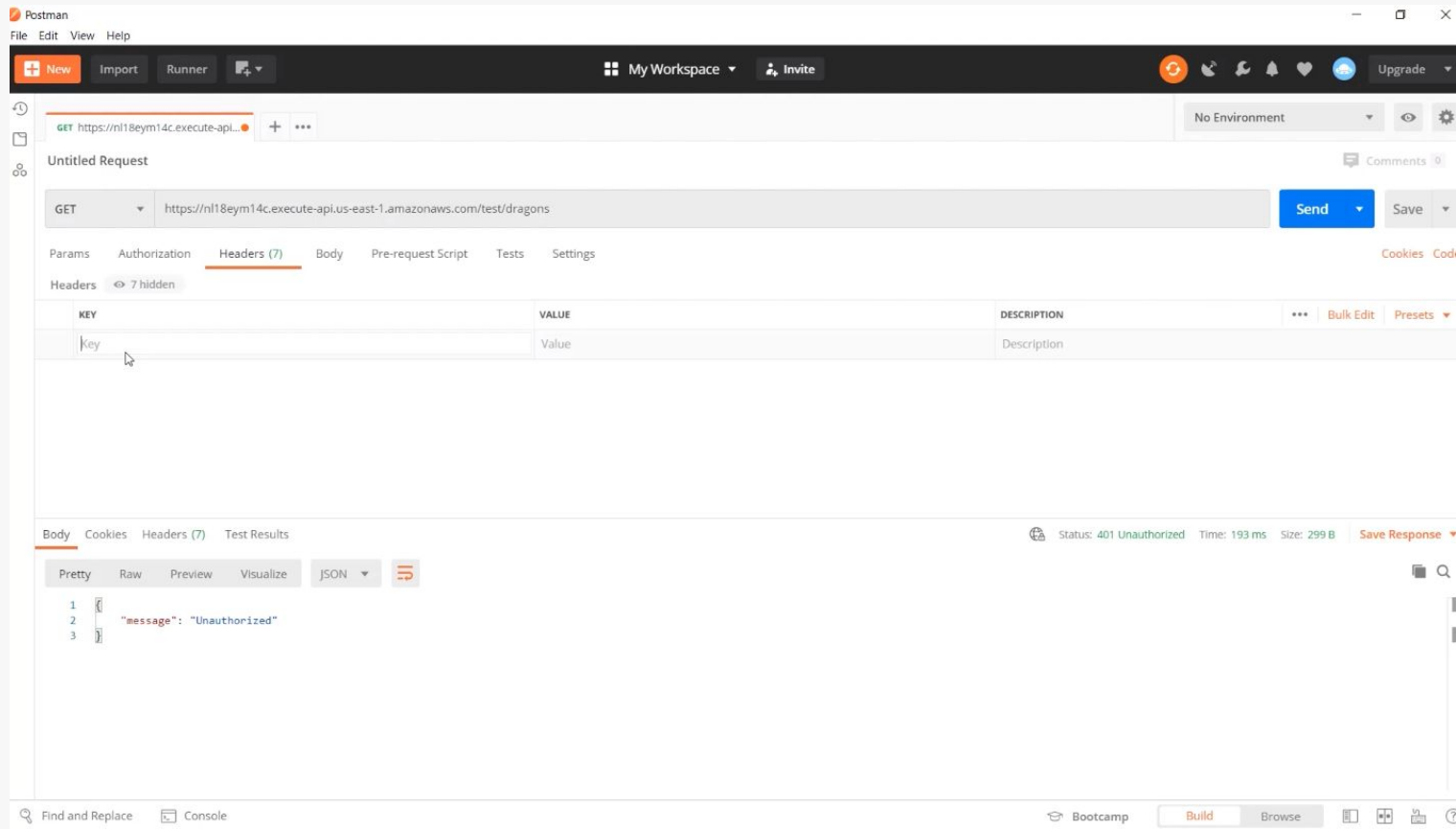
Let's use this JWT token inside a request via Postman.

The screenshot shows the AWS Management Console interface for the Amazon API Gateway service. The breadcrumb navigation indicates the path: **APIs** > **dragons (nl18eym14c)** > **Authorizers**. The main heading is **Authorizers**, with a sub-heading explaining that authorizers control access to APIs using Amazon Cognito User Pools or a Lambda function. A blue button labeled **+ Create New Authorizer** is visible. Below this, a table lists the existing authorizers:

Authorizer Name	Authorizer ID	Token Source	Token Validation
<b>demo</b>	3hqx6i	<b>Cognito User Pool</b> demo - w2Aolw50y (us-east-1)	-

At the bottom of the table row, there are **Edit** and **Test** buttons. The **Token Source** column is highlighted, and a mouse cursor is pointing at the **Authorization** text within the **demo** row.

Recap - (we selected token source as authorization. That's the name of the header.)



So inside of Postman, the same request we had, under headers, add a new key and they called that authorization, then add a value, which is the token that we copied.

Postman

File Edit View Help

New Import Runner + My Workspace Invite Upgrade

GET https://ml18eym14c.execute-api... No Environment

### Untitled Request

GET https://ml18eym14c.execute-api.us-east-1.amazonaws.com/test/dragons Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Code

Headers 7 hidden

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Authorization	eyJraWQlOjZlbnQzZGRlFwRmV2TGhOkMrMzZlTWNnZVlwYVwxCjcklobzR3ZiZQjAGllwYWxmjoUlMyNTYlIQ_ajhdF9oYXNoIjoIR50tSEFOcIFlN1YydTFfOZV4M3hjdylnN1YlI6ImlwMzdhZWI4LTl0ODQhNDJlZS1hY2ExLWQ1MDdmNDAxYWwMwZlsl	
Key		Description

Body Cookies Headers (7) Test Results Status: 401 Unauthorized Time: 193 ms Size: 299 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Unauthorized"
3 }
```

Find and Replace Console Bootcamp Build Browse

Postman

File Edit View Help

Now Import Runner My Workspace Invite Upgrade

GET https://ml18eym14c.execute-api... No Environment

### Untitled Request

GET https://ml18eym14c.execute-api.us-east-1.amazonaws.com/test/dragons Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies Code

Headers 7 hidden

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Authorization	eyJraWQjOjZGRVFWwRmV2TGVhQkMrMzZlTWNnZVJwVWwXC9jcklobzR3Z...	
Key	Value	Description

Body Cookies Headers (7) Test Results Status: 200 OK Time: 196 ms Size: 1.85 KB Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "description_str": "From the northern fire tribe, Atlas was born from the ashes of his fallen father in combat. He is fearless and does not fear battle.",
4     "dragon_name_str": "Atlas",
5     "family_str": "red",
6     "location_city_str": "anchorage",
7     "location_country_str": "usa",
8     "location_neighborhood_str": "w fireweed ln",
9     "location_state_str": "alaska"
10  },
11  {
12    "description_str": "Protheus is a wise and ancient dragon that serves on the grand council in the sky world. He uses his power to calm those near him.",
13    "dragon_name_str": "Protheus"
```

Done

We went through the entire flow of how to authenticate a client's request to an API Gateway using a Cognito User Pool.

## Lab 3 - API Authentication

### Exercise 3: Amazon Cognito Authentication

In this lab, you will continue to build the Dragons application in your AWS account.

You will start by upgrading your application programming interface (API) and application to use Amazon Cognito user pools for authentication.

You will then create the user pool and update the REST API to require authentication.

Finally, you will deploy an updated version of the application, which sends the user to a sign-in webpage.

[Exercise 3: Amazon Cognito Authentication](#)